

Pentest-Report ExpressVPN Android Client App & Integrations 08.2022

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Inführ, MSc. S. Moritz, BSc. C. Kean, Dipl.-Ing. A. Aranguren

Index

[Introduction](#)

[Scope](#)

[Severity Glossary](#)

[Table of Findings](#)

[Identified Vulnerabilities](#)

[EXP-10-001 WP1: Potential information leakage via absent security screen \(Low\)](#)

[EXP-10-002 WP1: Potential phishing via StrandHogg 2.0 on Android \(Medium\)](#)

[EXP-10-005 WP1: Several exported activities facilitate DoS \(Medium\)](#)

[Miscellaneous Issues](#)

[EXP-10-003 WP1: Insecure v1 signature support on Android \(Info\)](#)

[EXP-10-004 WP1: Absent root detection \(Info\)](#)

[EXP-10-006 WP1: Absent integrity protection for cipher texts \(Info\)](#)

[EXP-10-007 WP1: Potential clipboard tampering via exported activity \(Low\)](#)

[EXP-10-008 WP1: Android binary hardening recommendations \(Info\)](#)

[EXP-10-009 WP2: App crash via absent protocol verification \(Info\)](#)

[EXP-10-010 WP1: cleartextTrafficPermitted flag enabled for third-party domain \(Info\)](#)

[EXP-10-011 WP2: Overly-permissive URI parsing in autofill feature \(Info\)](#)

[EXP-10-012 WP1: Potential IP address correlation via third-party service \(Info\)](#)

[EXP-10-013 WP1: App storage HTTP cache uncleaned upon logout \(Info\)](#)

[Conclusions](#)

Introduction

“A VPN, or virtual private network, adds a layer of security between your Android and the internet. In addition to encrypting your online activity and protecting your personal information from third-party interception, ExpressVPN can also help you defeat censorship by making you appear to be in a different country.”

From <https://www.expressvpn.com/vpn-software/vpn-android>

This report - titled EXP-10 - details the scope, results, and conclusory summaries of a penetration test and source code audit against ExpressVPN's Android client and integrations with a particular focus on ExpressVPN Keys (the password manager integrated within ExpressVPN's mobile apps). The work was requested by ExpressVPN in June 2022 and initiated by Cure53 in August 2022, namely in CW31 and CW32. A total of nineteen days were invested to reach the coverage expected for this project. The testing conducted for EXP-10 was divided into three separate work packages (WPs) for execution efficiency, as follows:

- **WP1:** Source-code-assisted penetration tests against ExpressVPN Android application
- **WP2:** Source-code-assisted penetration tests against ExpressVPN Keys, ExpressVPN's password manager
- **WP3:** Source-code audits and reviews against VPN protocol integration and dependencies

To ensure a smooth audit, Cure53 was granted access to all Android mobile application binaries and integrations, plus all underlying source code. For these purposes, the methodology chosen was white box and a team of five senior testers was assigned to the project's preparation, execution, and finalization. All preparatory actions were completed in July 2022, namely in the week prior to testing (CW30), to ensure the review could proceed without hindrance or delay.

Communications were facilitated via a dedicated, shared Slack channel deployed to combine the workspaces of ExpressVPN and Cure53, thereby creating an optimal collaborative working environment. All participatory personnel from both parties were invited to partake throughout the test preparations and discussions.

In light of this, communications proceeded smoothly on the whole. The scope was well-prepared and transparent, no noteworthy roadblocks were encountered throughout testing, and cross-team queries remained minimal as a result. The developer team delivered excellent test preparation and assisted the Cure53 team in every respect to procure maximum coverage and depth levels for this exercise.

Cure53 gave frequent status updates concerning the test and any related findings, whilst simultaneously offering prompt queries and receiving efficient, effective answers from the maintainers. Top-line information relating to all pertinent findings was shared during the active testing phase in addition.

Regarding the findings in particular, the Cure53 team achieved comprehensive coverage over the WP1 through WP3 scope items, identifying a total of thirteen. Three of the findings were categorized as security vulnerabilities, whilst the remaining ten were deemed general weaknesses with lower exploitation potential. Generally speaking, despite the relatively high yield of findings, the overall impression gained by the testing team following this engagement is adequately positive. This primarily owes to the fact that the vast majority of findings are variations of common misconfigurations that are often present in Android applications.

This positive viewpoint is also corroborated by the fact that none of the aforementioned vulnerabilities can be directly abused to conduct successful attacks. As a result, leakage of sensitive information such as the user's clear-text IP address or any other form of de-anonymization was effectively deterred. Naturally, some alternative and significantly less impactful vectors documented in this report remain equally valid for review and mitigation, such as the various DOS vulnerabilities detailed in ticket [EXP-10-005](#) whereby exported activities are at fault. However, the general robustness of the underlying codebase and associated integration is undoubtedly high.

Nevertheless, all guidance offered should be considered and implemented to elevate the already strong security foundation to an exemplary and industry-leading standard.

The report will now shed more light on the scope and testing setup as well as provide a comprehensive breakdown of the available materials. Subsequently, the report will list all findings identified in chronological order, starting with the detected vulnerabilities and followed by the general weaknesses unearthed. Each finding will be accompanied by a technical description and Proof of Concepts (PoCs) where applicable, plus any relevant mitigatory or preventative advice to action.

In summation, the report will finalize with a conclusion in which the Cure53 team will elaborate on the impressions gained toward the general security posture of ExpressVPN's Android client and integrations in focus, giving high-level hardening advice where applicable.

Scope

- **Code audits and security assessments against ExpressVPN's Android client and integrations**
 - **WP1:** Source-code-assisted penetration tests against ExpressVPN Android application
 - **Primary audit focus:**
 - ExpressVPN Android applications.
 - Version: *v10.48.0.10480040.473546*.
 - **In-scope items:**
 - Remote Code Execution (RCE) vulnerabilities.
 - Privilege escalation vulnerabilities.
 - Identification of sensitive information disclosure findings, including client IP address leakage and DNS leakage, is in scope.
 - **Out-of-scope items:**
 - Build dependencies, build scripts, or internal development tools.
 - Code and dependencies used for tests (i.e. mocks, end to end (e2e) tests).
 - In-depth analysis of shared packages between applications, specifically *xvclient* and *lightway-core*.
 - Other third party dependencies included in the application.
 - Code not relevant to the Android application targeting other platforms (e.g. iOS, Windows, AirCove, Linux, macOS).
 - Testing of API servers. This includes the Password Manager APIs, the VPN APIs and VPN servers (both Lightway and OpenVPN).
 - Any exploit vector with the prerequisite that the device is rooted or any bypass vector for the root detection.
 - **WP2:** Source-code-assisted penetration tests against integrated password manager
 - **Primary audit focus:**
 - Keys a.k.a. password manager.
 - **In-scope items:**
 - Weaknesses in the autofill feature.
 - Incorrect domain matching through weaknesses in URL parsing.
 - Access to the secure vault via any remote means including IPC mechanisms.
 - Extraction/modification of secrets stored in the password manager vault.
 - **WP3:** Source-code audits and reviews against VPN protocol integration and dependencies
 - **Secondary audit focus:**
 - Core Lightway library / libhelium - "libheliumvpn.so"
 - xvclient - "libxvclient.so"
 - OpenVPN - "libopenvpnexec.so" and "libopenvpnutil.so"



Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53

Bielefelder Str. 14

D 10709 Berlin

cure53.de · mario@cure53.de

- **In-scope items:**
 - Usage of these dependencies and frameworks are in scope, but evaluated from a black-box perspective.
- **Test-user accounts were created and activated for the auditing team**
- **All binaries in scope were shared with Cure53**
- **Test-supporting material was shared with Cure53**
- **All relevant sources were made available for Cure53**

Severity Glossary

The following section details the varying severity levels assigned to the issues discovered in this report.

Critical: The highest possible severity level. Categorizes issues that allow attackers to achieve extensive access to sensitive areas, such as critical systems, applications, data or other pertinent components in scope.

High: Categorizes issues that allow attackers to achieve limited access to sensitive areas in scope. This also includes issues with limited exploitability that can facilitate a significant impact upon the target in scope.

Medium: Categorizes issues that do not incur major impact on the areas in scope. Additionally, issues requiring a more limited exploitation are graded as *Medium*.

Low: Categorizes issues that have a highly limited impact on the areas in scope. Mostly does not depend on the level of exploitation but rather on the minor severity of obtainable information or lower grade of damage targeting the areas in scope.

Info: Categorizes issues considered merely informational in nature. They are mostly considered as hardening recommendations or improvements that can generally enhance the security posture of the areas in scope.

Table of Findings

Identified Vulnerabilities

ID	Title	Severity
EXP-10-001	WP1: Potential information leakage via absent security screen	<i>Low</i>
EXP-10-002	WP1: Potential phishing via StrandHogg 2.0 on Android	<i>Medium</i>
EXP-10-005	WP1: Several exported activities facilitate DoS	<i>Medium</i>

Miscellaneous Issues

ID	Title	Severity
EXP-10-003	WP1: Insecure v1 signature support on Android	<i>Info</i>
EXP-10-004	WP1: Absent root detection	<i>Info</i>
EXP-10-006	WP1: Absent integrity protection for cipher texts	<i>Info</i>
EXP-10-007	WP1: Potential clipboard tampering via exported activity	<i>Low</i>
EXP-10-008	WP1: Android binary hardening recommendations	<i>Info</i>
EXP-10-009	WP2: App crash via absent protocol verification	<i>Info</i>
EXP-10-010	WP1: cleartextTrafficPermitted flag enabled for third-party domain	<i>Info</i>
EXP-10-011	WP2: Overly-permissive URI parsing in autofill feature	<i>Info</i>
EXP-10-012	WP1: Potential IP address correlation via third-party service	<i>Info</i>
EXP-10-013	WP1: App storage HTTP cache uncleared upon logout	<i>Info</i>

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified throughout the testing period. Please note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is given in brackets following the title heading for each vulnerability. Furthermore, each vulnerability is given a unique identifier (e.g., *EXP-10-001*) to facilitate any future follow-up correspondence.

EXP-10-001 WP1: Potential information leakage via absent security screen (*Low*)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

The Android app implements a security screen for the login screen only. However, the discovery was made that it fails to render a security screen when backgrounded on all other application screens. This allows attackers with physical access to an unlocked device to peruse and enumerate data displayed by the app before backgrounded. A malicious app or an attacker with physical access to the device could leverage this weakness to gain access to user information, such as VPN subscription details.

To replicate this issue, simply navigate to a screen displaying sensitive information, then send the application to the background. Subsequently, open the app and observe that one can read any information displayed. Notably, the text will remain readable even following a device reboot.

The following screenshot demonstrates the presence of a potential leak depending on the previous screen shown by the application and prior to being backgrounded:

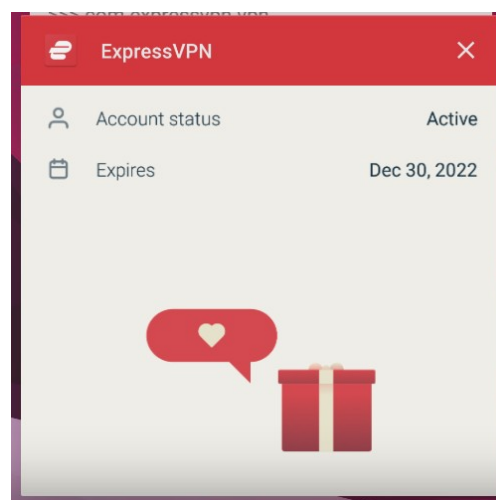


Fig.: Potential leakage via absent security screen.

The root cause of this issue originates from a lack of code within the Android app that captures background events to implement a security screen, which in turn explains the absence of a security screen displayed. This can be confirmed by searching globally for Android events in the source code provided, as well as the decompiled Android APK:

Command:

```
egrep -Ir '(onActivityPause|ON_PAUSE)' * |egrep -v  
"(androidx|google|android/support|launchdarkly)" |wc -l
```

Output:

0

To mitigate this issue, Cure53 recommends rendering a security screen overlay when the app is due to be backgrounded. In Android apps, this can be accomplished by implementing a security screen, thereby capturing the relevant backgrounding events; typically, *onActivityPause*¹ or the *ON_PAUSE* Lifecycle event² are used for such purposes. Subsequently, if possible, ensure that all views set the Android *FLAG_SECURE* flag³. This will guarantee that even apps running with root privileges are unable to directly capture information displayed by the app on screen. Alternatively, the base activity inherited by all application activities could be amended to always set this flag, regardless of the focus⁴.

¹ <https://developer.android.com/.../Application.ActivityLifecycleCallbacks#onActivityPaused...>

² <https://developer.android.com/reference/androidx/lifecycle/Lifecycle.Event>

³ http://developer.android.com/reference/android/view/Display.html#FLAG_SECURE

⁴ <https://gist.githubusercontent.com/jonaskuiler/.../raw/.../MainActivity.java>

EXP-10-002 WP1: Potential phishing via StrandHogg 2.0 on Android (Medium)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Testing confirmed that the Android app is currently vulnerable to a number of task hijacking attacks. The *launchMode* for the app-launcher activity is currently set to *singleTop*⁵, which mitigates task hijacking via *StrandHogg*⁶ and other older techniques documented since 2015⁷, while rendering the app vulnerable to *StrandHogg 2.0*⁸. This vulnerability affects Android versions 3-9.x⁹ but was only patched by Google on Android 8-9¹⁰. Since the app supports devices from Android 5 (API level 21), this renders all users running Android 5-7.x vulnerable, as well as users running unpatched Android 8-9.x devices, which remains common in the modern era.

A malicious app could leverage this weakness to manipulate the way in which users interact with the app. Specifically, this could be instigated by relocating a malicious attacker-controlled activity in the screen flow of the user, which may be useful toward instigating phishing or Denial-of-Service attacks, as well as user-credential theft. Notably, this issue has been exploited by a number of banking malware Trojans in the past¹¹.

Malicious applications typically exploit task hijacking by instigating one or a selection of the following techniques:

- **Task Affinity Manipulation:** The malicious application has two activities M1 and M2 wherein *M2.taskAffinity = com.victim.app* and *M2.allowTaskReparenting = true*. If the malicious app is opened on M2, M2 is relocated to the front and the user will interact with the malicious application once the victim application has initiated.
- **Single Task Mode:** If the victim application sets *launchMode* to *singleTask*, malicious applications can use *M2.taskAffinity = com.victim.app* to hijack the victim's application task stack.

⁵ <https://developer.android.com/guide/topics/manifest/activity-element#lmode>

⁶ <https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/>

⁷ <https://s2.ist.psu.edu/paper/usenix15-final-ren.pdf>

⁸ <https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/>

⁹ <https://www.xda-developers.com/strandhogg-2-0-android-vulnerability-explained-developer-mitigation/>

¹⁰ <https://source.android.com/security/bulletin/2020-05-01>

¹¹ <https://arstechnica.com/.../...fully-patched-android-phones-under-active-attack-by-bank-thieves/>

- **Task Reparenting:** If the victim application sets *taskReparenting* to *true*, malicious applications can move the victim's application task to the malicious application's stack.

However, in relation to StrandHogg 2.0, all exported activities without a *launchMode* of *singleTask* or *singleInstance* are affected on vulnerable Android versions¹². This issue can be confirmed by reviewing the *AndroidManifest* of the Android application.

Affected file:

AndroidManifest.xml

Affected code:

```
<application android:theme="@style/Fluffer_AppTheme"
android:label="@string/app_name" android:icon="@mipmap/ic_launcher"
android:name="com.expressvpn.vpn.ApplicationInstance"
android:allowBackup="false" android:supportsRtl="true"
android:banner="@drawable/fluffer_tv_banner" android:extractNativeLibs="true"
android:resizeableActivity="true"
android:networkSecurityConfig="@xml/network_security_config"
android:roundIcon="@mipmap/ic_launcher_round"
android:appComponentFactory="androidx.core.app.CoreComponentFactory">
[...]
    <activity android:theme="@style/Fluffer_AppTheme.Splash"
android:name="com.expressvpn.vpn.ui.SplashActivity" android:exported="true"
android:launchMode="singleTop" android:screenOrientation="behind"
android:configChanges="orientation">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Based on the above, we can deduce that the *launchMode* is set to *singleTop* and the *taskAffinity* property is not set at application or activity level. To provide additional clarity on this issue, an example of a malicious app was created to demonstrate the potential exploitability.

PoC Demo:

https://7as.es/ExpressVPN_FuF2X2zbW2/TaskHijacking_PoC.mp4

To mitigate this issue, one can recommend implementing as many of the following countermeasures as deemed feasible by the development team:

- The task affinity of exported application activities should be set to an empty string in the Android manifest. This will force the activities to use a randomly-generated

¹² <https://www.xda-developers.com/strandhogg-2-0.../>

task affinity rather than the package name. This would successfully prevent task hijacking, as malicious apps will not have a predictable task affinity to target.

- The `launchMode` should then be altered to `singleInstance` (rather than `singleTask`). This will ensure continuous mitigation in *StrandHogg 2.0*¹³ while improving security strength against older task hijacking techniques¹⁴.
- A custom `onBackPressed()` function could be implemented to override the default behavior.
- The `FLAG_ACTIVITY_NEW_TASK` should not be set in `activity launch` intents. If deemed required, one should use the aforementioned in combination with the `FLAG_ACTIVITY_CLEAR_TASK` flag¹⁵.

Affected file:

AndroidManifest.xml

Proposed fix:

```
<activity android:theme="@style/Fluffer_AppTheme.Splash"
android:name="com.expressvpn.vpn.ui.SplashActivity" android:exported="true"
android:launchMode="singleInstance" android:taskAffinity=""
android:screenOrientation="behind" android:configChanges="orientation">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

EXP-10-005 WP1: Several exported activities facilitate DoS (Medium)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Whilst auditing the ExpressVPN Android app's exported components, testing confirmed that sending a specially-crafted intent to the `SplashActivity`, `MagicUrlLoginActivity`, `OneLinkActivity`, and `CopyToClipboardActivity` exported activities causes the app to crash. This facilitates a scenario whereby malicious applications installed on the device can send crafted intents to the Android app in order to instigate a permanent crash. This would effectively prevent users from a prolonged engagement with the product.

PoC:

The following code snippets demonstrate the method by which one can send a serialized dummy Java object as an intent, resulting in an application crash.

¹³ <https://www.xda-developers.com/strandhogg-2-0-android-vulnerability-explained.../>

¹⁴ <http://blog.takemyhand.xyz/2021/02/android-task-hijacking-with.html>

¹⁵ <https://www.slideshare.net/phdays/android-task-hijacking>

Serializable class example:

```
import java.io.Serializable;

public class SerializableTest implements Serializable {
    private static final long serialVersionUID = 1L;
    boolean b;
    short i;
}
```

The following code highlights an example implementation of the *SerializableTest* class, which sends the intent to the ExpressVPN app's *MagicUrlLoginActivity* every ten seconds.

```
package com.example.maliciousapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ComponentName;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.widget.Toast;
import com.example.maliciousapp.SerializableTest;

public class MainActivity extends AppCompatActivity {

    Handler handler = new Handler();
    Runnable runnable;
    int delay = 10000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        AsyncTask.execute(new Runnable() {
            @Override
            public void run() {
                handler.postDelayed(runnable = new Runnable() {
                    public void run() {
                        handler.postDelayed(runnable, delay);
                        Toast.makeText(MainActivity.this, "This method is run every
                            10 seconds", Toast.LENGTH_SHORT).show();

                        // send serializable object via Intent
                        Intent intent = new Intent();
                        intent.setComponent(new
                            ComponentName("com.expressvpn.vpn", "com.expressvpn.vpn.ui.
```

```
        user.MagicUrlLoginActivity"));
        intent.putExtra("test", new SerializableTest());
        startActivity(intent);
    }
    }, delay);
}
});
}
}
```

Log excerpt:

2022-08-01 12:04:51.763 6093-6093/**com.expressvpn.vpn** E/AndroidRuntime: **FATAL**

EXCEPTION: main

```
Process: com.expressvpn.vpn, PID: 6093
java.lang.RuntimeException: Unable to start activity
ComponentInfo{com.expressvpn.vpn/com.expressvpn.vpn.ui.user.MagicUrlLoginActivit
y}: java.lang.RuntimeException: Parcelable encountered ClassNotFoundException
reading a Serializable object (name = com.example.maliciousapp.SerializableTest)
[...]
```

ExpressVPN keeps stopping



-  App info
-  Close app

Fig. Android crash dialog.

Notably, initiating activities from backgrounded applications is only possible on API level 28 and below. On newer Android versions, intents can only be sent if the app is in the foreground¹⁶. In addition, the issue is unable to deanonymize ExpressVPN users since the VPN connection persists following any app crash.

Steps to reproduce:

1. Create a new Android application and integrate the example implementation provided above.
2. Open the malicious app and send it to the background (best on Android 9).
3. Open the ExpressVPN app and observe the subsequent crash following the intent sent from the malicious app.

¹⁶ <https://developer.android.com/guide/components/activities/background-starts>



Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53

Bielefelder Str. 14

D 10709 Berlin

cure53.de · mario@cure53.de

To mitigate this issue, Cure53 advises correctly validating the data received via intents in order to ensure that intents received by the exported activities cannot result in a crash of the ExpressVPN app. This would ensure that any scenario whereby a malicious application attempts to cause the application to crash by sending an intent is avoided completely. Additionally and where possible, exported activities should be protected with Android permissions to ensure permitted applications only can invoke them.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not lead to an exploit but might assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

EXP-10-003 WP1: Insecure v1 signature support on Android ([Info](#))

Note from ExpressVPN: *All applications that are available on Android versions lower than Android 7.0 are vulnerable to this issue due to a weakness in the Android OS. While users on older versions of Android are encouraged to upgrade to a newer, more secure Android version, we recognize that not all users are able to do so. ExpressVPN will continue to be available to these users to ensure they have access to VPN protection. We've also added a security warning to make users aware of the issues related to using an older Android OS.*

Testing confirmed that the Android build provided for testing is signed with an insecure v1 APK signature. Usage of the v1 signature increases the application's susceptibility to the known Janus¹⁷ vulnerability on devices running Android versions lower than 7. This issue allows attackers to smuggle malicious code into the APK without breaking the signature. At the time of writing, the app supports a minimum SDK of 21 (Android 5), which also uses the v1 signature, hence remaining vulnerable to this type of attack. Furthermore, Android 5 devices no longer receive updates and are vulnerable to a host of well-known security issues. One can therefore assume that any malicious app installed may trivially gain root privileges on those devices via public exploits^{18 19 20}.

The existence of this flaw means that attackers could manipulate users into installing a malicious attacker-controlled APK matching the v1 APK signature of the legitimate Android application. As a result, a transparent update would be possible without any warning displayed, effectively taking over the existing application and all of its data.

To mitigate this issue, Cure53 recommends increasing the minimum supported SDK level to at least 24 (Android 7) to ensure that this known vulnerability cannot be exploited on devices running older and deprecated Android versions. In addition, future production builds should only be signed with an APK signature constituting v2 and greater.

¹⁷ <https://www.guardsquare.com/en/blog/new-android-vulnerability-allows-atta....affecting-their-signatures>

¹⁸ <https://www.exploit-db.com/exploits/35711>

¹⁹ <https://github.com/davidqphan/DirtyCow>

²⁰ https://en.wikipedia.org/wiki/Dirty_COW

EXP-10-004 WP1: Absent root detection ([Info](#))

Note from ExpressVPN: *The ExpressVPN Android application will continue to support users who choose to root their device. We've added a warning for users who choose to do this, so that they are made aware of the potentially increased security issues that arise from a rooted device.*

The discovery was made that the Android app does not currently implement any root detection feature on the majority of screens at the time of testing, with the Keys view representing the only exception. Hence, the applications fail to alert users concerning the security implications of operating the app in an environment of this nature²¹. This issue can be confirmed by installing the application on a rooted device and validating the complete lack of application warnings.

To mitigate this issue, Cure53 recommends implementing a root detection solution. Please note that, since the user has root access and the application does not, the application would always remain at a disadvantage in this scenario. Mechanisms such as these should always be considered bypassable in the hands of a skilled and dedicated attacker.

The freely available *rootbeer* library²² for Android could be considered for the purpose of alerting users on rooted devices. Whilst still bypassable, this would be sufficient to warn users of any associated risk of running the app on rooted devices.

EXP-10-006 WP1: Absent integrity protection for cipher texts ([Info](#))

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

During the deep-dive review of the sources, the confirmation was made that the Android app utilizes AES encryption in conjunction with the CBC padding algorithm. This mode of encryption is known to be weak against so-called Padding Oracle attacks²³. However, during the testing phase, no functioning oracle could be detected in order to be able to exploit the issue.

Affected file:

```
xv_android7/sharedandroid/src/main/java/com/expressvpn/sharedandroid/utills/  
KeystoreCrypt.kt
```

²¹ <https://www.bankinfosecurity.com/jailbreaking-ios-devices-risks-to-users-enterprises-a-8515>

²² <https://github.com/scottyab/rootbeer>

²³ <https://jiang-zhenghong.github.io/blogs/PaddingOracle.html>

Affected code:

```
private const val AES_MODE = "AES/CBC/PKCS7Padding"
[...]
object Api23 {
    fun encrypt(data: String, keyAlias: String): String {
        val key = getKey(keyAlias)
        val cipher = Cipher.getInstance(AES_MODE).apply {
            init(Cipher.ENCRYPT_MODE, key)
        }
    }
}
```

To mitigate this issue, Cure53 recommends utilizing a mode of operation that supports authenticated encryption such as GCM. This can be achieved via the *AES/GCM/NoPadding* algorithm, which is supported by the imported *javax.crypto.Cipher* package. Alternatively, if CBC mode is preferred, cipher texts can be protected via an HMAC or a signature to ensure the prevention of data tampering.

EXP-10-007 WP1: Potential clipboard tampering via exported activity (Low)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Testing confirmed that the ExpressVPN Android app exposes an activity that allows third-party apps to force the app into storing arbitrary data in its clipboard. A malicious app may leverage this weakness to disrupt users that utilize the clipboard. This issue can be confirmed by running the following ADB command:

ADB command:

```
adb shell am start -n
"com.expressvpn.vpn/com.expressvpn.vpn.util.CopyToClipboardActivity" --es
"android.intent.extra.TEXT" "test"
```

Affected file:

*xv_android7/ExpressVPNMobile/src/main/java/com/expressvpn/vpn/util/
CopyToClipboardActivity.java*

Affected code:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    String receivedText = getIntent().getStringExtra(Intent.EXTRA_TEXT);
    if (receivedText != null) {
        copyTextToClipboard(receivedText);
    }
}
```

To mitigate this issue, Cure53 recommends halting the export of these activities, since it is not specifically required by the application.

EXP-10-008 WP1: Android binary hardening recommendations ([Info](#))

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Testing confirmed that a number of binaries embedded into the Android application do not currently leverage the available compiler flags to mitigate potential memory corruption vulnerabilities. This unnecessarily increases the application's susceptibility to risk via these associated issues.

Issue 1: Lack of `-D_FORTIFY_SOURCE=2` usage

The absence of this flag means that common *libc* functions lack buffer overflow checks, which in turn renders the application prone to memory corruption vulnerabilities. The following is a reduced list of example binaries for the sake of brevity.

Example binaries (from decompiled production app):

```
lib/x86_64/libopvpnutil.so  
lib/arm64-v8a/libopvpnutil.so  
lib/x86/libopvpnutil.so
```

Issue 2: Absent binary stack canary

Testing confirmed that some binaries do not integrate a stack canary value to the stack. Stack canaries are leveraged to detect and prevent exploits from overwriting return addresses.

Example binaries (from decompiled app):

```
lib/x86_64/libpmcore.so  
lib/x86_64/libcrashlytics-trampoline.so  
lib/armeabi-v7a/libcrashlytics-trampoline.so  
lib/arm64-v8a/libpmcore.so  
lib/arm64-v8a/libcrashlytics-trampoline.so
```

To mitigate this issue, Cure53 recommends compiling all binaries using the `-D_FORTIFY_SOURCE=2` argument to ensure that common insecure *glibc* functions such as *memcpy* are automatically protected by buffer overflow checks.

EXP-10-009 WP2: App crash via absent protocol verification (*Info*)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

The password manager integrated in the ExpressVPN application allows users to store websites and associated credentials for autofill purposes in the web browser. In the eventuality a malformed URL is clicked in the password manager view, the application crashes.

Whilst testing this feature, the discovery was made that the app does not restrict the protocol schemes of the specified URL. In the eventuality a URL specifying an arbitrary protocol scheme is clicked, the application sends an internal *Intent* object, which is not handled by any activity. This behavior not only halts the app but also causes a disconnection of any established VPN tunnel.

Despite this impact, the issue was only assigned an *Info* severity rating, since the testing team could not locate any attack vector to inject an arbitrary URL into the password manager without the user's consent.

Steps to reproduce:

1. Open the Keys tab in the ExpressVPN application.
2. Click to add an additional entry.
3. Enter all the necessary details into the form and specify a website URL such as *abcd://test.com*, for example.
4. Save it.
5. Click the URL specified in Step 3, which should be displayed as a blue hyperlink.
6. Observe that the app crashes.

Affected file:

xv_android7/features/password-manager/src/main/java/com/expressvpn/pwm/ui/PasswordDetailScreen.kt

Affected code:

```
websiteEvent?.let { state ->
    val intent = Intent(Intent.ACTION_VIEW, Uri.parse(state.website))
    LocalContext.current.startActivity(intent)
}
```

Logcat output:

```
08-04 10:29:51.425 1247 1247 E AndroidRuntime: FATAL EXCEPTION: main
08-04 10:29:51.425 1247 1247 E AndroidRuntime: Process: com.expressvpn.vpn,
PID: 1247
```

```
08-04 10:29:51.425 1247 1247 E AndroidRuntime:  
android.content.ActivityNotFoundException: No Activity found to handle Intent  
{ act=android.intent.action.VIEW dat=javascript://alert(133) }  
08-04 10:29:51.425 1247 1247 E AndroidRuntime: at  
android.app.Instrumentation.checkStartActivityResult(Instrumentation.java:2007)
```

To mitigate this issue, Cure53 recommends implementing an allow list of protocol schemes that are supported by the password manager. This would prevent the aforementioned exception without harming usability.

EXP-10-010 WP1: *cleartextTrafficPermitted* flag enabled for third-party domain ([Info](#))

Note from ExpressVPN: *After attempts to use the HTTPS captive portal for Apple earlier this year to support network connectivity checks, we noted that the HTTPS captive portal provided by Apple is not stable, and as such, is unable to provide the required network check functionality. As a result of countries regularly enforcing blocks against domains we operate, we rely on Apple domains to ensure that network connectivity checks are reliable, and as such, we are forced to use the HTTP version of Apple's captive portal.*

Whilst reviewing the Android production binary, the observation was made that the Network Security Configuration defines clear-text permissions for certain third-party domains. This allows the app to connect to these domains without TLS. The impact of this issue was merely considered *Info* since the listed domains do not appear to comprise the ExpressVPN communications containing sensitive information.

Affected file:

expressvpn_android_10.48.0.10480040.473546_release_playstore/res/4u.xml

Affected code:

```
captive.apple.com\00cleartextTrafficPermitted\00connectivitycheck.gstatic.com\  
00domain\00domain-config\00includeSubdomains\00network-security-config
```

Affected file:

common/src/main/java/com/expressvpn/captiveportal/CaptivePortalChecker.kt

Affected code:

```
private const val APPLE_CAPTIVE_PORTAL_CHECK_URL = "http://captive.apple.com"
```

To mitigate this issue, Cure53 recommends setting the *cleartextTrafficPermitted* directive to *false*, since this will block requests without TLS to listed domains. The list can be expanded to include all domains used by the app domains, which would enforce TLS connections for a wider defense-in-depth implementation.

EXP-10-011 WP2: Overly-permissive URI parsing in autofill feature (*Info*)

Fix Note: *The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Whilst conducting an extensive review of the password manager's autofill feature to determine the presence of any issues in its domain matcher, the discovery was made that the current logic does not take the protocol or port into consideration. This behavior could potentially cause information leakage via an incorrect origin detection.

Allowing arbitrary protocol schemes whilst parsing an origin can potentially incur risk owing to *pseudo* protocols such as *about:*. This owes to the fact that the scheme is considered the same origin as the domain it was actually opened by, which the password manager would remain unaware of. The potential for information leakage via incorrect domain parsing is therefore raised.

Nevertheless, this issue was merely rated *Info* since modern browsers are sufficiently strict regarding the usage of pseudo protocols and the origin associated with them, therefore effectively preventing any leakage.

Affected file:

xv_android7/features/password-manager/src/main/java/com/expressvpn/pwm/autofill/AutoFillDomainMatcher.kt

Affected code:

```
internal fun String.getUrlHost(): String {
    return try {
        URI(this).host
    } catch (ex: Throwable) {
        this
    }
}
```

Example URL:

about://test.com/

Returned value:

test.com

Similarly to the guidance offered in ticket [EXP-10-009](#), Cure53 recommends implementing an allow list of protocol schemes or adapting the current logic to include the protocol in the domain matcher functionality.

EXP-10-012 WP1: Potential IP address correlation via third-party service (Info)

Fix Note: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.

The ExpressVPN Android app leverages the *LaunchDarkly* SDK for feature flags and throttled rollout. This causes the app to send a unique device fingerprint containing a unique identifier, device information, and the ExpressVPN app version to the third-party service. In addition, the confirmation was made via network sniffing that the *LaunchDarkly* API is contacted both with and without VPN connection. Thus, *LaunchDarkly* could infer from its HTTP access logs that a specific device with a real IP address is operating under a VPN IP address for a certain duration of time.

Affected file:

`com/launchdarkly/sdk/android/e0.java`

Affected code:

```
static final Uri C = Uri.parse("https://clientsdk.launchdarkly.com");  
static final Uri D = Uri.parse("https://mobile.launchdarkly.com");  
static final Uri E = Uri.parse("https://clientstream.launchdarkly.com");
```

The full request details can be observed either by reviewing the HTTP cache or the local storage.

Affected file:

`com.expressvpn.vpn/cache/com.launchdarkly.http-cache/0f6486f8161aba0ce146fe7c7cf40e2c.0`

Affected content:

```
https://clientsdk.launchdarkly.com/msdk/evalx/users/  
eyJhbm9ueW1vdXMiOmZhbHN1LCJrZXkiOiJkNDI5MmExMy04YzcyOjRlR1NTItOWRmMC1kYjZhMGFmNzI1MDciLCJjdXN0b20iOjV2Y2xpZW50X29zIjojYW5kcm9pZCI9ImFwcF92ZXJzaW9uIjojMTAuNDguMCI9Im9zIjozMiwizGV2aWN1IjojUGl4ZWwgNiBvcmlvbGUifX0=  
GET  
2  
Authorization: api_key mob-e04a4355-a5f5-46ea-b501-8e0006293fd5
```

Decoded JSON token:

```
{  
  "anonymous": false,  
  "key": "79b90403e5cdbe29bd1c62515897f20b001b8cfefab2f8e8a20aa0d25d93def",  
  "custom": {  
    "client_os": "android",  
    "app_version": "10.48.0",  
    "os": 32,  
    "device": "Pixel 6 oriole"  
  }  
}
```

To mitigate this issue, Cure53 recommends only using the *LaunchDarkly* API whilst connected to the VPN and only collecting information that is absolutely necessary for

feature management. In addition, one should consider configuring *LaunchDarkly* users as anonymous users, which is explicitly specified in the *LaunchDarkly* documentation²⁴.

This guidance will ensure that the device and IP address fingerprint created at *LaunchDarkly* is kept to a minimum. Alternatively, to reduce information sharing with third parties altogether, the ExpressVPN team could transition the entire feature management to a self-hosted and open-source solution²⁵.

EXP-10-013 WP1: App storage HTTP cache uncleared upon logout (*Info*)

Fix Note: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.

Testing confirmed that the ExpressVPN app does not completely wipe the local app storage following user logout. The information that persists in storage contains device-identifying information such as the *LaunchDarkly* token used during the user session.

The impact of this issue was considered *Info* since obtaining this information would require physical device access.

Affected file:

`com.expressvpn.vpn/cache/com.launchdarkly.http-cache/Of6486f8161aba0ce146fe7c7cf40e2c.0`

Affected content:

```
https://clientsdk.launchdarkly.com/msdk/evalx/users/  
eyJhbm9ueW1vdXMiOmZhbHN1LCJrZXkiOiJkNDI5MmExMy04YzcyLmR1NTI1tOWRmMC1kYjZhMGFmNzI1  
MDciLCJjdXN0b20iOnsiY2xpZW50X29zIjoIYW5kcm9pZCIiImFwcF92ZXJzaW9uIjoIbWtAuNDguMCIiS  
Im9zIjozMiwizGV2aWN1IjoIUGl4ZWwgNiBvcmlvbGUiX0=  
GET  
2  
Authorization: api_key mob-e04a4355-a5f5-46ea-b501-8e0006293fd5
```

To mitigate this issue, Cure53 recommends clearing the entire local-app storage post user logout to ensure that passwords or device-identifying information such as the *LaunchDarkly* session token cannot persist between user sessions.

²⁴ <https://docs.launchdarkly.com/home/users/anonymous-users>

²⁵ <https://featureflags.io/resources/>

Conclusions

The impressions gained during this report - which details and extrapolates on all findings identified during the CW31 and CW32 testing against ExpressVPN's Android client and integrations by the Cure53 team - will now be discussed at length. To summarize, the confirmation can be made that the components under scrutiny have garnered a sufficiently strong impression, though a host of general hardening and best practice measures can be implemented to elevate the scope to a first-rate standard.

This positive viewpoint primarily owes to the fact the average impact marker hovers between *Info* and *Medium* across all identified findings. This outcome provides ample evidence that the ExpressVPN team is not only acutely aware of the many problems that modern VPN applications tend to face, but also able to effectively counter them.

Generally speaking, the Android application provided a number of praiseworthy impressions during this assignment. The Android Keystore is correctly utilized to protect sensitive data on the device, whilst the lack of issues in relation to client-side injections also informs the strong security foundation observed.

The website shortcut feature was also subject to deep-dive evaluation to determine the presence of any Denial-of-Service vectors or user-tracking potential, since it fetches the HTML and favicon of a third-party website. Ultimately, no associated issues were detected in this feature.

Additionally, multiple DoS attempts against the VPN failed during this assignment, indicating that the application is resilient against attacks such as DNS spoofing. The deployed VPN configuration was also carefully assessed for any issues that may allow packets to escape the established tunnel. The fact that these attempts were largely unsuccessful confirms the ExpressVPN team's dedication toward safeguarding user-related information, which is also reflected in the soundly and transparently composed codebase.

However, the integration of third-party services facilitated additional privacy and security considerations regarding local storage (see [EXP-10-013](#)) as well as data in transit (see [EXP-10-010](#) and [EXP-10-012](#)).

Particular scrutiny was also placed on validating all privacy implications for the end user when relying on third-party services, since external privacy policies may significantly differ from ExpressVPN's strict no-logs policy.

In this regard, one should consider relinquishing third-party services altogether in favor of a self-hosted privacy ecosystem under the full control of ExpressVPN that can be configured to align with ExpressVPN's privacy policy.

Furthermore, the assessment of the ExpressVPN app's exported components revealed the presence of a selection of security weaknesses, as documented in tickets [EXP-10-002](#) and [EXP-10-005](#). Cure53 recommends addressing these issues to sufficiently protect ExpressVPN users against exploitation in real-world scenarios.

Elsewhere, the implemented password manager was subject to extensive examination by the testing team, with the deployed origin validation a particular area of interest. Here, two minor issues were identified within the functionality (see [EXP-10-009](#) and [EXP-10-011](#)), both of which are related to arbitrary protocol schemes.

The ExpressVPN Keys password manager was also deemed resilient against Unicode-related origin confusion attacks, although modern browsers already provide adequate protection against this type of attack by default. Given the importance of login credentials to the overall framework, the security of the data at rest was rigorously assessed. In this regard, the cryptographic functions utilized by the password manager to store the credentials garnered a solid impression on the whole.

Finally, the integration of libraries utilized by ExpressVPN was extensively reviewed. This included any storage of configuration files in insecure areas, file permission issues, parameter injection, path traversal, certificate validation in the utilized HTTP client, and plenty more potential issues. As reflected in this report, no issues were unearthed in this area, which is an exceptionally positive outcome for the developer team.

In general, Cure53 recommends that all issues identified in this report - including those considered minor and assigned *Informational* to *Low* severity ratings - are addressed where possible at the earliest possible convenience. This will not only strengthen the security posture of the platform, but also reduce the volume of tickets in future security engagements.

Moving forward, Cure53 also strongly advises conducting recurrent security assessments against the components in question, ideally at least once a year and/or prior to the rollout of significant framework alterations. This proven approach will ensure that both existing vulnerabilities and issues are sufficiently addressed, as well as ensure that newly-introduced functionalities cannot incur fresh vulnerabilities and attack vectors.

Cure53 would like to thank Timothy Tan, Brian Schirmacher and Harsh S. from the ExpressVPN team for their excellent project coordination, support and assistance, both before and during this assignment.