ASSURED

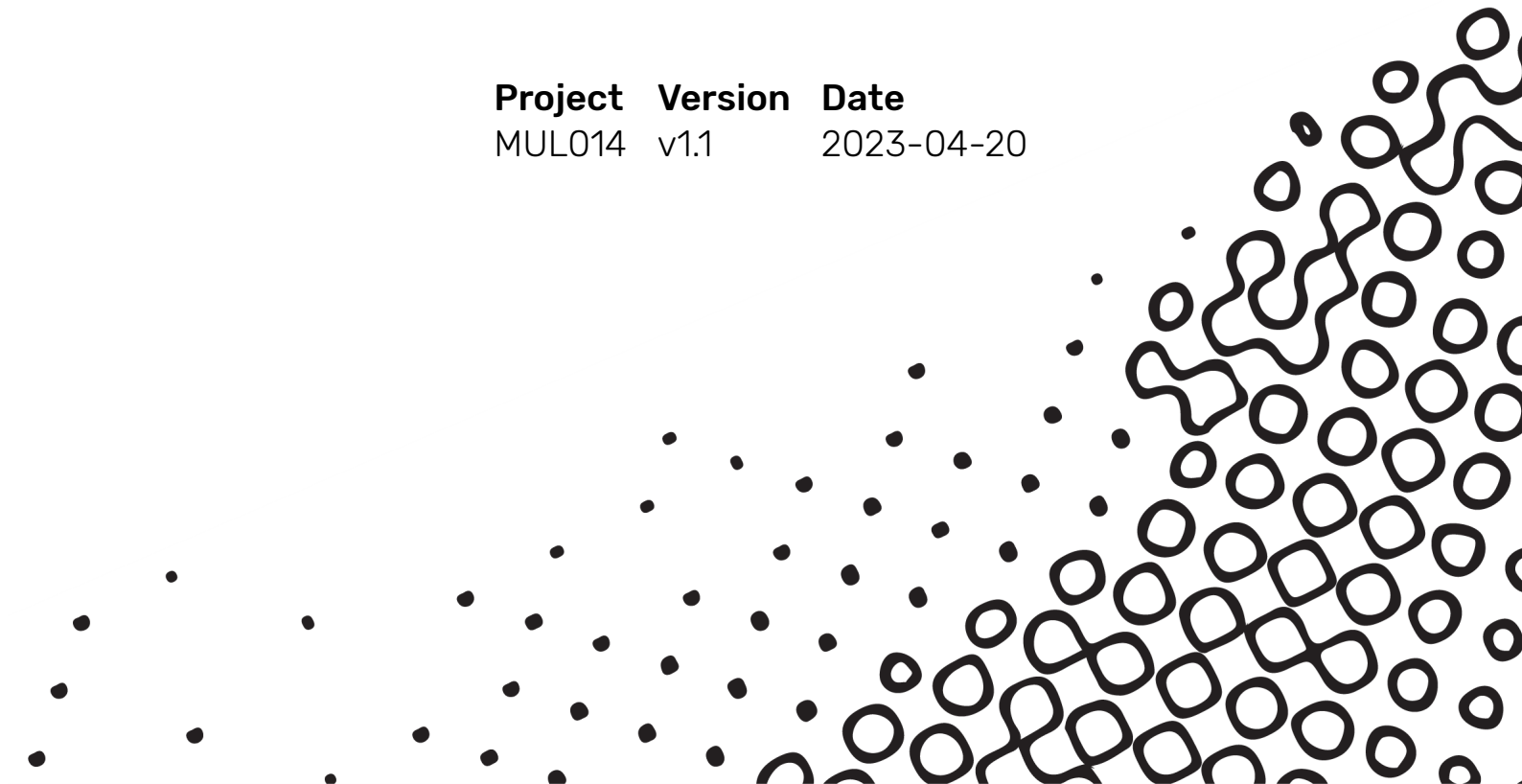
SECURITY CONSULTANTS

Report

Mullvad Leta Penetration Test

Albin Eldstål-Ahrens, Alexander Alasjö

Project	Version	Date
MUL014	v1.1	2023-04-20



ASSURED
SECURITY CONSULTANTS

REPORT

Project	Version	Date
MUL014	v1.1	2023-04-20

Executive summary

Between 2023-03-27 and 2023-03-31 Assured Security Consultants performed a web application penetration test on "Leta", a Google search proxy implementation by Mullvad VPN. Mullvad Leta (staging and production) was in scope. The test focused on two primary properties of the application: security and user privacy. Testing was carried out in accordance with OWASP Web Security Testing Guide.

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

Issues were found with the following risk severity assessments (number of issues):

The penetration test found two potential user privacy issues related to logging and search query caching. We also found a potential security issue with Google search results' HTML content being rendered in the Leta search results, with the risk of Cross-Site Scripting should the Google custom search API be manipulated to include harmful HTML or JavaScript.

Overall, Mullvad Leta is well contained with a small attack surface and good measures have been implemented to strengthen privacy as well as security.

Our recommendations for mitigating the identified security and/or privacy issues can be summarized as follows:

- Implement a Content Security Policy adhering to the principle of least privilege to avoid having untrusted third-party content injected into Leta.
- Consider generating search results based on plain text descriptions from the Google custom search API.
- Properly set the `Strict-Transport-Security` header.
- Restrict unnecessary logging of user data in production environments.
- Consider storing search terms in a hashed form to limit exposure of sensitive queries should the cache storage leak.
- Consider clearing cached search entries after expiration.

Assured would like to thank Mullvad representatives Hank, Oskar and Josh 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
MUL014	v1.1	2023-04-20

Contents

1 Introduction

1.1 Background

Assured AB (Assured) was contracted to perform a web application penetration test and web server security review of "Leta", a Google search proxy implemented by Mullvad VPN.

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 2023-03-27 and finished on 2023-03-31.

This report was last reviewed on 2023-04-20.

Involved in the penetration testing were Assured consultants Albin Eldstål-Ahrens and Alexander Alasjö.

1.4 Risk rating

1.4.1 OWASP Risk Rating Methodology

In this report we have assessed the severity of issues and identified vulnerabilities according to the OWASP Risk Rating Methodology [1].

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 determined 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 Scope and methodology

2.1 Scope

2.1.1 Penetration test of Mullvad Leta

The web application targets in scope were as follows:

- `https://leta.mullvad.net` (production)
- Leta, in a staging environment.

Source code, user accounts and root access to the staging web server were provided to the testers by Mullvad.

2.2 Methodology

2.2.1 Web application penetration test

Testing was carried out in accordance with the OWASP Web Security Testing Guide (WSTG, latest version) [2]. A complete WSTG checklist is included in Section 4.

2.2.2 Tools used

- Burp Suite Professional
- cURL
- subfinder
- ent
- testssl.sh

3 Observations

These are the observations made during the penetration test of Mullvad Leta.

3.1 Configuration and Deployment Management Testing

3.1.1 Low Content Security Policy (CSP) missing

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

The web application is served without an explicit Content Security Policy (CSP) [3]. CSP is a way for web applications to instruct the client to restrict access to dynamic content such as scripts and static content such as media, based on their origin. It can also limit which origins may embed the web page in an `iframe`. The purpose of this is to prevent the exploitation of script injection (XSS) vulnerabilities found elsewhere in the application, to prevent clickjacking, and to be able to report violations of the CSP. OWASP provides good practices on CSP in their cheat sheet series [4].

A missing CSP means the user's browser is allowed to include scripts and media from all sources, including external domains and scripts embedded in the document. The CSP presents an additional safeguard against script injections and other cross-origin interaction.

An optimal CSP allows only the application's own scripts. In addition, a *nonce* can be employed to further limit the possibility of including malicious scripts. The CSP can be implemented as a `Content-Security-Policy` header or a `meta http-equiv` tag.

We recommend configuring a Content Security Policy (CSP) for all documents, adhering to the principle of least privilege.

3.1.2 Low Partial logging of unique user ID

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

Account numbers and unique user identifiers are redacted when passed to the logging facility. The redaction removes all but the last four characters of either ID.

In production, the log level is set to INFO. At this level, a unique user UUID is logged when a user reaches their daily quota. The logged UUID is redacted, as shown in Example 1.

Example 1: Excerpt from application log

```
[08:03:42.027] INFO: User reached query quota
    userId: "********-****-****-****-********3c24"
```

The account number is only logged at the DEBUG level, and with similar redaction. DEBUG log entries are disabled in production. With four hexadecimal digits of a UUID visible, the

maximum entropy of the revealed data is 16 bits, i.e. 65 536 different combinations. This may not be sufficiently anonymized, in case log data is accessed by an unauthorized party.

We recommend disabling user identifiable log entries entirely in production, and removing the debug calls as soon as the product is ready for release. This is a preemptive measure to prevent accidental exposure in the future.

3.1.3 Note HTTP Strict Transport Security Header Missing

The responses from the web server do not contain a `Strict-Transport-Security` header. HTTP Strict Transport Security (HSTS) is a feature for browsers and HTTP clients to record whether a site or service should only be accessed over HTTPS. Once a service has been accessed over HTTPS with a `Strict-Transport-Security` response header, the client should remember this feature and henceforth never try to access the service over plain HTTP. See the OWASP Cheat Sheet on HSTS [5] for more information.

In cases where a sensitive cookie or header value is passed over plain HTTP, the sensitive content could be leaked to a network-adjacent attacker. The attacker could be passively sniffing for HTTP requests or use tools such as `sslstrip` to force clients to connect over plain HTTP to perform man-in-the-middle attacks.

While Leta does not serve any content over plain HTTP and the authentication cookie is only transmitted over HTTPS, it is still advised to serve relevant security headers. We noticed an intent to serve the HSTS header in the staging server's Nginx site configuration, but an edge-case misconfiguration omitted the header from the server's responses (for all but `TRACE` requests).

We recommend ensuring that the `Strict-Transport-Security` response header is properly set as it is good practice to serve this header to inform clients that they should only connect to the server over TLS (HTTPS).

3.2 Client-side Testing

3.2.1 Low Potential Cross-Site Scripting (XSS) via Google results

Likelihood: LOW (2), Impact: MEDIUM (4)

HTML snippets received from the Google API are inserted in the client side document without sanitization, as seen in Example 2. Should the HTML contain JavaScript elements, they will execute in the user's context as if they were received from the Leta domain.

While it is not likely that the Google custom search response API should contain harmful JavaScript or HTML, it cannot be ruled out: a recently published vulnerability in a Microsoft Azure service allowed anyone to change Bing search results for certain queries indeed resulting in Cross-Site Scripting in the Bing search results[1].

Example 2: SearchResults.svelte, raw HTML output from Google search result

```
28 {#if item.htmlSnippet}
29     <p>{@html item.htmlSnippet}</p>
30 {/if}
```

The potential impact of malicious JavaScript is limited in this application. The authentication token cookie is properly protected from JavaScript access, and the service itself exposes very few API endpoints. One possible attack is to silently redirect the user to a phishing page and trick them into providing their Mullvad account number to a third party.

We recommend using only the plain-text description from the Google results, rather than trusting HTML from an external party. A well-crafted CSP (see Finding 3.1.1) could also mitigate this issue to some extent.

[1] https://www.wiz.io/blog/azure-active-directory-bing-misconfiguration

3.3 Business Logic Testing

3.3.1 Note Search terms never removed from cache

All search queries and results are added to a cache database automatically. A limited time-to-live (TTL) is set on cached results, to avoid returning stale cached results on new queries. If a user specifies the option to only search in cached results, this TTL is ignored. No mechanism exists to remove expired searches and results from the database.

In the event of a security breach, an attacker is able to extract a history of search terms since the database was last restarted or purged. Each search entry is also indirectly associated with the timestamp of its last use (i.e. one TTL before the stored expiration date).

A more accessible avenue of enumeration is to query the API for cached results. This allows any user to determine if an exact search term is present in the cache, but does not reveal the last time of use.

We recommend setting a hard expiration time for new entries, and clearing entries from the database upon expiration. The built-in expiration mechanism of redis is already used to purge each user's quota entries at the end of each day, and should be suitable and robust for this purpose as well. If the presence of search terms (e.g. personally identifiable terms) is considered sensitive, we also recommend allowing users to exempt their searches from caching.

3.4 Other findings

3.4.1 Note Plaintext search queries in cache database

The search feature includes a redis database containing a cache of recent searches and their results. While the search is not mapped to any user, the search terms themselves may be considered sensitive information in some cases.

In the event of a security breach, an attacker is able to recover search terms and their associated time stamps.

We recommend hashing search terms before insertion/lookup in the cache database. Since search term cache lookups are only performed with exact matching, this should not affect functionality.

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 [2].

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
Information Gathering		
WSTG-INFO-01 Conduct Search Engine Discovery 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	N/A	
WSTG-INFO-10 Map Application Architecture	Pass	
Configuration and Deployment Management Testing		
WSTG-CONF-01 Test Network Infrastructure Configuration	Pass	
WSTG-CONF-02 Test Application Platform Configuration	Fail	3.1.2
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	Pass	
WSTG-CONF-10 Test for Subdomain Takeover	Pass	
WSTG-CONF-11 Test Cloud Storage	N/A	
WSTG-CONF-12 Testing for Content Security Policy	Fail	3.1.1
WSTG-CONF-13 Test Path Confusion	Pass	
Identity Management Testing		
WSTG-IDNT-01 Test Role Definitions	N/A	
WSTG-IDNT-02 Test User Registration Process	N/A	
WSTG-IDNT-03 Test Account Provisioning Process	N/A	
WSTG-IDNT-04 Testing for Account Enumeration and Guessable User Account	Fail	Accepted
WSTG-IDNT-05 Testing for Weak or Unenforced Username Policy	N/A	
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	N/A	
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 Weaknesses	Pass	
WSTG-ATHN-07 Testing for Weak Password Policy	N/A	
WSTG-ATHN-08 Testing for Weak Security Question Answer	N/A	
WSTG-ATHN-09 Testing for Weak Password Change or Reset Functionalities	N/A	

Section/Item	Status	Note
WSTG-ATHN-10 Testing for Weaker Authentication in Alternative Channel	Pass	
WSTG-ATHN-11 Testing Multi-Factor Authentication (MFA)	N/A	
Authorization Testing		
WSTG-ATHZ-01 Testing Directory Traversal File Include	N/A	
WSTG-ATHZ-02 Testing for Bypassing Authorization Schema	Pass	
WSTG-ATHZ-03 Testing for Privilege Escalation	N/A	
WSTG-ATHZ-04 Testing for Insecure Direct Object References	Pass	
WSTG-ATHZ-05 Testing for OAuth Weaknesses	N/A	
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	
Input Validation Testing		
WSTG-INPV-01 Testing for Reflected Cross Site Scripting	Pass	
WSTG-INPV-02 Testing for Stored Cross Site Scripting	Fail	3.2.1
WSTG-INPV-03 Testing for HTTP Verb Tampering	N/A	
WSTG-INPV-04 Testing for HTTP Parameter Pollution	Pass	
WSTG-INPV-05 Testing for SQL Injection	N/A	
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	Pass	
WSTG-INPV-12 Testing for Command Injection	Pass	
WSTG-INPV-13 Testing for Buffer Overflow	N/A	
WSTG-INPV-13 Testing for Format String Injection	Pass	
WSTG-INPV-14 Testing for Incubated Vulnerability	Pass	
WSTG-INPV-15 Testing for HTTP Splitting Smuggling	Pass	
WSTG-INPV-16 Testing for HTTP Incoming Requests	Pass	
WSTG-INPV-17 Testing for Host Header Injection	Pass	
WSTG-INPV-18 Testing for Server-side Template Injection	Pass	
WSTG-INPV-19 Testing for Server-Side Request Forgery	Pass	
WSTG-INPV-20 Testing for Mass Assignment	Pass	
Testing for Error Handling		
WSTG-ERRH-01 Testing for Improper Error Handling	Pass	
WSTG-ERRH-02 Testing for Stack Traces	Pass	
Testing for Weak 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	
Business Logic Testing		
WSTG-BUSL-01 Test Business Logic Data Validation	Pass	
WSTG-BUSL-02 Test Ability to Forge Requests	Pass	
WSTG-BUSL-03 Test Integrity Checks	Pass	
WSTG-BUSL-04 Test for Process Timing	Pass	
WSTG-BUSL-05 Test Number of Times a Function Can Be Used Limits	Pass	
WSTG-BUSL-06 Testing for the Circumvention of Work Flows	Pass	
WSTG-BUSL-07 Test Defenses Against Application Misuse	Fail	3.3.1
WSTG-BUSL-08 Test Upload of Unexpected File Types	N/A	
WSTG-BUSL-09 Test Upload of Malicious Files	N/A	
WSTG-BUSL-10 Test Payment Functionality	N/A	
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	Fail	3.2.1
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 Testing Cross Origin Resource Sharing	N/A	
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 Testing Web Messaging	N/A	
WSTG-CLNT-12 Testing Browser Storage	Pass	
WSTG-CLNT-13 Testing for Cross Site Script Inclusion	Fail	3.2.1
WSTG-CLNT-14 Testing for Reverse Tabnabbing	Pass	
API Testing		
WSTG-APIT-01 Testing GraphQL	N/A	

5 Conclusions and recommendations

Assured was tasked with conducting a penetration test on Mullvad Leta and to assess the web application with regards to security and privacy. Overall, Mullvad Leta is well contained with a small attack surface and good measures have been implemented to strengthen privacy as well as security.

5.1 Privacy

As for privacy issues, we found only two potential issues. First, logged in users are assigned a random UUID, which is necessary in order to maintain the daily request quota. When the quota for a user is exceeded, four characters of this UUID are written to an application log, with the remainder anonymized. While the partial ID is not linked to search activity, it is also not likely to be necessary in the log files of production servers.

Second, the search query/result caching feature is mainly implemented to increase performance and reduce cost. As a side effect it reduces privacy as it is possible to search the cache for exact queries which may inform a user if and if so when a certain search term was originally searched by another user.

Two pieces of information are stored client-side: a random authentication token and the user's preference for cache-only searches. No client fingerprinting or tracking was found.

The application prevents leakage of referrer data by applying the `noreferrer` property to outgoing links. This prevents the link destination from detecting that Leta has been used.

The nginx web server is configured to explicitly disable both access and error logging, preventing user activity such as search queries and IP addresses from being included in these logs.

While the application inherits guessable user account numbers from Mullvad's other services, a known and accepted risk, we found that the login endpoint is rate limited and searches are rate limited while also covered by quota (credits per 24 hours). This greatly increases the time it would take an attacker to effectively enumerate valid account numbers, and the implication is not privacy related as there is no search history or personal information to deduce from knowing an account number.

5.2 Security

Regarding security issues, no known vulnerabilities were identified in the server architecture or among the application's dependencies.

A potential security issue resides in the fact that the search feature in Leta employs the

Google custom search API which returns HTML content rendered directly in the Leta search results, making Leta vulnerable to Cross-Site Scripting as the content is out of Mullvad's control. A proper Content Security Policy and security headers could mitigate eventual attempts to introduce dynamic content in Leta via Google search results, but these are currently missing or lacking.

Origin headers are used to prevent cross-site POST requests to the search API, which would otherwise be a viable target for user quota exhaustion.

The only cookie stored client-side is the authentication token. This authentication cookie is properly configured with "Secure", "HttpOnly" and "SameSite=Strict" set, and sessions are properly invalidated on logout.

While we found that the Strict Transport Security header is not present, except for `TRACE` requests, the server does not expose any plain HTTP service and the TLS configuration is otherwise sound.

5.3 Recommendations

Our recommendations for mitigating the identified security and/or privacy issues can be summarized as follows:

- Implement a Content Security Policy adhering to the principle of least privilege to avoid having untrusted third-party content injected into Leta.
- Consider generating search results based on plain text descriptions from the Google custom search API.
- Properly set the `Strict-Transport-Security` header.
- Restrict unnecessary logging of user data in production environments.
- Consider storing search terms in a hashed form to limit exposure of sensitive queries should the cache storage leak.
- Consider clearing cached search entries after expiration.

References

[1] OWASP, "OWASP Risk Rating Methodology." https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology, 2019.

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

[3] Mozilla, "Content Security Policy (CSP)." https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP, 2023.

[4] OWASP, "Content Security Policy Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html, 2021.

[5] OWASP, "HTTP Strict Transport Security - OWASP Cheat Sheet Series." https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html, 2022.