NetProtect
Encrypt.Me
Final Report

Document Name:	Final Report
Date:	Sept. 18, 2019
Customer Contact:	Wayne MacLaurin <wayne.maclaurin@netprotect.com>
Author:	Rob Carey <rcarey@securityinnovation.com> Amy Jones <ajones@securityinnovation.com>
Project Manager:	Garrett Jaynes <gjaynes@securityinnovation.com>

BOSTON | SEATTLE | PUNE
187 Ballardvale St., Suite A195 ● Wilmington, MA 01887 ● Ph: +1.978.694.1008
getsecure@securityinnovation.com ● www.securityinnovation.com

Contact Information

Security Innovation

Business Contact

Zack Hatfield
Business Development Director
zhatfield@securityinnovation.com

Technical Contact

Joe Basirico
Senior VP of Engineering
jbasirico@securityinnovation.com
Mobile: +1.206.227.6458

Project Management Contact

Garrett Jaynes
Senior Project Manager
gjaynes@securityinnovation.com

NetProtect

Wayne MacLaurin
wayne.maclaurin@netprotect.com

Executive Summary

Security Innovation performed a security assessment of Encrypt.Me on behalf of NetProtect. This report summarizes the issues that were uncovered.

Encrypt.Me is a VPN application for iOS, Android, Firestick, Windows, and Mac, which allows users to secure their private data by automatically encrypting their communications.

Security Innovation performed extensive testing of Encrypt.Me. At a high level, client applications were thoroughly assessed for information leakage vulnerabilities common in VPN applications, such as DNS, UPnP, WebRTC, the existence of pre-defined persistent routes, and VPN killswitch effectiveness. Binary analysis was performed, including Windows reverse engineering, mobile communications, and android unpacked .so file inspection. Specific mobile platform testing was performed, such as examining secure handling of private user data. Host-specific testing such as examining DLL signatures, open ports and services, and general attack surface was also performed both on the Encrypt.Me application hosts and the provided private endpoint Docker instance. All protocols and communication used by Encrypt.Me were heavily analyzed using traffic inspection tools to identify any abnormalities, misconfigurations, or vulnerabilities.

The overall security of Encrypt.Me was found to be highly robust. No sensitive customer information leakages were identified, security controls were found to be thoroughly implemented, and best security practices consistently followed. Additionally, the attack surface of the VPN user was minimized by the absence of potentially-dangerous optional features, like port forwarding.

Major observations are as follows:

- A total of 2 security issues were identified:
  - PR1 - Information Disclosure
  - PR2 - DLL Hijacking
- Two additional informational observations were discovered.
- The most severe vulnerability identified was the use of unsigned DLL's
- From a **STRIDE** perspective, issues were found from the **T**ampering, **I**nformation Disclosure, and **E**levation of Privilege categories .
- If these vulnerabilities are not remediated, users of certain platforms may be at a greater risk of having their host machines compromised under very specific scenarios.

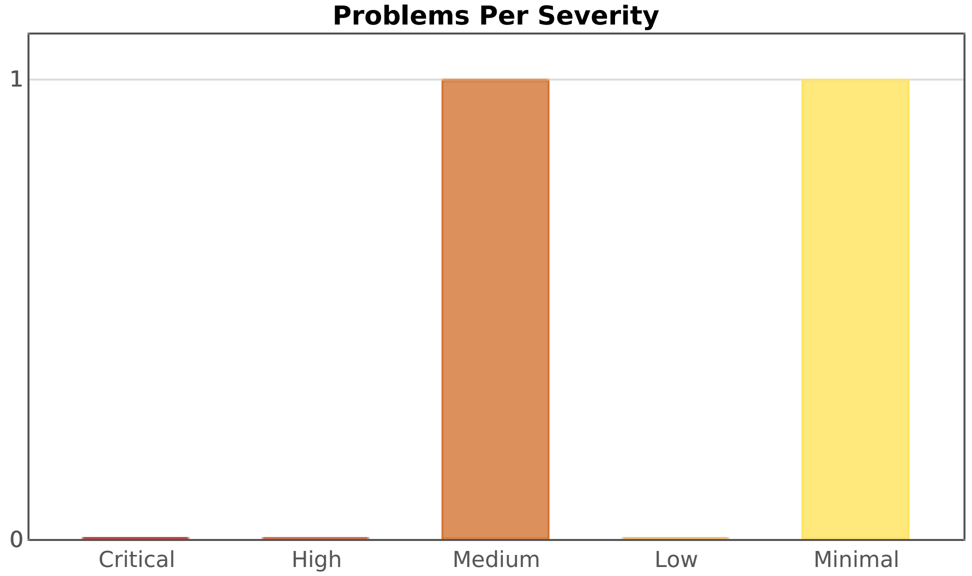
Introduction

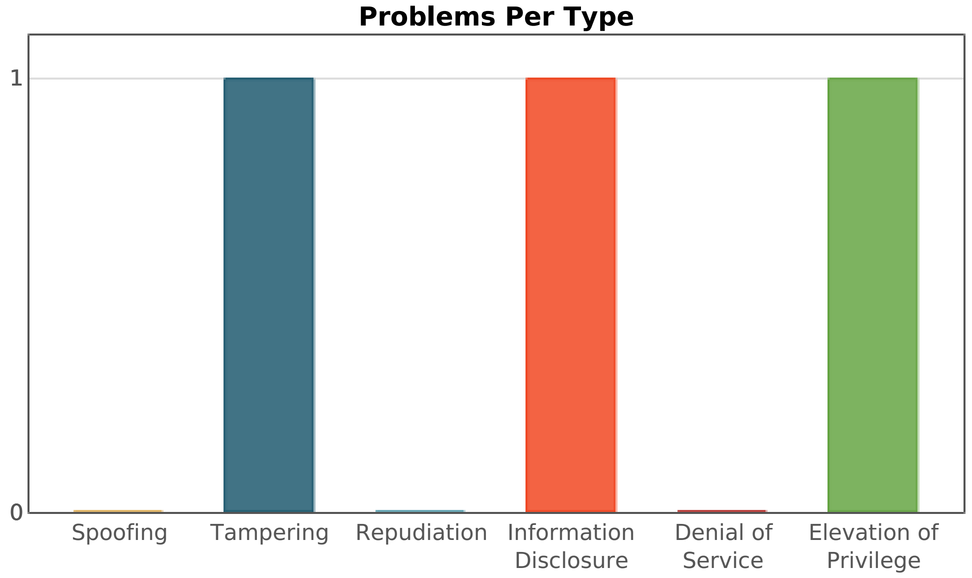
This report provides a summary of the findings discovered during the assessment. Each section is briefly described below:

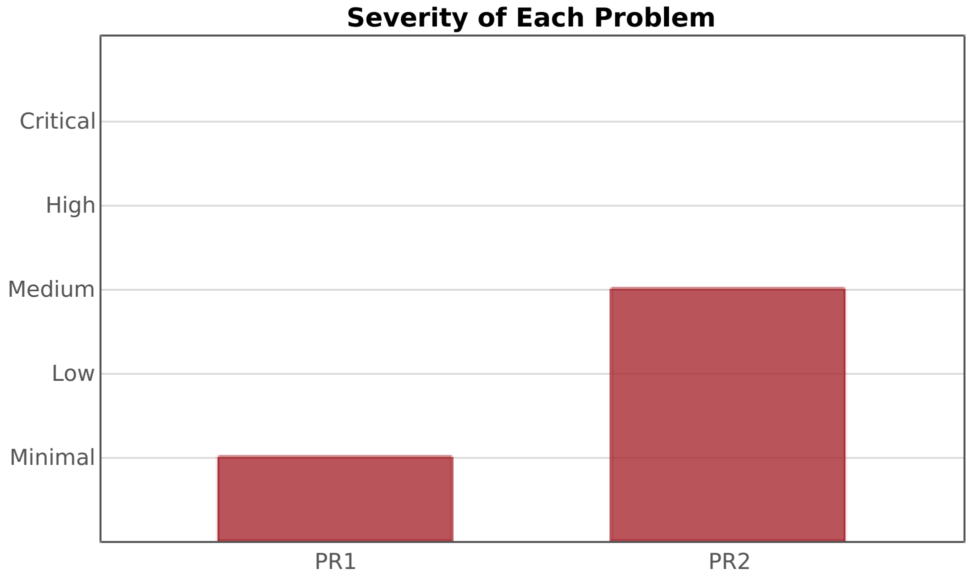
- The **Problem Report Summary** section summarizes the issues discovered during the engagement.
- The **Problem Reports** section contains the full text of each verified findings.
- The **Observations** section includes brief descriptions about less-severe or informational findings.
- The **Executed Test Cases** section lists a variety of performed tests and their observed results.
- The **Tools** section details the tools used during testing.
- The **Recommended Next Steps** section provides our recommendations for additional future testing of this system.

Problem Report Summary

A total of 2 problems were identified. This section describes, at a high level, each of the problems discovered. See the Problem Summaries section for a table of each problem discovered, its severity, description and consequences. The following charts display the number of problems for each level of severity, the number of problems for each STRIDE type (note that a problem may have more than one type), and each problem's overall severity.







Problem Summaries

The problem report summaries are sorted by problem report ID. The format of the problem report table is as follows:

- The problem report ID
- The component in which the issue was discovered
- The severity of the issue
- A short description of the issue
- The consequences of the issue

PR #	Component	Severity	Description	Consequence
1	Android Application	Minimal	The Encrypt.Me Android application does not obfuscate its screen when a user cycles through applications.	The application may leak sensitive information or reveal the fact that an individual using a VPN.
2	Windows Client	Medium	Encrypt.Me is vulnerable to DLL hijacking attacks.	An attacker can elevate their privileges by running malicious code within the Windows Client process.

Problem Reports

Below are the complete Problem Reports for all discovered issues.

Problem Report 1 - Information Disclosure

The Encrypt.Me Android application does not obfuscate its screen when a user cycles through applications. The application may leak sensitive information or reveal the fact that an individual using a VPN.

Component	Android Application
STRIDE	Information Disclosure
CWE	CWE-200: Information Exposure
CVSS v2 Score	2.1 (AV:L/AC:L/Au:N/C:P/I:N/A:N)
CVSS v3 Score	2.1 CVSS:3.0/AV:P/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N
OWASP Reference	OWASP Top 10 2017-A6: Security Misconfiguration

Overall Severity	**Minimal**
Vulnerability Type	**Defense in Depth**
Impact	**Low**
Confidentiality	The application leaks the fact that a VPN is in use and could potentially leak other information.
Integrity	This vulnerability does not impact data integrity.
Availability	This vulnerability does not impact data availability.
Exposure	This vulnerability represents a limited exposure to exploitation.
Affected Users	All users of the application are affected.
Likelihood	**Low**
Skill Required	No skill is required to exploit this vulnerability.
Conditions and Complexity	An attacker would need to be physically near a user.
Discoverability	This vulnerability is easy to discover.
Reproducibility	This vulnerability is easy reproducible in all cases.

Background Information

Privacy-oriented applications often blur the application window when a user cycles through applications in order to prevent the leakage of sensitive data. However, the presence of a specific application could itself be highly sensitive, as is the case for individuals using Tor or a VPN in countries with highly-oppressive governments.

Problem Details

During testing, Security Innovation observed that Encrypt.Me did not obfuscate its screen when cycling through applications.

Test Steps

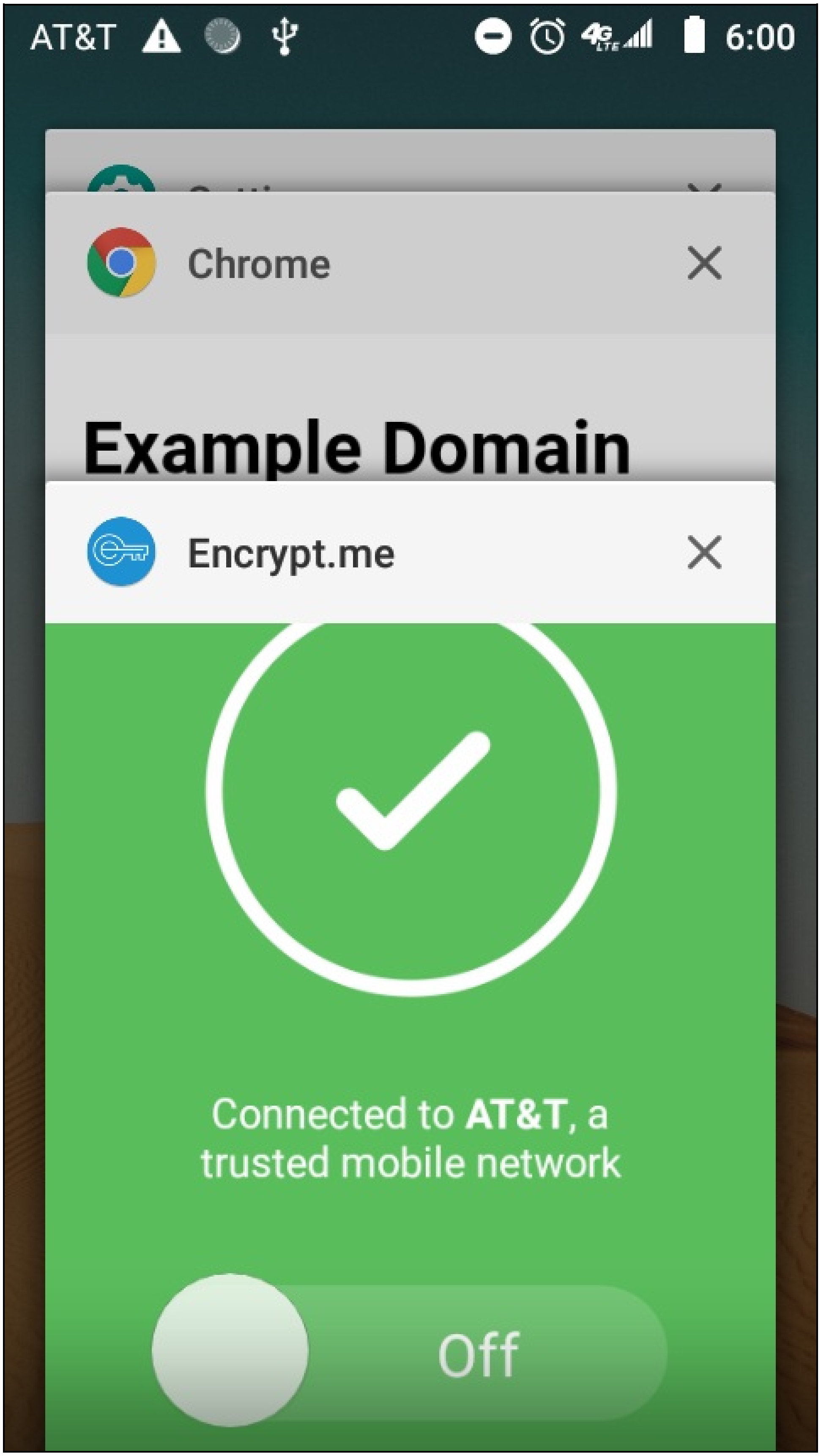
Test Configuration

The following is needed to reproduce this issue:

- Log in to the application on an Android device

Steps to Reproduce

1. This vulnerability may be observed by logging in to the application on an Android device and cycling through the application menus:



Connected to **AT&T**, a trusted mobile network

Remediation

Obfuscate the application. Security Innovation recommends blurring or otherwise obfuscating the application when cycling in order to avoid the leakage of potentially sensitive information.

Problem Report 2 - DLL Hijacking

The Encrypt.Me is vulnerable to DLL hijacking attacks. This allows an attacker to elevate their privileges by running malicious code within the Windows Client process.

Component	Windows Client
STRIDE	Tampering, Elevation of Privilege
CWE	CWE-427: Uncontrolled Search Path Element
CAPEC	CAPEC-251: Local Code Inclusion CAPEC-471: DLL Search Order Hijacking
CVSS v2 Score	6.2 (AV:L/AC:H/Au:N/C:C/I:C/A:C)
CVSS v3 Score	6.4 CVSS:3.0/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H

Overall Severity	**Medium**
Vulnerability Type	**Directly Exploitable**
Impact	**High**
Confidentiality	By exploiting this vulnerability, an attacker can read any data the application can access.
Integrity	By exploiting this vulnerability, an attacker can create, modify, or delete any data the application can access.
Availability	By modifying a DLL used by the application, an attacker can prevent the application from running.
Exposure	An attacker can elevate their privileges to those of the user running the Windows Client process.
Affected Users	This can affect any user of the application.
Likelihood	**Low**
Skill Required	Exploiting this vulnerability requires a skilled attacker to create the malicious DLL and install it in the proper location of the operating system.
Conditions and Complexity	This vulnerability requires the installation of a malicious DLL on the target operating system via social engineering, local access to the machine, or exploiting another vulnerability in the system.
Discoverability	This can be easily discovered by an attacker who has access to an installation of Encrypt.Me.
Reproducibility	This is 100% reproducible.

Background Information

Modern applications are generally not packaged as a single binary. Instead, they are composed of a main executable (on Windows generally ending in ".exe") and many dynamic linking libraries (generally ending in ".dll"). When loading a library, the executable will look first in the application directory for the relevant .dll file before looking in other areas of the file structure.

As a result, an attacker can force the application to load a malicious library by giving it the same name as a system library and placing it in the application's directory. Alternatively, unsigned DLLs may be replaced in-transit or after-deployment with malicious variants intended for privilege escalation or persistent access.

Problem Details

Multiple DLLs used by the Windows client are not signed. As a result, there is no way for the application to verify the integrity of DLLs.

Affected Areas

The following DLLs are not digitally signed:

- CertEnroll.dll
- CERTENROLLLib.dll
- CertificateHelper.dll
- DotRas.dll
- Hardcodet.Wpf.TaskbarNotification.dll
- IPCServiceLibrary.dll
- PropertyProviders.dll
- Serilog.dll
- Serilog.Sinks.File.dll
- Serilog.Sinks.Literate.dll
- Serilog.Sinks.RollingFile.dll
- Shared.dll
- SQLite-net.dll
- SQLitePCLRaw.batteries_green.dll
- SQLitePCL.core.dll
- SQLitePCLRaw.provider.e_sqlite3.dll
- TopShelf.dll
- Topshelf.Serilog.dll
- WinSparkle.DotNet.dll
- WpfPageTransitions.dll

This list may not include all instances of unsigned DLLs or shared objects. It is suggested that, upon remediation, the development team review other areas of Encrypt.Me where similar techniques are used in order to find and fix related vulnerabilities.

Remediation

Only load code from trusted sources. Before running code from an executable, library, or other component, check if the component is signed. If it is not signed or is signed with an invalid signature, do not load code from that component and fail over with the proper alerts.

Remove missing dependencies. References to libraries that are not used increase the attack surface of the application while providing no practical benefit. As such, it is recommended to remove references to DLLs that are not used by the application.

Windows - C/C++

Change the DLL search order. Applications can control the DLL search order and remove insecure directories altogether. Early in process execution, call `SetDllDirectory` with an empty string ("") to remove the current working directory from the DLL search order. In addition, use `SetSearchPathMode` to ensure that `SafeDllSearchMode` is enabled for the process.

For more information on securing DLLs, visit the following link: https://support.microsoft.com/en-us/help/2389418/secure-loading-of-libraries-to-prevent-dll-preloading-attacks

Observations

In addition to the Problem Reports listed above, two additional informational findings were observed. While not typically as severe as a Problem Report, Security Innovation recommends that these, too, be addressed.

Observation 1 - Weak Password Policy

During account registration, it was observed that users could set their password to very insecure words, such as the word "password." Although this is not directly related to the client, such a low-complexity password requirement may lead to user compromises.

It is generally recommended to enforce at least basic password policies, such as a minimum of eight characters and some mix of different types of characters.

Observation 2 - Trivial Decompilation

In testing the Windows client, it was observed that the client was trivially decompilable with free commercial software like DotPeek, with fully-named and unobfuscated functions. Since any user could do this to greatly expedite their ability to find vulnerabilities in the implementation across all clients, it is strongly recommended to employ some form of code obfuscation to achieve greater defense-in-depth.

Executed Test Cases

The following table shows the breakdown of executed test cases, including any problem reports relevant to that item, and gives a brief summary of the methodology used to check that item and any other observations.

Column descriptions are as follows:

- **Test ID** - An identifier for quick test case reference
- **Test Title** - A title describing the test case
- **Test Description** - A short description of the test case and why it was performed
- **Outcome**- Either 'Pass' or a reference to the Observation/PR Number

Test ID	Test Title	Test Description	Outcome
1	IP Leakage - DNS	Ensure that a malicious user cannot coerce a victim to reveal their true IP via DNS requests.	**PASS**
2	IP Leakage - UPNP	Confirm that UPNP cannot be used to set localized port forwarding that may be detected out-of-band to determine a user's identity.	**PASS**
3	IP Leakage - WebRTC/STUN	Verify that WebRTC requests cannot be made which provide malicious websites information about the user's true IP address.	**PASS**
4	Route Persistence	Determine if pre-existing or manually-created persistent routes with sufficiently high priority are ignored while the VPN is enabled.	**PASS**
5	IP Leakage - Port Forwarding/PortFail	Verify that the VPN provider does not allow for insecure port forwarding, which can lead to IP disclosure attacks such as PortFail.	**N/A**
6	IP Leakage - Protocol Changes	Check that server HTTP Protocol changes cannot be performed in such a manner as to coerce the browser to bypass the VPN tunnel.	**PASS**
7	IP Leakage - IPv6	Ensure that IPv6 traffic does not reveal the user's true identity.	**N/A**
8	Insecure Installation	Oversee the installation process to ensure that no timing attacks exist, sensible object permissions are established, and no unnecessary attack surface is added to the system.	**PASS**
9	Unsigned DLLs	Confirm that all DLLs and other shared object files in-use by the application have their integrity validated via digital signatures.	**PR 2**
10	Insecure Services/Daemons	Determine whether all services or daemons installed by the VPN are given appropriate permissions and secure paths.	**PASS**

Test ID	Test Title	Test Description	Outcome
11	VPN Killswitch Bypass	Identify if there is any way to induce a fail-insecure state, where the VPN is disabled due to a crash but the user may continue browsing exposed.	**PASS**
12	Insecure Updating	Validate the update methods used by the clients to ensure that they are done over secure, verified channels, such as TLS.	**PASS**
13	Insecure Support Channel	Confirm that any in-client support requests are conducted over TLS, do not disclose unnecessarily sensitive information about the end-user, and do not bypass the VPN functionality if it is currently running.	**PASS**
14	Transport Security of All Traffic	Verify that all web traffic in all clients is conducted over TLS when interfacing with any telemetry or reporting endpoints.	**PASS**
15	Local Storage of Credentials	Validate that local credentials are not stored insecurely, such as in plaintext/public locations.	**N/A**
16	Binary Analysis - Decompilation	Determine if the application may be trivially decompiled and reversed via free and easily-available software	**Observation 1**
17	Binary Analysis - Insecure Logging	Verify that logs created by the applications do not contain sensitive customer information or sensitive system information.	**PASS**
18	Binary Analysis - Hard Coded Secrets	Inspect the application to locate any existing hard coded secrets, such as credentials or connection strings.	**PASS**
19	Binary Analysis - Android .SO Decompilation	Discover if any secrets or easily reversed information can be gleamed by unzipping the android APK client and decompiling its shared objects.	**PASS**
20	Mobile Data Storage	Validate the security of locally stored data within the mobile application.	**PASS**
21	Mobile Privacy Testing	Ensure that any relevant privacy features are applied to the mobile clients to protect the user.	**PR 1**
22	Verbose Error Messages	Verify that the application cannot be made to crash or fail in such a way as to reveal sensitive user- or system-data to third parties, logs, or local users.	**PASS**

Tools

While testing Encrypt.Me, the following tools were employed:

Tool	Description	Link
Burp Suite Professional	Interactive HTTP/S proxy server that can intercept, inspect and modify data between the browser and target web server	https://portswigger.net/burp/
Wireshark	Gold standard of network sniffers and analysis	https://www.wireshark.org/
cURL	Command line tool for transferring files with URL syntax, supporting FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS and FILE	https://curl.haxx.se/
Python	High level, general purpose, scripting language	https://www.python.org
Detect-it-Easy	Windows application to determine compilation information about an executable	https://ntinfo.biz
x64dbg	Windows dynamic analysis and debugging tool for reverse engineering	https://x64dbg.com
DotPeek	DotNet framework decompiler for static analysis	https://www.jetbrains.com/decompiler/
IDA Pro	Cross-platform static- and dynamic-analysis code disassembler and decompiler	https://www.hex-rays.com/products/ida/

Recommended Next Steps

This section contains our recommendations for areas that may benefit from additional testing. For each section, we describe why it is important to test these sections, either more thoroughly or for the first time.

Code-Assisted Penetration Test - Security Innovation recommends a code-assisted penetration test of the Encrypt.Me application. This will allow for a more comprehensive audit of the application. In addition, it will potentially identify insecure coding patterns and other coding issues, which would not be possible in a black-box penetration test.

Server Review - Over the course of the penetration test, the various VPN clients were the primary targets of review. Although some time was allocated to inspect, stand up, and test the provided docker image to create a user-hosted VPN server, this was not the primary focus of the engagement. Since a compromise within a VPN server could potentially reveal sensitive information about all active users, this should be taken as a high priority target in future assessments.

Follow-up

Inconsistencies, errors, and reproducibility problems associated with this report should be directed through the customer contact person to the tester and preparer indicated at the beginning of this report.