Fine penetration tests for fine websites

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

Pentest-Report NordVPN Apps, UIs, Addons, APIs & IdP 06.2025

Cure53, Dr.-Ing. M. Heiderich, M. Wege, M. Piechota, A. Belkahla, M. Kinugawa, M. Rupp, MSc. H. Moesl-Canaval, MSc. A. Schloegl, BSc. D. Prodinger, BSc. C. Mayr, S. Possegger, D. Gstir, D. Albastroiu, L. Herrera

Index

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

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 large-scale security assessment of the wide array of the NordVPN applications, components and features, further specified below. The project, which included a penetration test and a dedicated source code audit, was conducted by Cure53 in May and June 2025.

The audit, registered as *NOR-22*, was requested by UAB 360 IT in March 2025 and then scheduled to take place in the late spring and early summer of 2025, thereby giving both parties time to prepare. The project belongs to a well-established cooperation between Cure53 and UAB 360 IT.

Since over twenty projects were performed by Cure53 for the customer, it is important to clarify how the current scope of *NOR-22* fits into the longer-term security-centered collaborative work. In fact, the security standing of the NordVPN application stack was previously assessed by Cure53 across various past projects, although the objectives were not as comprehensive in terms of coverage as in the current inspection. It can be noted that the latest test dedicated to the security of the NordVPN web application was completed by Cure53 about a year ago, i.e., in June 2024. The issues documented during that project can be perused via *NOR-16*.

In terms of the exact timeline and specific resources allocated to the current project, i.e., *NOR-22*, the Cure53 team has completed their research over several weeks from CW23 to CW26 in 2025, as scheduled. In order to achieve the expected coverage for this task, a total of ninety days were invested. Given the large scope and multiple components examined during *NOR-22,* the corresponding need for expertise led to a larger testing team. To be more precise, fourteen senior testers were engaged in this project and were responsible for its preparation, execution, documentation and delivery.

For optimal structuring and tracking of tasks, the assessment was divided into five work packages (WPs), ordered from WP1 to WP5. In addition to five main areas, sub-WPs were also formulated to ensure sufficient granularity and detail of coverage. As a result, WP3 included four detailed realms (WP3.1-WP3.4), while WP2 and WP4 spanned three subWPs each.

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

The breakdown of WPs and sub-WPs completed under *NOR-22* can be consulted next:

- **WP1**: White-box penetration tests against NordVPN browser addons & mobile applications

- **WP2**: White-box tests against NordVPN desktop apps for Win, Linux, macOS
  - **WP2.1**: White-box penetration test & deep dive against NordVPN Meshnet feature
  - **WP2.2**: White-box penetration test & deep dive against NordVPN TP Pro & Light
  - **WP2.3**: White-box penetration test & deep dive against NordVPN Darkweb Monitor

- **WP3**: White-box tests, audits & assessments against NordVPN APIs
  - **WP3.1**: White-box penetration test & deep dive against NordVPN VPN API
  - **WP3.2**: White-box penetration test deep dive against NordVPN TP API
  - **WP3.3**: White-box penetration test & deep dive against NordVPN PDP API
  - **WP3.4**: White-box penetration test & deep dive against NordVPN Pricing API

- **WP4**: White-box penetration tests & audits against NordVPN APIs & UIs
  - **WP4.1**: White-box penetration test & deep dive against NordCheckout UI & API
  - **WP4.2**: White-box penetration test & deep dive against NordVPN LP UI & API
  - **WP4.3**: White-box penetration test & deep dive against NordUCP UCP

- **WP5**: White-box penetration tests & assessments of NordAccount IdP

As can be deduced from the titles of the WPs, the white-box methodology was used. Cure53 was supplied with URLs, test-user credentials, as well as all further means of access required to complete the tests. In addition, all sources corresponding to the test targets were shared to ensure that the project could be executed in accordance with the agreed framework.

The project was completed without any major issues. To facilitate a smooth transition into the testing phase, all preparations were completed in the week prior to the start of the project, precisely in CW22. Throughout the engagement, communications were conducted through a private, dedicated, and shared Slack channel. Stakeholders - meaning the Cure53 testers and the internal staff from UAB 360 responsible for the various NordVPN components - were able to participate in discussions in this space.

Cure53 did not need to ask many questions, and the quality of all project-related interactions was consistently excellent. The testers offered frequent status updates on the examination and emerging findings. Moreover, live-reporting was requested and subsequently used during this project. In the frames of *NOR–22,* the details regarding the spotted issues were provided in the form of Markdown files.

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

Continuous communication contributed positively to the overall results of this project. Significant roadblocks were avoided thanks to clear and careful preparation of the scope, as well as through subsequent support.

The Cure53 team achieved very good coverage of the WP1-WP5 objectives. In the end, twenty-seven issues were discovered and documented as negatively affecting the NordVPN complex, including apps, UIs, addons, APIs, IdP and other components. Eleven problems were categorized as security vulnerabilities and sixteen can be seen as representing general weaknesses with lower exploitation potential.

While quantitatively the number of flaws may seem quite high, it needs to be read in context. Specifically, the scope for this audit was relatively very broad, which explains why a significant number of issues could be observed by the correspondingly large testing team. To properly assess the results of this audit, it is important to qualitatively note that no *Critical* issues were spotted, even though two items were marked as *High* and should not be ignored. More importantly, it is vital to differentiate between the varied outcomes of specific work packages.

The majority of the identified vulnerabilities, namely seven out of the ten, were discovered during the tests NordVPN browser addons and mobile apps (WP1). This showcases that the components targeted by this WP require comparably more improvements than other inspected aspects and elements of the complex.

For the second work package (WP2), a pattern is also visible, as Cure53 was able to unveil most of the general weaknesses. To that end, more comprehensive hardening needs to be incorporated here. In other words, while major threats envisioned for WP2 are prevented, the findings suggest that more attention to detail is warranted. The focus should be placed on the implementation of further best practices in this realm.

The components spanning work packages 3 and 4 revealed a relatively well-hardened security posture. Only a very small number of findings relate to each of these WPs. Last but not least, the tests against the NordAccount IdP (WP5), unveiled some areas which would benefit from further improvements. Resolving the *High*-ranking session fixation arising via trusted browser endpoint (see NOR-22-008) should be a top priority for the NordVPN team.

From a metalevel perspective, this assessment showcased that the inspected NordVPN applications and components differ strongly in their level of security. Cure53 recommends that all findings, regardless of their severity or level of exploitation, are tackled and eliminated in a timely manner in order to guarantee a good level of security and a safe user experience across all NordVPN apps, components and services.

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

The following sections first describe the scope and key test parameters, as well as how the work packages were structured and organized.

Next, all findings are discussed in grouped vulnerability and miscellaneous categories. The problems are then discussed chronologically within each category. In addition to technical descriptions, PoCs and mitigation advice are provided where applicable.

The report ends with general conclusions relevant to this June 2025 project. Based on the test team's observations and the evidence collected, Cure53 elaborates on the overall impressions and reiterates the verdict. The final section also includes tailored hardening recommendations for the NordVPN applications, components and features inspected during this *NOR-22* assignment.

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

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, every ticket has been given a unique identifier (e.g., *NOR-22-001*) to facilitate any follow-up correspondence in the future.

NOR-22-008 WP5: Session fixation via trusted browser endpoint *(High)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment of the NordAccount authentication mechanism, improper session management was identified, resulting in a session fixation vulnerability. User authentication for *my.nordaccount.com* uses OpenID Connect (OIDC) and OAuth2, with Ory Hydra as the authorization server and a custom login-app as the “identity provider”.

In order to facilitate a connection between these applications, a unique challenge code is created for each login. It is then used to redirect the user to the login-app and facilitate communications between the authorization server and the login-app.

For both password-based and passwordless code authentication, users are only redirected after providing valid credentials in combination with the corresponding challenge code. As a result, these endpoints are not susceptible to session fixation issues, since possession of the challenge code alone is not sufficient to obtain access.

For one-time password (OTP) login, however, the combination of the OTP with the challenge code is no longer sufficient. Hence, an additional token must be provided by the user. This item is only issued if the user has successfully completed a prior authentication step such as password authentication.

The vulnerability occurs at the */login/browser/trust* endpoint where users are asked if they want to skip MFA prompts in the current browser for future logins after successfully authenticating via the *login* application. This endpoint does not require an additional token like OTP during authentication. Instead, knowing the challenge code alone is sufficient to obtain access, provided that authentication has already been completed by the account-owner of said challenge code.

The existing parameters indicate that an attacker can initiate a login session for a victim’s email address, obtain the challenge code, forward a URL with this challenge code to the victim, wait for the victim to authenticate, and then take over the authenticated session. This can be achieved using nothing but the known challenge code and the */login/browser/trust* endpoint.

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

Some preconditions must be met for this exploit to succeed, mostly regarding a victim's MFA configuration. But because these preconditions are not uncommon, the attack is feasible both at scale and in targeted scenarios. Upon successful exploitation, an attacker could use this vulnerability to authenticate a NordVPN client and gain access to a victim's private Meshnet. As a result, this issue has been rated as *High* in terms of severity.

Preconditions:

- The victim's account has OTP configured
- The victim's browser is not trusted to skip MFA during sign-in (OTP is requested in the login flow)
- The victim is either logged in or has tried to sign in during their current session (meaning that the relevant *SAID* cookie is set).

NOR-22-013 WP5: Case-sensitive identifier allows OTP rate-limit bypass *(Medium)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment of the NordAccount authentication mechanism, an issue was identified in the rate-limiting implementation for *PasswordlessCode* login. The system employs one time password (OTP) authentication through the */login/code* endpoint, which validates 6-digit codes sent to users' email addresses. To prevent brute-force attacks, the endpoint implements rate-limiting based on two key identifiers: the challenge ID and the user's email address, with the latter used as the identifier.

The vulnerability occurs due to the missing normalization of the email identifier used as a rate-limiting key. The system extracts the email address from either the *SAID* cookie or, as a fallback, the email parameter. Unfortunately, it does not normalize the string to lowercase before using it as a rate-limiter key. This signifies that attackers can treat email addresses with different case variations as distinct identifiers for the same user account, effectively multiplying the allowed request rate.

For an email address containing alphabetic characters, an attacker can generate multiple case variations, each receiving its own rate-limit allocation of 5 requests per minute. This lets attackers significantly increase their attempt rate against the 6-digit OTP codes. While the intended rate-limit is five attempts per minute per account, this flaw indicates that attackers can multiply this rate exponentially: 2^n, where n is the number of letters in the email address. As a result, thousands of attempts per minute can take place instead of the intended five.

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

Additional rate-limiting measures exist at the infrastructure level. This entails stricter limits on *auth.nordaccount.com* for challenge ID retrieval and Cloudflare's protections on NordAccount endpoints. However, these can be circumvented through challenge ID accumulation over time and IPv6 subnet rotation combined with fingerprint-resistant HTTP clients that avoid Cloudflare Turnstile.

While this attack requires more resources and technical knowledge than a typical brute-force attempt, it could enable unauthorized account access without user interaction. This risk translated to a *High* severity rating.

NOR-22-014 WP1: Privacy settings disabled via non-encapsulated APIs *(Low)*

The NordVPN WebExtension may display a warning banner at the top of the displayed page to inform the user of the warning. To restrict access to the DOM of the warning banner, the HTML elements that make up the banner are placed in the closed Shadow DOM. Cure53 discovered that this restriction to DOM access could be easily bypassed in Firefox.

In particular, Firefox allows access to the elements on the Shadow DOM from the Light DOM via several event object properties. Specifically, *Event.prototype.originalTarget*[1], *Event.prototype.explicitOriginalTarget*[2], *UIEvent.prototype.rangeParent*[3] and so on are known to allow it. Due to this, a malicious page can disable the privacy settings by manipulating the warning banner UI on the shadow DOM via these JavaScript properties.

NOR-22-015 WP1: MitM attacks via HTTP URL of allowlisted hosts *(Medium)*

Certain hosts, such as **.nordvpn.com*, remain accessible without the VPN connection, even when the extension's *kill switch* setting is enabled. It was discovered that the hosts which can bypass the *kill switch* setting allow access not only via *https:* but also through *http:* protocols.

Notably, selected hosts on the allowlist have *HSTS* preloading enabled, but this does not apply to all of them. Any host listed here that does not have *HSTS* preloading enabled may be opened with the *http:* URL and may be susceptible to Man-in-the-Middle (MitM) attacks. These hosts include **.nordsecurity.com*, **.napps-1.com*, **.icpsuawn1zy5amys.com* and others.

Additionally, the domains added to the *Split Tunneling* settings to bypass the VPN tunnel are also allowed to be accessed via the *http:* protocol. Since there is no setting to disallow *http:* access, users will automatically allow *http:* access when they add domains to the *Split Tunneling* settings.

[1] https://developer.mozilla.org/en-US/docs/Web/API/Event/originalTarget
[2] https://developer.mozilla.org/en-US/docs/Web/API/Event/explicitOriginalTarget
[3] http://help.dottoro.com/ljifpseq.php

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

NOR-22-016 WP1: IP leak via clickjacking on NordVPN *info* API *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

Another issue was discovered on an endpoint previously spotted by Cure53 and reported as *NOR-15-021 WP1: IP leak via Clickjacking on NordVPN's Insights API (Low).* In fact, this flaw was fixed following a previous audit. However, a specific NordVPN endpoint, located at *https://web-api.nordvpn.com/v1/ips/info*, reflects the user's IP address in its response without implementing the necessary *X-Frame-Options* (XFO) header.

The observed behavior raises significant concerns, as it could be leveraged to deceive users into inadvertently revealing their IP addresses. The 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. Beyond circumventing the protections typically offered by the *kill switch* feature, this would also compromise user anonymity.

NOR-22-017 WP1: VPN bypass via direct access to top-level domain *(High)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

While reviewing the repairs introduced in response to *NOR-15-006*, it was discovered that the мон top-level domain (TLD) is active. It can be used to access websites outside the VPN by directly navigating to them.

Because this domain lacks dot characters, the NordVPN extension treats it as a private hostname. As a result, it facilitates a direct connection to an external host, effectively bypassing the proxy.

It is worth noting that ICANN decided in 2013 that TLDs are no longer permitted to have *A* or *AAAA* records. Nevertheless, some TLDs disregard this rule[4]. The мон TLD serves as a clear example, as it can be successfully loaded, demonstrating the bypass. Furthermore, access to this host remains possible even when the *kill switch* option is enabled.

NOR-22-018 WP1: Embedded modals removable via *document.write* *(Low)*

Following the discovery in NOR-22-014, it was also observed that the warning banner displayed by the NordVPN WebExtension can be dismissed by using the *document.write* JavaScript function, which completely rewrites the DOM. As a result, the previously created warning banner is removed from the page, preventing users from being notified about the related risks.

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

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

NOR-22-019 WP4.3: Missing rate-limiting enables DoS of MFA setup *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During auditing of the NordAccount web application, it was found that the enabled MFA OTP API endpoint lacks rate-limiting. Although authentication is required, a single client can send several hundred requests per second.

This problem can be exploited to degrade availability for other users, effectively causing a Denial-of-Service condition until accounts are manually blocked. During testing, which was limited to five second intervals to avoid production impact, Cure53 confirmed that flooding the endpoint blocks its availability for other users.

NOR-22-021 WP1: Clickjacking by applying CSS via *style* element *(Low)*

It was discovered that the issue reported as *NOR-15-018 WP1: Clickjacking attack allows disabling privacy settings (Low)* in the previous audit is not completely fixed. The fix was to set a *MutationObserver* and reject the attribute setting to the shadow host element that makes up the warning banner.

Unfortunately, this fix does not prevent styles from being applied via the <*style*> element. Therefore, clickjacking can still be achieved by making the banner transparent via the *opacity* property set in the <*style*> element.

NOR-22-022 WP5: Insufficient rate-limiting on MFA TOTP login verification *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment of the NordAccount authentication mechanism, an issue was identified in the rate-limiting implementation for verification of a multi-factor authentication (MFA). The */login/mfa/otp* endpoint validates 6-digit TOTP codes and implements rate-limiting of five attempts per minute based on the challenge ID.

However, the rate-limiter does not account for token reuse across multiple challenges. After successful password authentication, the obtained token remains valid and can be paired with new challenge IDs obtained through the standard login flow.

Since each challenge ID maintains its own independent rate-limit counter, an attacker can accumulate multiple challenge IDs and use them with the same token in order to circumvent the 5 TOTP attempts per minute. The endpoint does not implement rate-limiting based on the targeted account or token and no Cloudflare protections are triggered for the MFA verification endpoint. Consequently, sustained brute-force attacks from a single IP address can be feasible.

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

While this attack requires initial password knowledge or passwordless OTP code, its presence still reduces the effectiveness of the protection offered by the TOTP second authentication factor. Due to its difficulty, the attack's severity rating was set to *Low.*

NOR-22-024 WP1: User's real geolocation leaked via an iframe *(Low)*

It was observed that the issue previously reported as *NOR-15-017 WP1: User's real geolocation can be leaked via multiple ways (Low)* has not been fully resolved. Although PoC #2 is no longer functional, PoC #1 remains reproducible.

The content script of NordVPN includes a custom script that overwrites the methods of the Geolocation API on every web page load. This is intended to spoof the user's real location and prevent leaks. However, the configuration of this mechanism was found to be flawed. Specifically, the script is only injected into top-level frames.

The *all_frames*, *match_about_blank*, and *match_origin_as_fallback* flags were not set to *true* in the *manifest.json* file. Notably, the *csSpoofing.bundle.js* file is inserted by the background script. As a result, a malicious website can still access the Geolocation API via an iframe.

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 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-22-001 WP3.4: Potential panic on MQTT coupon consumer endpoints *(Info)*

***Note**: This finding has been determined to be a false positive. It is closed and no further remediation action will be taken.*

During a static analysis of the Golang pricing backend API, it was discovered that the *CouponPlanConsumer* and *RabbitMQConsumer* MQTT endpoints lack input validation. These items are used for batched coupon creation and events.

The noticed oversight could lead to a Denial-of-Service condition if malformed data is processed, causing the application to panic. However, these MQTT endpoints are not directly exposed to external networks and are only accessible within the NordVPN backend systems. Therefore, they are not considered as representing directly exploitable vulnerabilities from an external perspective.

NOR-22-002 WP5: Exposed *metrics* endpoint on NordAccount *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment of the NordAccount infrastructure, publicly accessible Prometheus *metrics* endpoints were identified across multiple NordAccount domains. The */metrics* endpoints on *nordaccount.com*, *ndaccount.com*, and *nordauth.com* are exposed without authentication and return detailed runtime information about the Go application. The disclosed data includes garbage collection statistics, Go routine counts, memory usage patterns, and HTTP request metrics with endpoint-specific performance data.

While these *metrics* do not directly expose sensitive user data or credentials, they provide unnecessary visibility into the application's internal state and operational characteristics. The exposed data includes detailed HTTP request counts and response times for all endpoints, memory allocation patterns, process information, and Go runtime statistics. The exposed information could assist attackers in reconnaissance activities, helping them understand traffic patterns, identify high-value endpoints, infer user behavior patterns, and potentially discover performance bottlenecks that could be exploited for denial-of-service attacks.

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

NOR-22-003 WP3.4: Potential unauthenticated DoS via GQL alias overload *(Info)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

Cure53 determined that malicious users may be able to exploit GraphQL alias overloading[5], which involves crafting requests that are commonly utilized to launch DoS attacks on GraphQL APIs. This strategy is particularly effective since rate-limiting measures often fail to detect these requests.

NOR-22-004 WP2: Linux client ships outdated OpenSSL with known CVEs *(Info)*

***Fix Note**: This issue has been mitigated by the development team and verified by Cure53 to be working as expected.*

Analyzing the Linux NordVPN client RPM, it was found that the OpenVPN binary is included in this component. However, it is statically linked to an outdated, vulnerable OpenSSL version.

Specifically, OpenSSL 3.0.14 contains multiple known CVEs which are potentially affecting the security of OpenSSL[6]. Among these is *CVE-2024-6119,* which describes a potential DoS on *X.509* name checks. It must be noted that a full impact analysis of the respective CVEs on OpenVPN was not performed due to time constraints. This is why the issue is only rated *Info.*

NOR-22-005 WP2: Weak configuration file encryption key derivation *(Info)*

When inspecting the configuration file handling, Cure53 found that the Linux client uses a custom file encryption logic. The observed configuration contains multiple weaknesses:

- The *settings.dat* file encryption key is generated using an insecure random number generator which makes it easier to guess.
- The key-encryption-key used for encrypting the *settings.dat* encryption key, stored as *install.dat*, is derived from a static value using MD5. The use of MD5 is considered insecure because the collision resistance of MD5 is broken.
- The secret used for deriving the key-encryption-key is a hardcoded value built into the binary at compile time. As this value is constant for every installation, it is possible to extract it from the binary and break every installation.

[5] https://portswigger.net/web-security/graphql
[6] https://openssl-library.org/news/vulnerabilities-3.0/index.html

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

It seems like this feature is intended as an obfuscation technique rather than a fully security-focused configuration file encryption. This is also underlined by the fact that multiple sections of this code contain comments indicating that this feature should be removed once the code is published openly. As the code is already publicly available but the feature is still present, the removal appears to not have been completed in the end.

NOR-22-006 WP5: Insecure random digit generation due to modulo bias *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

The audit of the six-digit code generator used for one-time passwords (OTP) looked at passwordless login and multi-factor authentication. It was found that digit generation from the random byte stream introduces entropy loss due to insecure sampling.

Specifically, using a modulo operation (e.g., byte % 10) results in a non-uniform distribution. The input byte ranges from 0 to 255. For example, the digit '4' is sampled in 26 out of 256 cases, while '8' is only sampled 25 times, thus making some digits slightly more likely to occur than others.

The impact of this finding is set to *Low* due to the small bias introduced. However, since a six-digit OTP already provides minimal entropy[7], even minor reductions can compromise its security margin. Therefore, there is little tolerance for suboptimal randomness in such constrained contexts.

Uniformity test results (n=10^7):

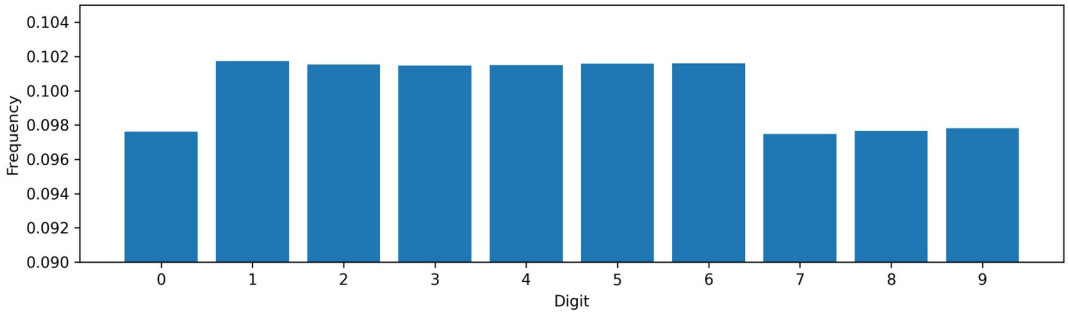

Fig.: Probability distribution for digits sampled from the random byte stream. Note that the higher probabilities range from digits ‘1’ to ‘6’ as the digit array starts with digit ‘1’

[7] https://pages.nist.gov/800-63-3/sp800-63b.html#sfotpa

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

NOR-22-007 WP2: Timing-unsafe verification code comparison in DWM *(Info)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

While inspecting the source code of the Dark Web Monitor, Cure53 found that the logic utilizes PHP string's equality operator in order to compare the email verification codes. This operator induces a linear time relationship regarding the number of equivalent prefix bytes between the code and the user input.

This relationship can act as a side-channel, facilitating non-negligible optimization of a naive brute-force attack. This attack requires measuring the time differences and is often heavily dependent on environmental factors such as processor and network load. This explains why this ticket has received a reduced severity rating of *Info*.

NOR-22-009 WP2: Usage of *bcrypt* for password hashing in TP admin API *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

While auditing the Threat Protection API source code, the observation was made that this backend API utilize *bcrypt*[8] for password hashing. Since the *bcrypt* hashing algorithm is no longer considered relevant[9], the development team should incorporate a contemporary password hashing algorithm with enhanced security assurances.

NOR-22-010 WP2: Basic authentication utilized in TP admin API *(Low)*

***Client Note:** The potential risk associated with this issue has been formally accepted by the client. Based on their assessment, the decision has been made to omit the fix, acknowledging the residual risk.*

Cure53 acknowledged that the Threat Protection admin API endpoints employ HTTP basic authentication, which transmits credentials in a Base64-encoded format that can be decoded with relative ease for each request. This introduces several potential interception risks. Even when using HTTPS, the credentials will remain at risk if secure connection is breached.

Additionally, HTTP basic authentication is prone to CSRF attacks and lacks sophisticated security features such as token expiration and MFA. These must be applied to elevate the security premise to a first-rate standard.

[8] https://en.wikipedia.org/wiki/Bcrypt
[9] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

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

The implementation of the middleware used in the API endpoints has certain characteristics. First, the router struct expects an *authorizationMiddleware* of type *echo.MiddlewareFunc*. Secon, this middleware is injected via dependency injection, using the name "*internal_authorization_middleware*". Third, the actual implementation of "*internal_authorization_middleware*" is set up in a particular manner presented next.

NOR-22-011 WP2: Privilege escalation via loading ML tagger in TP as DLL *(Medium)*

While auditing the Threat Protection component of the NordVPN desktop clients, the ML-based tagger was noticed. This tagger utilizes a TFLite[10] runtime to classify files with the aid of a machine learning (ML) model. The model is currently downloaded as a *.dll* dynamic library, which is loaded by the privileged NordVPN service.

The authenticity of the DLL is based on a SHA256 checksum received in response to a previous API call to *tp.nordvpn.com*. Communication with the NordVPN team indicated that these calls and their implementing C++ library do not utilize certificate pinning.

It is therefore possible for an attacker to utilize the ML model loader as a means to escalate privileges on a restricted machine. One scenario to consider would be of an attacker who controls the DNS server and the private key for a trusted certificate authority. These are two very strong preconditions, but ones that can be met, for instance by a malicious administrator in a corporate enterprise setting. To that end, this could even be used to achieve Remote Code Execution (RCE) with elevated privileges.

Unfortunately, the Cure53 team was unable to fully implement this attack during the time available for the assessment. While the issue has been filed as miscellaneous, it should be mitigated as soon as possible.

NOR-22-012 WP2: Unloading kernel minifilter undetected by Windows app *(Info)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During dynamic testing of the NordVPN Windows application, an observation was made regarding its antimalware feature. This feature relies on two main components: the *mshield* kernel minifilter and a userspace agent with a corresponding user interface. The application's UI indicates that the anti-malware feature is active and running.

The *mshield* minifilter is designed to scan and block malicious files during Internet downloads. However, it was found that this minifilter driver can be unloaded with administrative permissions. When the driver is unloaded, the antimalware scanner no longer functions as intended. Notably, the NordVPN application's UI does not reflect this change in *state*, continuing to display that antimalware protection is *active* when it is not.

[10] https://ai.google.dev/edge/litert

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

NOR-22-020 WP4: *Nonce* reuse facilitates exposure of OAuth *access* tokens *(Info)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During an audit of the token authentication mechanisms for the NordUCP and NordCheckout clients, testers were able to decrypt *access* tokens due to flawed encryption. In the OAuth authorization code flow, clients do not expose *access* tokens to the user-agent (browser) in order to reduce their overall exposure[11].

To that end, *access* tokens are encrypted by the backend client before they are provided to user-agents for authentication. The NordUCP and NordCheckout clients use Libsodium's secretbox authenticated encryption for these *access* tokens. Notably, Libsodium secretbox utilizes *XSalsa20*, which is a stream cipher that generates a key stream based on a key and a *nonce*, then XORs this steam with the plaintext[12].

It is explicitly noted in the documentation that *nonces* must never be reused with the same key, as this would result in the same key stream for different ciphertexts. Disregarding these requirements, the affected clients use a static *nonce*, meaning that every token is encrypted with the same *nonce* and key. Testers exploited this by performing statistical analysis on sampled tokens to recover the key stream, allowing them to decrypt any token and retrieve the raw OAuth token.

This issue could not be exploited because code review and client consultation confirmed that raw OAuth tokens are not used for authenticating to any resource server. Even if the message authentication code was to be compromised, there would be no impact. This is because all tokens are validated against a database before being accepted by the application. This evaluation would, of course, change if the decrypted OAuth *access* token were ever used directly for authentication. In this sense, the issue should still be remediated.

NOR-22-023 WP2: Command injection in URL handling *(High)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment, a command injection vulnerability was identified within the fallback logic of the *LaunchWebAddress* function. The root cause was a combination of permissive URL validation and insecure construction of a *shell* command. It was observed that the application's URL validator accepted schemas with leading whitespace characters, such as tabs or newlines. Although such a URL passes the host allowlist and schema checks, the primary process launch method fails because the operating system cannot find a handler for the malformed protocol.

[11] https://datatracker.ietf.org/doc/html/rfc6749#section-1.3.1
[12] https://libsodium.gitbook.io/doc/secret-key_cryptography/secretbox

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

This failure triggered a vulnerable fallback mechanism which attempted to execute the URL via *cmd.exe*. The input sanitization within this fallback, which replaced *&* with *^&*, was insufficient. By crafting a payload that combined leading whitespace with a pre-escaped command separator, it was possible to bypass both the validation and sanitization. Ultimately, arbitrary commands could be injected.

Initially, this issue was classified as a *Medium*-risk miscellaneous vulnerability because the vulnerable function was believed to exclusively process URLs from a configuration file only writable for an administrator. However, discovering that the same vulnerable function also processes URLs from application notifications significantly elevated the significance of this threat.

This created a viable attack vector where an adversary could deliver a malicious payload after finding a way to control notification content. A user interacting with such a notification would inadvertently trigger the command injection, leading to arbitrary code execution on their workstation with the privileges of their user account.

NOR-22-025 WP4.1: User ID disclosure via cart API endpoint *(Low)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

During the assessment of the NordCheckout application, an information disclosure vulnerability was identified in the cart retrieval functionality. The */api/carts/:cartId* endpoint returns cart data that includes the authenticated user's ID when accessed. This occurs because the frontend application does not remove the user's ID from the backend response. As a result, an attacker can send the victim of a NordCheckout URL a cart ID that they control in order to disclose the authenticated victim's user ID.

The vulnerability arises from the cart service implementation which queries the backend API with the authenticated user's ID as a parameter. When a logged-in user visits a checkout URL with an arbitrary cart ID, the application updates the cart data with the victim's user ID via session using a *PATCH* request. The latter exposes the victim's user ID in the *GET* API response. While this information does not directly compromise account security, it provides attackers with internal user identifiers that could facilitate user tracking or be leveraged for more sophisticated attack chains.

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

NOR-22-026 WP2: Arbitrary object instantiation from untrusted type name *(Info)*

***Fix Note**: This issue has been fixed by the development team and verified by Cure53 to be working as expected. The described issue no longer exists.*

An insecure code pattern was identified where the application used *Activator.CreateInstance* to instantiate objects based on a type-name provided as a string. The investigation confirmed that the parameter could be sourced from application notifications, potentially allowing attackers to specify an arbitrary class to be instantiated within the application's process.

While this pattern can lead to unintended code execution via class constructors, the immediate risk in this context was found to be significantly limited. This is because there were no arguments passed to the constructor, restricting the impact to classes with parameterless constructors that might have exploitable side effects. Due to the limited practical exploitability, this issue has been classified as *Info*.

NOR-22-027 WP2.2: Potential Threat Protection archive scanner DoS *(Medium)*

Archive scanning of the Threat Protection component was initially found to be susceptible to a DoS condition due to the insufficient enforcement of resource limits when processing recursively nested archives.

During testing, it was observed that the scanner's recursive processing logic, while correctly identifying and queuing nested archives for analysis, appeared to lack safeguards such as limits on maximum recursion depth and the aggregated size of all unpacked data across all layers. This led to a perceived DoS condition where the scanner process would terminate prematurely.

This premature termination could potentially be leveraged for a scanner bypass. During testing, an archive containing a standard EICAR antivirus test file in its deepest layer was used to attempt to trigger incomplete analysis.

However, further investigation revealed that the initial DoS observations were influenced by the concurrent operation of the Windows Search service, which probably tried to index unpacked archives. When the Windows Search service was disabled, the DoS condition was no longer reproducible. Nevertheless, even after disabling the Windows Search service, significant and prolonged increases in memory, disk, and CPU utilization by the Threat Protection processes were still observed.

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

In environments with high system load, this could still lead to a service malfunction. To definitively trigger the DoS, it would likely be necessary to further optimize the parameters of the crafted ZIP or to pack not just large files filled with null bytes, but also files specifically designed to be passed to Bastet taggers for further analysis, causing an even greater increase in resource consumption.

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

Conclusions

As noted in the *Introduction,* this large-scale *NOR-22* project entailed dedicated security examinations of multiple components within the NordVPN product- amily. The Cure53 team, tasked with assessing the scope with white-box methods in May and June 2025, found twenty-seven issues during this cooperation.

Rather than issuing a general verdict for the entire *NOR-22* inspection, it is more prudent and appropriate to recognize that the components targeted across five different work packages substantially differ in terms of the existing security posture and detailed implementational characteristics.

While some NordVPN components - especially the browser addons and mobile applications (WP1) still suffer from explicit dangers in the form of security bugs, other components managed by UAB 360 IT team demonstrate undeniable security strength and comprehensive deployment of protective measures.

Starting the discussion of specific approaches and outcomes, the NordVPN WebExtension and its associated components will be commented on first. The auditors reviewed the related *manifest.json* file to identify potentially insecure configurations, such as overly permissive permissions.

Particular attention was paid to files listed under *web_accessible_resources*, as they are common vectors for clickjacking attacks. It was noted that the extension enforces a strict *Content Security Policy (CSP)*, which significantly helps mitigate the risk of exploiting potential XSS vulnerabilities.

However, the background script was observed injecting the *csSpoofing.bundle.js* script to spoof geolocation. Notably, the *manifest.json* does not include the *all_frames*, *match_about_blank*, or *match_origin_as_fallback* flags. The absence of these parameters means the spoofing script does not execute in critical contexts such as iframes, about:blank, blobs, or data URIs, leaving a potential vector for geolocation leakage. This aligns with the previously reported vulnerability (*NOR-15-017*), which was refiled as only partially addressed in NOR-22-024.

An examination of the content script's behavior revealed certain flaws. Although the extension uses a *MutationObserver* to monitor and preserve its injected UI, it remains vulnerable to DOM manipulation. Specifically, it was found that calling *document.write* can completely overwrite the DOM, effectively removing security-related warnings displayed by the NordVPN extension. This issue is documented in NOR-22-018.

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

Additionally, a review of the request proxying logic revealed a recurrence of a prior vulnerability. The dotless domain мон is currently live but is not listed in the *ACTIVE_TLDS* array, which is used to differentiate public TLDs from private hostnames like localhost. This oversight allows external servers to potentially bypass the VPN, posing a significant risk. As such, this issue has been categorized as *High* in terms of severity. Detailed documentation can be perused via NOR-22-017.

The implementation of the warning banner inserted into pages by the extension was checked. The inside of the warning banner, which is placed in Shadow DOM with the intention of restricting access, retains accessibility via certain JavaScript properties. This could lead to privacy settings being changed unintentionally (NOR-22-014).

Further investigation determined that the previously reported clickjacking vector, which was effectuated by applying transparent styles, has not been eradicated fully, In fact, using the *style* element instead of the *style* attribute, one can succeed with the described approach (NOR-22-021). Since the Shadow DOM is not a reliable security boundary, the current implementation method should be reconsidered.

The PAC script used within the extension was checked. Due to the lack of the protocol check, MitM attacks were possible. In particular, the hosts allowed by the kill switch and Split Tunneling features outside the VPN connection are at risk in the described context (NOR-22-015).

Further review of the previously reported issues was conducted to compliment the already found recurrences. An issue was identified as potentially allowing an IP leak via clickjacking (NOR-22-016), though it is noted that its presence was confirmed on an endpoint different from the one reported before.

The Android application underwent a comprehensive security review, with particular emphasis on exported components, data storage practices, inter-app communication, and *deep link* handling. Activities such as *WelcomeActivity* and *ControlActivity* were thoroughly analyzed for input sanitization and proper handling of serialized extras.

Deep link flows, including those related to *nordvpn://connect*, *nordvpn://messages*, and *nordvpn://settings*, were all validated to ensure correct parsing and enforcement of input validation logic. Importantly, *deep link* parameters used for VPN connection setup are securely parsed by the *ConnectionLinkProcessor*, with no opportunity for injection or other misuse.

The application's *BrowserActivity*, which encapsulates all WebView logic, was found to have JavaScript enabled. Fortunately, it does not accept any user-controlled URLs, minimizing the risk of injection attacks.

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

Nonetheless, because of the default WebView configuration on API 28 and 29, where file access and universal access from file URLs are implicitly allowed, a minor risk remains. Particularly, it is worth considering a scenario of local file exposure, warranting attention in the context of device compromise.

Data storage across both *SharedPreferences* and local databases is encrypted using a custom implementation backed by hardware-keystore-based AES-GCM encryption. Sensitive clipboard operations are correctly flagged and time-limited, and tapjacking protections, screenshot restrictions, and biometric controls are properly enforced throughout the app.

Authentication and device binding flows also demonstrated solid implementation. The app uses a non-spoofable device identifier from *MediaDrm*, combined with *root* detection and asymmetric cryptographic verification. While the *root* checks are not tamper-proof, the use of public-key-based binding adds a strong layer of trust.

Similarly, MFA flows and token management benefit from strict input sanitization and encrypted storage, with shared use of *TextCipher* and *AndroidKeyStore*. API response signature verification using public keys further bolsters the trust boundary between client and server. Overall, the Android application presents a mature security posture with strong cryptographic hygiene, secure IPC handling, and general practices that minimize the attack surface.

For the iOS application, a deep static analysis revealed generally strong adherence to security best practices, including ATS enforcement, proper Keychain usage with restricted accessibility (*WhenPasscodeSetThisDeviceOnly*), and appropriate cleanup of session tokens upon logout. Jailbreak detection is present and triggers alerts post-login, although it was bypassed for analysis purposes. No sensitive data was found exposed in local storage, logs or user defaults.

Even though SSL pinning was successfully bypassed, no sensitive information was observed in transit. Hence, it is safe to say that the app does not disclose any secrets or credentials in an insecure manner.

The backend APIs underwent a thorough assessment with focus areas including OAuth authentication, user and referral management, subscription flows, and the Meshnet service. The OAuth implementation was reviewed in detail and found to follow industry best practices, including the use of PKCE to prevent interception of authorization codes.

In the user management and referral services, both access control and data integrity mechanisms were carefully evaluated. Access control lists (ACLs) were verified to be correctly enforced throughout the relevant endpoints. This effectively mitigates risks related to duplicate referrals, unauthorized link creation or privilege escalation.

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

The Meshnet API, responsible for device-to-device connectivity, was also carefully reviewed. *Authorization* tokens are strictly parsed and validated by a trusted internal SDK. While *scope* and *expiry* checks are not visible in the calling logic, the underlying SDK is assumed to enforce them correctly. Core functionalities - such as invitation management, peer connection setup, and token-based authentication - were all found to be implemented with proper validation and access controls. For example, they rely on UUIDv4 tokens and reject malformed or duplicate requests.

SQL query usage across critical paths was also audited, with attention to raw queries and potential injection vectors. No vulnerabilities were identified. Additionally, the credentials and token services demonstrated secure practices in key generation, cache handling, and session management, with proper time-to-live (TTL) logic applied to sensitive entries.

Overall, the API surface exhibits a solid and mature design. The absence of critical flaws, combined with thoughtful implementation details such as token signature validation and strict input sanitization, underscores the maintaining team’s strong commitment to security best practices for these NordVPN components.

The application's *deep link* handling, particularly the authentication flow, was subject to a thorough review. The primary line of investigation focused on a potential login CSRF scenario, where an attacker could craft a malicious *deep link* to log a victim into the attacker's account, potentially to manipulate settings or add the victim to the attacker's Meshnet. Practical tests to execute the login CSRF attack on a separate machine were unsuccessful.

Further analysis indicated the presence of a robust backend control mechanism. The authentication service appears to validate that the token is used by the same client session that initiated the login process, effectively rejecting tokens from unintended sources. This validation demonstrates a well-architected security control that successfully anticipates and mitigates potential issues.

The assessment included a review of how the application handles the creation of external processes, focusing on the *LaunchWebAddress* function. This investigation revealed a severe miscellaneous vulnerability related to opening URLs. A command injection flaw was identified within a fallback code path, which could be triggered by supplying a specially crafted URL designed to bypass the application's schema validation checks. Successful exploitation of this flaw led to a theoretical, one-click RCE scenario (see NOR-22-023).

The XPC interface of the privileged component of the macOS NordVPN client - named *Shield* - has been validated. It was confirmed that sufficient checks are in place that ensure only authorized entities can send messages to this backend API. This check is being enforced by inspecting the signing certificate of the calling process.

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

The kernel parts of the application, the filesystem monitor *mshield* and the sandbox minifilter drivers were strong subjects of the Cure53's analysis. While examining the components for potential crashes and out-of-bounds access vulnerabilities, it was noted that the NordVPN application did not correctly recognize the unloading of a minifilter driver, as documented in issue NOR-22-012.

The Cure53 testers reviewed the application's source code to identify potentially dangerous or insecure code segments, such as dangerous function usage and improper input handling. The application demonstrates a proper level of input sanitation and validation, which reduces the risk of injection-based attacks and memory-related vulnerabilities.

Cure53 also looked into the IPC communication between application and service running over named-pipes and *grpc.* The named-pipe communication boasts robust architecture and appears well-secured. ACLs are properly defined to ensure that only authorized users and processes can initiate or interact with the pipe. Communication over the pipe is encrypted to provide confidentiality and integrity for transmitted data. Furthermore, the maximum number of concurrent connections is explicitly set and managed appropriately, safeguarding the system against resource depletion or DoS attacks.

The file-sharing functionality was tested to ensure it could withstand abuse scenarios, especially those related to misconfiguring the *download* folder. The application correctly blocked attempts to redirect file downloads to privileged system paths. This behavior shows that privilege escalation through directory manipulation is not presently feasible.

The Linux client has been evaluated as clean and well-organized. Special focus was put on the Meshnet and FileShare features. Security in the entire Linux desktop application is heavily based on Linux file permissions, which are utilized well.

Attempts to escalate privileges for reading files using the FileShare service were unsuccessful, as this component is started from the user's perspective. Conversely, stopping the service also causes the opened firewall port to be closed, successfully remediating an issue pointed out in previous audits of the NordVPN complex.

Special attention was also given to the ThreatProtection feature available on macOS and Windows. This feature utilizes a set of taggers to classify files as malevolent or benign, where the classification strategy varies based on the specific tagger. The simple hash-based tagger uses a local cache and the Nord API to check signatures for known viruses. The ML based tagger uses a TFLite[13] runtime to classify files using the *quickscan* ML model.

[13] https://ai.google.dev/edge/litert

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

Interestingly, the quickscan model is loaded into the privileged NordVPN process as a DLL. This creates unnecessary risks for privilege escalation, which have been noted in NOR-22-011. The current strategy also might leak the model parameters to any malicious user of the feature with the know-how to deconstruct the DLL. Depending on the effort invested into such models, a different strategy might be required.

The Linux client makes heavy use of *exec.Command()* which is usually a common source for command line injections. However, a close review revealed that the NordVPN engineers understand this issue and no problems could be found there.

To ensure no leakage of network traffic outside of the VPN tunnel and to implement certain features like the Internet Kill Switch, *iptables* filter rules are set up by the client. These rules, setup and teardown code have been closely reviewed and found to be in good working order.

During dynamic tests of the installer, it was found that there is a negligible issue in the setup logic for *zypper*-based distributions like *openSUSE.* Specifically, the installer checks for the presence of a file path which is actually an HTTP URI. Should a path that matches this URI be present in the current directory, GPG checks for the *zypper* repository will be disabled. This appears to be a mistake in the installer, but triggering it is highly unlikely.

It was additionally found that an outdated version of OpenSSL containing known vulnerabilities is shipped as part of the OpenVPN binary. Overall, however, the Linux client is in very good shape, showcasing that the engineering team understands common vulnerabilities and how to avoid them.

The file sending mechanism in Meshnet underwent a tailored review, with particular attention paid to the *deep link* handler responsible for initiating file transfers. The assessment was focused on discovering potential path traversal vulnerabilities, wherein an attacker could craft a malicious *deep link* to force a file to be written to an arbitrary or privileged location on the user's filesystem. Fortunately, the investigation confirmed that the application is not vulnerable to this attack vector.

Instead of passing a direct or relative filename in the *deep link*, the application uses an abstracted file ID. A review of the code further revealed that this file ID is validated against a strict character-level allowlist, which permits only alphanumeric characters and a minimal, safe subset of symbols. This approach makes the injection of path traversal sequences impossible. As such, it positions the design as a robust and effective control that demonstrates a strong, security-first approach to file handling processes.

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

The assessment of the Libmangler library, the core engine for traffic inspection and modification, focused on whether its filter and modifier rules could be abused to cause information leaks. A combination of code review and manual testing was performed to probe for unintended side effects. No methods for bypassing the filtering mechanisms or exfiltrating the data could be crafted for successful use.

The Libmangler component demonstrates a robust design that effectively prevents its powerful features from being used as an attack vector against the user. While the assessment did not identify any direct security vulnerabilities, a specific investigation into potential XS-Leak scenarios was carried out. It was confirmed that Libmangler's functionality for filtering tracking parameters operates by completely removing them from URLs, which results in a measurable alteration of the final request's length.

Although no practical exploitation of this behavior was achieved during the engagement, this side-channel could theoretically be used to infer user activity in some edge cases. Therefore, as a proactive security hardening measure, it is worth considering a change in this implementation. A proposal would be to replace the tracking parameters with placeholder data of the exact same length, rather than removing these entirely. This would preserve the original URL length, fully mitigating this theoretical information leakage vector without impacting functionality.

An assessment of the Quickscan module was conducted as well. It centered on its static analysis engine for portable PE files, which leverages the *libfex* library and a machine learning model. The investigation pursued two primary scenarios: a potential TOCTOU vulnerability and a bypass of the scanning engine.

The TOCTOU investigation revolved around the initial check of the file's size, where the file could theoretically be swapped before further processing. However, it was confirmed that this had no security impact, as the reported filesize was overwritten later in the process and not used for any security-critical decisions.

The core of the testing focused on attempts to craft a malformed PE file that would fail to be parsed by *libfex,* thus avoiding ML analysis, but would still be successfully executed by the Windows OS loader. These tests demonstrated that files sufficiently corrupted to evade *libfex* parsing were also rendered non-executable by the operating system. The Quickscan module proved to be resilient against various bypass attempts, indicating that its parsing logic is well-aligned with that of the OS loader.

Furthermore, the security of the archive scanning capabilities was scrutinized, which led to the discovery of a potential DoS vulnerability. While the assessment confirmed that protections against simple *ZIP* bombs (via compression ratio) and path traversal are in place, the handling of nested ZIP archives was found to be risky.

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

The investigation demonstrated that a specially crafted archive with a deep level of recursion could trigger a DoS condition, which was observed when other system services, namely Windows Search, were also active. Without these additional system loads, the scanning service did not fully terminate prematurely, but still exhibited significant spikes in resource consumption.

The implication of this DoS could be relevant if the scanning service terminates its analysis prematurely. In this context, an unevaluated archive, potentially containing a malicious payload such as the EICAR test file, could be saved to the disk without proper inspection or detection (see NOR-22-027).

The Threat Protection backend API, developed in Golang, was subjected to a comprehensive security analysis. The codebase is well-structured and has optimal quality, with strict enforcement of authentication and authorization controls across all endpoints. The exposed API interface consistently implements robust input validation. Stringent file upload size restrictions are applied to mitigate abuse and resource exhaustion risks.

No actual issues classified as vulnerabilities were identified during the review of this component. At the same time, some miscellaneous findings were added to the report: among them, it was pointed out that the administrative Threat Protection API calls utilize HTTP basic authentication, as documented in finding NOR-22-010. While functional, HTTP basic authentication is generally considered less secure compared to more modern authentication mechanisms, particularly if not used in conjunction with secure transport (e.g., HTTPS) and additional hardening measures.

Relatedly, the credentials used for HTTP basic authentication are hashed using the *bcrypt* algorithm (see NOR-22-009). While *bcrypt* remains an industry-accepted password hashing function, its continued appropriateness should be periodically reviewed in the context of evolving cryptographic standards and organizational security requirements.

Furthermore, Cure53 scrutinized the NShield implementation within TP Pro. NShield is the concrete proxy implementation, intercepting and consequently inspecting the web traffic for malicious content. The library, implemented in C++, shows great maturity and comes with a fuzzing harness, emphasizing the security-focused development process employed at NordVPN.

In this area of the engagement, Cure53 utilized a hybrid approach, combining static application security testing and manual code review to identify potential issues. However, within the NShield implementation itself, the testers could not identify any security-related issues.

It can be clarified that specific assessments included - but were not limited to - process introspection and behavioral testing, fuzzing HTTP requests and responses in the hopes of crashing the NShield process, and attempts to bypass the TP Pro protection mechanisms.

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

The DarkWeb Monitor backend API interface, developed in PHP, underwent both static source code analysis and dynamic testing. The assessment placed particular emphasis on identifying common security issues associated with PHP backend applications, with a specific focus on authorization-related weaknesses. Throughout the course of this review, no vulnerabilities were discovered in the analyzed codebase or its exposed interfaces.

The only observation made pertains to a minor defense-in-depth hardening opportunity, as documented in NOR-22-007, calls for the revision of how the implementation uses a timing-unsafe comparison for the email verification code. While this does not present an immediately exploitable risk, replacing the timing-unsafe comparison with a timing-safe alternative would further strengthen the application's security posture.

The token handling mechanisms were scrutinized for logical flaws and bypasses, but none were discovered. All relevant HTTP endpoints within the API were extensively analyzed for common vulnerabilities, including missing input validation, logic errors, and authentication and authorization issues. No significant findings were identified in these areas.

It was confirmed that all queries and mutations had appropriate authorization and role configurations, and no bypasses were identified within the authentication middleware. Common GraphQL misconfigurations were also investigated, and it was found that unauthenticated alias overloading was possible on the GraphQL endpoint, as detailed in issue NOR-22-003. Additionally, the internal MQTT endpoints, implemented in Golang, were audited, revealing potential panic conditions. This was filed as NOR-22-001.

During analysis of the NordVPN web complex, the audit team sought to evaluate the state of existing endpoint functionalities and their associated environments. The focus was placed on whether these could withstand common attack vectors. Particular emphasis was given to analyzing typical vulnerabilities prevalent in modern applications, including various forms of injection attack. Furthermore, the assessment reviewed the potential for sensitive information leakages through the deployed API endpoints.

The testing team investigated the client-side code and frontend component functionality for any prevalent vulnerabilities. The team's core intention here was to discover any instances of XSS, DOM-based XSS, open redirection, template injection, Prototype Pollution, input manipulation, client-side faults, or subpar HTML handling. The frontend exposed by the complex left the team with a positive impression. Despite stringent efforts, no associated vulnerabilities were identified in this area.

To investigate the client-side issues such as DOM-based XSS, the client-side JavaScript was reviewed. The team carefully checked the XSS sinks and sources, including the query parameter handling, *postMessage* communication, as well as framework-specific properties and similar aspects. No issues were spotted.

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

Next, the team examined the server-side of the application. Since the API constitutes a key component for the NordVPN web complex, a host of testing techniques were applied against it in an attempt to leverage commonly found attack vectors. The queries and connected functionality were rigorously assessed. The validation and sanitization of untrusted user input was evaluated. The scope items were checked for unintentionally exposed resources, with the objective of leaking either user information or internal data.

The construct was also vetted to ascertain the persistence of any potentially dangerous calls, such as execution calls that may incur remote code execution, or similar activities on the project's backend areas. Despite stringent endeavors from the testing team, no associated faults were identified for these points.

Cure53 also attempted to locate any weaknesses essential to the implemented access control matrix in API endpoints. No access control matrix shortcomings were detected, demonstrating proper user access restrictions; the application endpoints transparently determine the user's input, and verify whether access is available, before acceptance of the final input.

The secure implementation of access tokens at the individual clients was also evaluated. Here, OAuth mechanisms are not used directly; instead, tokens are stored in a database and API calls are authorized based on the associated user ID. As such, the tokens themselves present minimal direct attack surface under the current usage model. However, potential risk arises if the encrypted OAuth access tokens are ever used directly for authentication. In that case, nonce reuse would break user-agent confidentiality, enabling possible exploitation (see NOR-22-020).

The NordCheckout application leverages Next.js and React to enhance security and functionality. These frameworks contribute to the prevention of common vulnerabilities by implementing features such as automatic output encoding and component isolation. Testing verified the mitigation of XSS attempts, parameter manipulation, and redirect attacks. For payment processing, Stripe and Adyen are integrated in line with industry standards, ensuring a clear separation between application logic and sensitive financial data.

Extensive testing addressed typical web vulnerability categories, including SQL injection, cross-site scripting, authentication bypass, and CSRF attacks. The application demonstrated robust input validation across all tested endpoints, secure session management, and consistent enforcement of authorization checks. API endpoints were designed to handle edge cases without exposing error details or stack traces that could assist attackers.

The information disclosure identified in the cart API (NOR-22-025) highlights a potential issue regarding the frontend application being a proxy to the internal API, which could expose more fields than intended as shown in the ticket. Although user IDs are exposed through arbitrary cart queries, proper enforcement of payment authorization safeguards sensitive operations without having attackers be able to influence much more.

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

The NordAccount identity provider exhibits strong security principles, utilizing Go and Ory Hydra effectively. Testing confirmed the absence of SQL injection vulnerabilities, the implementation of proper cryptographic practices, and consistent input validation across endpoints. Cloudflare integration enhances resilience against common attack vectors and distributed threats.

Testing for standard authentication vulnerabilities, account enumeration, password reset poisoning, and token prediction, showed that the system generates cryptographically secure tokens, ensures session isolation, and validates state-changing operations consistently.

Parameterized statements are used throughout database queries, demonstrating a systematic approach to injection prevention. The production environment uses Cloudflare with allow and blocklists which requires consideration when adding new endpoints in NordAccount API, as was the case with the publicly exposed Prometheus metrics (NOR-22-002) that should be restricted to the internal network.

The rate-limiting mechanism prioritizes user convenience by restricting attempts per challenge and IP via Cloudflare rather than per account. While this approach supports legitimate retries, it also allows attackers to escalate attempts by leveraging multiple challenges and IPv6 IPs (NOR-22-013 and NOR-22-022). Coupled with the inherent limitations of 6-digit OTP codes, the preventive measures should be even stricter in order to prevent abuse.

The NordAccount IdP is generally secure against common attacks against identity providers. However, the complexity introduced by features such as MFA, the trusted browser mechanism, incremental auth, SSO integration, and the Ory Hydra authorization server necessitates a clear, high-level understanding of the entire login flow to effectively mitigate potential threats.

The session pinning vulnerability identified during this assessment combines the Ory Hydra challenge codes with OTP MFA and incorrect state management in the trusted browser mechanism (see NOR-22-008). Hence this is not the result of a single flaw, but rather a weakness across multiple components of the login process, which have to be addressed individually.

The OTP login mechanism - relying on a 6-digit code sent via email - offers minimal entropy, with a theoretical success rate of 1 in 1,000,000 per attempt. While this is accepted in NIST recommendations[14], in practice it provides only a narrow margin for implementation errors. Two key issues were identified.

[14] https://pages.nist.gov/800-63-3/sp800-63b.html#sfotpa

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

First, rate-limiting - critical for any endpoint relying on low-entropy secrets - can be bypassed by altering the case of the email address, enabling attackers to circumvent brute-force protections (see NOR-22-013). While noisy brute-force attempts might trigger detection by operations teams, a distributed and slow attack across multiple IPs may evade monitoring systems.

Second, the entropy of the 6-digit code is further reduced due to modulo bias introduced during code generation (see NOR-22-006). Given these flaws, this mechanism presents probably the most likely single point of failure within the authentication flow. There is an argument to be made for this login flow to only be available to users that have MFA enabled due to its high risk.

The defined scope for this assessment was evidently wide, so best effort was made by Cure53 to cover as many aspects as possible within this *NOR-22* project. The team achieved reasonable coverage and managed to identify a wide range of issues across all work packages, with twenty-seven problems populating the list of findings.

Some aspects of the assignment were assessed as being in a better condition than others. At the same time, the security state of security observed on the scope of this May-June 2025 assessment is perceived as above average. This certainly does not mean that there is no room for improvement in its stance, which is why it is hoped that both vulnerabilities and general weaknesses spotted by Cure53 will be addressed by the UAB 360 IT in due course.

Cure53 would like to thank Kasparas Brazenas, Dainius Slezas, Alexander Evstigneev, Lukas Pukenis, Aleksas Golubevas, Vitalii Sheludchenkov, Asta Krasnickaite-Mickiene, Mrinmoy Ghosal, Andre Lopes, Domas Norkunas and Ignas Vienazindys from the UAB 360 IT team for their excellent project coordination, support, and assistance, both before and during this assignment.