Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Mullvad Apps, Clients & API 05.2020

Cure53, Dr.-Ing. M. Heiderich, J. Larsson, MSc. R. Peraglie, MSc. F. Fäßler,
MSc. S. Moritz, BSc. C. Kean

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

“Have you ever thought about how online ads tend to follow you from site to site? Or how Facebook knows exactly which ones to show you? We can assure you it's not magic. Think about all the intimate questions and concerns you ask Google on a daily basis. Or all of your personal shopping habits and interests that you feed to Amazon. Not to mention that Alexa is constantly listening to everything you say. While you may be doing this all from the privacy of your own home, your online life is, in contrast, very public.”

From https://mullvad.net/en/what-is-privacy/

This report describes the results of a thorough security assessment of the Mullvad complex. Carried out by Cure53 in May and June 2020, the project entailed both a penetration test and source code audits, specifically targeting selected Mullvad applications, clients and related API endpoints.

It needs to be underlined that the work was carefully scoped to cover only the client-side parts within the Mullvad scope items, concurrently indicating that nothing on a server-side received significant attention unless it has direct bearing on the apps. It is envisioned that a dedicated assessment focused on the server-side will ensue. Further of note is the fact that the project continues the cooperation between Cure53 and Mullvad initiated back in 2018. In particular, the Cure53 team examined similar scope for Mullvad in September 2018, with a report from this project headlined *MUL-01.*

Based on the previous experience and the discussion with the Mullvad team, Cure53 demarcated two work packages. In WP1, tests and audits of the Mullvad mobile applications for both Android and iOS were carried out. Similar approaches of security testing and auditing were deployed also in WP2 which nevertheless centered on the Mullvad VPN applications across three branches, namely Windows, Ubuntu and OSX.

The methods used during the assessment entail a white-box approach, dictated by the simple fact that all Mullvad software in scope is available as open source on GitHub. Cure53 had access not only to the sources but also to dedicated builds and threat modeling information. Further made available were test-users via a range of voucher codes that the Cure53 team could use and create new users. These are listed in the *Scope* chapter. Finally, the testers received a detailed briefing about the expectations set for this penetration test and audit from Mullvad's perspective.

The project started on time and progressed efficiently. In terms of timeline, resources and communications, it should be noted that six members of the Cure53 team were tasked with this project and given a budget of twenty days to reach the expected level of

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

coverage. All work was completed in late May and early June of 2020, with communications between Cure53 and Mullvad executed in a dedicated, private Slack channel created by Cure53 and then joined by relevant Mullvad personnel. Slack could be used for asking questions, as well as let the Cure53 team deliver status updates and data about the emerging and confirmed findings. All discussions were helpful, facilitating good flow of the project.

Moving on to findings, it can be noted that only seven items have been spotted and documented. Two problems represent security vulnerabilities while five have been classified as general weaknesses with lower or even non-existent exploitation potential. This is a good result for Mullvad, especially in the light of this project containing very thorough reviews and several deep-dive attempts at compromising the software in scope. Back in September 2018, the same number of seven issues was spotted, yet they included both *Critical*- and *High*-scoring bugs. This pattern has not recurred here as none of the findings stemming from this 2020 project exceeded a severity rank of *Medium.* All findings were live-reported to the Mullvad team via PGP-encrypted emails. Some fixes have been verified while the test was still ongoing.

In the following sections, the report will first shed light on the scope and key test parameters. After that, a dedicated chapter on the test coverage and methodology will furnish insights into what the Cure53 team looked at, even if a given approach yielded no results. Next, all findings will be discussed in a chronological order alongside technical descriptions, as well as PoC and mitigation advice when applicable. Finally, the report will close with broader conclusions about this 2020 penetration testing and auditing of the Mullvad scope. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team’s observations and collected evidence. Tailored hardening recommendations for Mullvad are also incorporated into the final section.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Source code audits & security assessments against Mullvad apps, clients & API**
  - **WP1**: Security tests & code audits against Mullvad mobile apps for Android & iOS
    - https://github.com/mullvad/mullvadvpn-app/
    - https://github.com/mullvad/mullvadvpn-app/releases/tag/2020.4
  - **WP2**: Security tests & code audits against Mullvad desktop apps for Windows, Ubuntu, OSX
    - https://github.com/mullvad/mullvadvpn-app/
  - **Test-users**
    - Test-users were created based on the voucher codes given to Cure53 and listed below
      - `KPPD-NRDM-MEK7-JYEJ`
      - `X86M-DMM3-68YN-MF76`
      - `DNPX-JAKF-9KHT-R3YD (unused)`
      - `P8AN-MKE8-F6K8-KXAK (unused)`
      - `83YD-FF7X-K93J-PM74 (unused)`
      - `N3TP-P479-64KX-HEX4 (unused)`
      - `KEJY-J7E9-4LEM-PR6X (unused)`
      - `98D9-K6PA-Y4TT-74AH (unused)`
      - `4YXY-MKYP-P474-9Y6X`
      - `KN87-EMXY-N3X9-L8TN`
- **Sources were shared with Cure53 via GitHub (OOS)**
- **Binaries were shared with Cure53 as well**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Coverage

This section describes the testing methodology and coverage. In terms of methods, an emphasis has been placed on what has been done in order to evaluate the current security posture of the Mullvad's applications, which as of this assessment offers client applications for Android, iOS, Windows and Ubuntu.

In order to provide a structured overview, the assessment has been divided into five sections: *Android Application*, *iOS Application*, *Windows Client, Ubuntu Client and macOS Client.* This assessment has been carried out with a white-box methodology and this has bearing on what type of actions and steps undertaken during this assessment.

Coverage for Android Application

A list of items below details noteworthy tasks undertaken during the Android application phase of this audit. This is to underline what the Cure53 testers covered during their analysis, especially in regards to items related to the Android application, further shedding light on the coverage and methodology applied during this phase of the assessment.

- The application was analyzed with an explicit focus on how the current version fits into the Android's ecosystem. In addition, Cure53 examined how communication with the Android's platform API is handled. The related attack surface is composed of one exposed activity (*MainActivity*) and two exposed services (*MullvadVpnService* and *MullvadTileService*), both protected through system permissions. It was investigated if and how the application is receiving data through registered custom schemes, data URLs, extra strings or *Parcelable* objects. It was found that none of these methods could be used to send malicious data to the Mullvad application. This drastically reduces the attack surface that other apps could misuse.
- The integrated permission model within the Android's ecosystem protects the access to the Mullvad data folder from other users. It was therefore checked whether the application processes files outside the protected data folder or reads data from files to which anyone has access. It was found that no other locations were used outside of the data folder, further reducing the attack surface. Moreover, filesystem permissions were analyzed in order to determine whether the written files within the data directory have world-readable permissions, making sure that access to the configuration and *shared_pref* files is prevented.
- Third-party libraries used by the application were verified for access to the protected data folder, which could lead to the theft of sensitive information as no encryption is used.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- The Android's application network communication was examined with a focus on interception of the TLS traffic. No plain-text HTTP traffic could be found.
- Cure53 checked if the application was allowed to create connections to servers having invalid or self-signed certificates. If successful, this would allow intercepting the traffic, yet it was found that the RPC client responsible for communication with the API prevents connections to such servers.
- The usage of the supported WireGuard protocol was examined for a possibility to leak information that would identify a user, for instance through outgoing DNS traffic. In addition, it was tested if the real IP could be observed via WebRTC. No issues were found leaking data to either ISP or other adversaries.
- The Android device logs were examined for potential information leaks from the app. It was discovered that the application logs debug information about the connection that is established via the tunnel interface. This leaks the used, internal and static IP addresses from the related Mullvad account.
- Additionally, it was checked whether the application uses insecure or weak cryptographic constructs. The private and public key creation was reviewed and no issues were found.
- Although every Android application is able to use a WebView, Mullvad makes no use of it, which significantly reduces the attack surface.
- The API consumed by the application on *api.mullvad.net* was also examined. It was checked if data from other users are accessible (*ACL/IDOR*). Cure53 focused on the backend application's ability to parse XML content (XXE) or general proneness to injection vulnerabilities such as SQLi or RCE. Additionally, a scan for sensitive web content was performed to gauge information disclosure potential linked to unprotected endpoints, files or administrative applications. In addition, the existing rate-limiting for corresponding endpoints was investigated with attempts at circumventing constraint, e.g. via the *X-Forwarded-For* header.

Coverage for iOS Application

A list of items below details the relevant tasks undertaken during the iOS Application phase of this audit. This is to underline what the Cure53 testers covered during their analysis, especially in regards to the iOS Application items, with the aim of showcasing coverage and methodology, as well as avoiding duplication of work in-house.

- The local storage of the Mullvad iOS application was examined via SSH connection[1] on a jailbroken device on version iOS 13.3.1 with the *checkra1n* exploit[2]. As iOS employs sandboxing to prevent applications from accessing other users' local storage, it can be assumed that the Mullvad local storage is

[1] https://cydia.saurik.com/package/openssh/
[2] https://checkra.in/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

secure when it comes to third-party application access. However, this countermeasure might vanish in a jailbroken or similarly altered iDevice. In addition, the application does not ask for any personal information or credentials which further reduces the potential attack surface.

- The Mullvad iOS application does not take advantage of native iOS filesystem protections and does not disable client-side caching via *NSURLRequest*[3].
- The iOS applications network communication was reviewed by intercepting connections and analyzing the traffic. It was found that plain-text HTTP communications are not in use. The team also attempted to intercept TLS traffic with invalid certificates, which the application correctly rejected. Furthermore, the Mulvad iOS application was found to have *App Transport Security (ATS)*[4] enabled and does not define ATS exceptions, thus avoiding insecure connections.
- The iOS device logs were examined for potential information leaks from the application. However, none were discovered during or after the usage of the application.

Coverage for Windows Client

A list of items below seeks to detail some of the noteworthy tasks undertaken during the Windows Client phase of this audit. This is to underline what the Cure53 testers covered during their analysis, especially in regards to the Windows Client items. In doing so, Cure53 elaborates on coverage and methodology applied during this phase of the audit.

- The IPC mechanisms of Mullvad were inspected and it was found that the *Named Pipe Mullvad VPN* was created with permissions, allowing read and write access to everyone. It was confirmed that the *Named Pipe* was exposed to the network by the *SMB* protocol for loose network configurations in Mullvad and Windows, which was then filed as MUL-02-007. Further, it was confirmed that a web browser could be used to open the *Named Pipe* and crash the Mullvad daemon in the 2020.4 release. Attempts to load the resource from an external webpage in a web browser were stopped by browser protocol-policies only permitting downloaded local files to exploit this bug. Further it was suggested that this bug was fixed in the 2020.5 release, preventing the flaw.
- The Firewall rules of Windows were examined and tested in-depth. For this purpose, a network sniffer was installed to monitor all outbound traffic exiting the computer. It was confirmed that most TCP connections were caught by the routing mechanism, wrapped and encrypted inside the tunnel. Exceptions to this rule were found to be controlled by the two-layer Firewall rules applied by

[3] https://github.com/nowsecure/secure-mobile-development/blob/master/e...tps-requests-responses.md
[4] https://developer.apple.com/documentation/bundleresources/information_...apptransportsecurity

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Mullvad VPN and being omnipresent in the connected state. It was found that the Firewall allowed a web browser to send TCP packets out of the tunnel to several whitelisted IP addresses, resulting in potential deanonymization reflected in MUL-02-002 and MUL-02-004.

- It was tested if enabling Mullvad's *local network sharing* feature could allow attackers to route one of the *Multicast* addresses outside of the local network to an attacker-controlled host with no success. To this extent, it was discussed if the Firewall rules allowing *DHCP* traffic to arbitrary hosts could allow deanonymization. It was evaluated that an application requires *root* privileges to bind to *DHCP* ports and performs this kind of deanonymization as a sufficient protection.

Coverage for Ubuntu Client

A list of items below seeks to detail some of the noteworthy tasks undertaken during the Ubuntu Client phase of this audit. This is to underline what the Cure53 testers covered during their analysis, allowing additional insights on coverage and methodology applied during this phase of the audit.

- The GUI is built with the Electron framework. Therefore, common pitfalls checks included the options passed to *BrowserWindow* instances, which were confirmed to run unsandboxed, unprotected by context-isolation with node integration turned on. It was confirmed that this issue was already listed in a previous report.
- Further, it was verified that the *AccountData* cache object does not store the account-data anywhere on the disk. Instead, the data was only held in-memory, thus preventing potential leaks through the filesystem.
- The GUI was checked for usage of operating system commands initiated with the *child_process* module. It was only found that the *mullvad-problem-report* binary was executed to pass user-controlled arguments, with no external injections found.
- Network-related attacks such as rogue and malicious upstream services, like *DHCP, DNS* and similar, were analyzed in order to look for potential attack vectors that could lead to out of tunnel communications or client leaks. No issues were deemed to affect the current threat model adopted by Mullvad.
- The current IPv6 implementation and configuration was assessed by looking for common misconfiguration issues that could lead to potential information disclosure as regards the end-user.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Coverage for macOS Client

A list of items below seeks to detail some of the noteworthy tasks undertaken during the macOS Client phase of this audit. This is to underline what the Cure53 testers covered during their analysis and as such elaborates on the methodology and coverage.

- One of the major risks for VPN applications is the communication between the unprivileged client and the privileged helper process, as it can lead to local privilege escalations. The *IPC* mechanism used does not build upon the OSX provided XPC APIs and can be reached by any process. This is not an issue as long as the functions implemented via *IPC* forbid escalating privileges.
- The exposed functionality was reviewed with a focus on file handling and process creation. All files touched by the daemon process are located in *root* or *SIP*-protected paths, thus a local attacker cannot use symlinks or other tricks to perform arbitrary or privileged actions.
- OpenVPN is a process executed by the privileged helper and it also is located in a safe location.
- It should be mentioned that the installation of the Mullvad app is done via a *.pkg* file and requires *root,* thus the installed *.app* package cannot be modified by a user. This is a good choice because if the *.app* package would be installed by drag and drop from a user, the *openvpn* binary could be potentially replaced to achieve privilege escalation.
- Another good design choice is the lack of an *auto updater.* While it would improve user-experience with hassle-free updating, an update mechanism introduces a lot of attack vectors. Telling the user to simply download the new version is a fair way to mitigate many issues.
- While the exposed functions seem safe, it was noticeable that many API HTTP requests not requiring *root* privileges are still handled by the daemon process. It does not directly introduce any issues, but it could be considered to keep the daemon as small as possible and further separate actions that require *root* privileges from those that do not.
- Besides the privileged helper, the *uninstall.sh* script was checked, especially as it has to be run as *root.* While it does touch user-owned folders and files, all used locations are protected by *SIP* and cannot be symlinked.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *MUL-02-001*) for the purpose of facilitating any future follow-up correspondence.

MUL-02-006 WP1: Blind HTML Injection via *Problem Report* (*Low*)

During the assessment of the Android application, it was discovered that the *Report a problem* feature is prone to Blind HTML injection attacks. The data is sent via the *RPC* client to the API endpoint called */v1/problem-report*. The content is included as HTML and gets displayed in a page of another web application without a sufficient degree of encoding. As such, an attacker can enter malicious HTML into this text-field and the payload will be executed on the affected domain's content.

This can signify that an attacker has a capacity to make arbitrary redirects to other domains, embed external content or steal sensitive data like tokens or other content from the loaded page, e.g. via a non-closed *img-src* attribute. Please note that this issue can lead to XSS attacks in certain scenarios as well, for instance when the content is displayed within a vulnerable web application.

The issue was rated *Low* because no sensitive information could be exfiltrated from the used Google Mail client.

Steps to Reproduce:

1. Open the Report Problem view under *Settings -> Report a problem*.
2. Insert into the description field the following HTML markup
   `<img src="//yourserver.com/pic.jpg">`
3. Click on Send and wait for a pingback.

Request received from the client:

```
GET /pic.jpg HTTP/1.1
Host: 6nuw7tsxcnuxi87froh763u5vw1qpf.burpcollaborator.net
Referer: http://mail.google.com/
From:
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246 Mozilla/5.0
[...]
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

It is recommended to make sure that proper encoding happens for all user-controlled data rendered within the browser. Please make note of the fact that further attacks like Cross-Site Scripting (XSS) could be possible and should also be mitigated.

Fix Note: *This issue was further investigated by Mullvad and was deemed as a non-applicable issue.The Gmail servers are responsible for automatically polling every URL outside of Mullvad's control. The injection does not affect Mullvad or its users*.

MUL-02-007 WP2: *Named Pipe* exposed via *SMB* accessible to everyone *(Medium)*

It was found that the permissions for the *Named Pipe Mullvad VPN* in Windows are set to being readable and writable for everyone. This is dangerous when the user's computer has lax network sharing options, allowing attackers on the same network to connect themselves to the Mullvad daemon via *SMB* through the *Named Pipe*. This could be abused to perform all actions that the GUI permits, for instance silently disconnecting the user mid-session.

Steps To Reproduce:

1. Connect to Mullvad VPN and enable *local network sharing* in settings
2. Visit *Control Panel>Network and Internet>Network and Sharing Center>Advanced sharing settings* and for *all networks* click on *turn off password protected sharing*
3. Attack with a Linux-based system with *smbclient:*
4. `echo '{"jsonrpc":"2.0","method":"disconnect","id":"1234"}' | smbclient -E -U vmuser '//192.168.56.102/IPC$' "" -c 'put /dev/fd/0 \\"Mullvad VPN"'`
5. Mullvad VPN of the targeted user should now disconnect.

It is recommended to tighten the permissions of the *Mullvad VPN Named Pipe* and deny access for *NT AUTHORITY\NETWORK* which will only permit local usage[5]. By doing so, attackers can no longer connect to the *Mullvad VPN Named Pipe* through the network interacting with the Mullvad daemon and, eventually, turning off the VPN remotely.

Fix Note: *This issue was reported during the audit and was fixed by Mullvad with the following PR https://github.com/mullvad/mullvadvpn-app/pull/1830*

[5] Read more about pipe permissions at the official Microsft MSDN documentation https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals 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.

MUL-02-001 iOS: Lack of filesystem protections *(Info)*

It was found that the iOS app does not take advantage of the native iOS filesystem protections and fails to fully protect some of its data files at rest. The affected files are only protected until the user authenticates for the first time after booting the phone. The problem is that the key to decrypt these files will remain readable in memory while the device is locked. The impact of this issue was evaluated as *Info* because no sensitive data is exposed. However, the exposed information includes a private IP address used inside the VPN.

Affected File:
Library/Caches/net.mullvad.MullvadVPN/Cache.db

Affected Content:
```
{"id": "A2810E5D-BA6D-4891-BA38-26A0FCCC4BBD", "jsonrpc": "2.0", "result":
{"ipv4_address": "10.66.15.111/32", "ipv6_address": "fc00:bbbb:bbbb:bb01::3:f6e/
128"}}
```

This issue requires actual, physical access to an iDevice set to a locked screen and a method of accessing the local storage. The latter could mean, for instance, an SSH connection established via a jailbreak. While the screen was locked, the files below represent some of the data that remained unprotected.

Command:
```
tar cvfz files_locked.tar.gz *
```

Output:
```
tar cvfz files_locked.tar.gz *
Library/Caches/net.mullvad.MullvadVPN/
Library/Caches/net.mullvad.MullvadVPN/Cache.db
Library/Caches/net.mullvad.MullvadVPN/Cache.db-wal
Library/Preferences/net.mullvad.MullvadVPN.plist
Library/Saved Application
State/net.mullvad.MullvadVPN.savedState/KnownSceneSessions/data.data
[...]
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

In order to solve the problem related to file-access, it is recommended to implement the *NSFileProtection-Complete* entitlement at the application-level[6]. To further harden the local storage, it should be considered to disable the client-side caching of requests[7].

Fix Note: *This issue was reported during the audit and was fixed by Mullvad with the following PR https://github.com/mullvad/mullvadvpn-app/pull/1808.*

MUL-02-002 WP2: Firewall allows deanonymization by eavesdropper *(Medium)*

It was found that the current configuration of Mullvad makes it possible for attackers to successfully and efficiently map the external Mullvad IP address and the real IP address of a user in a realistic scenario. The vulnerability requires Mullvad to choose TCP as the transport protocol and needs an attacker who is able to passively observe the outbound Internet traffic of the victim.

Additionally, the victim must browse to an attacker-controlled page while being protected by Mullvad VPN. In this scenario, adversaries can identify the user by sending an unencrypted request containing an identifier token from the victim's computer to the Mullvad entry relay. This identifier token is received through the secure VPN tunnel from the attacker's site and only sent to the victim. Therefore, it can be identified and associated directly with the user's machine sending the token outside of the VPN tunnel.

Steps To Reproduce:

1. Launch the Mullvad Desktop VPN on Ubuntu and configure it to always use OpenVPN on port 443 without a bridge and click on *connect*.
2. Simulate the passive network eavesdropper by launching *tcpdump* on the outbound network interface and searching for unencrypted requests with grep:
3. `sudo tcpdump -i enp0s3 -lA | grep -a token313373 -A 15`
4. Open the *poc_deanonymization.html* file in a browser.
5. Wait a few seconds until the page has loaded. *Tcpdump* should now have listed requests containing an identifier token sent unencrypted over the wire.

Mullvad in its current state allows no-tunnel-connections to be established to the entry gate of the currently active Mullvad network path. This applies to all applications. It is recommended that the Firewall rules tighten the conditions further, specifically to only permit connection attempts made by the Mullvad VPN service and all related applications like WireGuard and OpenVPN approved by Mullvad.

[6] https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html
[7] https://github.com/nowsecure/secure-mobile-development/blob/master/en/ios/av...ests-responses.md

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

By making use of the *FWPM_CONDITION_ALE_APP_ID* condition offered by Windows operating systems[8], all Firewall rules can be adjusted to precisely enforce that only Mullvad-related applications are permitted to connect to specific services.

Fix Note: *This issue was reported during the audit and was fixed by Mullvad with the following PR; https://github.com/mullvad/mullvadvpn-app/pull/1819, https://github.com/mullvad/mullvadvpn-app/pull/1827 and https://github.com/mullvad/mullvadvpn-app/pull/1829*

MUL-02-003 WP1: General hardening recommendations for Android app *(Info)*

During the assessment of the Mullvad Android app, it was discovered that not all security flags offered by Android are utilized. The absence of these flags does not introduce a security issue but could allow an attacker to exploit other problems more easily. As such, the flags described below should be considered as defense-in-depth mechanisms.

FLAG_SECURE

By setting the *FLAG_SECURE* for Android views, the app's windows can no longer be manually "*screenshotted*"[9]. Additionally, the items will be excluded from automatic screenshots or screen-recordings, which ultimately prevents screen data from being leaked to other apps. Including this flag is especially recommended for the implemented views that show sensitive data, such as the *Account* view that shows the *Account Number*.

filterTouchesWhenObscured

The *filterTouchesWhenObscured* security flag for views protects against so-called Tapjacking attacks. A malicious app can overlap the currently active app with a hidden screen overlay. The latter would need to have the ability to intercept data entered into the underlying app. Once the flag is set, a view will no longer receive touches when it is obscured by another window, therefore making this attack infeasible[10].

Fix Note: *This issue was reported during the audit and was fixed by Mullvad with the following PR https://github.com/mullvad/mullvadvpn-app/pull/1823 and https://github.com/mullvad/mullvadvpn-app/pull/1822*

[8] https://docs.microsoft.com/en-us/windows/win32/fwp/filtering-condition-identifiers-
[9] https://developer.android.com/reference/android/view/WindowManager.Layou...ml#FLAG_SECURE
[10] https://blog.devknox.io/tapjacking-android-prevent/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

MUL-02-004 WP2: Firewall allows TCP connections to WireGuard gateway *(Low)*

It was found that the Firewall allows non-tunneled TCP connections to the WireGuard gateway on port 53 which is actually serving on *UDP/53*. This introduces the risk of an attacker attempting a similar deanonymization attack as in MUL-02-002 by passively observing the network traffic of the victim and waiting for a failed TCP handshake performed with the WireGuard gateway. Since no identification token can be included in the TCP handshake, the mapping cannot be performed as reliably as in MUL-02-002, explaining the *Low* ranking of this flaw.

Affected File
windows/winfw/src/winfw/rules/dns/permitnontunnel.cpp

Affected Code

```
wfp::ConditionBuilder conditionBuilder(FWPM_LAYER_ALE_AUTH_CONNECT_V4);
conditionBuilder.add_condition(ConditionPort::Remote(DNS_SERVER_PORT));
for (const auto &host : m_hostsIpv4)
{
	conditionBuilder.add_condition(ConditionIp::Remote(host));
}
[...]
if (false == objectInstaller.addFilter(filterBuilder, conditionBuilder)){...}
```

It is recommended that the Firewall rules are tightened by adding a condition to the *PermitNonTunnel* rule, which enforces the UDP transport protocol before permitting a new connection. This will prevent attackers from initiating non-tunnel TCP handshakes by simply loading the Mullvad gateway address in a browser. Additionally, it is advisable to apply the recommendations from MUL-02-002 to prevent any other applications from potentially leaking.

Fix Note: *This issue was reported during the audit and was fixed by Mullvad with the following PR https://github.com/mullvad/mullvadvpn-app/pull/1827*

MUL-02-005 WP1: *VpnService* logs static internal IPs to Android's *syslog* *(Info)*

During the assessment of the Android app, it was found that the app leaks the static, internal IP addresses assigned to the account to the Android's *syslog*. The information is logged on a *debug* level when a new connection is established via WireGuard. In case an attacker has access to the Android's *syslog* (e.g. via *adb* command or on a rooted phone), s/he would be able to retrieve this kind of information (see below). Knowing the static internal IP might help an adversary to identify a user in combination with a stolen payment record[11].

[11] https://mullvad.net/en/help/why-wireguard/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Retrieve output from *syslog* via *adb* command:

```
adb logcat | grep `adb shell ps | grep net.mullvad.mullvadvpn | cut -c10-15`
```

Logcat output (excerpt):

```
2020-06-03 15:20:23.526 3479-5713/? D/VpnJni: Address added on tun0:
10.66.164.1/32
2020-06-03 15:20:23.528 3479-5713/? D/VpnJni: Address added on tun0:
fc00:bbbb:bbbb:bb01::3:a400/128
2020-06-03 15:20:23.532 3479-5713/? D/ConnectivityService: registerNetworkAgent
NetworkAgentInfo{ ni{[type: VPN[], state: CONNECTING/CONNECTING, reason:
(unspecified), extra: (none), failover: false, available: false, roaming:
false]}  network{115}  nethandle{493937674974}  lp{{InterfaceName: tun0
LinkAddresses: [10.66.164.1/32,fc00:bbbb:bbbb:bb01::3:a400/128,]  Routes:
[0.0.0.0/1 -> 0.0.0.0 tun0,128.0.0.0/1 -> 0.0.0.0 tun0,8000::/1 -> :: tun0,::/1
-> :: tun0,10.66.164.1/32 -> 0.0.0.0 tun0,fc00:bbbb:bbbb:bb01::3:a400/128 -> ::
tun0,] [...]
```

This demonstrates a bad practice on production releases, since it can reveal sensitive information. It needs to be ensured that neither information nor log-levels related to the development environments are appended into production releases.

Fix Note: *This issue was discussed with Mullvad and there is no meaningful mitigation for this issue because Android OS itself logs this metadata. Mullvad does not deem it a sensitive leak given the overall access needed to a device in order to leverage and extract the log files.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

The results of this May-June 2020 project targeting the Mullvad complex are quite positive. After spending twenty days on the scope, six members of the Cure53 team could only spot seven security-relevant items. Moreover, penetration tests and audits against application branches of Mullvad exclusively pointed to issues with limited severities, as demonstrated by the most impactful flaw scoring as *Medium* only.

As regards general impressions, Cure53 wishes to highlight that running the Mullvad daemon - especially the RPC services - as either system-, root- or administrator-user could potentially introduce unnecessary risks to the end-user in case an attacker was able to influence Mullvad applications communication. This was pointed out to Mullvad during this audit and the on-house team immediately started to map out the dependencies and permissions actually needed by the application, with the overarching goal of dropping certain privileges and adhering to the concept of least-privilege

The overall application ecosystem used by Mullvad leaves a sound and structured impression. The overall structure of the application makes it easy to roll out patches and fixes in a structured manner. More than anything, the findings spotted by Cure53 showcase the importance of constantly auditing and re-assessing the current leak vectors, in order to always ensure privacy of the end-users. With that being said, Mullvad does a great job protecting the end-user from common PII leaks and privacy related risks.

Majority of issues can be linked to the overly lax Firewall rules; they are permissions-related or concern missing hardening options. This indicates that these areas could benefit from some improvements. No PII leaks were found, which can be linked to the general concept of Mullvad neither requiring nor storing any PII data remotely or locally, with the exceptions of the anonymous account number, associated vouchers and current payment data.

It has to be noted, however, that a state-funded and persistent threat could effectively identify individual users with MUL-02-002 and MUL-02-007 acting inappropriately in their domain. Despite that, all issues require a strong attacker-model or significant misconfigurations, letting Cure53 conclude that Mullvad already offers sensible protections against most common attacks typically performed against commonly configured systems. Further, it can be said the current maturity level of the Mullvad's applications signals a minimal attack surface, hence furnishing a solid protection in connection to advanced threat actors targeting Mullvad clients.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

As far as OSX application is concerned, a lot of code is shared between the different operating systems, keeping the base minimal. The IPC mechanism is different from the recommended XPC, thus any process can interact with it. This is only an issue if very privileged functionality is exposed, which is not the case here. All files handled by the daemon are in *root*-owned locations or in SIP-protected paths. Thus, attacks via symlinks created by unprivileged users cannot be leveraged. Mullvad does not contain an automatic updater, which drastically eliminates pitfalls. While the functionality exposed by the daemon is not critical, it factually shows many functions that would not need to run as *root*. This mainly applies to sending HTTP requests to the API endpoints. While no serious risks were identified during this project, Mullvad could try to further separate actions that require *root* from those that do not.

Moving on to Android, this application makes a really good impression. No serious issues were spotted and the attack surface is kept very small in this case. The exported *Activity* and *Services* were examined but no data is received via intent calls. In addition, no file operations were found outside of the protected data folder from the *APK*. No other users would be able to inject data into the application's lifecycle, again eliminating a plethora of malicious approaches. The identified information disclosure issue concerns Android's *syslog* and indicates a missing best practice. Due to the very limited exploitation capabilities, this might cause problems only under certain circumstances (see MUL-02-005). However, it is recommended not to log this type of information and to give as little information as possible to the outside world.

To recover a forgotten account-number, several methods are available on Android. For example it is possible to get an account number via a used voucher code. This looked interesting but the voucher code has a good entropy and - given that rate-limiting is used as well - it would be hard to brute-force such voucher codes and acquire account ID. In addition, no public voucher codes were found on the Internet as an entry point to an account. In the end, the worst thing an attacker could do when an account ID is stolen is to add/remove ports and public keys. This seems to be an accepted risk for the chosen architecture. Nonetheless, it is recommended to invest some time into adding an additional layer of security to protect the user's privacy through *FLAG_SECURE* and protection against Tapjacking *a*ttacks (see MUL-02-003).

As a last point, Cure53 will comment on the iOS app which also seems robust and effective in eradicating attacks by default with limited attack surface. The single iOS issue documented during this May-June 2020 project refers to hardening measures and improving security for local storage, especially in the context of unauthorized physical access. The proposed measures should be interpreted as suggestions rather than necessary steps. At the same time, they will help harden the Mullvad iOS app further.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Bringing together evidence from different components clearly suggests that the Mullvad complex came out victorious from this Cure53 external assessment. Despite thorough penetration tests and dedicated audits against various Mullvad apps, clients and APIs, Cure53 was unable to compromise the complex. Mullvad clearly represents a mature design as a function of a sound development process. All findings found during this engagement were patched before the final stages of the project, which is also a very good indicator. The Mullvad complex is definitely on the right track from a security standpoint.

Cure53 would like to thank Linus Färnstrand and the rest of the involved Mullvad application team members for their excellent project coordination, support and assistance, both before and during this assignment.