ASSURED

SECURITY CONSULTANTS

Report

Mullvad VPN Web Application Penetration Test

Alexander Alasjö, Emilie Barse

Project	Version	Date
MUL022	4	2025-09-30

ASSURED
SECURITY CONSULTANTS

REPORT

Project	Version	Date
MUL022	4	2025-09-30

Executive summary

Between 2025-08-11 and 2025-08-22 Assured Security Consultants performed a penetration test of the Mullvad VPN web application on behalf of Mullvad VPN AB.

The web appliation, the Onion service setup for the web application, the rsync setup for synchronizing static content between web servers, and the CMS admin application were part of the scope. Backend APIs and payment services were not part of the scope.

This report lists the security issues found, along with recommendations for remediation or mitigation. In our conclusions, we discuss the issues and address apparent patterns in areas where security is lacking.

Overall, the Mullvad VPN web application and its administrative CMS application implement good security practices with only minor security-related issues found. Likewise, the Tor onion service setup and rsync implementation are both sound.

Observations were made with the following risk severity assessments (number of issues):

Critical 0 | High 0 | Medium 0 | Low 1 | Note 5 | Good 5

A verification test was performed between 2025-09-25 and 2025-09-26 and at this time, the finding with *Low* risk and four out of the five *Note* observations were fixed according to our recommendations. The last Note observation was accepted. The observations in this report are annotated with the outcome of the verification testing.

Assured would like to thank Mattias, William, Alexander and Hank for their support during this penetration test. We are happy to answer any questions and provide further advice.

ASSURED
SECURITY CONSULTANTS

REPORT

Project	Version	Date
MUL022	4	2025-09-30

Contents

1 Introduction

1.1 Background

Assured AB (Assured) was contracted to perform a penetration test of the Mullvad VPN web application on behalf of Mullvad VPN AB.

1.2 Constraints and disclaimer

This report contains a summary of the observations made during the project period. This report should not be considered as a complete list of all vulnerabilities, security flaws and/or misconfigurations.

1.3 Project period and staffing

Assured started the project on 2025-08-11 and finished on 2025-08-22.

An additional verification test was carried out between 2025-09-25 and 2025-09-26, to inspect the fixes and mitigations put in place after the original penetration test.

This report was last reviewed on 2025-09-30.

Involved in the penetration testing were Assured consultants Alexander Alasjö and Emilie Barse.

1.4 Assured Security Consultants

Assured Security Consultants (Assured AB[1]) was founded in 2015 with the mission to provide premier technical cybersecurity services as an independent consultancy, not affiliated with any vendor. Our team of experienced and dedicated cybersecurity specialists perform penetration testing, red team activities, secure design, embedded development, advisory, training and similar services.

We are committed to the security community as OWASP chapter leaders, event organizers and podcasters. We also take active part in security research projects into areas such as cryptography and automotive security.

[1]Assured AB, Org.nr. 556985-8276, registered in Sweden. www.assured.se

2 Scope and methodology

2.1 Scope

2.1.1 Penetration test of Mullvad VPN web application

The Mullvad VPN web application, including the (Tor) Onion service setup for the web application and the rsync setup for syncing static content between web servers were in scope for the test. Backend APIs and payment services were not part of the scope.

The web application was tested in a development/staging environment.

The Onion service version of the application uses the same code base, with some restrictions in functionality to avoid personal information leakage. For example are card payments disabled when accessing the application over the Tor network using the Onion URL.

The test was performed as a white-box test with access to source code and SSH access into the development web servers, where Docker setup, logs, and configuration files could be reviewed.

2.1.2 Penetration test of CMS application

The administrative CMS application was also part of the scope. The CMS admin application is used by Mullvad staff to manage content of the web application, for example to publish blog posts.

The test was done as a white-box test with access to source code and user credentials for the CMS application.

2.2 Methodology

Testing of the Mullvad VPN web application and CMS application was carried out in accordance with the OWASP Testing Guide [1].

Categories of testing include input validation, session management, authentication and authorization, identity management, error handling, business logic, client side testing, cryptography, information gathering and configuration and deployment management. The test case coverage is listed in Section 4.

A combination of dynamic testing and manual review of the code was used. In addition, static code analysis tools were used to search for vulnerable third party libraries, check configurations, and to find dangerous code patterns.

The Tor Onion service was reviewed by examining the Tor configuration files and Docker setup in the web server. The Onion version of the web application was tested both in Tor

browser inspecting the storage and network activity and in a ”normal” browser proxying the request to Burp Suite and then on to the Tor service proxy. Network traffic in the client computer was examined while using the Onion application to attempt to detect any side effects of the application which could reveal personal information.

Rsync is used over SSH to synchronize static content between the web servers. The rsync scripts and SSH setup were reviewed to make sure the SSH configuration was hardened, only the intended content was synchronized, and that the rsync service was well separated from the web application.

Logs were examined during the test to make sure they do not contain any personally identifiable information (PII) about clients accessing the web application, in line with Mullvad’s business and threat model.

2.2.1 Tools used

The following tools were used in the test:

- Burp Suite Professional
- Semgrep (community edition, with custom rules)
- Trivy
- Nmap
- Tor browser

2.3 Observation tagging and risk rating

2.3.1 OWASP Risk Rating Methodology

In this report we have assessed the severity of issues and identified vulnerabilities according to the OWASP Risk Rating Methodology [2].

Table 1: OWASP Risk Rating overall severity model

Overall risk severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

As Table 1 visualizes, the overall risk assessment is determined from a combined likelihood and impact of an identified vulnerability or security issue. A value from 0 to 9 is assessed for each variable, where 0-2 is designated LOW, 3-5 is MEDIUM and 6-9 is HIGH.

Likelihood is dependent on attributes related to threat actors and the identified vulnerability, with factors such as: the skill level and motivations of the threat agents; how easily the vulnerability can be found and exploited, and; how likely an exploit may be detected.

Impact depends on technical and business factors, such as: level of loss of confidentiality, integrity, availability and accountability; potential financial damage; potential brand damage, and; potential violations of privacy.

Please note that the severity assessment is made by Assured consultants and ratings may differ from the resource owners' ratings.

2.3.2 Observations without risk rating

Observations that do not pose a direct security threat, we mark with NOTE. These concern issues with very low impact and/or likelihood, which still may be interesting for developers to know about and consider fixing.

Observations concerning functionality or settings that we deem follow good security practice or aligning with provided requirements, we mark with GOOD.

2.3.3 Verification state tagging

When we conduct a verification test, we tag the re-tested observation with the state depending on whether they are fixed, partially fixed, remaining or accepted, with the following corresponding tags:

- FIXED for verified fixed findings.
- PARTIALLY FIXED for findings where we found partial mitigation in effect.
- REMAINING where the finding was verified still valid.
- ACCEPTED where the finding was valid but will not be fixed by the customer for some reason.

3 Observations

3.1 Mullvad.net Web Application

The web application under test contains Mullvad VPN static content (application downloads, blog, help guides and account management portal including payment interfaces). Note that backend APIs and third-party integrations (payment) was out of scope for testing. The web application is served on clearnet as well as over the Tor network as an Onion service. In scope was also a security review of the sshd-rsync implementation for synchronizing certain static content between web servers.

3.1.1 GOOD Security Headers

The web application shows good use of HTTP response security headers, with an A+ rating[2]. Key headers are present and correctly configured, providing protection against common web-based attacks. In addition to the standard best practices, the application also incorporates privacy-centric headers such as `permissions-policy` blocking browser features and API access.

Content Security Policy (CSP)

The `content-security-policy` header is an effective measure to protect a web application from XSS and other attacks. A proper configuration allows content only from trusted sources, effectively preventing the browser from loading potentially malicious assets. Mullvad VPN implements a sound CSP with allowed third-party integration domains, nonce-source, and blocks other origins to embed the web application.

HTTP Strict Transport Security

The `strict-transport-security` header tells the browser to enforce the use of HTTPS and is set according to good security practice.

Cross-Origin Opener Policy, Cross-Origin Resource Policy

Headers `cross-origin-opener-policy, cross-origin-resource-policy` as implemented by Mullvad prevent cross-origin pages from accessing Mullvad's `window.opener`, and prevent other sites to load resources such as scripts, images, fonts, etc.

Permissions Policy

Mullvad also implements `permissions-policy` to block features and APIs in the browser such as geolocation, camera, usb and microphone.

[2] From `https://securityheaders.com`

Referrer Policy

The `referrer-policy` header in use tells the browser to only include a referrer header when navigating on the same origin, sending no referrer information when navigating away from Mullvad.

X-Content-Type-Options

The `x-content-type-options` header is correctly set to `nosniff`, stopping the browser from guessing the response body content type, adhering to the content-type as set in the response header. This prevents unexpected interpretation of responses as HTML or JavaScript, for example.

X-Frame-Options

The `x-frame-options` header in use prevents the browser from framing the web application and mitigates attacks such as clickjacking.

3.1.2 GOOD Tor Onion service review

The review of the Tor/Onion service configuration and Onion version of the application did not result in any issues to report.

The Onion service is run in a separate Docker container, with minimal configuration redirecting incoming Onion application requests to the Nginx web server.

Nginx is configured to handle incoming Onion requests and scrubs the `X-Forwarded-For` header before forwarding the request to the web application.

The functionality excluded from the Onion version of the application is the Mullvad VPN connection check to avoid sending the client IP in the GET request URL, and blocking all payment methods which are not privacy friendly, like credit card payments.

3.1.3 GOOD Rsync service review

The rsync service for synchronizing static content added in run-time between the web servers was reviewed and no issues were found.

The rsync service is run in a separate Docker container and synchronizes the `/var/www` folder on the web servers.

Rsync is run over SSH, and the SSH configuration is hardened to only allow public key authentication for the rsync service user account, and does not permit login for any other user accounts.

The synced folders use a separate Docker volume mounted in the Nginx container, and is thus separated from the application code and other content.

3.1.4 FIXED LOW Input parameter length check missing

Likelihood: MEDIUM (3), Impact: LOW (2)

Verification note: Input validation of the account number and voucher code is implemented. The length is checked and only a limited character set is allowed. It is no longer possible to get a 500 error and an invalid account number will not be returned in the response. The issue is considered fixed.

The application fails to enforce input length limitation on string input parameters, and will return at least one of them in the error message.

For example, the account number submitted in `POST /en/account/login` can be around 500 000 bytes long and generate a response 2 000 000 bytes long.

Figure 1 shows a request where the request is around 100 000 bytes long, and the response is four times that size because of escaped special characters in the JSON response, generating four backslashes for each input backslash in the response.

Making a request with a sufficiently long input results in a `500 Internal Server Error` response. An even longer input (~1MB) will result in: `413 Request Entity Too Large`.

The voucher code submitted in `POST /account/payment/voucher` can also be very long and will generate the same type of error messages for the same input lengths. This parameter is, however, not reflected in the response.

An attacker can use this behaviour to consume unnecessary resources. We did not observe any denial of service, i.e. causing the Docker container to restart or otherwise disrupt user experience. Thus, the impact of this finding is very low.

We recommend enforcing input length at point of submission (before forwarding them to backend APIs), and consider providing error responses without reflecting input. Nginx con-

figuration can be used to restrict the client max body size selectively for these requests. A strict format check of input parameters can be alternative, but this will increase the complexity of the code when a format check is also done in the backend api, and may thus not be a suitable solution.

3.1.5 FIXED NOTE Invalid CSP returned in some cases

Verification note: The Content Security Policy (CSP) is removed for the pages previously returning an invalid CSP. Only pages with `Content-Type: text/html` return a CSP. The issue is considered fixed.

The Content Security Policy is correctly applied on pages and routes (see Observation 3.1.1). However, for certain requests returning non-HTML responses (e.g., fetch/XHR), the header in some cases includes *null* (see Figure 2) which is an invalid directive. This appears to stem from an issue in how the CSP is generated in the source code.

While this does not represent a direct security risk, it may indicate unexpected behavior. It is also worth noting that an invalid CSP header is ignored entirely by the browser, which then falls back to applying its default policy where applicable.

However, for responses that are not HTML documents (such as JSON or plain text), a CSP is usually not applicable anyway. Moreover, the `x-content-type-options: nosniff` response header tells the browser to adhere to the HTTP response `content-type`, not making assumptions by looking at the response body and potentially rendering the body as HTML or executing as JavaScript, which is good.

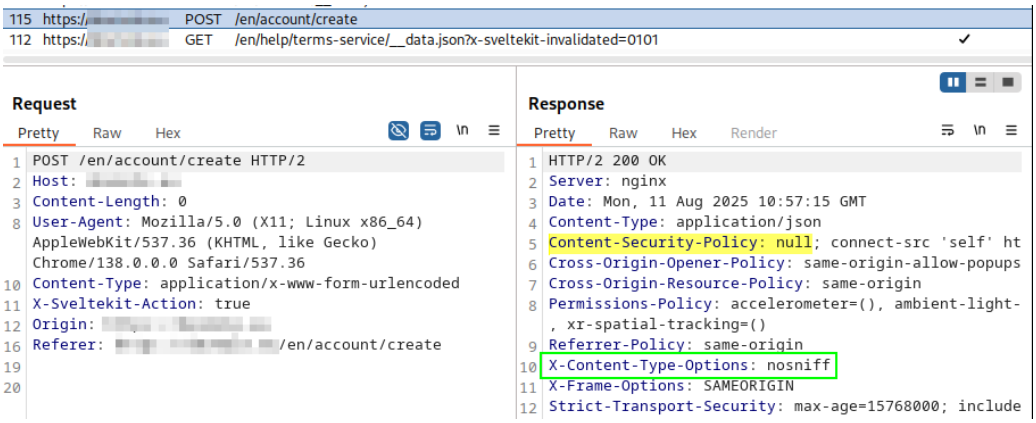

Figure 2: CSP includes null, an invalid directive

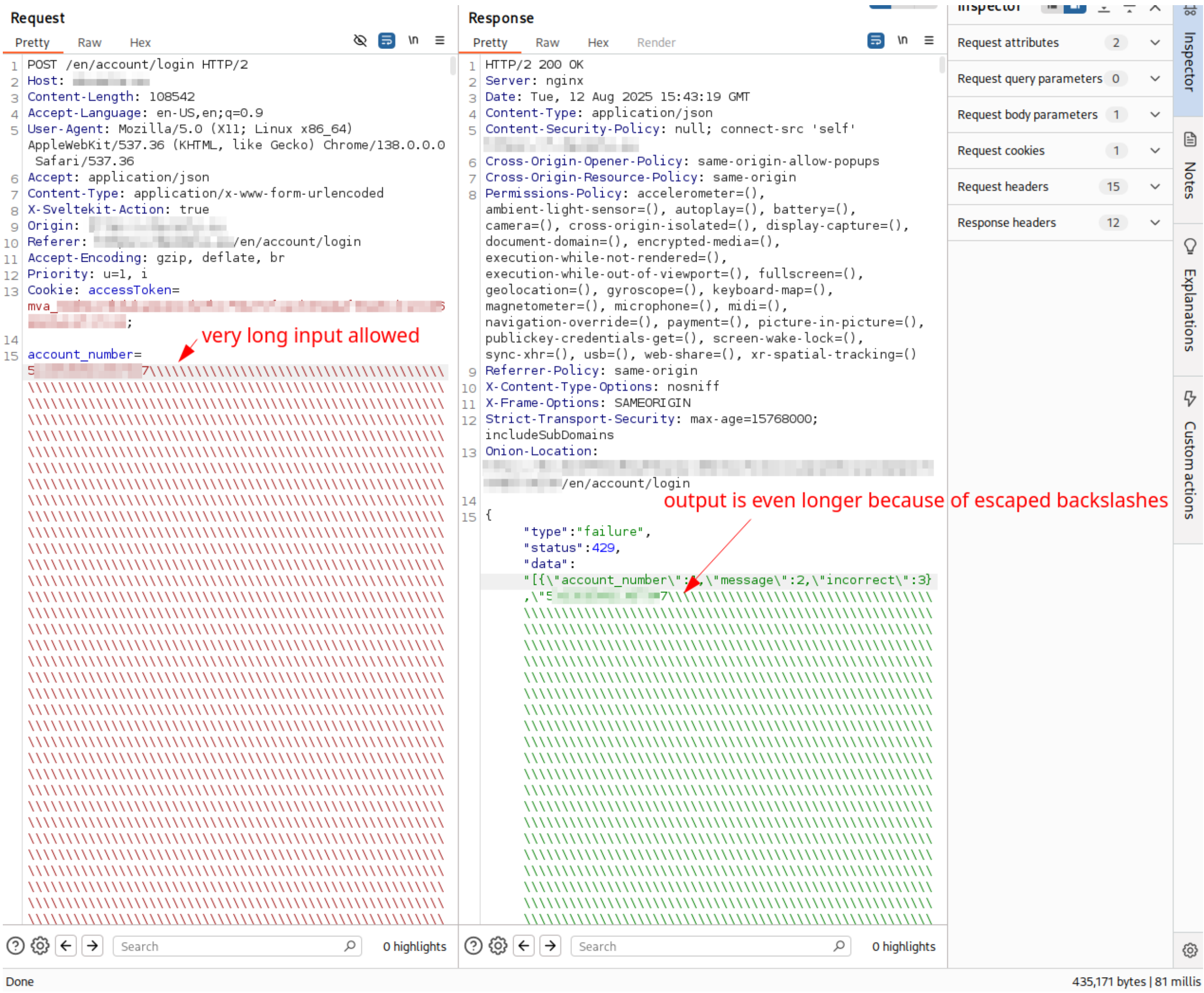

Figure 1: Very large account number can be submitted and will be reflected in error response

3.1.6 ACCEPTED NOTE Unhandled error for faulty Host header

Verification note: Issue is in SvelteKit and would require some effort to fix. Since it has no security impact, Mullvad consider it accepted.

Changing the Host header in a request to `POST /en/account/login` results in an unhandled Error in the client side code as can be seen in Figure 3.

The request gets an HTML error response, but the client side code tries to interpret the response as JSON data.

The edited request and the HTML response can be seen in Figure 4.

This issue does not have any security impact, but it is good security practice to handle error cases to avoid any potential issues in future versions of the code.

We recommend handling HTML error responses in the client side code.

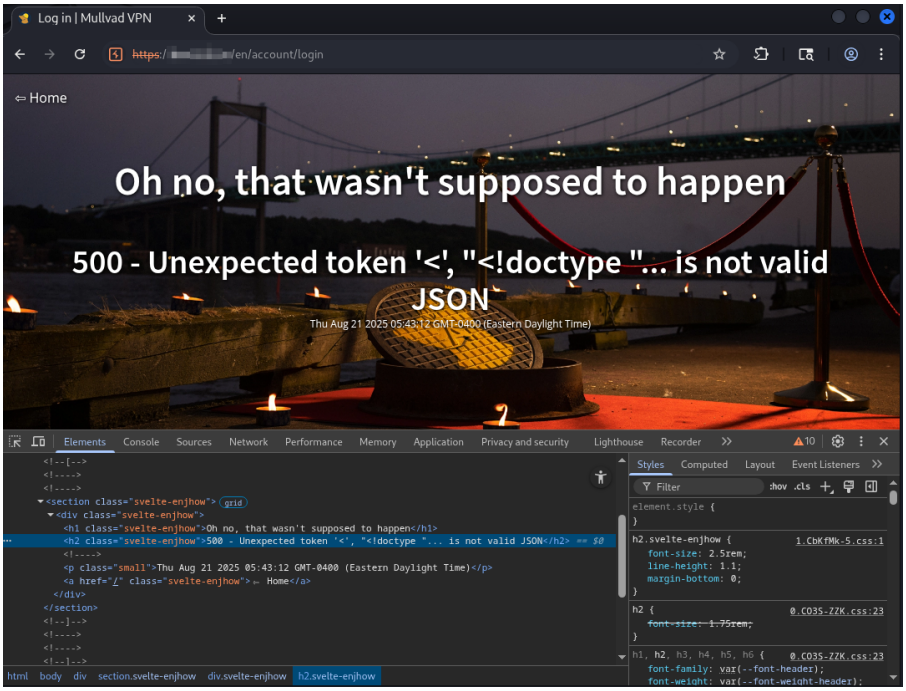

Figure 3: Error in browser when changing Host header in POST request for login

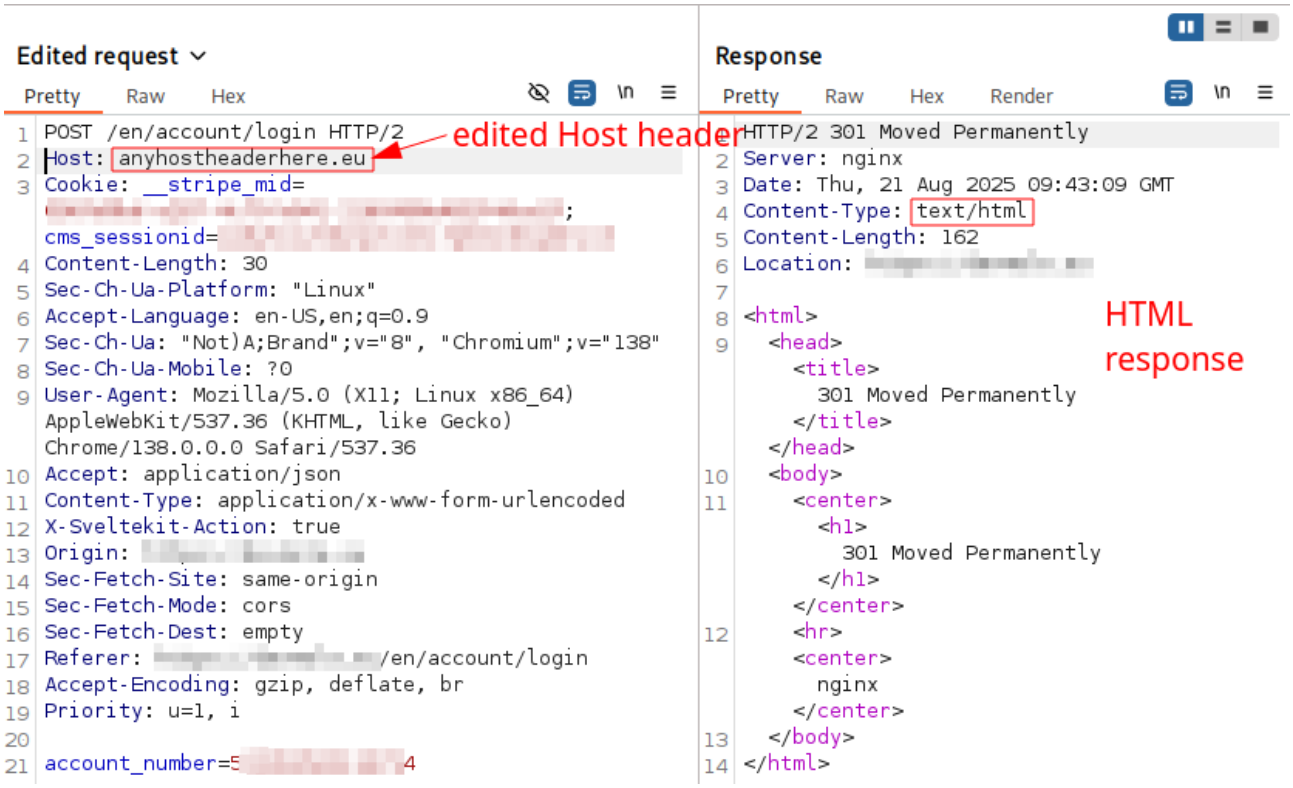

Figure 4: Request with edited Host header and HTML response causing the client side error

3.1.7 FIXED NOTE Unhandled error for Content-Type text/plain

Verification note: The application now returns `415 Unsupported Media Type` for `Content-Type: text/html` as for all other unsupported content types. The issue is considered fixed.

Using `Content-Type: text/html` in POST requests generates a `500 Internal Server Error` response.

The expected Content-Type for POST requests is `application/x-www-form-urlencoded`. All other content types generate a `415 Unsupported Media Type` error.

Figure 5 shows an example request and error response with the edited content type.

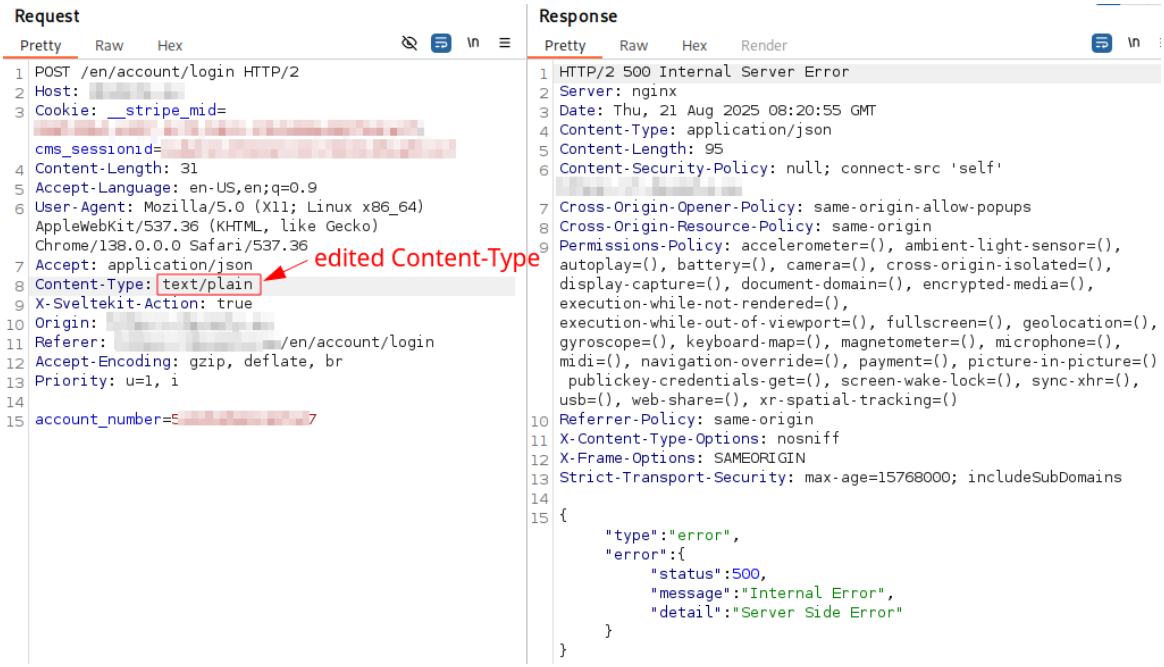

Figure 5: 500 Internal Server Error when setting Content-Type to text/plain

This issue does not really have an impact, but the error indicates that this case was not expected and could possibly cause trouble in future versions of the application.

We recommend implementing error handling for POST requests with the text/plain (or any unexpected) content type.

3.2 Mullvad.net CMS Web Application

The content management interface for the Mullvad VPN web application is a Django application that allows content administrators to manage the blog, help guides and similar articles.

3.2.1 GOOD Production CMS blocked from Internet/VPN

The production CMS web application is inaccessible both over Mullvad VPN and over Tor. Having administrative interfaces exposed only on dedicated, authorized networks is considered good security practice.

3.2.2 GOOD Patch level and compactness

The Django CMS application runs on Django[3] 4.2.22, albeit not the latest major version, it is still up to date and not vulnerable to known vulnerabilities at the time (source: Snyk[4]).

After inspection of the source code, we deduce that the application shows good use of Django's features resulting in a compact, simple and efficient application.

3.2.3 FIXED NOTE Insufficient Cross Origin Resource Sharing (CORS) policy

Verification note: The response header `Access-Control-Allow-Origin: *` is now removed, which means that the Same Origin Policy is enforced. The issue is considered fixed.

A Cross-Origin Resource Sharing (CORS) header is configured in the web server which allows resource sharing of CMS static content with any domain.

The server returned `Access-Control-Allow-Origin: *` for static content routes, meaning that any origin can embed static resources from the CMS application. Authenticated endpoints will not work though, as the server does not serve a `Access-Control-Allow-Credentials` header, and non-static resources are served with a stricter policy, which is good and makes this observation merely informational.

We recommend restricting the allowed origins in accordance with best practices. Typically, a strict allow-list of domains is preferred. If any type of pattern matching is needed (e.g. to allow wildcard subdomains), care must be taken to not accidentally match arbitrary domains.

[3] `https://www.djangoproject.com/`
[4] `https://security.snyk.io/package/pip/django/4.2`

3.2.4 FIXED NOTE Dev CMS: Onion service accessible but non-functional

Verification note: The path to the CMS application is now blocked when requesting it via the Onion service. The observation is considered fixed.

The Onion service of the staging/development CMS application is accessible over the Tor network (see Figure 6), however it is seemingly unusable due to failing CSRF check.

We recommend disallowing CMS access over Tor.

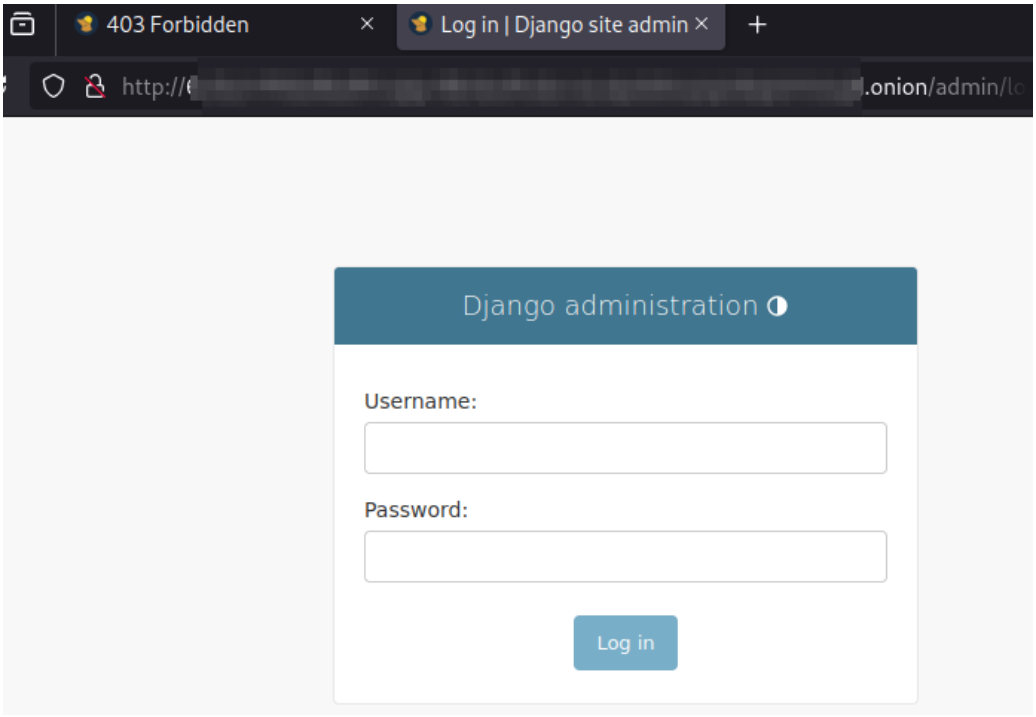

Figure 6: CMS (Dev) accessible over Tor/Onion service

4 Observations and coverage

The tables in this section cover the OWASP Web Security Testing Guide tests as in the latest version at the time of writing this report [1].

Status codes for each test are defined as:

- ”Pass”
- ”Fail” (issues found)
- ”N/A” (not applicable for this application)
- ”-” (test could not be fully carried out due to time constraint, missing requisites or being out of scope for this test)

We may have findings even for items that pass tests.

Section/Item	Status	Note
WSTG-INFO Information Gathering		
WSTG-INFO-01 Conduct Search Engine Discovery and Reconnaissance for Information Leakage	Pass	
WSTG-INFO-02 Fingerprint Web Server	Pass	
WSTG-INFO-03 Review Webserver Metafiles for Information Leakage	Pass	
WSTG-INFO-04 Enumerate Applications on Webserver	Pass	
WSTG-INFO-05 Review Webpage Content for Information Leakage	Pass	
WSTG-INFO-06 Identify Application Entry Points	Pass	
WSTG-INFO-07 Map Execution Paths Through Application	Pass	
WSTG-INFO-08 Fingerprint Web Application Framework	Pass	
WSTG-INFO-09 Fingerprint Web Application	Pass	
WSTG-INFO-10 Map Application Architecture	Pass	
WSTG-CONF Configuration and Deploy Management Testing		
WSTG-CONF-01 Test Network Infrastructure Configuration	-	
WSTG-CONF-02 Test Application Platform Configuration	Pass	
WSTG-CONF-03 Test File Extensions Handling for Sensitive Information	Pass	
WSTG-CONF-04 Review Old Backup and Unreferenced Files for Sensitive Information	Pass	
WSTG-CONF-05 Enumerate Infrastructure and Application Admin Interfaces	Pass	
WSTG-CONF-06 Test HTTP Methods	Pass	
WSTG-CONF-07 Test HTTP Strict Transport Security	Pass	
WSTG-CONF-08 Test RIA Cross Domain Policy	N/A	
WSTG-CONF-09 Test File Permission	-	
WSTG-CONF-10 Test for Subdomain Takeover	-	
WSTG-CONF-11 Test Cloud Storage	N/A	
WSTG-CONF-12 Testing for Content Security Policy	Pass	
WSTG-CONF-13 Test Path Confusion	Pass	
WSTG-IDNT Identity Management Testing		
WSTG-IDNT-01 Test Role Definitions	N/A	
WSTG-IDNT-02 Test User Registration Process	-	
WSTG-IDNT-03 Test Account Provisioning Process	N/A	
WSTG-IDNT-04 Testing for Account Enumeration and Guessable User Account	-	
WSTG-IDNT-05 Testing for Weak or Unenforced Username Policy	N/A	
WSTG-ATHN Authentication Testing		
WSTG-ATHN-01 Testing for Credentials Transported over an Encrypted Channel	Pass	
WSTG-ATHN-02 Testing for Default Credentials	Pass	
WSTG-ATHN-03 Testing for Weak Lock Out Mechanism	Pass	
WSTG-ATHN-04 Testing for Bypassing Authentication Schema	Pass	
WSTG-ATHN-05 Testing for Vulnerable Remember Password	N/A	
WSTG-ATHN-06 Testing for Browser Cache Weakness	Pass	
WSTG-ATHN-07 Testing for Weak Password Policy	N/A	
WSTG-ATHN-08 Testing for Weak Security Question Answer	N/A	

Section/Item	Status	Note
WSTG-ATHN-09 Testing for Weak Password Change or Reset Functionalities	-	
WSTG-ATHN-10 Testing for Weaker Authentication in Alternative Channel	Pass	
WSTG-ATHN-11 Testing Multi-Factor Authentication (MFA)	N/A	
WSTG-ATHZ Authorization Testing		
WSTG-ATHZ-01 Testing Directory Traversal File Include	Pass	
WSTG-ATHZ-02 Testing for Bypassing Authorization Schema	Pass	
WSTG-ATHZ-03 Testing for Privilege Escalation	Pass	
WSTG-ATHZ-04 Testing for Insecure Direct Object References	Pass	
WSTG-ATHZ-05 Testing for OAuth Weaknesses	N/A	
WSTG-SESS Session Management Testing		
WSTG-SESS-01 Testing for Session Management Schema	Pass	
WSTG-SESS-02 Testing for Cookies Attributes	Pass	
WSTG-SESS-03 Testing for Session Fixation	Pass	
WSTG-SESS-04 Testing for Exposed Session Variables	Pass	
WSTG-SESS-05 Testing for Cross Site Request Forgery	Pass	
WSTG-SESS-06 Testing for Logout Functionality	Pass	
WSTG-SESS-07 Testing Session Timeout	Pass	
WSTG-SESS-08 Testing for Session Puzzling	Pass	
WSTG-SESS-09 Testing for Session Hijacking	Pass	
WSTG-SESS-10 Testing JSON Web Tokens	N/A	
WSTG-INPV Input Validation Testing		
WSTG-INPV-01 Testing for Reflected Cross Site Scripting	Pass	
WSTG-INPV-02 Testing for Stored Cross Site Scripting	Pass	
WSTG-INPV-03 Testing for HTTP Verb Tampering	Pass	
WSTG-INPV-04 Testing for HTTP Parameter pollution	Pass	
WSTG-INPV-05 Testing for SQL Injection	Pass	
WSTG-INPV-06 Testing for LDAP Injection	N/A	
WSTG-INPV-07 Testing for XML Injection	N/A	
WSTG-INPV-08 Testing for SSI Injection	N/A	
WSTG-INPV-09 Testing for XPath Injection	N/A	
WSTG-INPV-10 Testing for IMAP SMTP Injection	N/A	
WSTG-INPV-11 Testing for Code Injection	-	
WSTG-INPV-12 Testing for Command Injection	-	
WSTG-INPV-13 Testing for Format String Injection	-	
WSTG-INPV-14 Testing for Incubated Vulnerabilities	-	
WSTG-INPV-15 Testing for HTTP Splitting Smuggling	Pass	
WSTG-INPV-16 Testing for HTTP Incoming Requests	-	
WSTG-INPV-17 Testing for Host Header Injection	Pass	Note: 3.1.6
WSTG-INPV-18 Testing for Server-Side Template Injection	Pass	
WSTG-INPV-19 Testing for Server-Side Request Forgery	-	
WSTG-INPV-20 Testing for Mass Assignment	-	
WSTG-ERRH Error Handling		
WSTG-ERRH-01 Testing for Improper Error Handling	Pass	Note: 3.1.5, 3.1.6, 3.1.7
WSTG-ERRH-02 Testing for Stack Traces	Pass	
WSTG-CRYP Cryptography		
WSTG-CRYP-01 Testing for Weak Transport Layer Security	Pass	
WSTG-CRYP-02 Testing for Padding Oracle	N/A	
WSTG-CRYP-03 Testing for Sensitive Information Sent Via Unencrypted Channels	Pass	
WSTG-CRYP-04 Testing for Weak Encryption	Pass	
WSTG-BUSLOGIC Business Logic Testing		
WSTG-BUSL-01 Test Business Logic Data Validation	Pass	(Fixed) Low:3.1.4
WSTG-BUSL-02 Test Ability to Forge Requests	Pass	
WSTG-BUSL-03 Test Integrity Checks	Pass	
WSTG-BUSL-04 Test for Process Timing	-	
WSTG-BUSL-05 Test Number of Times a Function Can Be Used Limits	Pass	
WSTG-BUSL-06 Testing for the Circumvention of Work Flows	-	
WSTG-BUSL-07 Test Defenses Against Application Misuse	Pass	
WSTG-BUSL-08 Test Upload of Unexpected File Types	-	
WSTG-BUSL-09 Test Upload of Malicious Files	-	
WSTG-BUSL-10 Test Payment Functionality	-	
WSTG-CLIENT Client-side Testing		

Section/Item	Status	Note
WSTG-CLNT-01 Testing for DOM Based Cross Site Scripting	Pass	
WSTG-CLNT-02 Testing for JavaScript Execution	Pass	
WSTG-CLNT-03 Testing for HTML Injection	Pass	
WSTG-CLNT-04 Testing for Client-Side URL Redirect	Pass	
WSTG-CLNT-05 Testing for CSS Injection	Pass	
WSTG-CLNT-06 Testing for Client-Side Resource Manipulation	Pass	
WSTG-CLNT-07 Test Cross Origin Resource Sharing	Pass	Note: 3.2.3
WSTG-CLNT-08 Testing for Cross Site Flashing	N/A	
WSTG-CLNT-09 Testing for Clickjacking	Pass	
WSTG-CLNT-10 Testing WebSockets	N/A	
WSTG-CLNT-11 Test Web Messaging	N/A	
WSTG-CLNT-12 Test Browser Storage	Pass	
WSTG-CLNT-13 Testing for Cross Site Script Inclusion	Pass	
WSTG-CLNT-14 Testing for Reverse Tabnabbing	N/A	
WSTG-APIT API Testing		
WSTG-APIT-01 Testing GraphQL	N/A	

5 Conclusions and recommendations

Assured Security Consultants performed a web application penetration test of the Mullvad VPN application and the CMS application in the development environment. In addition, the Tor service setup and an rsync service was reviewed.

Only minor issues were observed, where only one was considered to be of low security risk and the others added as notes, where the latter do not have any security impact, but can be good to know about. In essence, the developers may want to improve on error handling and application behaviour.

Good security practice is followed in all parts of the reviewed web applications and also in the additionally reviewed services for access to the web application over Tor and rsync of run-time created static content.

Our recommendations can be summarized as follows:

- Enforce input length limitations server-side, before forwarding to internal APIs.
- Review the cases where a *null* Content Security Policy is returned, and consider handling these cases either by correcting or by removing the policy, depending on applicability.
- Add handling of the case where an HTML error response causes a client side error.
- Add handling of the case when an unexpected Content-Type is used in a POST request.

Recommendations for the CMS web application:

- Restrict the allowed origins for the CMS application in accordance with best practices.
- Consider disallowing access to the CMS application over Tor (in staging/development).

The result of the verification test was that all observation were fixed according to the recommendations, except one Note observation which Mullvad decide to accept without any fix due to the issue being in Sveltekit. This Note observation does not have any security impact, and thus we have no further recommendations.

References

[1] OWASP, "OWASP Web Security Testing Guide (latest)."
`https://owasp.org/www-project-web-security-testing-guide/latest/`, 2023.

[2] OWASP, "OWASP Risk Rating Methodology."
`https://owasp.org/www-community/OWASP_Risk_Rating_Methodology`, 2023.