X41 D-Sec

**Security Review
for Mullvad VPN AB**

Final Report and Management Summary

2024-12-10

PUBLIC

X41 D-Sec GmbH
Krefelder Str. 123
D-52070 Aachen
Amtsgericht Aachen: HRB19989

`https://x41-dsec.de/`
`info@x41-dsec.de`

Revision	Date	Change	Author(s)
1	2024-11-30	Final Report and Management Summary	Eric Sesterhenn, JM, Markus Vervier, and Robert Femmer
2	2024-12-10	Improved wording according to feedback	JM

Contents

Dashboard

Target

Customer	Mullvad VPN AB
Name	Mullvad VPN Application
Type	Desktop and Phone Applications
Version	Android: 2024.8-beta1, iOS: 2024.8, Desktop: 2024.6

Engagement

Type	White Box Penetration Test
Consultants	4: Eric Sesterhenn, JM, Markus Vervier, and Robert Femmer
Engagement Effort	30 person-days, 2024-10-23 to 2024-11-28

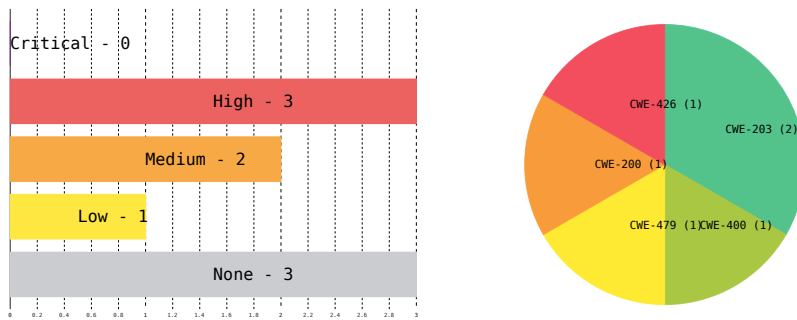Total issues found   6

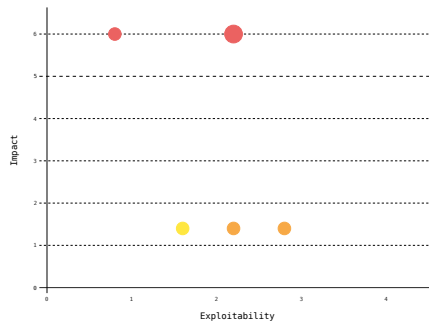**Figure 1:** Issue Overview (l: Severity, r: CWE Distribution)

Figure 2: CVSS Impact and Exploitability Distribution

1   Executive Summary

In October and November 2024, X41 D-Sec GmbH performed a white box penetration test with source code access against the Mullvad VPN Application. The efforts included formulating a light threat model.

A total of six vulnerabilities were discovered during the test by X41. None were rated as having a critical severity, three as high, two as medium, and one as low. Additionally, three issues without a direct security impact were identified.
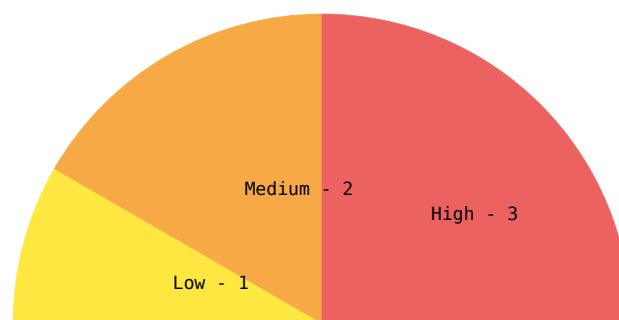
Figure 1.1: Issues and Severity

The Mullvad VPN Application connects a user to the Mullvad VPN service. Internet traffic generated by the user is then routed through the servers operated by Mullvad. The service masquerades the public IP address of the user, which is considered a proxy to their real identity. The application is available for Linux, macOS, Windows, Android and iOS.

This audit investigated how attackers could use the client software and the service for privilege

escalation attacks and to investigate ways in which the real identity of a user could be leaked, or in which way the behavior of a user could be monitored by third parties.

In a white box penetration test, the testers receive all available information about the target, including source code. The test was performed by four experienced security experts between 2024-10-23 and 2024-11-28.

Overall, the Mullvad VPN Application appear to have a high security level and are well positioned to protect from the threat model proposed in this report. The use of safe coding and design patterns in combination with regular audits and penetration tests led to a very hardened environment.

The most serious vulnerabilities are considered to be race conditions and temporal safety violations leading to memory corruption issues in the signal handler code. While exploitation of the signal handler code once triggered seems not unlikely, the fact that an attacker first needs to trigger a signal via another fault reduces the severity of the issues. Other vulnerabilities allow leaking information about the identity of a user by network adjacent attackers and to perform side channel attacks that could in specific circumstances reveal which site a client is currently accessing.

The aspect of side channel attacks is mitigated in most parts, except for protocol level attacks that are not within the control of Mullvad VPN AB because they root from a combination of different technologies such as NAT and modern variants of the HTTP protocol. The introduction of obfuscation technologies and proxy services within the protected VPN is an option for users with higher security and privacy demands.

In conclusion, the client applications exposed a limited number of relevant vulnerabilities. Mullvad VPN AB addressed them swiftly and the fixes were audited to be working properly.

2   Introduction

X41 reviewed the Mullvad Client applications used to connect to the Mullvad VPN[1] service. The product is designed to protect the user's privacy by obscuring the IP[2] address of the user's Internet connections and encrypt the user's traffic on the way between the user and Mullvad's servers. The Internet providers and network adjacent observers are prevented from learning which sites the user visits and which hosts the user connects to.

The confidentiality, integrity and availability of these connections are considered sensitive because users rely on them to protect against the surveillance of employers, ISPs[3], or government entities. They may also be used by citizens of countries with authoritarian governments to access information or services that may be considered illegal. A failure to provide adequate anonymity to the user may lead to their prosecution.

---

[1] Virtual Private Network
[2] Internet Protocol
[3] Internet service providers

2.1 Threat Model

The following light threat model was created during this security audit and should serve as a benchmark for rating the findings. It is intended to be a base for security audits and the rating of vulnerabilities and can be extended over time in future security audits.

2.1.1 Rationale

1. End-users want to make informed choices about whether or not a VPN service is able to cover their perceived threat. A documented threat model can help making these choices.

2. Whether or not a certain behavior of the software is a security issue or not, may come down to nuance. It is advantageous to check these against a documented threat model. This is helpful for developers and future security reviews.

The Mullvad VPN Application is available on Windows, Linux, MacOS, iOS and Android. Every platform differs in interfacing with the operating system, the network device, the firewall and other parts of the system. While potentially hostile code (3rd party applications) is part of the threat model on iOS and Android, it is out of scope for Linux, Windows and MacOS, where the environment is generally considered trusted. Therefore, there must be one threat model per distinct platform. Some threats are shared across all platforms. Here, the threats either apply for all platforms or for mobile platforms.

2.1.2 Threats

This subsection lists the nomenclature used in this threat model and the assumed relevant threats.

2.1.2.1 Nomenclature

- User: The person operating the client.
- Peer: One Internet server, which represents the ultimate destination of the data connection from the client.
- Client: The VPN client software (Mullvad Daemon) run to connect to a relay and establishing a tunnel, which is then used to facilitate the connection from the user to the peer.
- Tunnel: The encrypted connection between the client and the last Mullvad exit relay server.
- Leak: Traffic that was intended to be sent through the tunnel connection, but was accidentally not or can be inferred using other methods.

- Network adjacent entity: An entity on the same local network as the client.
- Relay: VPN peer of the VPN client, first hop of the VPN tunnel.
- Exit relay: The server that connects and routes traffic to the peer. It's IP address will appear as the source for connection to the peer.
- Active entity: an entity that may generate or modify traffic on a network.
- Passive entity: an entity that may observe traffic on a network.

2.1.2.2 Identities

The following are sets of confidential information that attackers may seek to learn.

2.1.2.2.1 User Identity

- Any information that may be used as a proxy to the real user identity.
- The virtual IP address of the tunnel service.
- The account number used to log in to the Mullvad VPN service.

2.1.2.2.2 Tunnel Identity

- The user identity.
- Public user IP: The Internet-facing IP address routing traffic to and from the client.

2.1.2.2.3 Peer Identity

- The Internet-facing IP address (or domain) of the destination of a client connection.

2.1.2.2.4 E2E[4] Identity

- The user identity.
- The peer identity.

2.1.2.3 Attacker Capabilities

The following profiles define information and malicious capabilities (active/passive) that different attackers have.

---

[4] End to End

2.1.2.3.1   Local System Unprivileged Attacker

- Executes code on the same system as different user (unprivileged).
- Can observe limited network information (source port and destination port of connections).
- Can initiate own connections.

2.1.2.3.2   Global Network

- Can passively observe all Internet traffic including connection metadata and payloads.
- Cannot decrypt or deduct details of traffic inside the tunnel.

2.1.2.3.3   Peer Traffic

- Can observe unencrypted or encrypted traffic originating from the VPN client / user's system.
- Cannot observe traffic of the tunnel (also not even encrypted).
- Cannot observe user's upstream traffic between the users public IP address and the relay.

2.1.2.3.4   Network Adjacent

- Can observe traffic between the VPN client and the initial relay.
- Does not know the exit public IP address.
- Can actively interfere or tamper with the traffic between the VPN client and the relay.

2.1.2.3.5   Physical Attacker

- Access to the physical device for some time.

2.1.2.3.6   Attacker on Mullvad Servers

- The attacker has full or partial control of Mullvad Servers. *Such attackers are not in scope, since Mullvad employees are trusted.*

2.1.2.4   Example Threat Scenario

A user seeks to exchange information with a peer over the Internet using some software on their system that initiates the connection and performs communication. This software generates traffic that is to be protected by the Mullvad VPN service.

In a concrete example, a user connects to a website using their browser. Their system may request the destination IP address by sending a DNS[5] request and then establish the connection, exchange data and then close the connection. For this purpose, the system is configured to use the Mullvad VPN client to protect data from leaking and to securely route the connection through the VPN tunnel. The protection also includes metadata.

A network diagram illustrating the scenario is given in figure 2.1.
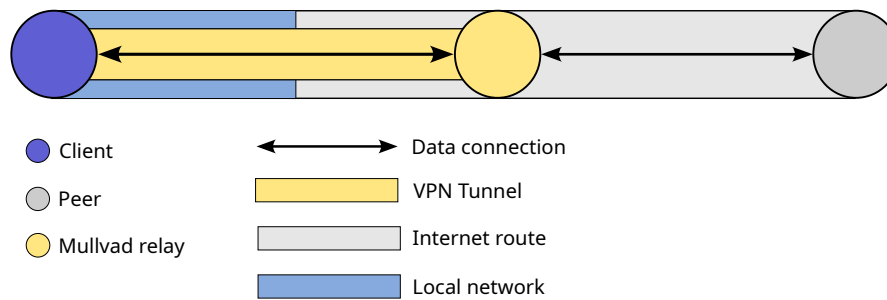

Figure 2.1: Network Diagram Illustrating the Scenario

2.1.2.5   General Assumptions

1. Mullvad ensures that the tunnel is encrypted and integrity protected.

2. Mullvad ensures that an active or passive actor operating a network hop forwarding tunnel traffic is not able to learn the E2E identity.

   - DAITA[6,7] protects against traffic pattern matching attacks.

3. Mullvad ensures that the peer does not learn the tunnel identity of the client.

4. Mullvad ensures that an active or passive network adjacent entity does not learn the user identity of the client.

5. Mullvad ensures that no traffic leaks if the client software crashes.

2.1.2.6   Limitations

1. There is no protection against side channels that exist in the protected payloads themselves (e.g. a browser inserting a unique identifier).

---

[5] Domain Name System
[6] Defense against AI-guided Traffic Analysis
[7] `https://mullvad.net/en/vpn/daita`

2. A compromised client cannot be protected.

3. A compromised VPN tunnel (encryption compromised) cannot be protected.

4. A network observer can learn about the fact that a VPN is used.
   Mullvad's obfuscation techniques – when enabled – attempt to be transparent to technical restrictions such as firewalls, not to observers aimed at learning whether a VPN service is used.

5. An actor with enough resources to observe and analyze traffic on a global scale may be able to correlate tunneled traffic with traffic between the exit relay and the peer and therefore learn the E2E identity. This is a limitation of network communication in general, and not of the Mullvad VPN Service.

2.1.2.7   Mobile Platforms

1. Mullvad ensures that the third party apps can not accidentally or intentionally learn the E2E identity.

2. When the phone is locked, Mullvad ensures a physical attacker can not learn the E2E identity.

2.2   Methodology

The review was conducted as a white box penetration test.

A manual approach for penetration tests and for code reviews is used by X41. This process is supported by tools such as static code analyzers and industry standard security tools.

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[8] standards and the *Study - A Penetration Testing Model*[9] of the German Federal Office for Information Security.

The workflow of source code reviews is shown in figure 2.2. In an initial workshop regarding the design and architecture of the application a light threat model is created. This is used to explore the source code for interesting attack surface and code paths. These are then audited manually and with the help of tools such as static analyzers and fuzzers. The identified issues are documented and can be used in a GAP analysis to highlight changes to previous audits.

---

[8] `https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards`
[9] `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1`

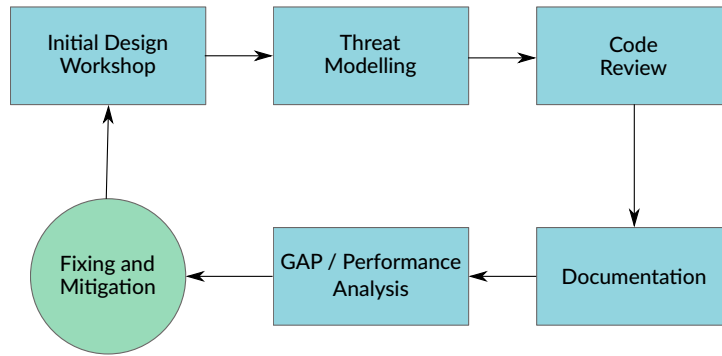**Figure 2.2:** Code Review Methodology

2.3   Findings Overview

DESCRIPTION	SEVERITY	ID	REF
Signal Handler's Alternate Stack Too Small	HIGH	MLLVD-CR-24-01	4.1.1
Signal Handler Uses Non-Async-Safe Functions	HIGH	MLLVD-CR-24-02	4.1.2
Virtual IP Address of Tunnel Device Leaks to Network Adjacent Participant	MEDIUM	MLLVD-CR-24-03	4.1.3
Deanonymization Through NAT	MEDIUM	MLLVD-CR-24-04	4.1.4
Deanonymization Through MTU	LOW	MLLVD-CR-24-05	4.1.5
Sideloading Into Setup Process	HIGH	MLLVD-CR-24-06	4.1.6
Publisher Not Set	NONE	MLLVD-CR-24-100	4.2.1
Binary Hardening	NONE	MLLVD-CR-24-101	4.2.2
IOCTL Unrestricted Kernel Pool Memory Allocation	NONE	MLLVD-CR-24-102	4.2.3

Table 2.1: Security-Relevant Findings

2.4   Scope

The scope of this audit was defined to be the Mullvad client application including the daemon for Linux, Windows, MacOS and Android as well as their respective GUI[10] and CLI[11] components. Additionally the iOS app and packet tunnel component were part of the scope. The following repositories with their respective releases and commit hashes were reviewed:

Component	Version	Commit
Android	2024.8-beta1	f482e3bfb2c6930f4610e84b83e036058f4cb8cf
iOS	2024.8	1991d182f614215ba80a41a733fe29f3dab2a7cb
Desktop	2024.6	ef6c862071b26023802b00d6e1dc6ca53d1ab3e6

All releases refer to the repository hosted at `https://github.com/mullvad/mullvadvpn-app`. A Slack channel was used for communication between the developers and the testers. Additionally, testers were on-site to facilitate direct interaction with the development teams. Valid accounts for the Mullvad VPN were provided. Further, a list of major changes since the last external code audit was provided to guide the efforts of this project.

---

[10] Graphical User Interface
[11] Command line interpreter

2.5   Coverage

In a first step, a light threat model was created to establish a benchmark that possible issues can be measured against. Very subtle issues that may seem inconsequential at first may have undesired consequences for users who are impacted by them. A threat model aids in determining how severe an issue is. Furthermore, it helps a user to make an informed decision whether the VPN service can protect them from their perceived threat.

A security assessment seeks to find the most important issues or as many issues as possible, though it is practically impossible to rule out the existence of additional weaknesses being found in the future. The time allocated to X41 for this assessment was sufficient to yield a reasonable coverage of the given scope.

The Mullvad daemon was covered on Linux, Windows, MacOS and Android with focus on components that were deemed at risk of leaking traffic outside of the tunnel, were exposed to user inputs or other external data that may crash the daemon or lead to local privilege escalation. This included for instance the split tunneling features on all supported platforms, which were audited to ensure that no processes whose egress traffic is supposed to be tunneled through the VPN will end up outside of the tunnel. The firewall rules were audited for completeness, ensuring that all traffic will be routed through the VPN tunnel. This included the implementation of the interface with the respective operating system, where applicable. In addition, the effectiveness of firewalling measures were tested by observing the network traffic after attempting attacks such as using publicly routable IP addresses on the local network interface and/or as additional static routes, DNS resolvers within the local network, or TLDs[12] configured as search domains. The implementation of DAITA was audited to ensure that there is no way to disable DAITA. The efficacy of DAITA itself was not tested and was considered out of scope for this audit.

The Mullvad daemon was audited for common security vulnerabilities with regards to file and signal handling, in particular race conditions in that regard. The daemon and applications were also investigated regarding memory safety issues that could result from unsafe code, usage of FFIs[13] or other limitations of the security guarantees of the used frameworks and programming languages. Used libraries were scanned for known vulnerabilities. Additionally, binaries were checked for common compile time hardening flags.

Denial of service attacks were investigated both from local and remote attack positions. While all system users on desktop systems are declared to be trustworthy to manage the VPN connection as per the threat model, local system privilege escalation was investigated as relevant attack vector. Attacks from untrustworthy applications or websites were also investigated on both mobile and desktop systems.

---

[12] Top-level domains
[13] Foreign Function Interfaces

2.6   Recommended Further Tests

The high security level of the Mullvad VPN Application is a direct result of regular security reviews every two years, and X41 recommends to continue this approach. We recommend to consider additional security reviews of the VPN infrastructure, and for major new features that may impact the security model and may be introduced in the future and fall too far in between two regularly scheduled reviews.

X41 recommends to mitigate the issues described in this report. Afterwards, CVE[14] IDs[15] should be requested and customers be informed (e.g. via a changelog or a special note for issues with higher severity) to ensure that they can make an informed decision about upgrading or other possible mitigations.

---

[14] Common Vulnerabilities and Exposures
[15] Identifiers

3   Rating Methodology

Security vulnerabilities are given a purely technical rating by the testers when they are discovered during a test. Business factors and financial risks for Mullvad VPN AB are beyond the scope of a penetration test, which focuses entirely on technical factors. However, technical results from a penetration test may be an integral part of a general risk assessment. A penetration test is based on a limited time frame and only covers vulnerabilities and security issues which have been found in the given time, there is no claim for full coverage.

The CVSS[1] is used to score all findings relevant to security. The resulting CVSS score is mapped to qualitative ratings as shown below.

3.1   CVSS

All findings relevant to security are rated by the testers using the CVSS industry standard version 3.1, revision 1.

Vulnerabilities scored with CVSS get a numeric value based on several metrics ranging from 0.0 (least worst) to 10.0 (worst).

The score captures different factors that express the impact and the ease of exploitation of a vulnerability among other factors. For a detailed description of how the scores are calculated, please see the CVSS version 3.1 specification.[2]

The metrics used to calculate the final score are grouped into three different categories.

---

[1] Common Vulnerability Scoring System
[2] `https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf`

The *Base Metric Group* represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. It captures the following metrics:

- Attack Vector (AV)
- Attack Complexity (AC)
- Privileges Required (PR)
- User Interaction (UI)
- Scope (S)
- Confidentiality Impact (C)
- Integrity Impact (I)
- Availability Impact (A)

The *Temporal Metric Group* represents the characteristics of a vulnerability that change over time but not among user environments. The following metrics are covered by it:

- Exploitability (E)
- Remediation Level (RL)
- Report Confidence (RC)

The *Environmental Metric Group* represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. It includes the following metrics:

- Attack Vector (MAV)
- Attack Complexity (MAC)
- Privileges Required (MPR)
- User Interaction (MUI)
- Confidentiality Requirement (MCR)
- Integrity Requirement (MIR)
- Availability Requirement (MAR)
- Scope (MS)
- Confidentiality Impact (MC)
- Integrity Impact (MI)
- Availability Impact (MA)

A CVSS vector defines a specific set of metrics and their values, and it can be used to reproduce and assess a given score. It is rendered as a string that exactly reproduces a score.

For example, the vector `CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N` defines a base score metric with the following parameters:

- Attack Vector: Network
- Attack Complexity: High
- Privileges Required: Low
- User Interaction: Required
- Scope: Changed
- Confidentiality Impact: High
- Integrity Impact: Low
- Availability Impact: None

In this example, a network-based attacker performs a complex attack after gaining access to some privileges, by tricking a user into performing some actions. This allows the attacker to read confidential data and change some parts of that data.

The detailed scores are the following:

Metric	Score
CVSS Base Score	6.5
Impact Sub-Score	4.7
Exploitability Sub-Score	1.3
CVSS Temporal Score	Not Available
CVSS Environmental Score	Not Available
Modified Impact Sub-Score	Not Available
Overall CVSS Score	6.5

CVSS vectors can be automatically parsed to recreate the score, for example, with the CVSS calculator provided by FIRST, the organization behind CVSS: `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N`.

3.2   Severity Mapping

To help in understanding the results of a test, numeric CVSS scores are mapped to qualitative ratings as follows:

Severity Rating	CVSS Score
NONE	0.0
LOW	0.1–3.9
MEDIUM	4.0–6.9
HIGH	7.0–8.9
CRITICAL	9.0–10.0

3.3   Common Weakness Enumeration

The CWE[3] is a set of software weaknesses that allows vulnerabilities and weaknesses in software to be categorized. If applicable, X41 gives a CWE ID for each vulnerability that is discovered during a test.

CWE is a very powerful method for categorizing a vulnerability. It gives general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*.[4] More information can be found on the CWE site at `https://cwe.mitre.org/`.

---

[3] Common Weakness Enumeration
[4] `https://www.mitre.org`

4   Results

This chapter describes the results of this test. The security-relevant findings are documented in Section 4.1. Additionally, findings without a direct security impact are documented in Section 4.2.

4.1   Findings

The following subsections describe findings with a direct security impact that were discovered during the test.

4.1.1   MLLVD-CR-24-01: Signal Handler's Alternate Stack Too Small

Severity:	HIGH
CVSS v3.1 Total Score:	8.3
CVSS v3.1 Vector:	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H/E:U/RC:C
CWE:	400 – Uncontrolled Resource Consumption ('Resource Exhaustion')
Affected Component:	mullvad-daemon/src/exception_logging/unix.rs:enable()

4.1.1.1   Description

During the review X41 discovered that heap corruption could occur when the alternate stack that is defined for exception handlers is exhausted.

As seen in listing 4.1, an exception handler is installed and given a memory area as its alternate stack using the *POSIX* function ***sigaltstack()***[1]:

```
1   /// Installs a signal handler.
2   pub fn enable() {
3     INIT_ONCE.call_once(|| {
4       // Setup alt stack for signal handlers to be executed in.
5       // If the daemon ever needs to be compiled for architectures where memory can't be writeable
6       // and executable, the following block of code has to be disabled. This will also mean that
7       // stack overflows may be silent and undetectable in logs.
8       let sig_handler_flags = {
9         // The kernel will use the first properly aligned address, so alignment is not an issue.
10        let alt_stack = vec![0u8; libc::SIGSTKSZ]; // MARK heap memory backing the alt stack
11        let stack_t = libc::stack_t {
12          ss_sp: alt_stack.as_ptr() as *mut c_void,
13          ss_flags: 0,
14          ss_size: alt_stack.len(),
15        };
16        let ret = unsafe { libc::sigaltstack(&stack_t, std::ptr::null_mut()) }; ⌋
↪   // MARK alt stack in effect
17        if ret != 0 {
18          log::error!(
19            "Failed to set alternative stack: {}",
20            std::io::Error::last_os_error()
21          );
22          SaFlags::empty()
23        } else {
24          std::mem::forget(alt_stack);
25          SaFlags::SA_ONSTACK
26        }
27      };
```

----

[1] https://pubs.opengroup.org/onlinepubs/009696799/functions/sigaltstack.html

```
28
29      let signal_action = SigAction::new(
30        SigHandler::SigAction(fault_handler),
31        sig_handler_flags,
32        SigSet::empty(),
33      );
34
35      for signal in &FAULT_SIGNALS {
36        if let Err(err) = unsafe { sigaction(*signal, &signal_action) } {
37          log::error!("Failed to install signal handler for {}: {}", signal, err);
38        }
39      }
40    });
41  }
```

Listing 4.1: Signal Handler Installed with Insufficient Alt-Stack

On a standard Debian-12 Linux system running on *x86_64*, the $SIGSTKSZ$ is defined as only two pages (8KB). The exception handler is installed for the following signals:

- SIGBUS
- SIGFPE
- SIGILL
- SIGSEGV
- SIGSYS

When any such signal appears due to a runtime condition in the daemon process, the handler is run and is subject to a stack exhaustion due to the insufficient size of its alternate stack.

Since the alternate stack is located on the heap and is not separated by a guard page from other pages or metadata, such pages and data are corrupted due to out of bounds writes as seen in listing 4.2.

```
1   Thread 1 "mullvad-daemon" received signal SIGFPE, Arithmetic exception.
2   syscall () at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
3   38          ../sysdeps/unix/sysv/linux/x86_64/syscall.S: No such file or directory.
4   (gdb) c
5   Continuing.
6   [2024-11-11 16:08:16.478][mullvad_daemon::exception_logging::unix][ERROR] Caught signal SIGFPE
7   [Thread 0x7fffa9bf96c0 (LWP 19363) exited]
8
9   Thread 1 "mullvad-daemon" received signal SIGSEGV, Segmentation fault.
10  core::intrinsics::copy_nonoverlapping<u8> (src=0x55440218730, dst=0x555558a1c990, count=16)
11      at /home/user/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib/src/rust/⌋
    ↪   library/core/src/intrinsics.rs:3298
```

```
12   3298          unsafe { copy_nonoverlapping(src, dst, count) }
13   (gdb)
```

Listing 4.2: Memory Corruption

Note that even though the segmentation violation occurs in an unsafe code block, this code block is in the Rust standard library.

While exploitation for code execution is expected to be non-trivial, the fact that the alt stack collides with the heap of concurrently running processes makes exploitation a possibility if an attacker is able to trigger a signal in the right context.

4.1.1.2   Solution Advice

X41 recommends to use memory backed by *mmap()* and isolated from other memory areas. Additionally it is advised to define the alternate stack size with an appropriate safety margin such as 8 times `libc::SIGSTKSZ` and introduce a guard page below the stack. This size is estimated to be sufficient for the code observed to be running.

4.1.2   MLLVD-CR-24-02: Signal Handler Uses Non-Async-Safe Functions

Severity:	HIGH
CVSS v3.1 Total Score:	8.3
CVSS v3.1 Vector:	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H/E:U/RC:C
CWE:	479 – Signal Handler Use of a Non-reentrant Function
Affected Component:	mullvad-daemon/src/exception_logging/unix.rs:fault_handler()

4.1.2.1   Description

The installed signal handler calls library functions that are not *async-signal-safe*[2]. In an example scenario, when an allocation function enters a critical section and is pre-empted by a signal delivered by the kernel and the signal handler enters the same function a deadlock could occur.

In particular the functions related to the logging of a backtrace invoke the ***log*** crate, which uses heap allocations internally. The calls can be observed in listing 4.3.

```
1  log::error!("Caught signal {}", signal);
2  log::error!("Backtrace:");
3  for line in format!("{}", Backtrace::force_capture()).lines() {
4      log::error!("{line}");
5  }
6  std::process::exit(2);
```

Listing 4.3: Non-Signal-Safe Function Calls

Since the lock protecting the function is re-entrant, it is also permitted to execute the function again from within itself in the same thread. However, the function may have been preempted while the managed allocation resources are in an unsafe state. If another allocation function is called by signal handler, it might proceed to operate on invalid data. This may lead to exploitable conditions, if the state can be controlled enough and the signal handler can be externally triggered.

4.1.2.2   Solution Advice

X41 recommends to only use signal-safe functions in the signal handler and to limit the amount of complex code running within the handler to a minimum.

---

[2] https://man7.org/linux/man-pages/man7/signal-safety.7.html

4.1.3   MLLVD-CR-24-03: Virtual IP Address of Tunnel Device Leaks to Network Adjacent Participant

Severity:	MEDIUM
CVSS v3.1 Total Score:	4.3
CVSS v3.1 Vector:	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
CWE:	200 – Exposure of Sensitive Information to an Unauthorized Actor
Affected Component:	Linux and Android Network Stack

4.1.3.1   Description

Due to the way the Linux (and consequently the Android) network stacks work, an interface may answer to ARP[3] requests for an IP address configured to one interface on a different interface. This lets an attacker learn the virtual IP address of the Wireguard tunnel device by enumerating IP addresses of the Mullvad tunnel subnet and sending ARP request to each of them. If the IP address is configured on the device, it will reply with a corresponding ARP reply.

In the context of this audit, the virtual IP assigned to a specific user was static for a certain amount of time. This could give a network adjacent attacker the ability to learn that the same user was active at different times.

4.1.3.2   Solution Advice

X41 recommends to mitigate the issue by setting the kernel parameter arp_ignore[4] to 1 on Linux. On Android, the application does not have sufficient permissions to do so and hence, Mullvad has to rely on mobile phone vendors to implement this fix. It is also recommended to randomize the virtual IP address for each user on each connection if possible.

---

[3] Address Resolution Protocol
[4] https://www.kernel.org/doc/html/latest/networking/ip-sysctl.html

4.1.4   MLLVD-CR-24-04: Deanonymization Through NAT

Severity:	MEDIUM
CVSS v3.1 Total Score:	4.0
CVSS v3.1 Vector:	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N
CWE:	203 – Observable Discrepancy
Affected Component:	Network

4.1.4.1   Description

An attacker able to monitor encrypted tunnel packets and also to spoof UDP[5] packets with a certain peer's IP address can learn whether the Mullvad client has sent UDP packets to the peer. A common scenario would be a Mullvad user visiting a website using WebRTC[6] or HTTP/3[7] over QUIC[8]. Since UDP is used, the attacker could learn that there has been a connection to it, for example *google.com*.

When sending a UDP packet through the Mullvad VPN tunnel, the Mullvad exit relay rewrites the packet's source IP address and source port, and records it with the client's IP address, peer's IP address, source port, and destination port in its NAT[9] table.

As long as the entry is kept alive in the NAT table, an attacker could then spoof UDP packets of a particular size with the peer's IP address and port and send them to all known exit relays and all high ports. When an entry in the NAT table is matched, the packet will be rewritten and sent to the Mullvad client through the tunnel. The attacker may then observe the packet with a particular size. Alternatively, the attacker could flood the relay's ports one by one, and observe a spike in traffic volume when targeting the right port.

The attack works against UDP because it is stateless and does not have sequence numbers like TCP[10]. NAT gateways that do not perform TCP connection tracking would be similarly effected. X41 found Mullvad's relays to be unaffected when using TCP.

Security features built into higher level UDP protocols such as HTTP/3 or DNS would reject the packets sent by the attacker, but only after reaching the client through the encrypted tunnel.

While IP source address spoofing is prevented on most consumer providers, some hosting providers do not effectively prevent this. The attack may be even more feasible when the attacker uses

---

[5] User Datagram Protocol
[6] Web Real Time Communication
[7] HyperText Transfer Protocol version 3
[8] Quick UDP Internet Connections
[9] Network Address Translation
[10] Transmission Control Protocol

the same hosting provider as the targeted peer.

The attack was previously described as "Server-side Attacks" in the USENIX paper "Blind In/On-Path Attacks and Applications to VPNs"[11].

In essence an attacker that would want to know if a certain encrypted VPN connection is currently communicating with for example `https://google.com` would need only the following:

- Ability to observe the local client's network traffic
- A list of potential Mullvad exit IPs
- The ability to spoof IP packets with a source IP of google.com and send them to the Mullvad exit IPs

The combination of the stateless nature of UDP and the more widespread use of UDP based protocols that implement state on a higher layer makes the exploitation of this issue easier than in classical HTTP[12] version 1.1 or 2.0 scenarios.

4.1.4.2   Solution Advice

The issue can be partly mitigated by avoiding the use of UDP between the software and the exit relay and by enabling stateful connection tracking for TCP connections. In that case web browsers could be configured to use a (TCP based) HTTP proxy where the proxy initiates the HTTP requests for the client. Alternatively, a SOCKS[13] proxy that does *not* support the UDP could be used. Both options would result in the web browsers not using UDP, mitigating the issue for the web browser. A more aggressive approach would be an option to block UDP at the client and/or relay. This would prevent the issue while breaking applications that rely on the use of UDP and is therefore not a realistic option. Using a full HTTP proxy could also mitigate this issue partly.

When enabled the use of DAITA mitigates the possibility to observe particular packet sizes or single packet signals reliably. However, it does not protect against observing traffic spikes, given that the traffic volume is high enough.

---

[11] `https://www.usenix.org/system/files/sec21fall-tolley.pdf`
[12] HyperText Transfer Protocol
[13] `https://datatracker.ietf.org/doc/html/rfc1928`

4.1.5   MLLVD-CR-24-05: Deanonymization Through MTU

Severity:	<mark>LOW</mark>
CVSS v3.1 Total Score:	3.4
CVSS v3.1 Vector:	CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N
CWE:	203 – Observable Discrepancy
Affected Component:	Network

4.1.5.1   Description

Lowering the MTU[14] by a certain amount on the upstream network device which the Mullvad exit relay connects to (such as a web server), will lead to a stream of network packets with their size lowered by approximately that amount on the tunnel. An attacker able to control the MTU and simultaneously able to observe the encrypted tunnel can correlate the connections and thereby deanonymize the client.

DAITA mitigates this by enforcing uniform packet sizes.

It should be noted that similar attacks can be performed by delaying packets, dropping packets, or causing traffic bursts en route and correlating the timing of packets. DAITA attempts to mitigate this to some degree, but depending on the magnitude of manipulation may not be able to reliably suppress every signal introduced into network traffic.

4.1.5.2   Solution Advice

X41 recommends to enforce the use of DAITA to mitigate the MTU issue, being a reasonable attempt to mitigate related issues.

---

[14] Maximum Transmission Unit

4.1.6   MLLVD-CR-24-06: Sideloading Into Setup Process

Severity:	**HIGH**
CVSS v3.1 Total Score:	7.6
CVSS v3.1 Vector:	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:H
CWE:	426 – Untrusted Search Path
Affected Component:	Windows Installer

4.1.6.1   Description

During the installation process, the setup process executes the program `taskkill` with no abso-
lute path to the program. A binary called *taskkill.exe* in the same directory as the setup executable
(e.g. the user's `Downloads` directory) is preferred over one elsewhere in the search path. If a ma-
licious actor is able to place a binary of that name there, it will be executed by the setup process.
This is similar to a DLL[15] sideloading attack[16].

In figure 4.1 the Windows `calc.exe` was renamed to `taskkill.exe` and placed in the same folder.
It was executed during the setup procedure.
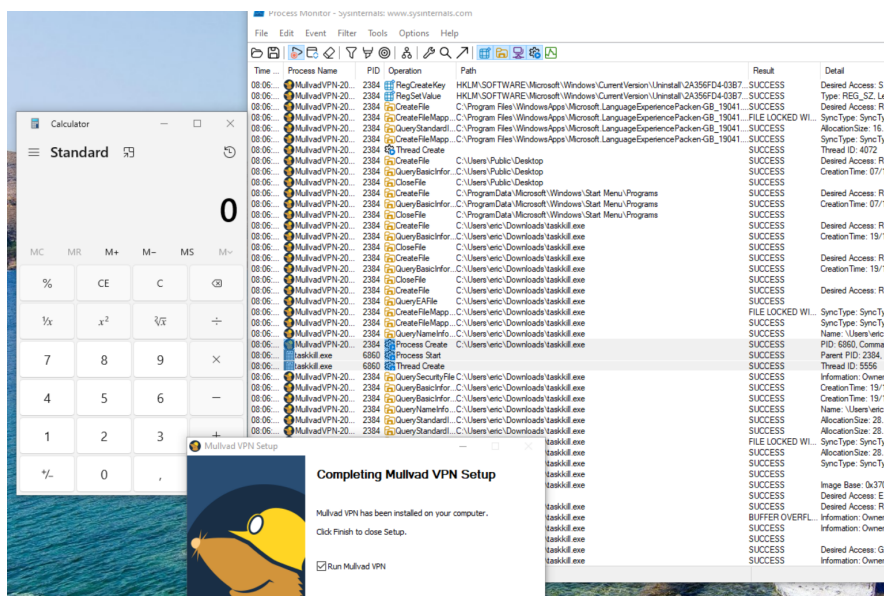
Figure 4.1: taskkill.exe Started by Installer

---

[15] Dynamic Link Library
[16] https://attack.mitre.org/techniques/T1574/002/

4.1.6.2 Solution Advice

X41 recommends to only execute binaries in trusted code path. In this case, the binary in
`C:\Windows\System32` should be started instead.

4.2   Informational Notes

The following observations do not have a direct security impact, but are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

4.2.1   MLLVD-CR-24-100: Publisher Not Set

Affected Component:	Uninstaller

4.2.1.1   Description

During the uninstallation process, a popup asks the user to confirm the removal. That popup does not have the publisher's information set (as seen in image 4.2).
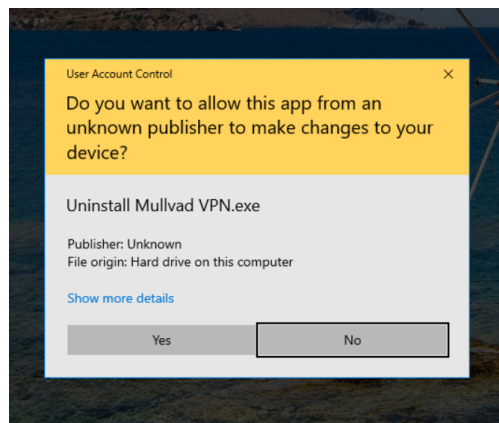
Figure 4.2: Publisher Popup

This might spark doubt in the user about whether it was a legitimate version of the Mullvad VPN client that was installed. Setting the publisher's information is best practice.

4.2.1.2   Solution Advice

X41 recommends to set the publisher's information for the uninstall components.

4.2.2   MLLVD-CR-24-101: Binary Hardening

Affected Component:	Windows Binaries

4.2.2.1   Description

Some of the binaries shipped with the Windows version of Mullvad VPN do not seem to be built from Go or Rust sources. Not all of these have all the binary hardening options enabled that are available. A truncated output of binary-security-check[17] can be found in listing 4.4

```
1  resources/elevate.exe:      +DATA-EXEC-PREVENT !CONTROL-FLOW-GUARD ~ASLR-LT-2GB +SAFE-SEH
2  resources/openvpn.exe:      +DATA-EXEC-PREVENT !CONTROL-FLOW-GUARD !ASLR +SAFE-SEH
3  resources/winfw.dll:        +DATA-EXEC-PREVENT !CONTROL-FLOW-GUARD +ASLR +SAFE-SEH
4  Uninstall Mullvad VPN.exe: +DATA-EXEC-PREVENT !CONTROL-FLOW-GUARD !ASLR !SAFE-SEH
```

Listing 4.4: Binary Hardening Flags (Shortened Output)

Since these flags are not security bugs themselves, but hardening measures, this is reported as an informational finding.

4.2.2.2   Solution Advice

X41 recommends to enable CFI[18], safe SEH[19] and ASLR[20] for all shipped binaries.

---

[17] `https://github.com/koutheir/binary-security-check`
[18] Control Flow Integrity
[19] Structured Exception Handler
[20] Address Space Layout Randomization

4.2.3   MLLVD-CR-24-102: IOCTL Unrestricted Kernel Pool Memory Alloca-
tion

Affected Component:	win-split-tunnel/src/ioctl.cpp

4.2.3.1   Description

It was found that the Windows filtering driver does not set appropriate limits on the number of
configuration entries. A malicious low-privileged user could add an unrestricted number of split
tunneling process entries via the daemon. Subsequently the daemon would send a configuration
request that is parsed by the kernel filtering driver, which allocates memory from a pool as seen
in the following listing:

```
1   NTSTATUS
2   AddEntryInner
3   (
4     CONTEXT *Context,
5     LOWER_UNICODE_STRING *ImageName
6   )
7   {
8     //
9     // Make a single allocation for the struct and string buffer.
10    //
11
12    auto offsetStringBuffer = util::RoundToMultiple(sizeof(REGISTERED_IMAGE_ENTRY), 8);
13
14    auto allocationSize = offsetStringBuffer + ImageName->Length;
15
16    const auto poolType = (Context->Pageable == ST_PAGEABLE::YES) ? PagedPool : NonPagedPool;
17
18      auto record = (REGISTERED_IMAGE_ENTRY*)
19      ExAllocatePoolUninitialized(poolType, allocationSize, ST_POOL_TAG);
```

Listing 4.5: Memory Allocation per Entry

The code shown above is called within a loop from function *SetConfigurationPrepare*:

```
1   //
2   // SetConfigurationPrepare()
3   //
4   // Validate and repackage configuration data into new registered image instance.
5   //
```

```
 6    // This runs at PASSIVE, in order to be able to downcase the strings.
 7    //
 8    NTSTATUS
 9    SetConfigurationPrepare
10    (
11        WDFREQUEST Request,
12        registeredimage::CONTEXT **Imageset
13    )
14    {
15        *Imageset = NULL;
16
17        PVOID buffer;
18        size_t bufferLength;
19    ...
20    for (auto i = 0; i < header->NumEntries; ++i, ++entry)
21    {
22        UNICODE_STRING s;
23
24        s.Length = entry->ImageNameLength;
25        s.MaximumLength = entry->ImageNameLength;
26        s.Buffer = (WCHAR*)(stringBuffer + entry->ImageNameOffset);
27
28        status = registeredimage::AddEntry(imageset, &s);
29
30        if (!NT_SUCCESS(status))
31        {
32            DbgPrint("Could not insert new entry into registered image instance: 0x%X\n", status);
33
34            registeredimage::TearDown(&imageset);
35
36            return status;
37        }
38    }
```

Listing 4.6: Unrestricted Entry Processing

Since the number of configuration entries within the request buffer is not limited and the type of *header->NumEntries* is of type *size_t*, an attacker could cause an exhaustion of kernel pool memory by adding a large number of configuration entries. The memory needed in userspace to trigger is similar to the memory usage caused in the kernel pool. Still an attacker might be able to cause system stability issues by increasing the kernel memory usage.

4.2.3.2   Solution Advice

X41 recommends to restrict the number of configuration entries to a sane limit that cannot cause large memory usage in kernel pool memory.

5   About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security and penetration testing services. Having extensive industry experience and expertise in the area of information security, a strong core security team of world-class security experts enables X41 D-Sec GmbH to perform premium security services.

X41 has the following references that show their experience in the field:

- Source code audit of ISC BIND9 DNS server[1]
- Source code audit of the Git source code version control system[2]
- Review of the Mozilla Firefox updater[3]
- X41 Browser Security White Paper[4]
- Review of Cryptographic Protocols (Wire)[5]
- Identification of flaws in Fax Machines[6,7]
- Smartcard Stack Fuzzing[8]

The testers at X41 have extensive experience with penetration testing and red teaming exercises in complex environments. This includes enterprise environments with thousands of users and vendor infrastructures such as the Mozilla Firefox Updater (Balrog).

Fields of expertise in the area of application security encompass security-centered code reviews, binary reverse-engineering and vulnerability-discovery. Custom research and IT security consulting, as well as support services, are the core competencies of X41. The team has a strong technical background and performs security reviews of complex and high-profile applications such as Google Chrome and Microsoft Edge web browsers.

X41 D-Sec GmbH can be reached via `https://x41-dsec.de` or `mailto:info@x41-dsec.de`.

---

[1] `https://x41-dsec.de/news/security/research/source-code-audit/2024/02/13/bind9-security-audit/`
[2] `https://x41-dsec.de/security/research/news/2023/01/17/git-security-audit-ostif/`
[3] `https://blog.mozilla.org/security/2018/10/09/trusting-the-delivery-of-firefox-updates/`
[4] `https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf`
[5] `https://www.x41-dsec.de/reports/Kudelski-X41-Wire-Report-phase1-20170208.pdf`
[6] `https://www.x41-dsec.de/lab/blog/fax/`
[7] `https://2018.zeronights.ru/en/reports/zero-fax-given/`
[8] `https://www.x41-dsec.de/lab/blog/smartcards/`

Acronyms