Security Review
for Mullvad VPN AB

Final Report and Management Summary

2026-01-20

PUBLIC

X41 D-Sec GmbH
Soerser Weg 20
D-52070 Aachen
Amtsgericht Aachen: HRB19989

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

Revision	*Date*	*Change*	*Author(s)*
1	2025-10-30	Final Draft Report	JM and Robert Femmer
2	2025-11-10	Final Redacted Draft Report	JM and Robert Femmer
3	2025-11-21	Final Redacted Report	JM and Robert Femmer

Contents

Dashboard

Target	
Customer	Mullvad VPN AB
Name	Mullvad API
Type	API and Backend Services
Version	git tag *x41-api-audit-2025*

Engagement	
Type	White-Box Penetration Test
Consultants	2: JM and Robert Femmer
Engagement Effort	24 person-days, 2025-10-13 to 2025-10-31

Total issues found	5

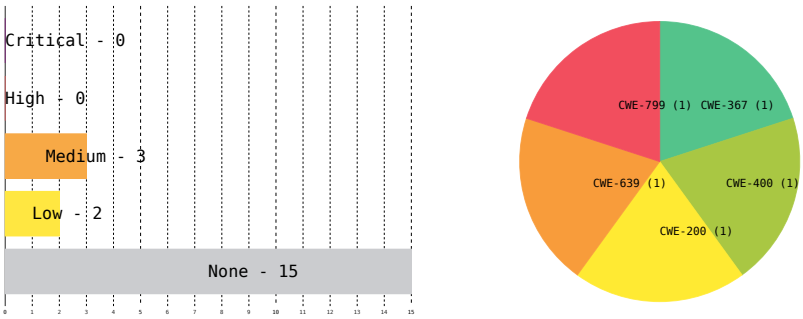

Figure 1: Issue Overview (l: Severity, r: CWE Distribution)

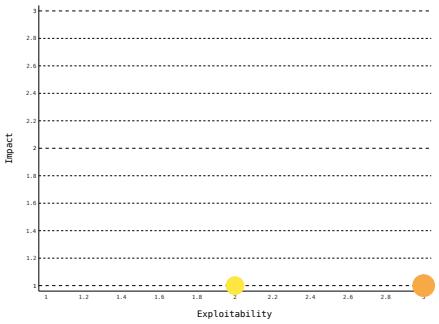

Figure 2: CVSS Impact and Exploitability Distribution

1 Executive Summary

In October 2025, X41 D-Sec GmbH performed a source code audit of the Mullvad services to identify vulnerabilities and weaknesses in the public API and connected services.

A total of five vulnerabilities were discovered during the test by X41. None were rated as having a critical or high severity, three as medium, and two as low. Additionally, 15 issues without a direct security impact were identified. The severity classification of the vulnerabilities is restricted to their technical properties. The impact of the vulnerabilities on user privacy, business and financial risk was rated as low.

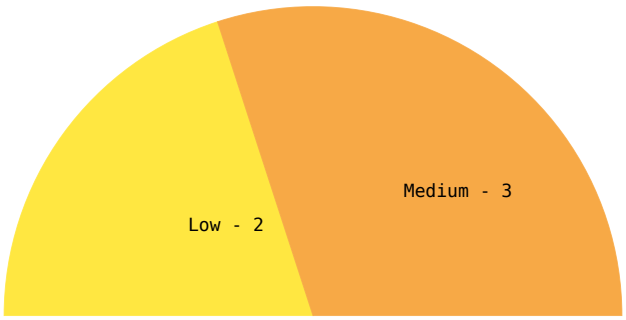

Figure 1.1: Issues and Severity

The Mullvad API is a set of services facilitating authentication, device setup and payment for the network tunneling service provided by Mullvad VPN AB. Vulnerabilities in the application would allow an attacker to access sensitive customer data, like Internet connection metadata and payment information.

In a source code audit, all information about the system is made available. The test was performed by two experienced security experts between 2025-10-13 and 2025-10-31.

The most severe issue discovered allows an attacker to redeem a voucher for two or more different accounts by racing the redemption request. This allows them to extend a paid service for multiple accounts with only one payment.

Four issues were redacted from the public report because they may be used to cause denial of service conditions. These issues do not affect the confidentiality or integrity of customer data.

The code base in general suffers from problems that arise from concurrent database accesses. While none were found to be severe security problems, it is clear that time of check/time of use issues and race conditions were not initially considered while designing the application.

X41 recommends refactoring the business logic into simple functions, providing database transactions that can be reasoned about easily in a multi-tasking application. Locking operations should be used where necessary.

The system is designed in a way that an outage of the API has minimal impact on existing customers and VPN connections. The reviewed code and configuration shows that the service attempts to keep as little data about customers and payments as possible, thereby minimizing the risk of potential security exploits. Many tasks were designed as microservices that act as additional security boundaries and only expose minimal information to other internal services. For example, the VPN relays only have a list of allowed WireGuard keys and do not know which account these keys belong to. The keys are frequently rotated by the clients. On the other hand, the Mullvad API does not see any VPN traffic. A service related to the API is able to see the amount of connections on a relay and can ban WireGuard keys that are used on multiple relays, but the service only sees an obfuscated version of the WireGuard keys and cannot learn the real keys used on a relay. Additionally, log messages and statistics are designed in a way that they do not contain account information.

In conclusion, the reviewed scope appears to be on a good security level compared to systems of similar size and complexity. X41 recommends improving the code base with regards to concurrency issues.

2 Introduction

X41 reviewed the Mullvad API[1] and backend services which are involved in processing payments of customers, managing customer devices and distributing WireGuard public keys to relays. Further, the API serves as data source for the public website.

Confidentiality, integrity and availability of the stored data and functionality constitute the core of the business and any compromise would expose sensitive customer data. Since confidentiality is a major selling point of the product, a compromise constitutes an existential business risk.

For example, attackers could try to attack the API in a way that will expose payment information or IP[2] addresses, which may serve as a proxy to the customer's identity. One of Mullvad's use cases provides citizens of authoritarian governments with a way to circumvent censorship. A failure to uphold anonymity guarantees may lead to prosecution.

2.1 Threat Model

The following informal threat model was created during this security audit and can serve as a benchmark for rating the findings. It is intended to be extended over time and can also serve as guidance for future security audits. The scope of the threat model covers the software systems providing the Mullvad API. A threat model for the client applications was published in a previous security audit[3] by X41.

After motivating the creation of a threat model, the assets to protect and the possible threats are listed. Given the security review, a list of general assumptions is given which we believe to be justified. The threat model is finalized by a list of limitations, which the user of the service should consider with regards to their individual risk profile and perceived threat.

[1] Application Programming Interface
[2] Internet Protocol
[3] `https://x41-dsec.de/news/2024/12/11/mullvad/`

2.1.1 Rationale

1. End-users want to make informed choices about whether or not a VPN[4] service is able to cover their perceived threat. An 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 may be helpful for developers and future security reviews to check these against a documented threat model.

2.1.2 Threats

The following is a list of possible threat actors who could attempt to undermine the security of the service:

- Compromised partner company with access to partner API
- Compromised employee seeking access to a different role
- Network attacker
- Supply chain attacker
- Advanced law enforcement agency

2.1.3 General Assumptions

X41 assumes the following statements to hold at the time of this audit. Note that every security audit is a best-effort assessment and does not translate to absolute safety.

1. Mullvad ensures that the user network activity and connection metadata is not exposed or logged through the API.
2. Mullvad ensures that the user IP address is not exposed or logged through the API.
3. Mullvad ensures that the payer metadata is not exposed or logged through the API and scrubbed from the database after an initial amount of time that still allows processing refunds.
4. Mullvad ensures that account numbers are not exposed or logged through the API.
5. Relay service availability does not immediately depend on the API availability.
6. User account time credit is sufficiently protected from tampering.

[4] Virtual Private Network

7. Relays are deployed and operated by dedicated staff.
8. Support staff can access production systems only in controlled environments.

2.1.4 Limitations

The following limitations need to be considered when deciding whether to use the service under a perceived threat model or not.

1. Mullvad cannot hide the fact that an IP addresses uses Mullvad Services.
2. External payment providers learn and keep payer metadata for an extended period of time and provide Mullvad VPN AB access to this data. Mullvad cannot delete this data with the payment providers where it could become compromised or provided to law enforcement agencies.
3. The privacy of payment transactions is subject to the limitations of the employed payment method.
4. Mullvad is subject to Swedish and European Law. Any data that is retained could potentially be obtained by law enforcement agencies.
5. Mullvad is subject to software and hardware supply chains which are not directly controlled by Mullvad.

2.2 Methodology

The review was conducted as a source code audit. In an audit, reviewers are given access to the source code and the environment it is running in. This allows identifying deeply-hidden vulnerabilities in a more efficient way.

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 web application security tools[5].

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[6] standards and the *Study - A Penetration Testing Model*[7] of the German Federal Office for Information Security.

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

The workflow of source code reviews is shown in figure 2.1. In an initial, informal workshop regarding the design and architecture of the application, a basic 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.

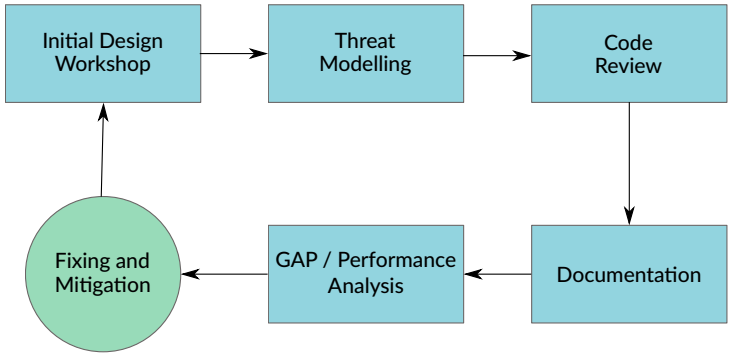

Figure 2.1: Code Review Methodology

2.3 Findings Overview

DESCRIPTION	SEVERITY	ID	REF
Race Condition when Submitting Vouchers	MEDIUM	MLLVD-CR-25-01	4.1.1
Redacted Issue	MEDIUM	MLLVD-CR-25-02	4.1.2
Possible Leak of Existence of Payment Hash of Lightning Invoice	LOW	MLLVD-CR-25-03	4.1.3
Authorization Bypass in Peerkaboo	LOW	MLLVD-CR-25-04	4.1.4
Redacted Issue	MEDIUM	MLLVD-CR-25-05	4.1.5
Possible Race Condition Between Payment and Account Number Rotation	NONE	MLLVD-CR-25-100	4.2.1
Possible Time of Check/Time of Use Issue Between Account Deletion and Device Registration	NONE	MLLVD-CR-25-101	4.2.2
Duplicate Code for in Payment Handlers	NONE	MLLVD-CR-25-102	4.2.3
Account Extended Twice during Payment Processing	NONE	MLLVD-CR-25-103	4.2.4
Supply Chain Exposure Could Be Decreased	NONE	MLLVD-CR-25-104	4.2.5
CAA Resource Records	NONE	MLLVD-CR-25-105	4.2.6
Redacted Issue	NONE	MLLVD-CR-25-106	4.2.7
Delayed HTTP Body	NONE	MLLVD-CR-25-107	4.2.8
Missing mTLS	NONE	MLLVD-CR-25-108	4.2.9
Redacted Issue	NONE	MLLVD-CR-25-109	4.2.10
Missing Type Hints	NONE	MLLVD-CR-25-110	4.2.11
Nginx Configuration Complexity	NONE	MLLVD-CR-25-111	4.2.12
Unsigned Relay List	NONE	MLLVD-CR-25-112	4.2.13
Container Hardening	NONE	MLLVD-CR-25-113	4.2.14
Nginx Listening on IPv6	NONE	MLLVD-CR-25-114	4.2.15

Table 2.1: Security-Relevant Findings

2.4 Scope

X41 was given access to several source code repositories. Each repository had a version tag `x41-api-audit-2025` which was based on the latest commit and used by the auditors. The exact commit references are listed below.

Repository	Commit
`mullvad/docker-pgbouncer`	`3768b7b77942ab3ecf9054b24f51e284ee818de0`
`mullvad/docker-postfix`	`6f6aee2a30b5c4b74333d29699c8e756a70e2491`
`mullvad/docker-redis-exporter`	`0e781ec8360d7f6cbdc722eef69a874da094144d`
`mullvad/frootloot`	`5ea561c4d0309950a0edf62fa7389c193316dc51`
`mullvad/lightning-payment-proxy`	`0cb1b1ecccbd8cae2960d192e52996f37750861d`
`mullvad/mullvad-api`	`86521b21d9e22105f40927b1b41d234605a5dff5`

mullvad/nginx-geomancer	*c1cf1da7d5dc5040e7222c0c13f185f73a946d82*
mullvad/peerkaboo	*371e2f520a105e62d458e6cb8f24ad3917a5805f*
mullvad/scroogle	*3026a3fe219293a9003b7c29ff8b30d7b8b8878c*
mullvad/yellow-pages	*a68b20225aa36c10c15eae42eca2c4db88267d0f*

The repositories also contained Dockerfiles and docker-compose files.

Additionally, X41 had root access to a development setup of the API and backend services including the nginx[8] configuration.

The first week of the audit was conducted in the offices of Mullvad, in Gothenburg, Sweden. This on-site testing was beneficial to the audit due to the direct communication and discussions about Mullvad's security posture. Questions regarding both in-scope and out-of-scope subjects were answered in-depth which allowed the testers to quickly gain a good understanding of the service.

A Slack channel was also set up for further communication between the developers and the testers.

2.5 Coverage

A security assessment attempts to find the most important or sometimes as many of the existing problems as possible, though it is practically never possible to rule out the possibility of additional weaknesses being identified in the future.

The time allocated to X41 for this assessment was sufficient to yield a good coverage of the given scope.

X41 audited the API middleware and the authentication scheme. The API endpoints were reviewed for possible TOCTOU[9] issues and race conditions, which may affect the consistency of the database and business logic. Further, the code base was audited for ways to cause DoS[10] conditions. Due to the sensitivity of some of the data, the API was checked for side-channel attacks that may leak customer data.

The same treatment was given to payment proxies, which run in separate processes and communicate with the API server via RPCs[11].

The internal services *Yellow Pages* and *Peerkaboo* were reviewed for security issues. An emphasize was given to how the relays communicate with *Peerkaboo*, the authentication scheme

[8] `https://nginx.org`
[9] Time of Check / Time of Use
[10] Denial of Service
[11] Remote Procedure Calls

and the nginx configuration which ties both together. This included, among other things, path traversal issues and HTTP[12] request smuggling attacks.

All code was reviewed for common security issues and issues which may impact performance or availability of the service. All calls to logging functions were reviewed for potentially leaking sensitive information. The dependency graph was investigated for potential supply chain issues. The tests were reviewed for coverage and correctness. The output of the static analyzer `Bandit` was analyzed for issues.

A threat model covering this scope was created.

2.6 Recommended Further Tests

X41 recommends mitigating the issues described in this report and conducting a security review of the code base after major changes to the code base.

Further tests could cover the source code of the relay software, and infrastructure such as internal networks and services, CI/CD[13] infrastructure, and relay setup.

[12] HyperText Transfer Protocol
[13] Continous Integration/Continous Delivery

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

Testers rate all security-relevant findings using the CVSS industry standard version 4.

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 4.0 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/v4.0/specification-document`

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)
- Attack Requirements (AT)
- Privileges Required (PR)
- User Interaction (UI)
- Vulnerable/Subsequent System Confidentiality (VC/SC)
- Vulnerable/Subsequent System Integrity (VI/SI)
- Vulnerable/Subsequent System Availability (VA/SA)

The *Threat Metric Group* represents the current state of exploit techniques and the availability of proof of concepts. It captures the following metric:

- Exploit Maturity (E)

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:

- Confidentiality Requirement (CR)
- Integrity Requirement (IR)
- Availability Requirement (AR)
- Modified Attack Vector (MAV)
- Modified Attack Complexity (MAC)
- Modified Attack Requirements (MAC)
- Modified Privileges Required (MPR)
- Modified User Interaction (MUI)
- Modified Vulnerable System Confidentiality (MVC)
- Modified Vulnerable System Integrity (MVI)
- Modified Vulnerable System Availability (MVA)
- Modified Subsequent System Confidentiality (MSC)
- Modified Subsequent System Integrity (MSI)
- Modified Subsequent System Availability (MSA)

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:4.0/AV:N/AC:H/AT:N/PR:L/UI:A/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N` defines a base score metric with the following parameters:

- Attack Vector: Network
- Attack Complexity: High
- Attack Requirements: None
- Privileges Required: Low
- User Interaction: Active
- Vulnerable System Confidentiality: High
- Vulnerable System Integrity: Low
- Vulnerable System Availability: None
- Subsequent System Confidentiality: None
- Subsequent System Integrity: None
- Subsequent System Availability: 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
Exploitability	Medium
Complexity	Medium
Vulnerable system	Medium
Subsequent system	Low
Exploitation	High
CVSS Score	5.8 (Medium)

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/4.0#CVSS:4.0/AV:N/AC:H/AT:N/PR:L/UI:A/VC:H/VI:L/VA:N/SC:N/SI:N/SA: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-25-01: Race Condition when Submitting Vouchers

Severity:	MEDIUM / 6.3
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N
CWE:	367 – Time-of-check Time-of-use (TOCTOU) Race Condition
Component:	`https://github.com/mullvad/mullvad-api/blob/x41-api-audit-2025/payments/models.py#L716-L744`

4.1.1.1 Description

The ***VoucherManager.submit()*** method uses the ***@transaction.atomic()*** decorator[1], which does not protect against TOCTOU issues.

Concurrent voucher submissions can result in the application fetching the "unused" voucher code multiple times concurrently, extending the account expiry, and only then marking the voucher as used.

[1] `https://docs.djangoproject.com/en/4.2/topics/db/transactions/#django.db.transaction.atomic`

The effect is negligible when doing this for the same account, as the calculated time extension would be the same for each concurrent request, but it allows applying a single voucher code to multiple accounts.

To simulate and verify the race condition, X41 inserted an artificial delay between the check if the voucher was used in line 5 and calling the ***save()*** routine on the voucher in line 13 of listing 4.1.

```
1  @transaction.atomic(using='accounts')
2  def submit(self, code, account):
3      voucher = self.find_voucher(code)
4
5      if voucher.used:
6          raise exceptions.VoucherAlreadyUsed
7
8      if voucher.group.include_in_paid_stats:
9          voucher.payment = Payment.objects.create(...) # omitted by X41 for brevity
10     voucher.account = account
11     voucher.date_used = timezone.now()
12     voucher.used = True
13     voucher.save()
14
15     account.extend_expiry(voucher.value())
16
17     return int(voucher.value().total_seconds())
```

Listing 4.1: TOCTOU Issue During Voucher Redemption

X41 later verified the issue in the production environment, where a single voucher was successfully redeemed to 16 different accounts. The attack made use of techniques described in sections 4.2.7 and 4.2.8.

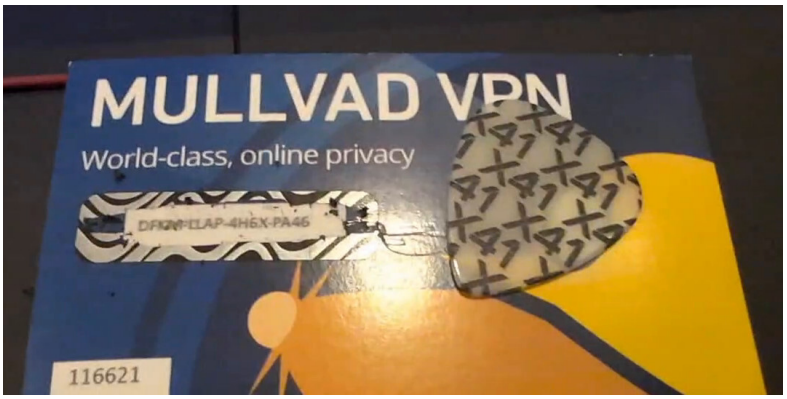

Figure 4.1: Voucher DFKM-LLAP-4H6X-PA46 which was used 16 times by X41

4.1.1.2 Solution Advice

X41 recommends using ***select_for_update()***[2] in the ***find_voucher()*** method and has verified that it fixes the vulnerability when using PostgreSQL. The database will delay other transactions that select the row until the current transaction was committed, see also the `SELECT FOR UPDATE` documentation[3].

X41 further recommends adding a test case that ensures the fix is working correctly in order to protect against regressions.

This issue was fixed during the audit by locking the database row for an update during the operation.

[2] `https://docs.djangoproject.com/en/4.2/ref/models/querysets/#select-for-update`
[3] `https://www.postgresql.org/docs/16/explicit-locking.html#LOCKING-ROWS`

4.1.2 MLLVD-CR-25-02: Redacted Issue

Severity:	MEDIUM / 6.3
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:L
CWE:	400 – Uncontrolled Resource Consumption ('Resource Exhaustion')
Component:	

4.1.2.1 Description

This issue describes a method which can be used to bypass defensive mechanisms put in place to protect the availability of the Mullvad API. It does not affect the confidentiality or integrity of customer data.

4.1.3 MLLVD-CR-25-03: Possible Leak of Existence of Payment Hash of Lightning Invoice

Severity:	LOW / 2.3
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:P/VC:N/VI:N/VA:N/SC:L/SI:N/SA:N
CWE:	200 – Exposure of Sensitive Information to an Unauthorized Actor
Component:	`https://github.com/mullvad/mullvad-api/blob/x41-api-audit-2025/payments/lightning/services.py#L112-L116`

4.1.3.1 Description

The function ***check()*** retrieves the payment status of a Lightning payment by looking up the payment hash in a local database. If the payment hash exists, another query ensures that the payment is connected to the authenticated account issuing the check before returning the status information.

An attacker with knowledge of the payment hash could use this endpoint to probe if the payment hash is contained in the database by measuring the response times of the endpoint. A lookup for a hash that exists in the invoice database will incur an additional database query to check that the associated account is the one that created the payment process. The listing 4.2 shows the two responsible operations. Each operation may throw an exception, which would then return that the payment was not found, however, their execution time will differ depending on the existence of the payment hash.

The impact is limited by the fact that the payment hash is not public information, but is only known to intermediate routing nodes of the Lightning Network.

```
1 try:
2     check_payment_response = proxy.check_payment(provider_id)
3     payment_token = PaymentToken.objects.get(
4         account=account, value=check_payment_response.payment_token
5     )
```

Listing 4.2: Timing Differential when Checking Payments

4.1.3.2 Solution Advice

X41 recommends using a constant time operation to facilitate the lookup of the payment hash. For this, even if the payment was not found initially, the endpoint should check for a random payment token, disregard the result and return that the payment was not found. Alternatively, it may be possible to use the payment token, which is only known to authorized parties and contains sufficient entropy to not be guessed in a brute-force attack.

This issue was fixed during the audit by using an identifier for lightning invoices that cannot be derived from the payment hash by an observer of the payment.

4.1.4 MLLVD-CR-25-04: Authorization Bypass in Peerkaboo

Severity:	LOW / 2.1
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:P/PR:H/UI:N/VC:N/VI:L/VA:L/SC:N/SI:N/SA:N
CWE:	639 – Authorization Bypass Through User-Controlled Key
Component:	`https://github.com/mullvad/peerkaboo/blob/x41-api-audit-2025/connbouncer/handler.go#L14`

4.1.4.1 Description

Requests from relays to Peerkaboo are authenticated in *`mullvad-api`* using the *`Authorization: Basic`* header. In Peerkaboo, the relay is then identified using the *`X-Relay-Hostname`* header.

A compromised relay or an attacker with knowledge of a relay's password could send multiple requests with different *`X-Relay-Hostname`* headers, and would thus be able to get a WireGuard peer banned by reporting that the WireGuard key is used on multiple relays at the same time.

A successfull attack depends on knowledge of the hash, which is calculated on relays using the WireGuard public key and a shared secret which the *`mullvad-api`* does not know.

4.1.4.2 Solution Advice

X41 recommends using mTLS[4] to authenticate and identify relays, which would also protect against observed credentials being reused by attackers.

As a short-term mitigation, the relay could be identified in Peerkaboo using the *`Authorization: Basic`* header.

This issue was fixed during the audit by relying on the authorization header to identify the relay which the request originated from.

[4] Mutual Transport Layer Security

4.1.5 MLLVD-CR-25-05: Redacted Issue

Severity:	MEDIUM / 6.9
CVSS Vector:	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:L/SC:N/SI:N/SA:N
CWE:	799 – Improper Control of Interaction Frequency
Component:	

4.1.5.1 Description

This issue describes a method which can be used to bypass defensive mechanisms put in place to protect the availability of the Mullvad API. It does not affect the confidentiality or integrity of customer data.

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-25-100: Possible Race Condition Between Payment and Account Number Rotation

Component:	mullvad_api/accounts/models.py

4.2.1.1 Description

For an existing account, the function ***change_account_token()*** creates a new account, migrates all attributes of the existing account to the new account and deletes the existing account. If a payment process finishes between migrating all existing payments and migrating all payment tokens to the new account, the payment will be created and associated with the existing account, which is about to be deleted. The deletion will set the associated account for the payment to `null`. The probability of this happening is very low, since changing the account token is a manual process triggered by support staff only, whereas the time of the payment is largely controlled by the user. Further, the window to hit this condition is small.

4.2.1.2 Solution Advice

X41 recommends using ***select_for_update()***[5] for the payment tokens to ensure that a new payment will either be associated with the newly created account, or with the old account, but will be subject to the migration. For this, the payment tokens should be migrated before the payments.

[5] `https://docs.djangoproject.com/en/4.2/ref/models/querysets/#select-for-update`

4.2.2 MLLVD-CR-25-101: Possible Time of Check/Time of Use Issue Between Account Deletion and Device Registration

Component:	mullvad_api/accounts/services.py

4.2.2.1 Description

When an account is deleted by a user, it is soft deleted, i.e. the account's deleted field is set to true, but the row is not immediately removed from the database. If an account is soft-deleted, a user cannot authenticate using this account anymore. However, existing sessions that are authenticated can still proceed. These can reliably be produced by sending the complete headers including the `Authorization` header and delay the sending of the body. One operation that is possible is registering a new device. This issues does not have any impact on operations or other limits. The device will be removed when the account is finally deleted after a configured period.

4.2.2.2 Solution Advice

X41 recommends fixing this issue by filtering for non-deleted accounts in ***register_device()***.

4.2.3 MLLVD-CR-25-102: Duplicate Code for in Payment Handlers

Component:	mullvad_api/payments

4.2.3.1 Description

Both the Swish and PayPal payment processes rely on callbacks from the respective payment processor signaling a change of state of the payment process. To protect the API from arbitrary connections, the endpoint is restricted to IP addresses from the range of the provider. Both payment processors implement their own method to check if the source IP address is contained in the whitelisted network:

- ***is_remote_ip_valid()*** in *payments/paypal/utils.py*
- ***_get_remote_ip()***, ***callback()*** and ***refund_callback()*** in *payments/swish/callbacks.py*

4.2.3.2 Solution Advice

X41 recommends implementing the method once to increase maintainability.

4.2.4 MLLVD-CR-25-103: Account Extended Twice during Payment Processing

Component:	mullvad_api/payments

4.2.4.1 Description

When a payment from Google Play or Apple is processed via their respective payment proxies, an *GooglePlayPayment* or *ApplePayment* is created. Both classes derive from *Payment*, which implements a ***save()*** handler that extends the expiration date for the associated account by the purchased amount. The same extension is, however, explicitly executed in the special handlers that create the payment objects in the first place. Listing 4.3 shows the function ***_credit_payment()*** in *mullvad-api/payments/googleplay/services.py*.

```
1  def _credit_payment(*, payment_token, payment_spec, client):
2    [..]
3      _, created = GooglePlayPayment.objects.get_or_create(
4          provider_id=payment_spec.get('provider_id'),
5          defaults={
6              'token': payment_token,
7              'account': payment_token.account,
8              'time_added': value,
9          },
10     )
11     if created:
12         payment_token.account.extend_expiry(value)
13   [..]
```

Listing 4.3: Redundant Account Extension Calls

GooglePlayPayment.objects.get_or_create() will internally call the ***save()*** method, which then extends one instance of the account. The call to ***payment_token.account.extend_expiry()*** will extend another instance of the same account. Since both account instances have the same expiration date, the account does not receive twice the amount of time. Therefore the impact is one superfluous *UPDATE* operation executed against the database. Further, two concurrent payments may race and not extend the account expiry twice, but only one payment will be effective.

4.2.4.2 Solution Advice

X41 recommends making business logic as explicit as possible. All database operations for one business event should reside in the same function, and ideally the same transaction. This simplifies reasoning about database accesses, especially when concurrency issues play a role in the code base.

4.2.5 MLLVD-CR-25-104: Supply Chain Exposure Could Be Decreased

Component:	mullvad_api

4.2.5.1 Description

The list of dependencies for core API, which has access to the database is already quite minimized due to an effort to place handling of payment provider communication into their own microservices. A supply chain attack on one of the dependencies of third-party libraries will then be confined to the boundaries of the microservices. This practice could be extended to the Stripe payment provider, as well as anything using the `fpdf2` package. This dependency is currently used to generate invoices and payment receipts and could be segregated to another microservice which would remove the exposure of supply chain attacks further. Another dependency that may be removed completely (or constrained to only the dev environment) is `sshpubkeys`, as it is only used in tests.

4.2.5.2 Solution Advice

X41 recommends placing the logic generating invoices and receipts as well as Stripe payments into their own microservice in order to mitigate supply chain attacks by isolating the impact.

4.2.6 MLLVD-CR-25-105: CAA Resource Records

Component: DNS

4.2.6.1 Description

The security of the API depends on TLS[6] and on attackers not being able to obtain a valid TLS certificate for host names like `mullvad.net`, `api.mullvad.net`, and other subdomains.

Mullvad has defined the following CAA[7] DNS[8] resource records.

```
1 mullvad.net.        1161    IN    CAA    128 issue "letsencrypt.org"
2 mullvad.net.        1161    IN    CAA    128 issuewild "letsencrypt.org"
3 mullvad.net.        1161    IN    CAA    128 iodef "mailto:support@mullvaddns.net"
```

Listing 4.4: CAA Resource Records

This restricts the certificate issuance to Let's Encrypt. However, Let's Encrypt can be freely used by attackers in an automated manner. Additionally, the policy allows the issuance of wildcard certificates despite Mullvad showing very limited usage of wildcard certificates. No CAA resource records exist for unused subdomains, such as `support.mullvad.net`.

Additionally, X41 noticed that `mullvaddns.net` has no *MX*, *A*, or *AAAA* resource record. As such, emails sent to `support@mullvaddns.net` cannot be received.

4.2.6.2 Solution Advice

X41 recommends a stricter policy.

Let's Encrypt supports[9] the *validationmethods* and *accounturi* parameters which restrict issuance to certain validation methods and ACME[10] Account key.

The *issuewild* property should be set to *";"* in order to prevent wildcard certificate issuance. Exceptions can be made for domain names where wildcard certificates are used, such as `*.dnsleak.am.i.mullvad.net`.

[6] Transport Layer Security
[7] Certification Authority Authorization
[8] Domain Name System
[9] `https://letsencrypt.org/docs/caa/`
[10] Automatic Certificate Management Environment

Domains that are not used with TLS certificates should have both the `issue` and `issuewild` set to `";"`. This can be achieved using a wildcard DNS record such as `*.mullvad.net`.

4.2.7 MLLVD-CR-25-106: Redacted Issue

Component:	Network Infrastructure, mullvad-api

4.2.7.1 Description

This issue describes a method which can be used to bypass defensive mechanisms put in place to protect the availability of the Mullvad API. It does not affect the confidentiality or integrity of customer data.

4.2.8 MLLVD-CR-25-107: Delayed HTTP Body

Component: Nginx, mullvad-api

4.2.8.1 Description

HTTP requests can be split into multiple TCP[11] packets which are sent at different times. Due to the way HTTP and Django are designed, the Django application processes requests as soon as the HTTP request header is fully received. The request body is then exposed as an I/O[12] stream on the `HttpRequest` object; the read will block until the full body is available.

In `mullvad-api` the rate limiting and the authentication take place as soon as the header is fully received, where the authenticated account is then stored as `request.account`. This variable is sometimes used when further processing the request. When processing depends on reading the request body, an attacker may induce arbitrary delays while the application waits for the body to become available. The underlying database object may then have been modified or deleted in the meantime.

Attackers may send HTTP request headers to the server, keep the connection open, and delay sending the body at a later time. This may be used to abuse possible TOCTOU issues, as described in the informational note 4.2.2 or to send multiple requests at nearly the same time while circumventing the rate limit.

X41 used this technique for the finding described in section 4.1.1.

4.2.8.2 Solution Advice

X41 recommends storing the unique IDs[13] in request variables rather than full objects, and possibly using ***select_for_update()***[14] to fetch these objects again where required.

Where no large HTTP request bodies such as large file uploads or WebSocket connections are expected, the `proxy_request_buffering on`[15] directive may be used in Nginx in order to delay forwarding the request to the Django application until the body is available. It is important to combine this with `proxy_http_version 1.0` to prevent chunked transfer encoding, and to set a reasonable `client_max_body_size` and relevant timeouts.

[11] Transmission Control Protocol
[12] Input/Output
[13] Identifiers
[14] `https://docs.djangoproject.com/en/4.2/ref/models/querysets/#select-for-update`
[15] `https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_request_buffering`

4.2.9 MLLVD-CR-25-108: Missing mTLS

Component: Infrastructure

4.2.9.1 Description

The infrastructure is made up of various microservices which communicate with each other. The services may reside on the same physical server or on the same network. However, not all of the communication is mutually authenticated. This includes the connections between the Django server and the payment proxies and the database (including Redis), connections between Peerkaboo and the database, and between Yellow Pages and Netbox. An attacker gaining control of one of the services, say a payment proxy, could perform further attacks on other parts of the infrastructure. Some of the communication believed to be on the same host is currently not encrypted. An attacker gaining control of one of the services may be able to observe unencrypted traffic on the same physical host.

4.2.9.2 Solution Advice

X41 recommends employing mTLS to secure all internal connections as part of a defense in depth approach. This furthers service isolation as not all services may communicate with any other service unauthenticated. As a side effect, all communication will be encrypted by default.

4.2.10 MLLVD-CR-25-109: Redacted Issue

Component: mullvad-api

4.2.10.1 Description

This issue describes a method which can be used to bypass defensive mechanisms put in place to protect the availability of the Mullvad API. It does not affect the confidentiality or integrity of customer data.

4.2.11 MLLVD-CR-25-110: Missing Type Hints

Component:	frootloot, mullvad-api

4.2.11.1 Description

The Python projects *Frootloot* and *mullvad-api* do not make use of Python's type hints[16], which can help developers to better understand the code, refactoring, and spotting mistakes using type checkers and linters.

X41 noticed that the *Scroogle* project does make use of type hints.

4.2.11.2 Solution Advice

X41 recommends using type hints in Python projects.

[16] https://docs.python.org/3.14/library/typing.html

4.2.12 MLLVD-CR-25-111: Nginx Configuration Complexity

Component: Nginx

4.2.12.1 Description

The Nginx configuration that was made available to X41 showed a high complexity. The configuration handles requests from end users, relays, and Mullvad staff and includes request modifications, routing decisions, and authorization, among others.

While no immediate vulnerabilities were found, X41 believes that the complexity comprises a risk and future changes may be subject to common pitfalls in Nginx configuration.

For example, `proxy_pass http://upstream;` will forward a request with the original, unmodified path (which may contain directory traversal attempts) while `proxy_pass http://upstream/;` will first resolve and normalize the path before forwarding. When combined with a `location` directive used as a security boundary, this may lead to issues.

4.2.12.2 Solution Advice

X41 recommends simplifying the configuration as much as possible to remove complex logic from the reverse proxy. This includes replacing rewrites with redirects to subdomains and implementing further routing in the upstream services. Conditional constructs (using the keyword *if*) should be avoided. WireGuard keys should not be part of a URL[17] path and only be accepted in POST request bodies. For relay authentication mTLS should be preferred over the `auth_request` directive.

[17] Uniform Resource Locator

4.2.13 MLLVD-CR-25-112: Unsigned Relay List

Component: Yellow Pages

4.2.13.1 Description

The list of relays is not signed. If an attacker is able to modify the list of relays, the client or the Mullvad operators may not be able to detect a rogue relay before clients connect to it and use it for some period of time, exposing their connection data.

4.2.13.2 Solution Advice

X41 is aware of plans to add transparency logs and recommends following that plan.

In the meantime, X41 recommends cryptographically signing the list of relays and implementing changes in the Mullvad clients to verify the signatures.

4.2.14 MLLVD-CR-25-113: Container Hardening

Component: Docker

4.2.14.1 Description

In the development environment that X41 had access to, the Docker daemon ran with root permissions. Although the processes are restricted to dedicated namespaces and a cgroup, the root user still holds more permissions than the services need to run. Additionally, the containers had Internet access without an apparent need to access the Internet. This may be used by attackers to abuse SSRF[18] attacks, exfiltrate data, or download malware.

4.2.14.2 Solution Advice

X41 recommends running the container deaemon unprivileged and using UID mapping. Internet access should be disabled except where needed, and then firewalled to necessary IP addresses, protocols, and ports. Processes should be executing using unprivileged accounts inside the containers. Containers should run with all capabilities removed and only those necessary added. Additionally, reasonable resource limits should be applied to each container.

[18] Server-Side Request Forgery

4.2.15 MLLVD-CR-25-114: Nginx Listening on IPv6

Component: Nginx

4.2.15.1 Description

Nginx was configured to listen on IPv6[19] although the API domain name has no IPv6 address. While it is generally desirable to use IPv6, some features such as rate limiting do not currently account for IPv6.

All of the four API hosts appeared to be blocked by a firewall when accessing them via IPv6 on TCP port 443.

4.2.15.2 Solution Advice

X41 recommends adding support for IPv6. Until then, listening on IPv6 should be disabled to prevent accidental exposure of the endpoints via IPv6.

[19] Internet Protocol Version 6

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

ACME Automatic Certificate Management Environment 31
API Application Programming Interface 6
CAA Certification Authority Authorization 31
CI/CD Continous Integration/Continous Delivery 12
CVSS Common Vulnerability Scoring System 13
CWE Common Weakness Enumeration 16
DNS Domain Name System 31
DoS Denial of Service 11
HTTP HyperText Transfer Protocol 12
ID Identifier 34
I/O Input/Output 34
IP Internet Protocol 6
IPv6 Internet Protocol Version 6 41
mTLS Mutual Transport Layer Security 23
RPC Remote Procedure Call 11
SSRF Server-Side Request Forgery 40
TCP Transmission Control Protocol 34
TLS Transport Layer Security 31
TOCTOU Time of Check / Time of Use 11
URL Uniform Resource Locator 38
VPN Virtual Private Network 7