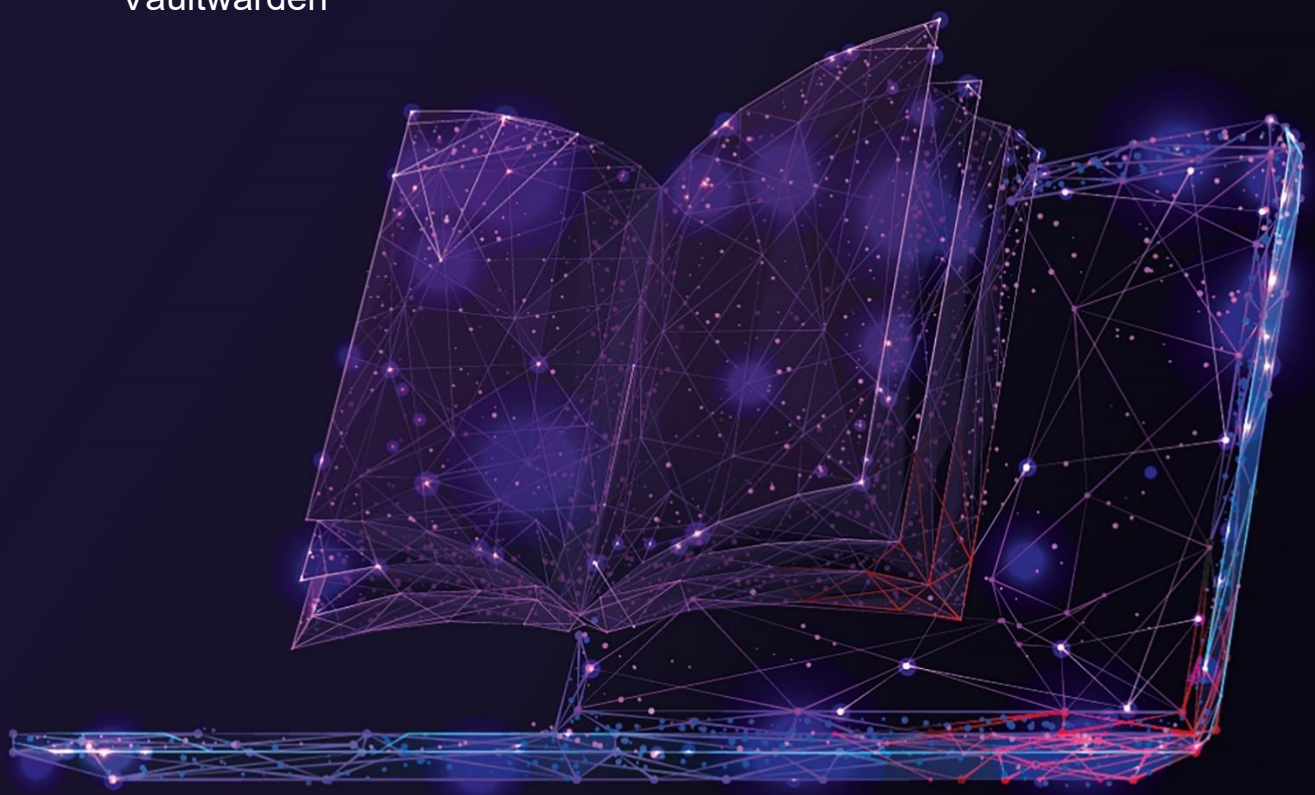
Bundesamt
für Sicherheit in der
Informationstechnik

Deutschland
Digital•Sicher•BSI•

Ergebnisse der Studie zur statischen Codeanalyse ausgewählter Opensource Software CAOS 3.0

Vaultwarden



mgm security partners

Ergebnisbericht
Security Quellcodeanalyse und Penetration Test

Projekt 604 – Codeanalyse für Opensource Software (CAOS3)

Analyseergebnisse Arbeitspaket 4: „Vaultwarden“

12.06.2024, Version 1.1, Status: Final

mgm security partners GmbH
Taunusstraße 23
80807 München
Tel.: +49/89/358680-880
E-Mail: info@mgm-sp.com
https://www.mgm-sp.com

A. Zusammenfassung und Bewertung

Sprungtabelle

Table 1: Sprungtabelle für die wichtigsten Bereiche in diesem Dokument

Auf den folgenden Seiten sind die Ergebnisse der Sicherheitsanalyse zusammengefasst.

Alle Verweise in diesem Dokument sind aktive Links, d. h. können per Mausklick direkt angesprungen werden.

A1 Inhaltsverzeichnis

A2 Management Summary

Der vorliegende Bericht fasst die Ergebnisse der Untersuchung der Applikation **Vaultwarden in der Version 1.30.3** und der **Bitwarden-Client Browser-Extension in der Version 2024.3.1** zusammen (Details siehe A4.1)

Die Anwendungen wurden einer umfassenden werkzeuggestützten statischen Quellcode-Analyse, sowie einer begleitenden dynamischen Analyse unterzogen (s. A4.4). Die Untersuchung erfolgte nach der Whitebox-Methode, bei der die Tester sowohl Zugriff auf den Code, als auch auf laufende Instanzen hatten.

Die Analysen fanden im Zeitraum vom 12.02.2024 bis zum 16.05.2024 in und aus den Räumlichkeiten des Auftragnehmers statt.

Die Vaultwarden-Serveranwendung weist je zwei

Sicherheitslücken mittlerer bzw. hoher Kritikalität

auf, mit denen ein Angreifer gezielt Benutzer und die Anwendung kompromittieren kann.

Vaultwarden sieht keinen Offboarding-Prozess für Mitglieder vor, die die Organisation verlassen. Das bedeutet, dass die für den Datenzugriff notwendigen Master-Schlüssel in diesem Fall nicht ausgetauscht werden. Folglich besitzt das ausscheidende Mitglied, dem der Zugang eigentlich entzogen wurde, weiterhin den kryptografischen Schlüssel zu den Daten der Organisation (E2.1). In Kombination mit der Schwachstelle, dass unberechtigt auf verschlüsselte Daten anderer Organisationen zugegriffen werden kann (E2.2), behält dieses ehemalige Mitglied somit weiterhin unberechtigten Zugriff auf alle (auch später erzeugten) Geheimnisse der entsprechenden Organisation im Klartext.

Weiterhin wird beim Ändern der Metadaten eines eingerichteten Notfall-Zugriffs die Berechtigung des zugreifenden Nutzers nicht überprüft (E1.1). Über den Endpunkt können die Bedingungen des eingerichteten Notfall-Zugriffs, einschließlich der Zugriffsebene und der Wartezeit, nachträglich geändert werden. Dadurch könnte ein Angreifer, dem auf diesem Wege Zugriff auf ein Konto gewährt wurde, sowohl auf die Daten des Kontos mit einer höheren Zugriffsstufe zugreifen, als auch die durch den Besitzer festgelegte Wartezeit (standardmäßig 7 Tage) beliebig verkürzen.

Das Admin-Dashboard ist anfällig für HTML-Injection-Angriffe. Durch das Einfügen von HTML-Tags ist es möglich, die Optik und die Inhalte der Seite zu verändern und z.B. Links zu bösartigen Seiten einzubetten, oder unter Umständen Skripte auszuführen (E3.1).

Die umfassende statische Quellcodeanalyse führte zu einem als [hoch] und 6 als [info] eingestuften SAST-, sowie einem als [hoch] eingestuftem SCA-Finding.
Das als [hoch] eingestufte SAST-Finding wurde im Nachgang per dynamischer Analyse verifiziert und bestätigt (E3.1).

Für die Bitwarden-Client Browser-Extension konnten, abgesehen von drei als [niedrig] und einem als [info] eingestuften SAST-Findings (A3.2), keine Schwachstellen identifiziert werden.

Hinweis: Für die vier oben erwähnten Sicherheitslücken mittlerer bzw. hoher Kritikalität wurden insgesamt drei CVE-Einträge (Common Vulnerabilities and Exposure) beantragt. Aufgrund der hohen Korrelation in der Ausnutzbarkeit der beiden Schwachstellen E2.1 und E2.2, wurde für die beiden Findings ein gemeinsamer CVE-Eintrag beantragt.
Zum Zeitpunkt der Fertigstellung des Berichts lagen noch keine CVE-Nummern vor.

A3 Übersicht aller Findings

A3.1 Toolbasierte Findings

Dem in B1.3 beschriebenen Vorgehen zur Bestimmung des Gefährdungspotentials folgend, werden in diesem Kapitel alle Ergebnisse automatischer Toolanalysen zusammengefasst dargestellt.

Hinweis: Die SARIF-Exporte unterscheiden sich in der Menge von den Findings im Excel-Report, da wir dort bestimmte Findings (Details dazu siehe Abschnitt D) von einer Bewertung ausgeschlossen haben.

A3.2 SAST-Analysen

Auf Basis der durchgeführten semiautomatischen SAST- und SECRETS-Analysen (s. Abschnitt D1) ergeben sich, nach Abbildung auf unser Gefährdungspotential, folgende Mengen an Findings:

Vaultwarden Server:

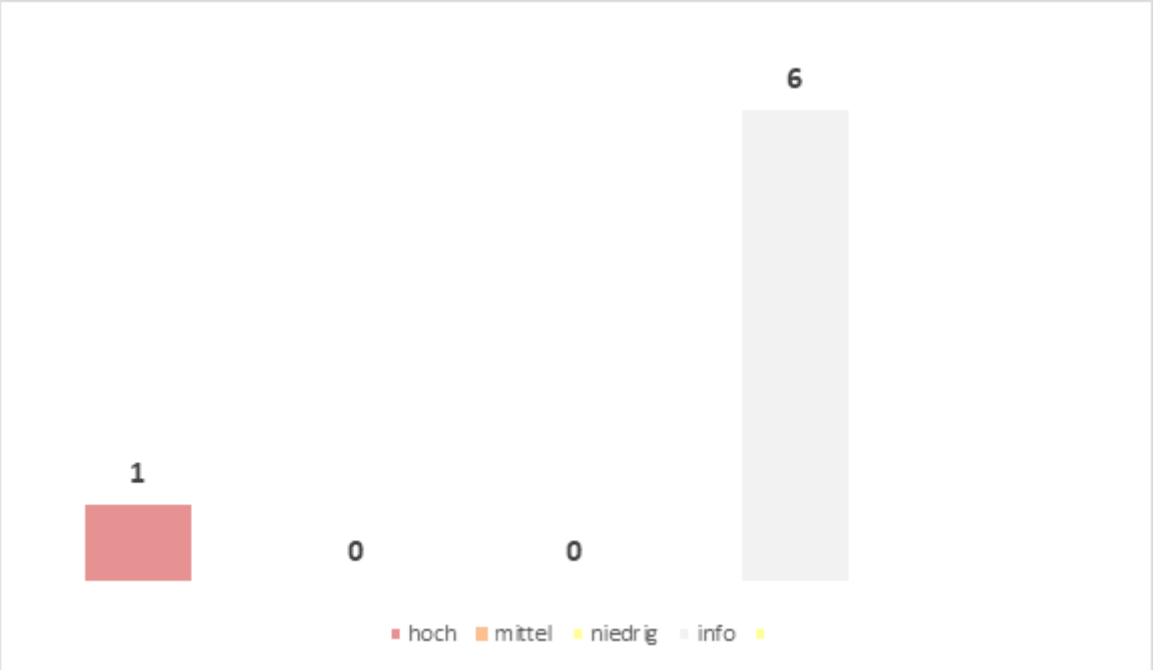


Abbildung 1: Übersicht SAST-Findings Vaultwarden-Server

Bitwarden-Client Browser-Extension:

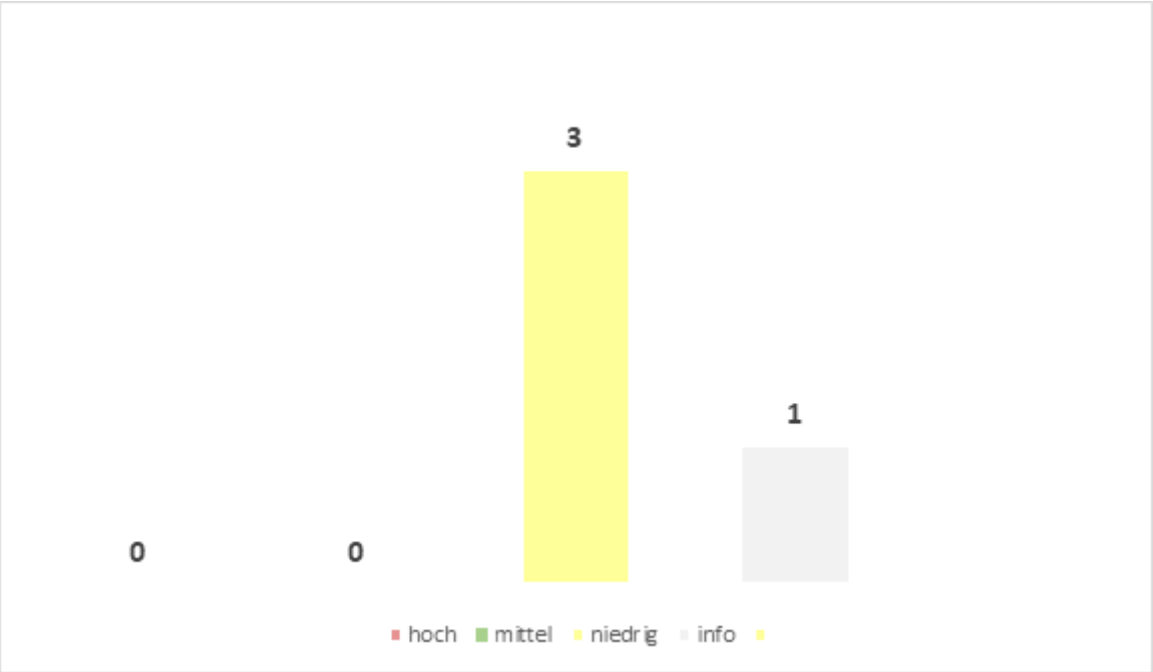


Abbildung 2: Übersicht SAST-Findings Bitwarden-Client Browser-Extension

A3.3 SCA-Analyse

Auf Basis der automatischen SCA-Analyse (s. Abschnitt D2) ergeben sich, nach Abbildung auf unser Gefährdungspotential, folgende Mengen an Findings für den Vaultwarden Server:

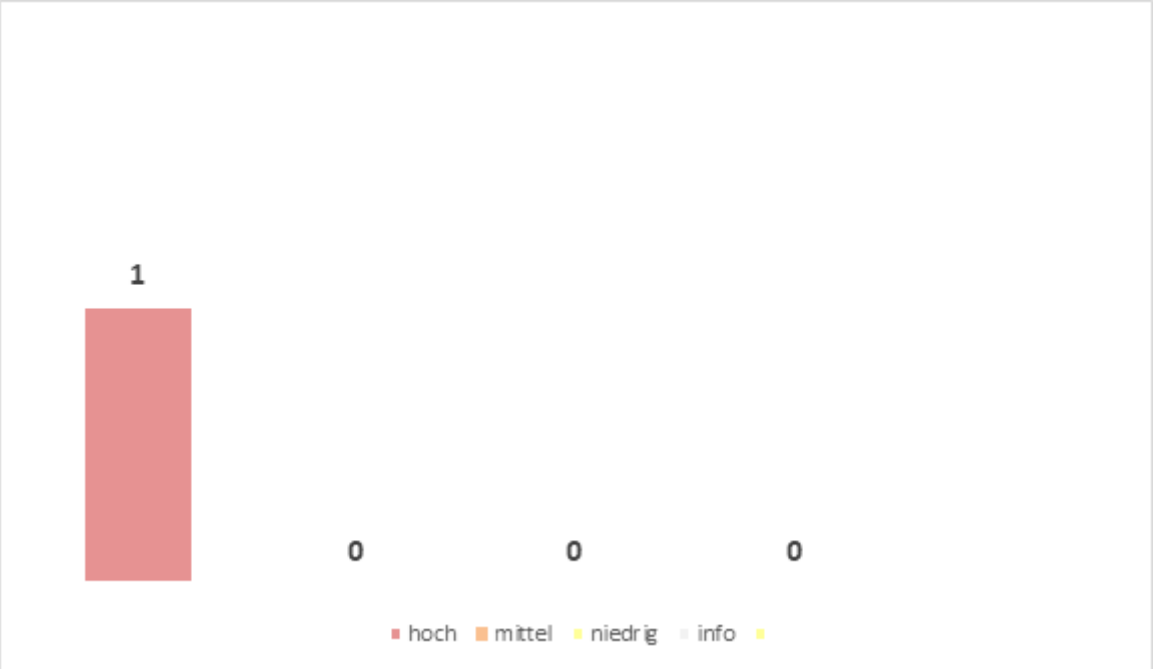


Abbildung 3: Übersicht SCA-Findings Vaultwarden-Server

Bei dem als [hoch] eingestuften SCA-Finding handelt es sich um keine direkte, sondern eine transitive Abhängigkeit.

Hinweis: Für die Bitwarden-Client Browser-Extension erfolgte keine SCA-Analyse, da die externen Abhängigkeiten direkt durch den jeweiligen Browser verwaltet und bereitgestellt werden.

A3.4 Findings mit ausführlicher Beschreibung in diesem Dokument

In der folgenden Übersicht sind die Bedrohungen und Schwachstellen aufgeführt, welche durch manuelle SAST- bzw. DAST-Ansätze aufgefunden wurden, bzw. bei den automatischen SAST-Analysen von uns mit dem Gefährdungspotential „kritisch“ oder „hoch“ eingeschätzt wurden.

Die folgende Übersicht listet dabei zunächst alle Schwachstellen, welche mit niedrig, mittel, oder hoch eingeschätzt wurden.

Table 2: Übersicht über ausführlich beschriebene Schwachstellen

Schwachstelle	[niedrig]	[mittel]	[hoch]
Authentisierung			
Unbefugte Änderung der Metadaten eines Emergency Access			E1.1
Zugriffskontrolle			
Fehlende Rotation der Organisationsschlüssel			E2.1
Unautorisierter Zugriff auf verschlüsselte Daten		E2.2	
Mögliche Härtungsmaßnahmen der Dock-erumgebung	E2.3		
Cross-Site-Scripting			
HTML-Injection möglich		E3.1	
Server-Side Request Forgery			
Server-Side-Request-Forgery möglich	E4.1		
Session-ID im URL oder Referrer			
Übertragung sensibler Daten in der URL	E5.1		
Logik			
Denial-of-Service: IP-basierte Benutzersper-rung möglich	E6.1		
Unzureichende Anti-Automatisierung	E6.2		
Offenlegung von Informationen			
Offenlegung von Informationen	E7.1		
Session-Management			
Unzureichende Cookie-Konfiguration im Admin-Dashboard	E8.1		
Datenvalidierung			

Schwachstelle	[niedrig]	[mittel]	[hoch]
Log-Injektion	E9.1		

In der folgenden Übersicht sind die Beaobachtungen aufgeführt, die das Gefährdungspotenzial [info] besitzen.

Table 3: Übersicht über ausführlich beschriebene info-Schwachstellen

[info]	Kapitel
Upload beliebiger Dateiformate möglich	E6.3

Übersicht

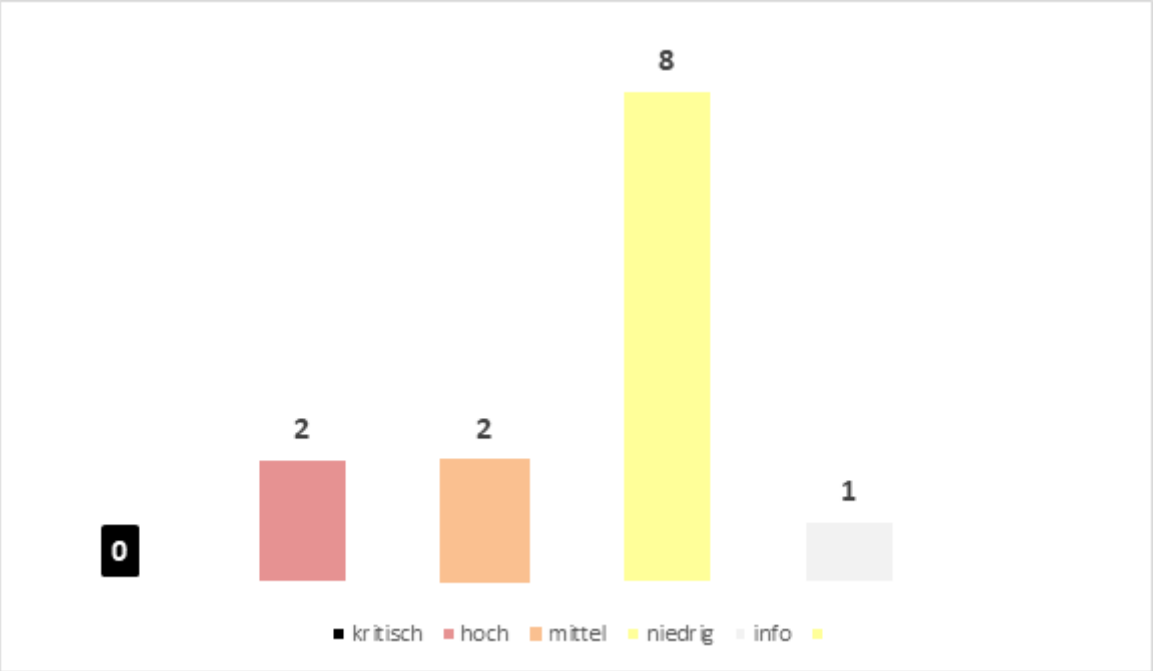


Abbildung 4: Statistik über Anzahl ausführlich beschriebener Findings im Dokument

Hinweis: In der Bitwarden-Client Browser-Extension wurden keine Schwachstellen festgestellt.

A4 Assessment Details

A4.1 Getestete Applikationen

Bei der vorliegenden Sicherheitsüberprüfung wurden folgende Komponenten, sowohl dediziert, als auch dynamisch in Kombination auf Schwachstellen hin untersucht:

1. Vaultwarden Server (https://github.com/dani-garcia/vaultwarden) unter Verwendung des Tags `v1.30.3` (https://github.com/dani-garcia/vaultwarden/releases/tag/1.30.3)
2. Bitwarden-Client Browser-Extension (Unterordner `apps/browser` von https://github.com/bitwarden/clients) unter Verwendung des Tags `browser-v2024.3.1` (https://github.com/bitwarden/clients/tree/browser-v2024.3.1)

Die zugrundeliegenden Mengengerüste der zum Einsatz kommenden Sprachen & Technologien (erstellt mit dem Werkzeug cloc - https://github.com/AlDanial/cloc) stellen sich folgendermaßen dar:

```
github.com/AlDanial/cloc v 1.98  T=1.70 s (159.3 files/s, 46524.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Rust                            52           3426           1692          21399
JavaScript                       9           4704           9716          18895
CSS                              4            917             48          11629
Handlebars                      62             68             60           1394
SQL                            107            145             47           1065
JSON                             1              0              0            975
Text                             1            117              0            544
YAML                            10             70             85            506
SVG                              6              1              1            425
Markdown                         4             97             45            254
Bourne Shell                     5             28             53            162
HCL                              1             43             37            155
TOML                             4             45             56            104
Python                           2             17             20             75
INI                              1              5              0             18
make                             1              0              0              4
Dockerfile                       1              0              0              1
-------------------------------------------------------------------------------
SUM:                           271           9683          11860          57605
-------------------------------------------------------------------------------
```

Abbildung 5: Übersicht über die Anzahl Dateien & Codezeilen per verwendeter Technologie im Vaultwarden-Server

```
github.com/AlDanial/cloc v 2.00  T=6.16 s (92.2 files/s, 43559.5 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JSON                            70              0              0         187374
TypeScript                     319           7591           4528          45296
HTML                            84             68             46           7511
XML                             60           1327           2451           6418
SCSS                            14            564             82           3247
JavaScript                       7            108            242            789
Swift                            3             38              5            230
Markdown                         2             49              0            117
SVG                              6              0              0            110
CSS                              2              9              0             39
YAML                             1              0              0             28
-------------------------------------------------------------------------------
SUM:                           568           9754           7354         251159
-------------------------------------------------------------------------------
```

Abbildung 6: Übersicht über die Anzahl Dateien & Codezeilen per verwendeter Technologie in der Browser-Extension

Für die dynamischen Analyseanteile wurden beide Komponenten auf dedizierten Testsystemen installiert. Sämtliche (Haupt-) Funktionalitäten wurden durch den Tester dabei manuell getriggert und untersucht (s. auch unsere Methodik-Beschreibung in Abschnitt B), sowie alle mindestens als [hoch] eingestuften statischen Findings hinsichtlich ihrer tatsächlichen Ausnutzbarkeit nachgetestet.

Hinweis: Der Bitwarden Web-Client (https://github.com/bitwarden/clients/tree/web-v2024.3.1) stand nicht im Fokus der Tests, wurde aber für den Penetrationstest des Servers und der Extension benötigt. Da sich hier ebenfalls wichtige Logik befindet, wird empfohlen, diesen in Zukunft ebenfalls tiefgreifenderen Untersuchungen zu unterziehen.

A4.2 Testzeitraum

Die Tests wurden im Zeitraum von 12.02.2024 bis 16.05.2024 durchgeführt.

A4.3 Einschränkende Rahmenbedingungen

A4.3.1 Quellcodeanalyse

Im Vorfeld der Analyse wurden folgende Festlegungen hinsichtlich der Durchführung getroffen:

- keine spezifische Untersuchung von Vagrant-/Docker-Releases + Konfigurationen
- keine spezifische Untersuchung des Quellcodes von 3rd-Party-Abhängigkeiten
- keine Durchführung von Befragungen/Interviews der Projektbeteiligten (Leiter, Entwickler etc.)
- keine Untersuchung von mobile Code-Anteilen (Android-, iOS-Code etc.)
- keine Untersuchung von Demo- oder Test-Code bzw. Dokumentationen

- keine Bewertung von Findings in Entwicklungsskripten und Utilities, welche nicht für den produktiven Einsatz vorgesehen sind
- freie Auswahl der freien und kommerziellen Analysewerkzeuge

A4.3.2 Penetration Test

Im Vorfeld der Analyse wurden folgende Festlegungen hinsichtlich der Durchführung getroffen:

- Module zur dynamischen Analyse werden durch den Auftragnehmer festgelegt
- Installation der Anwendungen durch den Auftragnehmer auf eigener Infrastruktur
- Schwachstellenfokussierung wird durch Auftragnehmer festgelegt (siehe auch die spezifischen aufgeführten Bedrohungen in Kapitel A4.5)

Hinweis: Ergebnisse, die durch diesen Ansatz aufgefunden wurden, können dem Kapitel E entnommen werden.

A4.4 Toolgestützte Analysen

Beide Anwendungen wurden einer eingehenden Sicherheitsuntersuchung unterzogen. Dafür kam eine Kombination folgender Herangehensweisen zum Einsatz:

- Automatische und manuelle Quellcodeanalyse (SAST + SECRETS + SCA)
- Automatische und manuelle Penetrationstests (DAST)

Die konkret im jeweiligen Ansatz eingesetzten Werkzeuge werden in den folgenden Abschnitten aufgelistet. Alle Tool-Resultate sind diesem Bericht sowohl in toolspezifischem RAW-Format, als auch in Berichtsform (im Excel-Format) als Anhang (s. auch Abschnitt D) beigefügt.

Alle Ergebnisberichte sind vollständig auditiert und alle Findings wurden bewertet (unsere Bewertungsvorgehen ist in B1.3 beschrieben).

Eine grobgranulare Finding-Übersicht zu den initial vom Werkzeug gelieferten Findings (inkl. ihrer toolspezifischen Kritikalitätseinstufung), sowie zu den durch den Auditor vorgenommenen Bewertungen, sind dem Abschnitt D zu entnehmen.

Hinweis: Nicht alle Werkzeuge sind in der Lage, alle Sprachen und Frameworks umfassend zu analysieren. Die Werkzeugauswahl erfolgte durch den Auftragnehmer, jeweils passend zu den verwendeten Programmiersprachen und Frameworks.

A4.4.1 SAST: Eingesetzte Werkzeuge

Die zu analysierenden Module wurden mit folgenden SAST-Werkzeugen einem Scan unterzogen:

- Synopsys Coverity (https://www.synopsys.com/software-integrity/static-analysis-tools-sast/coverity.html) – Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.1

- Semgrep Professional (https://semgrep.dev/products/pro-engine/semgrep) – Standard Regelsätze, Stand Februar 2024
    - Ergebnisse siehe Kapitel D1.2
- Devskim (https://github.com/microsoft/DevSkim) – Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.3
- Checkov (https://www.checkov.io/) – Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.4
- Bearer (https://github.com/Bearer/bearer) - Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.5
- Snyk Code (https://snyk.io/de/) - Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.6
- CodeQL (https://codeql.github.com/) - Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.7
- Trivy (https://github.com/aquasecurity/trivy) - Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.8
- Application Inspector (https://github.com/microsoft/ApplicationInspector) - Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.9

A4.4.2 SECRETS: Eingesetzte Werkzeuge

Eine Prüfung auf mögliche geleakte Geheimnisse im Quellcode wurde mit folgenden Werkzeugen durchgeführt:

- GitLeaks (https://gitleaks.io/) – Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.10
- CredScan (als Teil von SAST-Scan - https://github.com/ShiftLeftSecurity/sast-scan) – Standard-Regelsatz, Stand April 2024
    - Ergebnisse siehe Kapitel D1.11

A4.4.3 SCA: Eingesetzte Werkzeuge

Die Abhängigkeiten der Vaultwarden-Serveranwendung wurden mit folgenden SCA-Werkzeugen einer Known-Vulnerability-Prüfung unterzogen:

- Semgrep Supply Chain (https://semgrep.dev/products/semgrep-supply-chain/) – Known-Vulnerability-Datenbankstände vom 15.03.2024
    - Ergebnisse siehe Kapitel D2.1
- Syft/Grype (https://github.com/anchore/syft, https://github.com/anchore/grype) – Known-Vulnerability-Datenbankstände vom 15.03.2024
    - Ergebnisse siehe Kapitel D2.2

A4.4.4 DAST: Eingesetzte Werkzeuge

Für die dynamischen Analysen kam die Burp Suite Professional (https://portswigger.net/burp/pro) in der Version v.2024.4.5 zum Einsatz. Sämtliche (Haupt-) Funktionalitäten wurden hierbei durch den Tester manuell ausgelöst und untersucht. Hierbei kam der „Burp Web Vulnerability Scanner“ (aktiv und passiv) umfassend zum Einsatz.

Alle darüber hinaus vom Tester als „aus Sicherheitssicht interessant“ eingestuften Requests wurden manuell (mit automatischer Werkzeug-Unterstützung) tiefgreifender analysiert. Dabei kamen (neben den eingebauten Hilfstools, wie Repeater oder Intruder) auch die folgenden Erweiterungen zum Einsatz:

- Autorize v1.7
- Turbo Intruder v1.42
- Param Miner v1.4f
- Active Scan++ v1.0.24
- CSP-Bypass v1.0
- 403 Bypasser v1.2

Im Datei-Anhang sind die vollständigen Arbeitsprotokolle hinterlegt.

Als Browser kam für die dynamischen Server- als auch Extension-Tests der Google-Chrome in der Version 124 (mit mehrmaligen Aktualisierungen innerhalb des Testzeitraumes) zum Einsatz.

Hinweis: Gewonnene Erkenntnisse, welche nach unseren Auswertungen ggf. zu validen Findings/Schwachstellen führten, werden für dieses Werkzeug nicht explizit im Anhang als RAW-Format oder Report aufgeführt, sondern führen zu Finding-Beschreibungen in Kapitel E.

A4.5 Manuelle Analysen

Im Rahmen der Untersuchungen wurden ebenfalls umfangreiche manuelle Code-Reviews durchgeführt, welche aufgrund der LoC (insgesamt ca. 300.000) nicht vollumfassend möglich waren. Es wurden daher zielgerichtete Untersuchungen an neuralgischen Stellen durchgeführt. Folgende, gemeinsam mit dem Auftraggeber initial festgelegten Priorisierungen und Fokussierungen lagen dabei zugrunde:

A4.5.1 Vaultwarden Server

Der Vaultwarden Server selbst enthält keine sensiblen Klartext-Informationen, da aufgrund der Ende-zu-Ende-Verschlüsselung alle auf ihm gespeicherten Geheimnisse dem aktuellen Stand entsprechend ausreichend verschlüsselt abgelegt sind. Das Hauptaugenmerk liegt daher auf dem Server und den verwendeten kryptografischen Algorithmen für die Verschlüsselung.

- Spezifische Bedrohungen:
  - Browserbasierten Bedrohungen und Angriffe (z.B. Server-Side-Request-Forgery, Cross-Site-Scripting …)
  - Serverbasierte Bedrohungen und Angriffe (z.B. Command-Injection, SQL-Injection, …)
  - Fehlerhafte Zugriffkontrolle (z.B. unbefugter Zugriff auf nicht zugewiesene *Collections* innerhalb einer *Organization*, unbefugter Zugriff auf Benutzerverwaltungsfunktionen (e.g. Admin-Only-Feature),…)
  - Angriffe auf das Konzept des *Shared Access* innerhalb *Organization Vaults* und weitere wichtige Funktionen (z. B. *Sends*, *Emergency Access*)
  - Angriffe auf die 2-Faktorauthentifizierung (z.B. Brute-Force-Angriffe, Bypasses, …)
  - Kryptografische Fehlkonfigurationen (z.B. Verwendung von im Vergleich zu BSI TR-02102-2 schwachen kryptografischen Algorithmen oder fehlerhafte Initialisierung)
  - Race-Conditions, welche potenziell zu einem ausnutzbaren Fehlerzustand führen können

A4.5.2 Bitwarden-Client Browser-Extension

Die Bitwarden-Browsererweiterung verwendet für die *auto-fill*-Funktion Cross-Origin Kommunikation. Die Kommunikation ist mit JavaScript *postMessage* realisiert. Aufgrund des hohen Schutzbedarfs bei der Übergabe sensibler Daten an Dritte, lag ein großer Fokus auf der sicheren Authentifizierung der empfangen *postMessage*-Nachrichten.

- Weitere spezifische Bedrohungen:
  - Clientseitige Schwachstellen (z.B. Cross-Site-Scripting, Dom-based open redirects, fehlerhafte Content Security Policy, Clickjacking …)
  - Fehlkonfiguration innerhalb der `manifest.json` (z.B. *permissions, host_permissions*, *web_accessible_resources*, …)
  - Offenlegung von Informationen im Code bzw. Speicher der Browsererweiterung
  - Fehler in der Implementierung kryptografischer Funktionen

- Kryptografische Fehlkonfigurationen
- Angriffe auf die Ablauf- und Geschäftslogik

Für beide Module erfolgte zudem ein Review der CVEs (beginnend am 01.01.2023). Dabei wurden primär die Art und Weise des Umgangs mit den gemeldeten Schwachstellen, sowie die Qualität der vom Projekt gelieferten Korrektur (siehe eigenständiges Kapitel C) analysiert und bewertet.

Hinweis: Bei der Bewertung der mit den automatischen Werkzeugen aufgezeigten Findings (s. Kapitel D) schaut sich der Auditor auch immer den „umgebenden Code“ mit an und erkennt so ggf. anderweitige Schwachstellen oder sogenannte „Code-Smells“. Er lernt dabei, wie die einzelnen Entwickler programmieren, welche Vorlieben sie haben und welche Fehler man von ihnen „erwarten“ könnte. Die Bewertung dieser Findings ist somit auch immer eine Form von „geführter manueller Analyse“.

B. Methodik und Bewertung

B1 Beschreibung der Methodik

B1.1 Testverfahren (Whiteboxtest)

Die Anwendung wurde sowohl manuellen als auch automatischen statischen und dynamischen Analysen unterzogen.

B1.1.1 Art der Schwachstellenuntersuchung

Wir betrachten die im folgenden Bild gezeigten Ebenen (Aspekte) 2 bis 5 der Sicherheit der Webanwendungen:

Table 4: Übersicht über das Ebenenmodell

	Ebene	Inhalt (Beispiele)
5	**Semantik**	Schutz vor Täuschung und Betrug - Informationen ermöglichen Social-Engineering-Angriffe - Gebrauch von Popups u. ä. erleichtern Phishing-Angriffe - Keine Absicherung für den Fall der Fälschung der Website
4	**Logik**	Absicherung von Prozessen und Workflows als Ganzes - Verwendung unsicherer E-Mail in einem ansonsten gesicherten Workflow - Angreifbarkeit des Passworts durch nachlässig gestaltete „Passwort vergessen"-Funktion - Die Verwendung sicherer Passwörter wird nicht erzwungen
3	**Implementierung**	Vermeiden von Programmierfehlern, die zu Schwachstellen führen - Cross-Site Scripting - SQL-Injection - Cross-Site Request Forgery (Session Riding)
2	**Technologie**	Richtige Wahl und sicherer Einsatz von Methoden - unverschlüsselte Übertragung sensitiver Daten - Authentisierungsverfahren, die dem Schutzbedarf nicht angemessen sind - Ungenügende Entropie von Tokens

1	**System**	Absicherung der auf der Systemplattform eingesetzten Software - Fehler in der Konfiguration des Webservers - Bekannte Schwachstellen in den eingesetzten Softwareprodukten - Mangelnder Zugriffsschutz in der Datenbank
0	**Netzwerk & Host**	Absicherung von Host und Netzwerk

Abbildung B1.1.1: Ebenen der Web Application Security

Ebene 0 – Netzwerk und Host (nur teilweise Gegenstand der Untersuchung)

Die Websicherheit steht auch in Abhängigkeit zur Sicherheit von Netzwerk, Hardware und Host. Wir betrachten die Berührungspunkte der beiden Ebenen, soweit dies erforderlich ist.

Ebene 1 – Systemebene (nur teilweise Gegenstand der Untersuchung)

Ebene 1 beinhaltet all jene Software, die eine Webanwendung benötigt, um überhaupt laufen zu können. Dazu gehören der Webserver und der Applikationsserver, aber auch die Datenbank und beteiligte Backend-Systeme. Alle diese Komponenten müssen bei der Betrachtung der Sicherheit einer Webanwendung mit einbezogen werden. Eine Webanwendung A, die frei von Sicherheitsmängeln programmiert ist, ist trotzdem unsicher, wenn die von ihr verwendete Datenbank über einen anderen Kanal, etwa den nicht ausreichend abgesicherten und für ‚Innentäter‘ zugänglichen Direktzugriff per Database-Client manipulierbar ist und diese Manipulation im Web von einem Angreifer ausgenutzt werden kann.

Ein besonderes Feld sind die sog. Known-Vulnerabilities, die wir ebenfalls der Systemebene zuordnen.

Ebene 2 – Technik

In diesem Bereich geht es um die Auswahl der für den jeweiligen Zweck und Schutzbedarf richtigen Technik – und um deren richtigen Einsatz. Eine Webanwendung, die sensible Daten unverschlüsselt über das Internet transferiert, setzt nicht die richtige Technik ein. Und eine Webanwendung, die Passwörter zwar verschlüsselt, dies aber mit einem zu kurzen Schlüssel tut, setzt die richtige Technik falsch ein. Die erste Anwendung ist gegen Ausspähen auf dem Übertragungswege nicht geschützt, die andere nicht ausreichend gegen Passwort-Cracking. Beide sind also unsicher, auch wenn sie im Programmcode keine Sicherheitslücken enthalten.

Ebene 3 – Implementierung

Die Implementierungsebene ist die offensichtliche Ebene der Web Application Security. Dies ist der Bereich, in dem unbeabsichtigte Programmierfehler (Bugs) auftreten und zu Sicherheitsproblemen führen oder aber fehlerhafte Programmierung, wie nicht vorhandene oder ungenügende Datenvalidierung, stattfindet.

Logik (Ebene 4) und Semantik (Ebene 5)

Diese beiden Ebenen sind diejenigen, die gegenwärtig im Sicherheitskontext noch kaum beachtet werden. Dabei sind sie von großer Wichtigkeit – und werden es im Zeitalter von Phishing und Identitätsdiebstahl immer stärker werden – wenn man die Sicherheit von Webanwendungen nicht allein als den Schutz des Servers vor Eindringversuchen versteht, sondern sie umfassend begreift. Eine Webanwendung ist sicher, wenn sie selbst mitsamt dem sie beherbergenden System sicher ist, und wenn sie die Benutzer und deren vertrauenswürdige Daten vor Schaden bewahrt. Der Anbieter einer Webanwendung trägt also nicht nur die Verantwortung für sein eigenes System, sondern auch für alle an der Nutzung Beteiligten. Auf den Ebenen der Logik und der Semantik kommt dieser letzte Aspekt ganz besonders zum Tragen.

Ebene 4 – Logik

Diese Ebene betrifft die Logik der Abläufe in einer Anwendung – die Anwendungs- und Business-Logik – mit dem Benutzer. Ist diese zu ‚zweckorientiert‘ implementiert, d. h. ist die Möglichkeit, dass sie anders als beabsichtigt genutzt wird, zu wenig berücksichtigt, dann ist häufig eine Angreifbarkeit gegeben. Sind z. B. Mechanismen eingebaut, welche bei unsachgemäßer Bedienung (Zustände, die nur unter Umgehung der Browserfunktionalität möglich sind und damit eine missbräuchliche Verwendung eindeutig anzeigen) den Benutzer ausloggen oder andere Formen des Schutzes betreiben?

Ebene 5 – Semantik

Die semantische Ebene der Web Application Security umfasst inhaltliche und die Kommunikation betreffende Aspekte. Dies betrifft die Art der Informationen, die dem Benutzer gegeben werden, wie ihm Inhalte präsentiert werden und wie mit ihm umgegangen wird. Dieser Bereich kann in seiner Betrachtung selten auf eine einzelne Anwendung beschränkt bleiben. Er ist in der Regel vielmehr website- oder unternehmensübergreifend zu definieren und ähnlich, wie die Vorgabe einer CI von allen Kommunikationsmedien einzuhalten ist, von allen Anwendungen zu befolgen.

Ein fehlerhafter Umgang mit den semantischen Aspekten der Web Application Security erleichtert insbesondere Angriffe auf den Benutzer, was wiederum dem Vertrauen des Benutzers in die Anwendung und das Unternehmen sowie das Web insgesamt schadet. Zu derartigen Angriffen zählen Social-Engineering, Phishing, Identitätsdiebstahl, Täuschung, Fälschung, Betrug, sowie Aufbrechen des Datenschutzes und des Schutzes der Privatsphäre.

B1.2 Bewertungsschema

B1.2.1 Gefahrenpotenzial

Die durchgeführten Tests wurden jeweils mit einem Gefahrenpotenzial bewertet. Das Gefahrenpotenzial ist ein abstraktes Maß für die Klassifizierung einer Schwachstelle (Vulnerability) und der durch sie entstehenden Bedrohung (Threat).

Das Gefahrenpotenzial wird zu jeder getesteten Schwachstelle in der rechten Spalte genannt. Wir unterscheiden:

[OK] die genannte Schwachstelle liegt nicht vor

[kritisch] (siehe unten Gefährdungs-Matrix)

[hoch] (siehe unten Gefährdungs-Matrix)

[mittel] (siehe unten Gefährdungs-Matrix)

[niedrig] (siehe unten Gefährdungs-Matrix)

[info] hierbei handelt es sich um Informationen, nicht um eine Schwachstelle

Das Gefahrenpotenzial ist kein Maß für das Risiko, das eine solche Schwachstelle darstellt. Das Risiko (= Schadenshöhe * Eintrittswahrscheinlichkeit), welches dieses Gefahrenpotenzial darstellt, wird hier nicht bewertet, da zu dessen Beurteilung weitere Informationen (z. B. Schadenspotenzial) vom Auftraggeber nötig sind.

B1.2.2 Eintrittswahrscheinlichkeit

Die Eintrittswahrscheinlichkeit ist im Wesentlichen von folgenden Faktoren bestimmt:

- Wie einfach ist die Schwachstelle zu finden (Visibility)?
- Wie einfach ist die Schwachstelle auszunutzen (Exploitability)?
- Wie bekannt ist die Schwachstelle (Publicly Known)?
- Welche technischen Kenntnisse sind zum Ausnutzen nötig?
- Welche Zugriffsmöglichkeiten bestehen (remote vs. local)?
- Welchen Typs ist die Schwachstelle (Vulnerability Type)?

Vereinfacht kann man sagen, dass die Eintrittswahrscheinlichkeit von der Schwierigkeit des Angriffs abhängt.

Die Einschätzung der Eintrittswahrscheinlichkeit sollte nur mit großer Vorsicht dazu herangezogen werden, um über die Notwendigkeit der Beseitigung einer Schwachstelle zu entscheiden. Ein entsprechend motivierter Angreifer mit hinreichendem Kenntnisstand ist einem Pentester in der Regel überlegen, wenn er nur bereit ist, genügend Zeit zu investieren. Die Situation verschlechtert sich weiter, wenn man annehmen muss, dass eine Vielzahl von Angreifern nach Schwachstellen sucht.

B1.2.3 Schadenspotenzial

Das Schadenspotenzial ist vor allem die Folge, die sich aus der Ausnutzung der Schwachstelle ergeben kann:

- Verlust der Verfügbarkeit (Availability)
- Verlust der Vertraulichkeit (Confidentiality)
- Verlust der Vollständigkeit, bzw. Unversehrtheit (Integrity)
- Verlust von Schutzmechanismen

— Erweiterung der Zugriffsberechtigungen.

Das konkrete Schadenspotenzial kann i. A. nur vom Auftraggeber bestimmt werden.

Risiko-/Gefährdungs-Matrix

Es ist allgemein üblich, das Risiko nach der Formel

Risiko = Schadenspotenzial * Eintrittswahrscheinlichkeit

zu bestimmen. Aus den Beschreibungen zur Eintrittswahrscheinlichkeit und zum Schadenspotenzial wird ersichtlich, dass sich beide Faktoren ebenfalls aus mehreren Werten zusammensetzen. Z. B. hat eine Schwachstelle, die allgemein bekannt ist und für die bereits fertige Exploits existieren, eine hohe Eintrittswahrscheinlichkeit. Ist dadurch dann z. B. die Übernahme des Systems möglich oder können vertrauliche Daten eingesehen werden, dann ist auch das Schadenspotenzial hoch.

Um das Produkt aus Eintrittswahrscheinlichkeit und Schadenspotenzial mit ihren jeweils eigenen Faktoren zu berechnen, gibt es verschiedene Modelle (CVSS, DREAD, STRIDE, OCTAVE), die die einzelnen Faktoren unterschiedlich gewichten.

Um eine genaue Risikoabschätzung zu erhalten, müsste das Risiko mit der subjektiven Gewichtung des Auftraggebers nach den bekannten Modellen berechnet werden. Diese muss für jede in diesem Bericht aufgeführte Schwachstelle einzeln erfolgen.

Um die Schwachstellen einfach und schnell zu klassifizieren, wird für das Gefahrenpotenzial folgende vereinfachte Matrix verwendet:

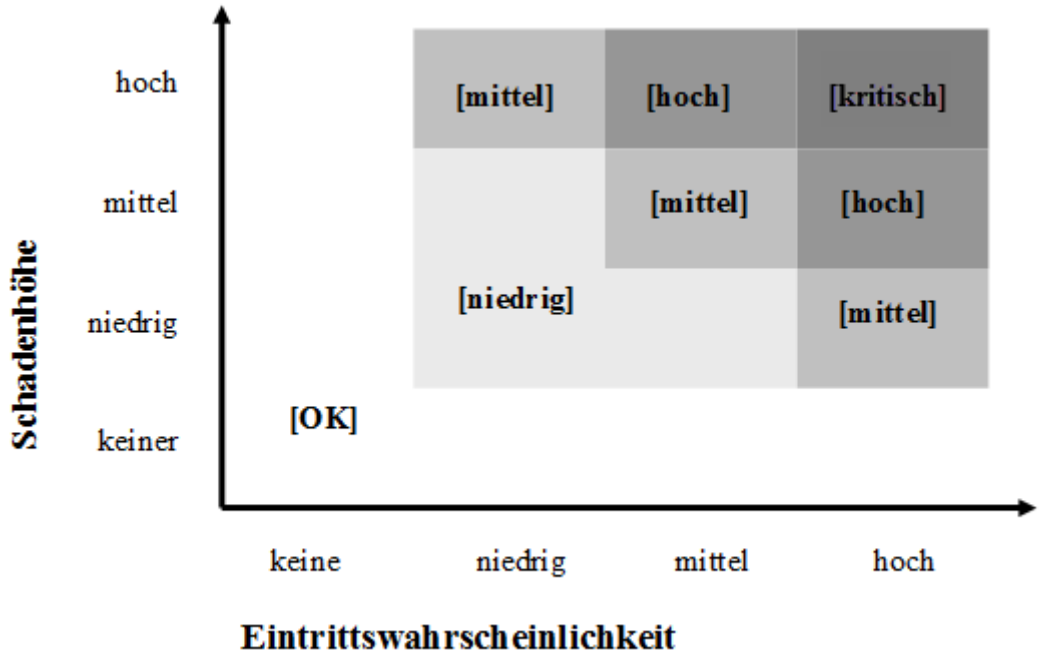


Abbildung 7: Bestimmung Gefahrenpotential

Aus dieser Beurteilung kann auch eine Handlungspriorität abgeleitet werden:

[kritisch] – Es besteht sofortiger Handlungsbedarf; der Applikation, der Datenbank oder dem System kann schwerwiegender Schaden zugefügt werden; ggf. ist das System abzuschalten

[hoch] – Es besteht dringender Handlungsbedarf, die Webanwendung/das System und/oder die Daten sind bestandsgefährdet;

[mittel] – Es besteht Handlungsbedarf, die Webanwendung/das System muss z. B. durch einspielen von Sicherheitsupdates oder durch zusätzliche Sicherungsmaßnahmen (z. B. WAF) abgesichert werden.

[niedrig] – Der Handlungsbedarf liegt im Ermessen des Auftraggebers.

Hinweis

Obwohl wir bei der Bewertung objektive Maßstäbe bzw. als allgemeingültig anerkannte Grundsätze zugrunde gelegt haben und nach bestem Wissen und Gewissen versucht haben, die Vor- und Nachteile gegeneinander abzuwägen, kann eine solche Bewertung immer nur subjektiv sein. Uns nicht bekannte Details oder unterschiedliche Interpretation der Sicherheitsanforderungen können zu einer anderen, als der hier jeweils abgegebenen Bewertung führen. Daher ist der Auftraggeber angehalten, anhand der gegebenen Informationen eine eigene Bewertung vorzunehmen, bzw. die Dringlichkeit der Behebung einzelner Mängel selbst zu beurteilen.

Gerne stehen wir dabei beratend zur Seite.

B1.3 Aufbau der Testergebnisse der toolbasierten Findings

Alle von den Werkzeugen gemeldetet Schwachstellen wurden bewertet und Ergebnisberichte (falls möglich bzw. notwendig) liegen im Anhang (s. jeweiliges Werkzeug in Abschnitt D bzw. grobe Übersicht in Abschnitt E) bei.

Die tooleigenen Kritikalititätseinstufungen wurden von uns folgendermaßen interpretiert:

- **Kritisch** (in den Tools: „Critical“)
- **Hoch** (in den Tools: „High“ oder „Error“)
- **Mittel** (in den Tools: „Medium“, „Moderate“ oder „Warning“)
- **Niedrig** (in den Tools: „Low“, „Note“ oder „Recommendation“)

Die eigentlichen Bewertungen erfolgten entweder in der vom Werkzeug bereitgestellten Oberfläche oder in von uns aus den RAW-Ergebnissen erstellten Excel-Dokumenten. Dabei musste bei der Bewertung teilweise auf tool-eigenes „Wording“ zurückgegriffen werden. Die Bewertungen in den Reports sind folgendermaßen zu interpretieren:

- **Bestätigt** (in den Bewertungen: „Exploitable“ oder „Confirmed“)
    - Finding wurde bestätigt (mit „kritisch“ oder „hoch“ bewertete Findings wurden hinsichtlich ihrer Ausnutzbarkeit untersucht und dann in Abschnitt E näher beschrieben)
- **Verdächtig** (in den Bewertungen: „Suspicious“)

- Finding konnte nicht 100%ig bestätigt oder entkräftet werden (weder statisch noch durch DAST-Analysen), könnte aber ein (zukünftiges) Sicherheitsproblem darstellen, wessen sich das Projekt bewusst sein sollte. Hier muss die Bewertung und Behandlung von einem Projekt-Insider mit Sicherheitshintergrund entschieden bzw. abgeschlossen werden.

— **Schlechte Praxis** (in den Bewertungen: „Bad Practice“)

- Finding wurde als Programmierfehler oder schlechte Code-Qualität eingestuft, welcher z.B. sicherheitsrelevante Konsequenzen oder logische Fehlinterpretationen zur Folge haben könnte. Die Wahrscheinlichkeit dafür wurde aber als sehr gering eingestuft.

— **Kein Problem** (in den Bewertungen: „Not an issue“)

- Finding wurde als irrelevant oder False Positive eingestuft.

Auf Basis dieser beiden Einstufungen ergibt sich die folgende Interpretation zur in 0 definierten Riskio-/Gefährdungsmatrix (die Statistik-Übersicht in A3.1 basiert auf dieser Zuordnung):

[kritisch]

— Tooleinstufung: kritisch X Auditoreinstufung bestätigt

[hoch]

— Tooleinstufung: hoch X Auditoreinstufung bestätigt

[mittel]

— Tooleinstufung: kritisch X Auditoreinstufung verdächtig

— Tooleinstufung: mittel X Auditoreinstufung bestätigt

[niedrig]

— Tooleinstufung: kritisch X Auditoreinstufung schlechte Praxis

— Tooleinstufung: hoch X Auditoreinstufung verdächtig

— Tooleinstufung: niedrig X Auditoreinstufung bestätigt

[info]

— Tooleinstufung: hoch X Auditoreinstufung schlechte Praxis

— Tooleinstufung: mittel X Auditoreinstufung verdächtig

— Tooleinstufung: mittel X Auditoreinstufung schlechte Praxis

— Tooleinstufung: niedrig X Auditoreinstufung verdächtig

— Tooleinstufung: niedrig X Auditoreinstufung schlechte Praxis

B1.4 Aufbau der Testergebnisse in diesem Dokument (manuelle SAST/DAST)

Für jede relevante Schwachstelle gibt es jeweils ein eigenes Unterkapitel, wobei deren Überschriften Auskunft über die Art der Schwachstelle geben.

Eine solche Überschrift sieht dann z. B. so aus:

- n.m Cross-Site-Scripting
- **n.m** ist die Unterkapitelnummer
- **Cross-Site-Scripting** ist die genaue Schwachstellenbezeichnung

Jedes dieser Unterkapitel ist dann gegliedert in:

- **Bedrohung**
  Die Schwachstelle bzw. Angriffstechnik wird allgemeinverständlich erklärt. Ggf. mit Links zu weiterführenden Informationsquellen.

- **Beobachtung**
  Ist die Anwendung gegen die beschriebene Bedrohung gefeit, wird dies hier begründet und belegt.

  Ist die Anwendung gegenüber der beschriebenen Bedrohung anfällig, wird die Beobachtung in einem eigenen Unterkapitel wie folgt ausgeführt und belegt.

n.m.o Reflected-Cross-Site-Scripting stark verbreitet [hoch]

- **Reflected-Cross-Site-Scripting stark verbreitet**
  ist eine knappe Zusammenfassung der gefundenen Schwachstelle

- **hoch**
  ist die Bewertung des Gefahrenpotenzials

Die Gliederung sieht hier folgendermaßen aus:

- **Beobachtung**
  Der Test und die Beobachtung der Reaktion der Anwendung werden beschrieben. Damit soll (1) der Leser in die Lage versetzt werden, den Angriff zu verstehen, um das Risiko selbst besser beurteilen zu können, und (2) der Angriff nachvollziehbar werden. Es ist nicht immer möglich, (2) zu gewährleisten, da für das Nachstellen u. U. eine umfangreichere Vorbereitung oder zeitliche Korrelation mit anderen Bedingungen erfüllt sein muss.

- **Reproduktionsschritte** (optional)
  Falls die konkrete Ausnutzung der Schwachstelle detailliertere Anweisungen benötigt, die nicht sinnvoll über die Beobachtung abgedeckt werden können, werden hier die einzelnen Schritte zur Nachstellung der Schwachstelle beschrieben.

- **Schwachstelle im Code** (optional)
  In diesem Abschnitt wird auf die Zeile(n) im Code bzw. die Codeabschnitte verwiesen, welche verantwortlich für die gezeigte Schwachstelle sind. Falls nötig, werden einzelne Elemente des Codes kurz erklärt. Der Abschnitt ist nur

vorhanden, wenn die Schwachstelle im Quellcode des Untersuchungsgegenstandes aufgetreten ist.

— **Ausnutzbarkeit**
Hier wird beschrieben, welche Voraussetzungen für einen Angriff gegeben sein müssen, in welcher Form ein Angreifer die Schwachstelle ausnutzen kann und mit welchen Konsequenzen und Risiken zu rechnen ist. Generell legen wir einen Worst-Case Ansatz zugrunde, d. h. schildern im Zweifel das bedrohlichere Szenario. Diese Schilderung ist unabhängig von der Eintrittswahrscheinlichkeit. Eine Bewertung der Eintrittswahrscheinlichkeit muss separat erfolgen.

— **Hinweis / Bemerkung** (optional)
Hier werden Besonderheiten und/oder bestimmte Bedingungen zur gefundenen Schwachstelle beschrieben, z. B. wenn diese nicht reproduzierbar auftrat oder nur mit bestimmten Browsern. Oft sind solche Anmerkungen zur Beurteilung der Bewertung (siehe oben) wichtig.

— **Maßnahme**
Es ist zwischen grundlegenden Maßnahmen und individuellen Maßnahmen zu unterscheiden. Grundlegende Maßnahmen sind meistens bedrohungsbezogen, wohingegen individuelle Maßnahmen eher schwachstellenbezogen sind. Grundlegende Maßnahmen lassen sich unabhängig von der jeweiligen Anwendung formulieren und haben universelle Gültigkeit.

Bei Webanwendungen, die ohne Berücksichtigung der grundlegenden Schutzmaßnahmen entwickelt worden sind, kann die nachträgliche Anwendung grundlegender Maßnahmen zu unvertretbar hohem Aufwand führen. Hinzu kommt, dass eine Anwendung, die die beschriebene Maßnahme nicht umgesetzt hat, trotzdem nicht zwingend anfällig für einen Angriff sein muss, weil dies bewusst durch andere Maßnahmen sichergestellt worden ist, oder weil es sich eher zufällig so verhält. In diesem Fall wäre es nicht angebracht, die Umsetzung der genannten Maßnahme zu fordern.

Maßnahmen aus Sicht des Penetrationstests sind daher zumeist individuelle schwachstellenbezogene Maßnahmen, die erst bei Entdeckung einer Schwachstelle relevant werden. Dabei kommt zur Behebung der Schwachstelle häufig nicht genau eine Maßnahme in Betracht, sondern eine ganze Palette von möglichen Maßnahmen. Die Entscheidung über die richtige Maßnahme ist für jeden Anwendungsfall individuell zu treffen und abhängig von den softwaretechnischen Randbedingungen, der verwendeten Technik, der Art der Realisierung, den Realisierungskosten usw.

— Eine tatsächliche Bewertung und Entscheidung darüber, wie eine Schwachstelle zu beseitigen ist, muss der Auftraggeber treffen oder sie wird in einem abschließenden Workshop zusammen mit den Entwicklern bzw. Anwendungsverantwortlichen getroffen.

C. CVE-Reviews

Im Rahmen dieser Analyse wurden alle gemeldeten CVEs (01.01.2023 bis 16.05.2024) hinsichtlich Ihrer Korrektur überprüft und kommentiert.

Im oben genannten Zeitraum wurde eine CVE für die Browser-Extension gemeldet: CVE-2023-27974

C1 CVE-2023-27974

Table 5: Steckbrief des CVE-Eintrags

Fix-Bewertung	OK – wir stützen die Haltung des Herstellers.
CWE	nicht klassifiziert
CVE-Kurzbeschreibung	Bitwarden through 2023.2.1 offers password auto-fill when the second-level domain matches, e.g., a password stored for an example.com hosting provider when customer-website.example.com is visited. NOTE: the vendor's position is that "Auto-fill on page load" is not enabled by default.
CVE registriert	n/a
CVE veröffentlicht	08.03.2023
CVE-Details	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-27974 https://www.cve.org/CVERecord?id=CVE-2023-27974 https://nvd.nist.gov/vuln/detail/CVE-2023-27974
CVSS-Base-Score (NIST)	n/a
Zusammenfassung	Ein Angreifer könnte eine Subdomain einer bekannten/vertrauenswürdigen Domain (z.B. `evil.example.com`, unter der Annahme `example.com` wäre vertrauenswürdig) verwenden, um Passwörter durch das `auto-fill` Feature zu stehlen. Wäre ein bösartiges Formular über ein iframe in einer legitimen Seite mit Formular eingebettet, würde das `auto-fill` Feature beide Formulare befüllen.
Betroffene Versionen	<= 2023.2.1
Fix-Datum	n/a
Fix-Commit	n/a

C1.1 Korrektur-Informationen

Wurde bisher nicht korrigiert.

C1.2 Bewertung

Das Risiko des gemeldeten Sicherheitsproblems ist gering, da es für einen erfolgreichen Angriff gegen Bitwarden-Nutzer nötig ist, ein bösartiges `iframe` auf einer legitimen Anmeldeseite einzuschleusen.

Falls die Anmeldeseite zudem gängige Einschränkungen für `iframes` anwendet, würde dies das Problem zudem deutlich abmildern.

D. Testergebnisse automatische Werkzeuge

Im Folgenden werden die Untersuchungsergebnisse, unterteilt nach dem automatischen Analysewerkzeug, zusammengefasst dargestellt. Detaillierte Ergebnisdetails können den Ergebnisberichten und -dokumenten in der Anlage entnommen werden.

Nach der Durchführung des Scans wurden Findings in folgenden Ordnern von Review und Bewertung ausgenommen:

- .github/ (Github-Arbeitsdateien)
- .vscode (Visual-Studio Arbeitsdateien)
- spec/ (Tests und Fixtures)

Hinweis: Im Finding-Rohmaterial (SARIF- oder JSON-Dateien) sind möglicherweise auch Ergebnisse für die o.g. Ordner zu finden.

D1 SAST- und SECRET-Werkzeuge

Alle bewerteten und durch uns kommentierten Findings sind in folgenden Excel-Ergebnisdokumenten ersichtlich:

- *vaultwarden-server-audited.xlsx*
- *vaultwarden-browser-extension-audited.xlsx*

Vom Review ausgeschlossene, sowie als Duplikat erkannte Findings, sind in folgenden separaten Excel-Dateien dokumentiert (in nachfolgenden Beschreibungen werden betroffene Findings allgemein als „von der Bewertung ausgenommen“ bezeichnet):

- *vaultwarden-server-excluded.xlsx*
- *vaultwarden-browser-extension-excluded.xlsx*

D1.1 Synopsys Coverity

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 2023.12.0 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Coverity Standard-Bewertung (high, medium, low). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.1.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-coverity-raw.json*
- *raw/vaultwarden-browser-extension-coverity-raw.json*

D1.1.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 1x „medium“
- 3x „low“

Browser-Extension

- 29x „medium“ (32 gefunden und 3 von der Bewertung ausgenommen)
- 78x „low“ (80 gefunden und 2 von der Bewertung ausgenommen)

D1.1.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 1 bewertetes „medium“-Finding resultierte in folgender Einstufung:
  - 1 wurde als `Kein Problem` eingestuft
- 3 bewertete „low“-Findings resultierten in folgenden Einstufungen:
  - 2 wurden als `Schlechte Praxis` eingestuft
  - 1 wurde als `Kein Problem` eingestuft

Browser-Extension

- 29 bewertete „medium“-Findings resultierten in folgenden Einstufungen:
  - 29 wurden als `Kein Problem` eingestuft
- 78 bewertete „low“-Findings resultierten in folgenden Einstufungen:
  - 78 wurden als `Kein Problem` eingestuft

D1.2 Semgrep Professional

Für diese Scans kamen folgende Regelsätze (Stand Mai 2024) zum Einsatz:

- p/owasp-top-ten
- p/cwe-top-25
- p/default
- p/comment
- p/r2c-security-audit
- p/command-injection
- p/secrets
- p/sql-injection
- p/xss
- p/bandit
- p/eslint
- p/brakeman
- p/security-code-scan
- p/security-audit
- p/supply-chain
- p/findsecbugs
- p/gosec
- p/jwt

Die Scanner-Engine kam dabei in der Version 1.67.0 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Semgrep Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.2.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-semgrep-pro-raw.sarif*
- *raw/vaultwarden-browser-extension-semgrep-pro-raw.sarif*

D1.2.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 3x „error“ (9 gefunden und 6 von der Bewertung ausgenommen)
- 141x „warning“ (416 gefunden und 275 von der Bewertung ausgenommen)

Browser-Extension

- 8x „error“
- 209x „warning“ (224 gefunden und 15 von der Bewertung ausgenommen)

D1.2.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 3 bewertete „error“-Findings resultierten in folgenden Einstufungen:
  - 1 wurde als `Bestätigt` eingestuft
  - 2 wurden als `Kein Problem` eingestuft
- 141 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 141 wurden als `Kein Problem` eingestuft

Browser-Extension

- 8 bewertete „error“-Findings resultierten in folgenden Einstufungen:
  - 8 wurden als `Kein Problem` eingestuft
- 209 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 209 wurden als `Kein Problem` eingestuft

D1.3 Devskim

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 1.0.20 bzw. 1.0.32 (Browser-Extension) zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Devskim Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.3.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-devskim-raw.sarif*
- *raw/vaultwarden-browser-extension-devskim-raw.sarif*

D1.3.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 5x „error“ (6 gefunden und 1 von der Bewertung ausgenommen)
- 0x „warning“ (2 gefunden und 2 von der Bewertung ausgenommen)
- 44x „note“ (57 gefunden und 13 von der Bewertung ausgenommen)

Browser-Extension

- 2x „error“
- 11x „warning“
- 354x „note“

D1.3.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 5 bewertete „error“-Findings resultierten in folgenden Einstufungen:

- 5 wurden als `Kein Problem` eingestuft

- 44 bewertete „note“-Findings resultierten in folgenden Einstufungen:
  - 44 wurden als `Kein Problem` eingestuft

Browser-Extension

- 2 bewertete „error“-Findings resultierten in folgenden Einstufungen:
  - 2 wurden als `Kein Problem` eingestuft
- 11 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 11 wurden als `Kein Problem` eingestuft
- 354 bewertete „note“-Findings resultierten in folgenden Einstufungen:
  - 354 wurden als `Kein Problem` eingestuft

D1.4 Checkov

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 3.2.22 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Checkov Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.4.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-checkov-raw.sarif*
- *raw/vaultwarden-browser-extension-checkov-raw.sarif*

Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 6x „error“ (11 gefunden und 5 von der Bewertung ausgenommen)

Browser-Extension

— 1x „error“

D1.4.2 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

— 6 bewertete „error“-Findings resultierten in folgenden Einstufungen:

  — 4 wurden als `Schlechte Praxis` eingestuft

  — 2 wurden als `Kein Problem` eingestuft

Browser-Extension

— 1 bewertetes „error“-Finding resultierte in folgender Einstufung:

  — 1 wurde als `Kein Problem` eingestuft

D1.5 Bearer

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 1.27.1 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Bearer Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.5.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

— *raw/vaultwarden-server-bearer-raw.sarif*

— *raw/vaultwarden-browser-extension-bearer-raw.sarif*

D1.5.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

— 2x „error“ (3 gefunden und 1 von der Bewertung ausgenommen)

Browser-Extension

— 33x „error“

D1.5.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

— 2 bewertete „error“-Findings resultierten in folgenden Einstufungen:

  — 2 wurden als `Kein Problem` eingestuft

Browser-Extension

— 33 bewertete „error“-Findings resultierten in folgenden Einstufungen:

  — 33 wurden als `Kein Problem` eingestuft

D1.6 Snyk Code

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 1.1284.0 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Snyk Code Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.6.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

— *raw/vaultwarden-server-snyk-code-raw.sarif*

— *raw/vaultwarden-browser-extension-snyk-code-raw.sarif*

D1.6.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

— 2x „warning“

- 0x „note“ (2 gefunden und 2 von der Bewertung ausgenommen)

Browser-Extension

- 9x „warning“ (14 gefunden und 5 von der Bewertung ausgenommen)

D1.6.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 2 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 2 wurden als `Kein Problem` eingestuft

Browser-Extension

- 9 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 9 wurden als `Kein Problem` eingestuft

D1.7 CodeQL

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 2.15.1 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der CodeQL Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.7.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-codeql-js-raw.sarif*
- *raw/vaultwarden-browser-extension-codeql-js-raw.sarif*

D1.7.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

— 0x „warning“ (13 gefunden und 13 von der Bewertung ausgenommen)

Browser-Extension

— 1x „warning“ (2 gefunden und 1 von der Bewertung ausgenommen)

D1.7.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

— Keine Bewertung erfolgt

Browser-Extension

— 1 bewertetes „warning“-Finding resultierte in folgender Einstufung:

  — 1 wurden als `Kein Problem` eingestuft

D1.8 Trivy

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 0.49.1 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Trivy Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.8.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

— *raw/vaultwarden-server-trivy-raw.sarif*

— *raw/vaultwarden-browser-extension-trivy-raw.sarif*

D1.8.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl der Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

— **0x „error“ (2 gefunden und 2 von der Bewertung ausgenommen)**

— 1x „warning“

Browser-Extension

— 3x „error“

— 3x „warning“ (4 gefunden und 1 von der Bewertung ausgenommen)

— 3x „note“

D1.8.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

— 1 bewertetes „warning“-Finding resultierte in folgender Einstufung:

  — 1 wurde als `Kein Problem` eingestuft

Browser-Extension

— 3 bewertete „error“-Findings resultierten in folgenden Einstufungen:

  — 1 wurde als `Schlechte Praxis` eingestuft

  — 2 wurden als `Kein Problem` eingestuft

— 3 bewertete „warning“-Findings resultierten in folgenden Einstufungen:

  — 3 wurden als `Kein Problem` eingestuft

— 3 bewertete „note“-Findings resultierten in folgenden Einstufungen:

  — 3 wurden als `Kein Problem` eingestuft

D1.9 Application Inspector

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in den Versionen 1.9.10 für den Vaultwarden-Server und 1.9.22 für die Browser-Extension zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Application Inspector Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.9.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-appinspector-raw.sarif*
- *raw/vaultwarden-browser-extension-appinspector-raw.sarif*

D1.9.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 5.151x „note“ (6.773 gefunden und 1.622 von der Bewertung ausgenommen)

Browser-Extension

- 7.702x „note“ (11.729 gefunden und 4.027 von der Bewertung ausgenommen)

D1.9.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 5.151 bewertete „note“-Findings resultierten in folgenden Einstufungen:
  - 5.151 wurden als `Kein Problem` eingestuft

Browser-Extension

- 7.702 bewertete „note“-Findings resultierten in folgenden Einstufungen:
  - 7.702 wurden als `Kein Problem` eingestuft

D1.10 GitLeaks

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 8.18.0 zum Einsatz.

Durch GitLeaks erfolgt keine initiale Finding-Einstufung hinsichtlich der Kritikalität. Alle vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung gleichberechtigt in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

Hinweis: Die exportierten SARIF-Dateien wurden von uns um die zugehörigen Code-Zeilen, mit 5 Zeilen Kontext davor und danach, angereichert, um eine Verwendung der Ergebnisse zu erleichtern (Grund: Findings beziehen sich oft auf Commits aus der Vergangenheit).
Diese erfolgte im `results`-Array bei den einzelnen Ergebnis-Einträgen unter dem JSON-Pfad `locations > physicalLocation > artifactLocation > description > text`. Dabei wurde die Schema-Konformität bzgl. des Sarif-Formates in der Version 2.1.0 erhalten.

D1.10.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-gitleaks-raw.sarif*
- *raw/vaultwarden-browser-extension-gitleaks-raw.sarif*

D1.10.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings, sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet (Hinweis: Gitleaks nimmt keine Kritkalitätseinstufung vor):

Vaultwarden Server

- 1x „warning“ (Einstufung erfolgte durch Auditor)

Browser-Extension

- 729x „warning“ (929 gefunden und 200 von der Bewertung ausgenommen) (Einstufung erfolgte durch Auditor)

D1.10.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 1 bewertetes „warning“-Finding resultierte in folgender Einstufung:
  - 1 wurden als `Kein Problem` eingestuft

Browser-Extension

- 729 bewertete „warning“-Findings resultierten in folgenden Einstufungen:
  - 3 wurden als `Verdächtig` eingestuft
  - 726 wurden als `Kein Problem` eingestuft

D1.11 CredScan

Diese Scans wurden mit den Standard-Regeln (Stand April 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 1.0.0-scan (als Teil von SAST-Scan Version 2.1.2) zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der CredScan Standard-Bewertung (error, warning, note). Die vom Werkzeug gelieferten, nicht durch uns ausgeschlossenen Findings wurden für die Auditierung in das zentrale Excel-Ergebnisdokument überführt, in welchem die Bewertung erfolgte.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D1.11.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-credscan-raw.sarif*
- *raw/vaultwarden-browser-extension-credscan-raw.sarif*

D1.11.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl zu bewertender Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 1x „error“

Browser-Extension

- 2x „error“

D1.11.3 Übersicht über Bewertung durch den Auditor

Zusammenfassung der Bewertung durch den Auditor (Details siehe Excel-Ergebnisdokument im Anhang):

Vaultwarden Server

- 1 bewertetes „error“-Finding resultierte in folgender Einstufung:
  - 1 wurden als `Kein Problem` eingestuft

Browser-Extension

- 2 bewertete „error“-Findings resultierten in folgenden Einstufungen:
  - 2 wurden als `Kein Problem` eingestuft

D2 SCA-Werkzeuge

Alle aufgefundenen (und um Duplikate bereinigte) Findings sind in folgenden Excel-Ergebnisdokumenten ersichtlich:

- *vaultwarden-server-sca.xlsx*

Als Duplikat erkannte Findings, sind in folgenden separaten Excel-Dateien dokumentiert:

- *vaultwarden-server-sca-excluded.xlsx*

Hinweis: Da die von den Tools gewonnen Informationen auf Basis bekannter und verbreiteter Datenbanken beruhen, nehmen wir alle Findings als „bestätigt“ an. Der Absatz „Übersicht über Bewertung durch den Auditor“ entfällt daher für diese Werzeugkategorie.
Eine feingranulare Bewertung (z.B. durch Nichtverwendung vulnerabler Bibliotheksanteile) kann aus unserer Erfahrung nur durch Projekt-Insider erfolgen.

D2.1 Semgrep Supply Chain

Für diese Scans kamen folgende Regelsätze (Stand März 2024) zum Einsatz:

- p/supply-chain

Die Scanner-Engine kam dabei in der Version 1.67.0 zum Einsatz.

Die initiale Finding-Einstufung erfolgte auf Basis der Semgrep Standard-Bewertung (error, warning, note). Alle Ergebnisse wurden in das SCA-Ergebnisdokument übernommen.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D2.1.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können den folgenden Dateien entnommen werden:

- *raw/vaultwarden-server-semgrep-supply-chain-raw.sarif*

D2.1.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl der Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 1 mit Tooleinstufung „warning“ (zu „error“ hochgestuft – siehe https://github.com/advisories/GHSA-r8w9-5wcg-vfj7)

Browser-Extension

Entfällt.

D2.2 Syft / Grype

Diese Scans wurden mit den Standard-Regeln (Stand März 2024) durchgeführt. Die Scanner-Engine kam dabei in der Version 0.89.0 (Syft) bzw. 0.66.0 (Grype) zum Einsatz.

Die initiale Finding-Einstufung erfolgte ohne Standard-Bewertung. Ergebnisse, welche neue Erkenntnisse gegenüber „Semgrep Supply Chain“ erbrachten, wurden in das SCA-Ergebnisdokument übernommen.

Im Folgenden werden die aus dem Scan gewonnenen Erkenntnisse zusammenfassend kurz dargestellt.

D2.2.1 Mitgelieferte Report-Artefakte

Die vom Werkzeug gelieferten (unauditierten) Roh-Daten können der folgenden Datei entnommen werden:

- *raw/vaultwarden_server_grype_raw.sarif*

D2.2.2 Grobübersicht über das rohe Scan-Ergebnis des Werkzeugs

Im Folgenden wird die Anzahl der Findings (pro Standard-Finding-Einstufung), sowie in Klammern die tatsächliche Anzahl gefundener, respektive der Anzahl ausgenommener Findings aufgelistet:

Vaultwarden Server

- 0 mit Tooleinstufung „error“ (1 gefunden und 1 von der Bewertung ausgenommen)

Browser-Extension

Entfällt.

E. Testergebnisse DAST-Analysen detailliert

In diesem Kapitel werden Beobachtungen aufgelistet, welche durch manuelle DAST-aufgefunden bzw. als Folge der Nachtests von hoch oder kritisch eingestuften SAST-Findings ausführlich beschrieben wurden.

E1 Authentisierung

Bedrohung

Unzureichende oder fehlerhafte Authentifizierung bleibt eine weitverbreitete Schwachstelle in Webanwendungen.

Authentifizierung bezeichnet die Bestätigung der Identität einer Entität, bspw. einer Person oder eines Gerätes. Hierfür können die Kenntnis einer Information, der Besitz an einer Sache oder biometrische Merkmale genutzt werden. Kenntnis-basierte Authentifizierung mittels Benutzername und Passwort ist nach wie vor die am häufigsten auftretende Form der Authentifizierung. Um die Sicherheit der Authentifizierung zu erhöhen, werden aber zunehmend mehrere Merkmale kombiniert, was als Mehrfaktorauthentifizierung (MFA) bezeichnet wird.

Im Zuge der Authentifizierung wird lediglich die Identität überprüft, es findet allerdings keine Kontrolle statt, ob der Zugriff auch erlaubt ist. Diese Überprüfung muss über die Autorisierung erfolgen.

E1.1 Unbefugte Änderung der Metadaten eines Emergency Access [hoch]

Beobachtung

Emergency Access ist eine Funktion von Vaultwarden, die es einem Benutzer (A*ccess Grantor*) erlaubt, einen vertrauenswürdigen Kontakt (G*rantee*) zu bestimmen, der im Notfall Zugang zu seinen Vault beantragen kann.

Beim Einrichten eines *Emergency Access* kann der A*ccess Grantor* einige Bedingungen für den gewährten Zugriff festlegen, darunter:

- *Access Level*: Bestimmt, ob der *Grantee* nur Lesezugriff auf die Daten des *Access Grantors* hat oder ob der *Grantee* das komplette Konto des *Access Grantors* übernehmen darf.
- Wartezeit: Definiert, wie lang der *Grantee* warten muss, bis er auf die Daten des *Access Grantor* zugreifen kann, nachdem der *Access Grantor* einen *Emergency Access* eingeleitet hat. Diese Wartezeit gibt dem *Access Grantor* ein Zeitfenster, in dem der *Access Grantor* den *Emergency Access* widerrufen kann.

Weitere Informationen zum Vaultwarden-*Emergency Access* können unter folgendem Link eingesehen werden:

https://bitwarden.com/help/emergency-access/

Sobald der *Access Grantor* einen *Emergency Access* erstellt, können die Metadaten des *Emergency Access* über den API-Endpunkt `/api/emergency-access/<access_UUID>` geändert werden. Dies ermöglicht es dem *Access Grantor*, die Bedingungen für den erstellten *Emergency Access* zu ändern. Einschließlich des *Access Level* und der Wartezeit.

Vaultwarden hat keinerlei Zugriffskontrolle an diesem API-Endpunkt zur Modifizierung der Metadaten implementiert. Dadurch könnte sogar ein nicht authentifizierter Benutzer die Metadaten eines *Emergency Access* ändern, falls ihm die UUID des

Emergency Access bekannt ist. Der folgende Screenshot veranschaulicht das Problem.

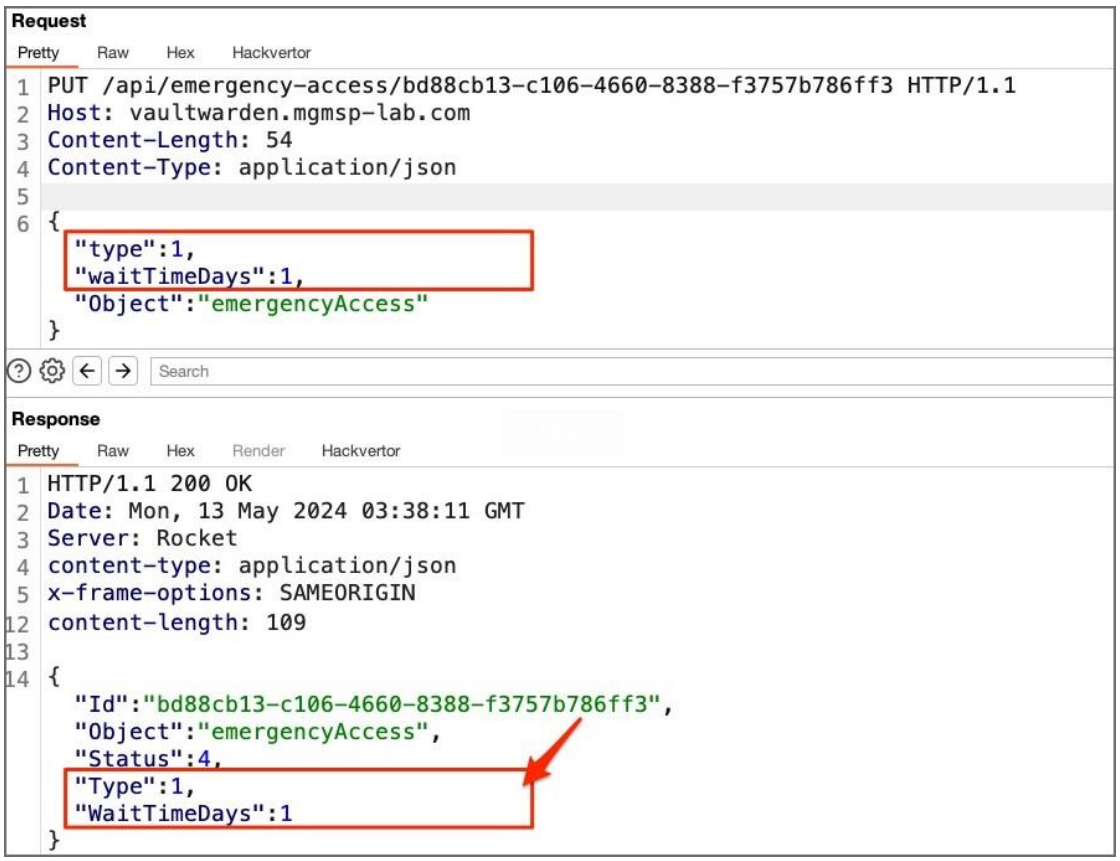


Abbildung 8: Fehlende Authentifizierungsprüfung im MetadataEndpunkt des Emergency Access

Um die Metadaten eines *Emergency Access* zu ändern, muss die UUID des *Emergency Access* mit angegeben werden. Vaultwarden verwendet UUID Version 4 für Entity IDs, die schwer enumerierbar sind. Für einen Angriff durch den *Grantee* ist dies jedoch irrelevant, da er diese Information in der entsprechenden Benachrichtigungsmail erhält.

Abbildung 9: Benachrichtigungsemail bei Bestimmung zum Grantee

Ausnutzbarkeit

Sobald ein *Emergency Access* eingerichtet ist, kann der *Grantee* die Zugangsbedingungen ändern, einschließlich des *Access Level* und der Wartezeit. Dadurch kann der *Grantee* auf die Daten des *Access Grantor* mit einer höheren Zugriffsstufe zugreifen. Das bedeutet, es wäre die vollständige Übernahme des Kontos anstelle des reinen Lesezugriffs möglich, ohne den Ablauf der festgelegten Wartezeit (standardmäßig 7 Tage) abwarten zu müssen. Dadurch werden die vom *Access Grantor* festgelegten Zugriffsbeschränkungen unwirksam.

Anmerkung: Die Schwachstelle betrifft nur den Metadaten-Endpunkt des *Emergency Access*. Der Endpunkt zur Erstellung eines *Emergency Access* prüft die benötigten Rechte korrekt. Die Ausnutzbarkeit dieser Schwachstelle ist daher darauf beschränkt, dass der *Grantee* sich höhere Privilegien für den Zugriff auf die Daten des *Access Grantor* geben kann.

Schwachstelle im Code

— Quelle: https://github.com/dani-garcia/vaultwarden/blob/1.30.3/src/api/core/emergency_access.rs#L115-L148

1.30.3 vaultwarden / src / api / core / emergency_access.rs ↑ Top

Code Blame Raw

```rust
#[put("/emergency-access/<emer_id>", data = "<data>")]
async fn put_emergency_access(emer_id: &str, data: JsonUpcase<EmergencyAccessUpdateData>, conn:
    post_emergency_access(emer_id, data, conn).await
}

#[post("/emergency-access/<emer_id>", data = "<data>")]
async fn post_emergency_access(
    emer_id: &str,
    data: JsonUpcase<EmergencyAccessUpdateData>,
    mut conn: DbConn,
) -> JsonResult {
    check_emergency_access_enabled()?;

    let data: EmergencyAccessUpdateData = data.into_inner().data;

    let mut emergency_access = match EmergencyAccess::find_by_uuid(emer_id, &mut conn).await {
        Some(emergency_access) => emergency_access,
        None => err!("Emergency access not valid."),
    };

    let new_type = match EmergencyAccessType::from_str(&data.Type.into_string()) {
        Some(new_type) => new_type as i32,
        None => err!("Invalid emergency access type."),
    };

    emergency_access.atype = new_type;
    emergency_access.wait_time_days = data.WaitTimeDays;
    if data.KeyEncrypted.is_some() {
        emergency_access.key_encrypted = data.KeyEncrypted;
    }

    emergency_access.save(&mut conn).await?;
    Ok(Json(emergency_access.to_json()))
}
```

Abbildung 10: Schwachstelle im Code

Maßnahme

Vaultwarden muss die benötigten Zugriffsrechte am Metadaten-Endpunkt, wie an anderen API-Endpunkten bereits geschehen, korrekt überprüfen. Nur der *Access Grantor* sollte die Metadaten eines *Emergency Access* ändern dürfen.

E2 Zugriffskontrolle

Bedrohung

Während bei der Authentifizierung die Identität eines Benutzers (oder einer technischen Entität) überprüft wird, werden bei der Zugriffskontrolle Richtlinien (Policies) durchgesetzt, die verhindern, dass Benutzer außerhalb ihrer vorgesehenen Berechtigungen handeln. Fehlende oder falsch implementierte Berechtigungsprüfungen können dem unerlaubten Zugriff auf Ressourcen Vorschub leisten, was häufig zur Offenlegung von Informationen oder der unerwünschten Änderung von Daten führt.

Eine fehlende Autorisierung resultiert zudem häufig in der Ausweitung von Privilegien. Darunter versteht man im Allgemeinen den unbefugten Zugriff auf höhere Privilegien innerhalb eines Systems.

E2.1 Fehlende Rotation der Organisationsschlüssel [hoch]

Beobachtung

Vaultwarden ermöglicht es Mitgliedern einer Organisation Daten (*Secrets*) miteinander zu teilen. Dazu werden alle Daten einer Organisation mit einem symmetrischen Schlüssel (dem Organisationsschlüssel) verschlüsselt, bevor sie serverseitig gespeichert werden. Jedes Mitglied einer Organisation erhält den Organisationsschlüssel während des Onboarding-Prozesses über einen Key-Sharing-Mechanismus.

Vaultwarden sieht jedoch keinen Offboarding-Prozess für Mitglieder, die eine Organisation verlassen, vor. Das bedeutet, dass der Organisationsschlüssel nicht rotiert wird, wenn ein Mitglied eine Organisation verlässt. Folglich besitzt das ausscheidende Mitglied, dem der Zugang eigentlich entzogen wurde, immer noch den verwendeten Organisationsschlüssel.

Das Fehlen der Schlüsselrotation kann mit den folgenden Schritten nachgestellt werden:

1. Erstellen einer Organisation mit mehreren Mitgliedern und *Secrets*
2. Notieren der verschlüsselten *Secrets*, die auf dem Server gespeichert sind
3. Entziehen des Zugangs eines Mitglieds zur Organisation
4. Die in Schritt 2 notierten, verschlüsselten *Secrets* wurden nach Schritt 3 nicht mehr geändert

Durch Ausnutzung der Schwachstelle E2.2 kann ein Benutzer, der eine Organisation verlassen hat, immer noch auf die verschlüsselten *Secrets* der Organisation zugreifen. Da der Benutzer, aufgrund der fehlenden Schlüsselrotation, aber immer noch in Besitz des Organisationsschlüssel ist, kann dieser die über E2.2 abgerufenen, verschlüsselten Daten wieder entschlüsseln. Dadurch erhält er unbefugt Zugriff auf die *Secrets* der Organisation. Dieser unbefugte Zugriff umfasst dabei auch *Secrets*, die nach dem Ausscheiden des Benutzers aus der Organisation erstellt bzw. aktualisiert wurden.

Der Angriff wird im Folgenden beispielhaft demonstriert.

Zunächst wird einem Testbenutzer (*Mallory* - die Angreiferin) Zugang zu einer Organisation, *Foobar*, gewährt.

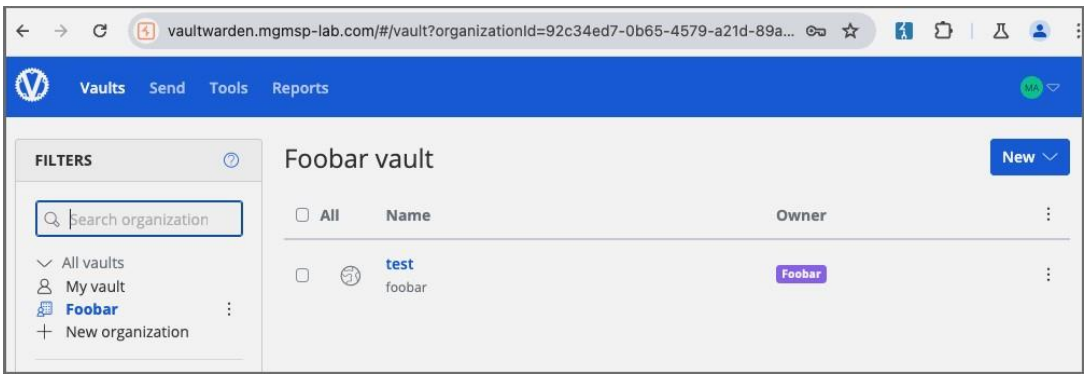


Abbildung 11: Mallory erhält Zugang zur Organisation Foobar

Durch den Onboarding-Prozesses erhält *Mallory* den Organisationsschlüssel. Sie notiert sich die Organisations-ID und den (verschlüsselten) Organisationsschlüssel aus der Antwort des GET-Endpunkts `/api/sync`.

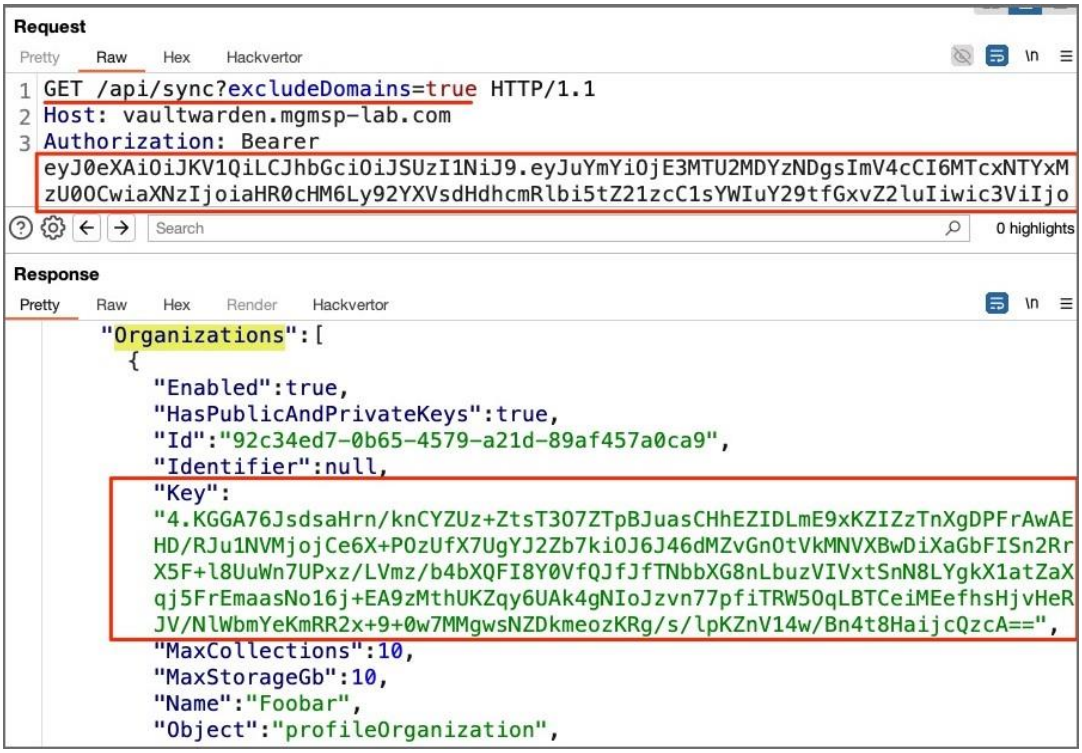


Abbildung 12: Mallory notiert sich die Organisations-ID und den Organisationsschlüssel

Als Nächstes entfernt *Alice* (eine Organisationsadministratorin) *Mallory* aus der *Foobar*-Organisation. Dadurch hat *Mallory* keinen Zugriff mehr auf die *Secrets* der Organisation.

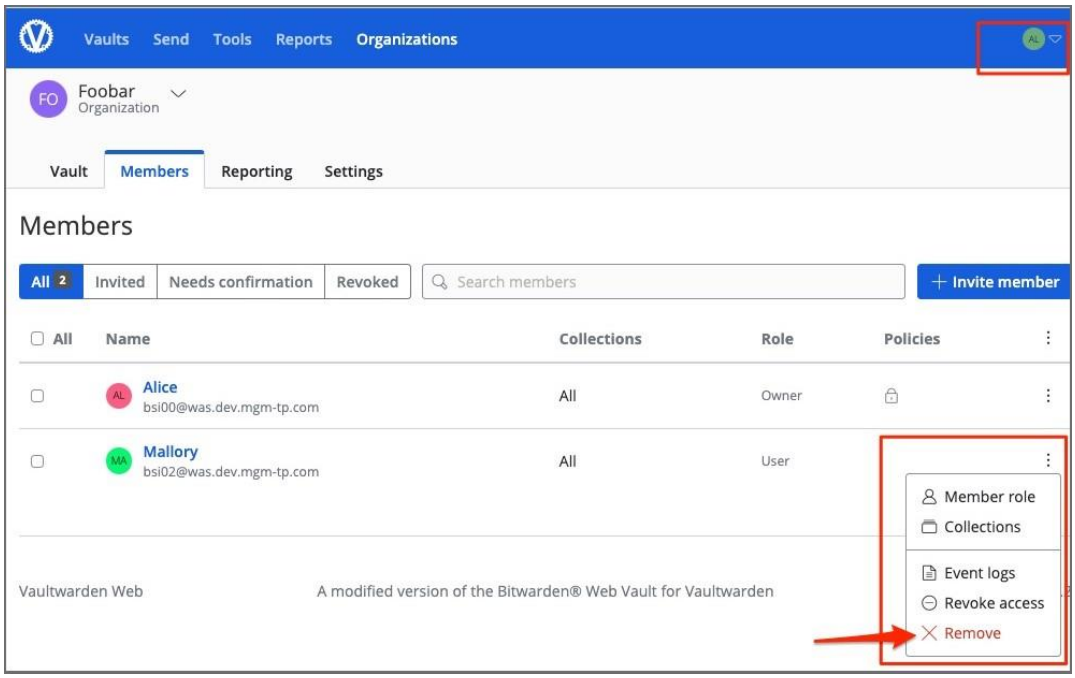


Abbildung 13: Admin der Organisation entfernt Mallorys Zugang

Als nächstes fügt *Alice* der *Foobar*-Organisation neue Secrets hinzu.

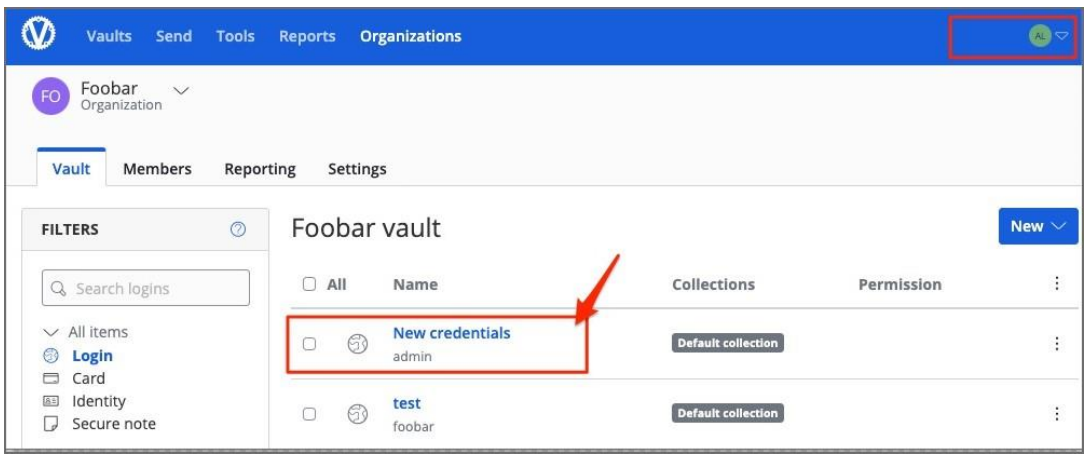


Abbildung 14: Ein neues Geheimnis wird dem Foobar-Organisationstresor hinzugefügt

Es kann über die Benutzeroberfläche von Vaultwarden bestätigt werden, dass *Mallory* keinen Zugriff mehr auf die *Secrets* der *Foobar*-Organisation haben sollte.

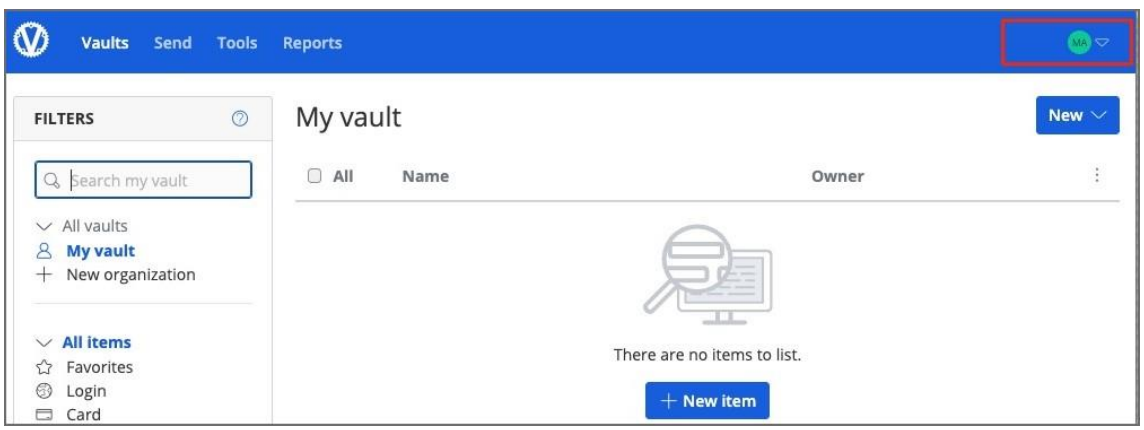


Abbildung 15: Fehlender Zugriff auf die Foobar-Organisation durch Mallory

Mallory nutzt nun erst die Schwachstelle E2.2 aus und gelangt so in Besitz der neu hinzugefügten (verschlüsselten) *Secrets* der *Foobar*-Organisation.

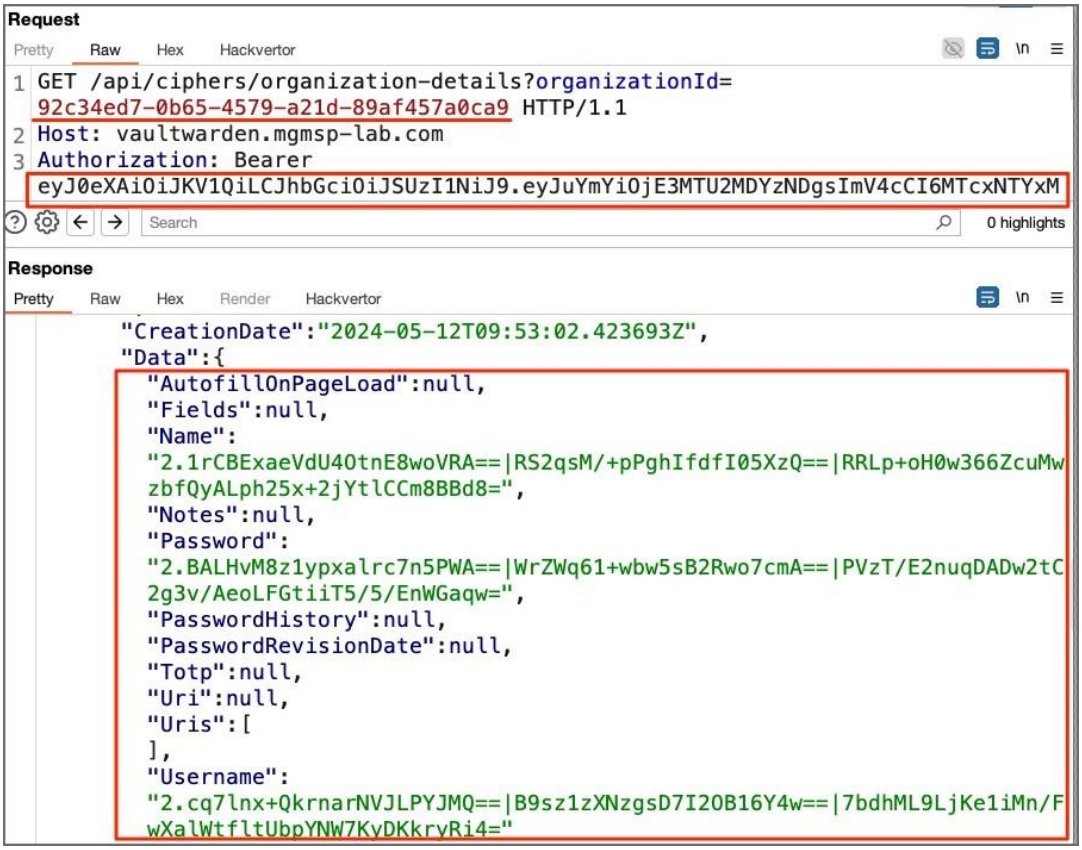


Abbildung 16: Mallory nutzt den „Unbefugten Zugriff auf verschlüsselte Unternehmensdaten“ aus

Mallory kann die so erhaltenen, verschlüsselten *Secrets* mit dem notierten Organisationsschlüssel wieder entschlüsseln. Dies kann auf verschiedene Weise geschehen:

1. In Anlehnung an das Bitwarden-Whitepaper könnte *Mallory* den verschlüsselten Organisationsschlüssel mit ihrem persönlichen Schlüssel entschlüsseln und dann den Klartext-Organisationsschlüssel zur Entschlüsselung der Organisationsdaten verwenden.
2. *Mallory* könnte den eigenen Datenverkehr mit der Anwendung manipulieren und so die Anwendung dazu bringen, die Daten für sie zu entschlüsseln. (Anmerkung: Da die Entschlüsselung anschließend clientseitig durchgeführt wird, könnte man die Entschlüsselung auch direkt über das lokale JavaScript durchführen.)

Für einen Proof-of-Concept wird die Methode #2 verwendet.

Mallory führt einen Refresh der Anwendung durch und manipuliert die Antwort auf den Synchronisations-Requests `/api/sync`. Im folgenden Screenshot ist der Parameter `Cipher` des Original-Request leer, da Mallory keinen Zugriff auf ein Secret hat (siehe oben). Da ihr auch der Zugriff auf die *Foobar*-Organisation entzogen wurde, ist auch das Feld `Organizations` leer. *Mallory* muss diese beiden Parameter im abgefangenen Request manuell mit Daten aus den vorangegangenen Schritten füllen.

- `Ciphers`: Enthält das Array der verschlüsselter Daten, die *Mallory* durch Ausnutzung der Schwachstelle E2.2 erhalten hat.
- `Organizations`: Enthält die Metadaten der *Foobar*-Organisation, die *Mallory* noch aus der Zeit, als sie noch Mitglied der Organisation war kennt.

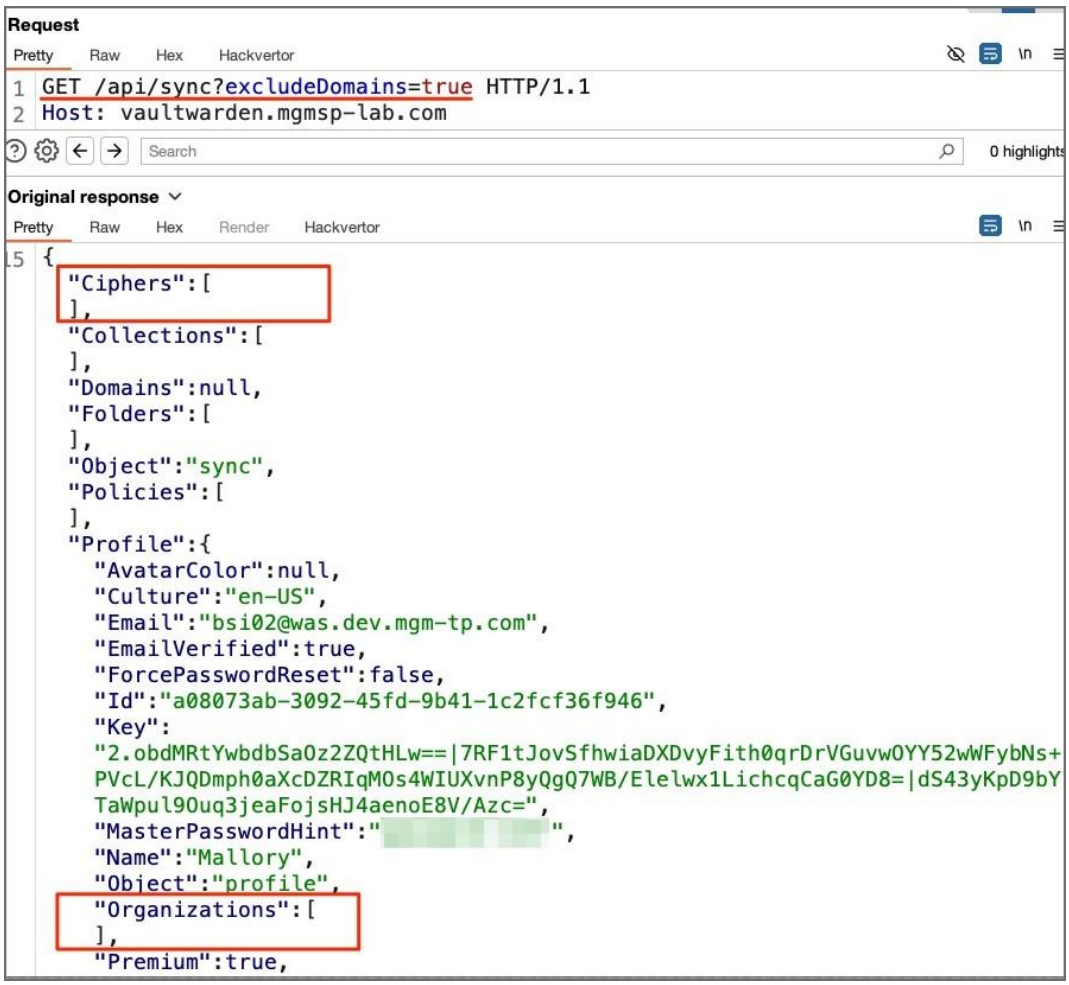


Abbildung 17: Parameter, die im abgefangenen Request manuell ausgefüllt werden müssen

Nachdem die erforderlichen Parameter entsprechend ersetzt wurden, wird die Antwort an einen Client weitergeleitet. Der Client führt anschließend automatisch die Entschlüsselung der Daten durch. Schließlich sieht *Mallory* die entschlüsselten Daten in der Benutzeroberfläche, mit allen *Secrets* der *Foobar*-Organisation.

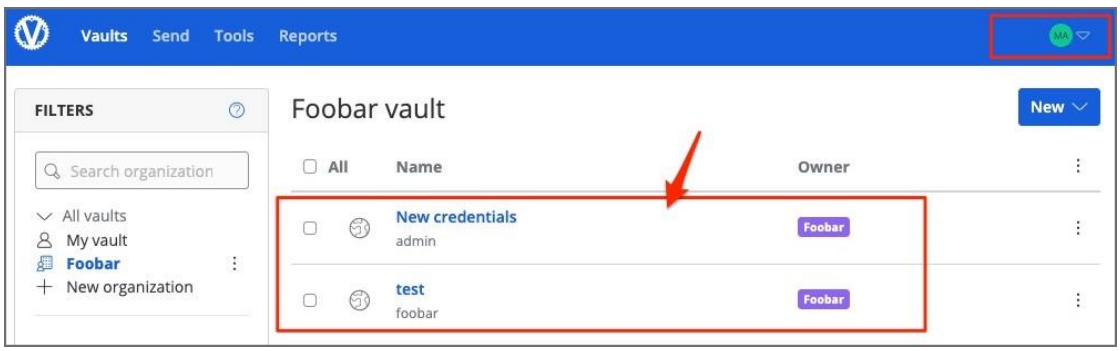


Abbildung 18: Mallory sieht alle Geheimnisse von Foobar im Klartext

Ausnutzbarkeit

Benutzer, denen der Zugang zu einer Organisation entzogen wurde, können immer noch Zugang zu den *Secrets* dieser Organisation erhalten. Einschließlich des Zugangs zu *Secrets*, die nach dem Zeitpunkt des Entzugs des Zugangs hinzugefügt wurden.

Schwachstelle im Code

Die fehlende Rotation der Schlüssel kann nicht mit einem konkreten Code-Snippet nachgewiesen werden. Deswegen kann hier trotz der Kritikalität *[hoch]* für dieses Finding keine Quelle im Code angegeben werden.

Maßnahme

Wenn ein Organisationsadministrator die Zugriffsrechte eines Mitglieds ändert, d.h. einem Mitglied den Zugriff entzieht, müssen die Organisationsdaten mit einem neu generierten Schlüssel neu verschlüsselt werden (Schlüsselrotation des Organisationsschlüssels). Dieser Prozess muss vom Client des Organisationsadministrators, der die Änderung vornimmt, durchgeführt werden. Der neue Organisationsschlüssel muss mit den anderen Organisationsmitgliedern über den bereits bestehenden Schlüsselaustauschmechanismus synchronisiert werden.

E2.2 Unautorisierter Zugriff auf verschlüsselte Daten [mittel]

Beobachtung

Vaultwarden schützt einige verschlüsselte, auf dem Server gespeicherte Daten nicht ausreichend.

Ein authentifizierter Benutzer könnte sich so unbefugten Zugriff auf verschlüsselte Daten einer beliebigen Organisation verschaffen, auch wenn der Benutzer nicht Mitglied der anvisierten Organisation ist. Jedoch müsste dem Nutzer die entsprechende `organizationId` bekannt sein. Dies ist im folgenden Screenshot zu erkennen. Der Nutzer „*Flapper*“ des Bearer-Tokens sollte eigentlich keinen Zugriff auf die gezeigten Informationen besitzen.

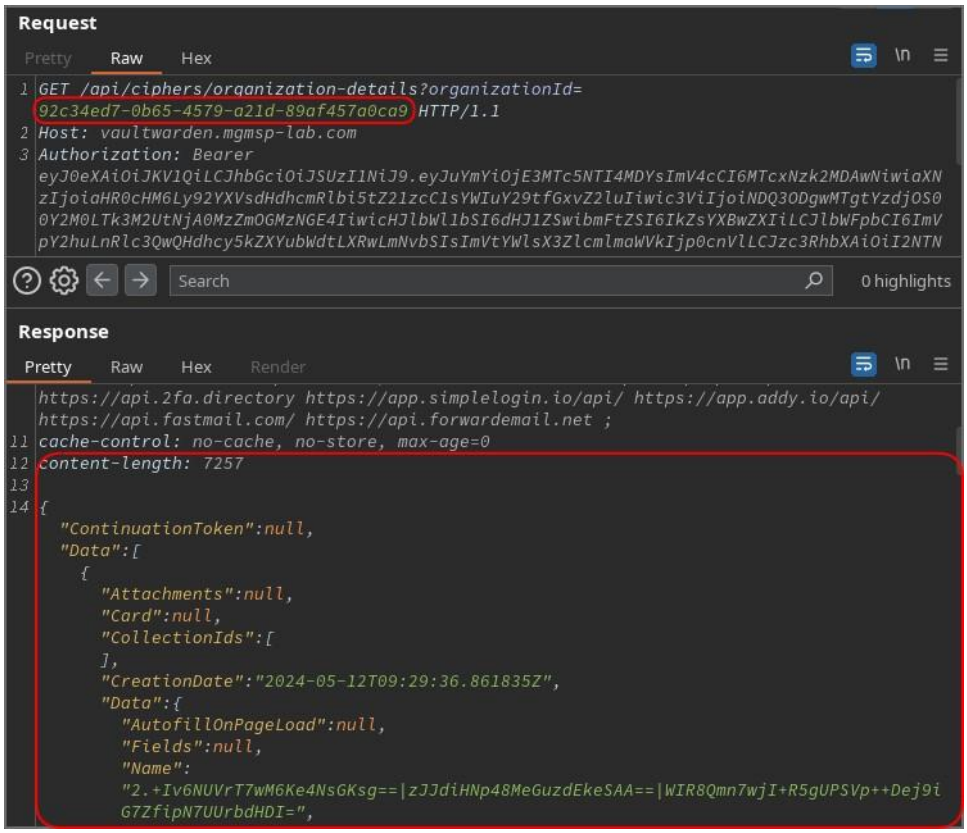


Abbildung 19: Rückgabe von Organisationsdaten ohne entsprechende Autorisierung

Außerdem gibt der API-Endpunkt `/api/organizations/< organizationId >/keys`, der neben dem *Public Key* auch den verschlüsselten *Private Key* der Organisation enthält, die angefragten Daten ohne vorherige Authentifizierung des Nutzers zurück.

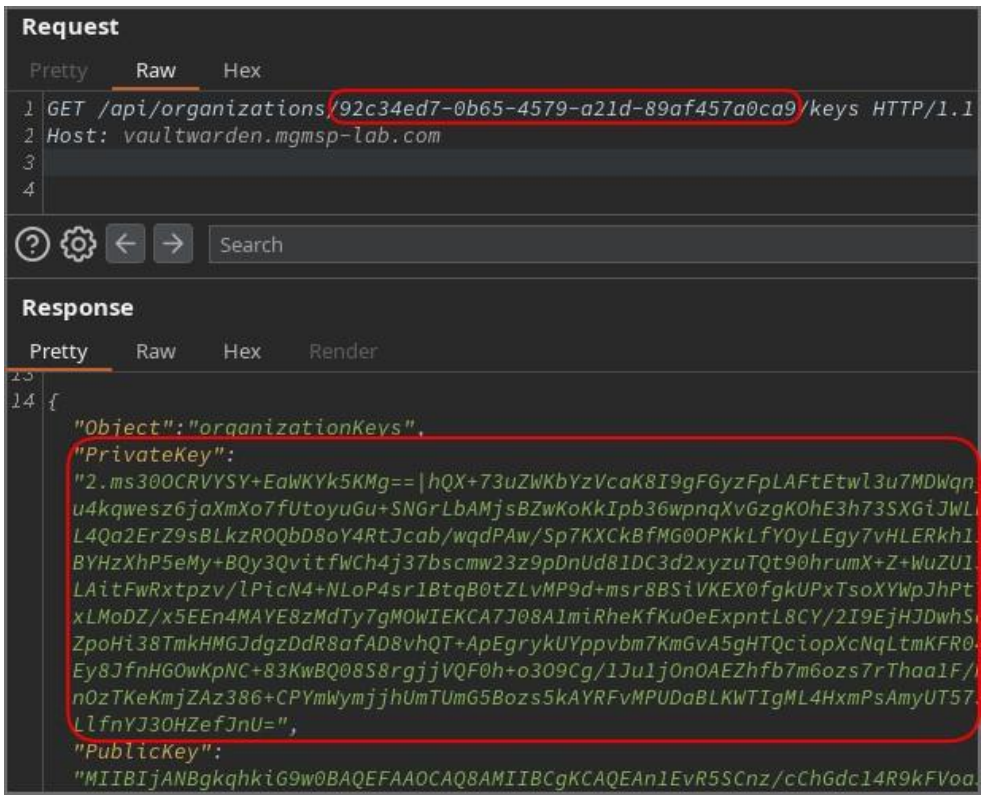


Abbildung 20: Rückgabe das verschlüsselten Private Key ohne Authentifizierung

Anmerkung: Die angefragten, unberechtigt erlangten Daten sind verschlüsselt und somit eigentlich nicht für einen Angreifer lesbar (siehe Finding E2.1)

Ausnutzbarkeit

Die geleakten Daten werden mit einer starken Verschlüsselung (AES 256-Bit-Verschlüsselung mit kryptografisch sicheren, zufällig generierten Schlüsseln) verschlüsselt. Damit ist es allein mit offengelegten Daten praktisch nicht möglich, diese zu entschlüsseln. Auch nicht in einem Offline-Brute-Force-Angriff.

Jedoch rotiert Vaultwarden die zur Verschlüsselung verwendeten Organisationsschlüssel nicht (siehe Finding E2.1). Im speziellen Fall, dass einem Benutzer der Zugang zu den Unternehmensdaten entzogen wurde, aber er bereits im Besitz des Organisationsschlüssels ist, könnte dieser die Schlüssel zur Entschlüsselung der geleakten Daten verwenden.

Anmerkung: Aufgrund der erhöhten Ausnutzbarkeit der Schwachstelle durch das Finding E2.1, sowie dem speziellen Schutzbedarf von Vaultwarden in Bezug auf die Verwaltung von Secrets, wird dieses Finding mit der Kritikalität *mittel* eingestuft.

Maßnahme

Die Zugriffskontrolle muss am API-Endpunkt `/api/ciphers/organization-details` wie im Rest der Anwendung implementiert werden, um sicherzustellen, dass nur Daten zurückgegeben werden, für die der anfragende Benutzer auch eine entsprechende Zugriffsberechtigung hat.

Der verschlüsselte *Private Key* sollte aus der Antwort des API-Endpunkts `/api/organizations/< organizationId >/keys` entfernt werden. Der *Private Key* sollte nur über einen anderen API-Endpunkt mit entsprechender Zugriffskontrolle zurückgegeben werden.

E2.3 Mögliche Härtungsmaßnahmen der Dockerumgebung [niedrig]

Beobachtung

Bei Verwendung des offiziellen Vaultwarden-Docker-Image wird der gestartete Serverdienst standardmäßig mit dem Benutzer `root` ausgeführt.

https://hub.docker.com/r/vaultwarden/server

```
[redacted]@BSI-CAOS3:~$ sudo docker container top vaultwarden -o user,pid,command,%cpu,%mem
USER                PID                 COMMAND             %CPU                %MEM
root                379887              /vaultwarden        0.0                 1.6
```

Abbildung 21: Vom Root-Nutzer ausgeführter Vaultwarden-Prozess

Dieses Verhalten kann auch durch die Analyse der Vaultwarden-Dockerdateien bestätigt werden:

https://github.com/dani-garcia/vaultwarden/blob/main/docker/Dockerfile.debian

Ausnutzbarkeit

Wird in der Dockerfile kein USER angegeben, werden Prozesse innerhalb des Containers mit `root`-Rechten gestartet. Die Ausführung von Anwendungen innerhalb

des Docker-Containers mit `root`-Rechten erhöht die Angriffsfläche und erleichtert die Ausnutzung bei einer erfolgreichen Kompromittierung des Systems.

Maßnahme

Docker-Container sollten nicht unter dem `root`-Nutzer, sondern einem Nutzer mit eingeschränkten Rechten, laufen.

Der Vaultwarden-Serverdienst kann ohne Root-Rechte ausgeführt werden. Entsprechende Dokumentation ist unter folgendem Link zu finden.

https://docs.docker.com/develop/develop-images/instructions/#user

E3 Cross-Site-Scripting

Bedrohung

Cross-Site-Scripting (XSS) bezeichnet allgemein das Ausnutzen einer Schwachstelle, indem Daten aus einem Kontext, in dem sie nicht vertrauenswürdig sind, z. B. Benutzereingaben, in einen anderen Kontext eingefügt werden, in dem sie als vertrauenswürdig eingestuft sind. Beispielsweise kann so gefährlicher JavaScript-Code ohne Prüfung in die HTML-Seiten eingebettet und im Kontext des Browsers eines Nutzers zur Ausführung gebracht werden.

XSS-Schwachstellen können vielfältig ausgenutzt werden, z. B.:

- Session-Hijacking (Das Stehlen der Session-ID und somit das Eindringen in den Account eines anderen Benutzers unter Umgehung des Logins).
- Website-Spoofing (Fälschung einer Website) zum Phishing (Stehlen von Login-Daten und persönlichen Daten) oder anderen Täuschungs- und Betrugsformen. Hierbei wird weiterhin die original URL der Website angezeigt und auch das Zertifikat bei einer HTTPS-Verbindung bezeugt weiterhin die Echtheit der Site – es bleibt unverletzt.
- Ausspionieren oder Manipulation der Daten auf dem System des Benutzers.
- Angriff auf den Browser um die vollständige Kontrolle des Clientsystems zu erhalten durch frei verfügbare Angriffstools (z. B. BeEF).

Es gibt verschiedene Formen von Cross-Site-Scripting. Beim **Stored-Cross-Site-Scripting** wird der Schadcode vom Angreifer innerhalb der Anwendung gespeichert und kommt jedes Mal zur Ausführung, wenn ein Benutzer die gespeicherten Daten aufruft.

Beim **Reflected-Cross-Site-Scripting** wird der Schadcode nicht gespeichert, sondern direkt in die Server-Response eingebettet. Die Ausführung eines solchen Angriffs geschieht am einfachsten durch Versenden einer E-Mail mit einem Link, auf den der Benutzer klicken muss, damit der Angriff ausgeführt wird. Bei bestimmten Einstellungen des E-Mail-Clients ist in manchen Fällen nicht mal dieser Klick nötig.

Als Sonderform gilt **DOM-Based-Cross-Site-Scripting**, bei welchem der Schadcode nicht durch den Server sondern auf Seite des Clients durch eine Modifizierung des DOMs in die Seite eingebettet und ausgeführt wird.

Wegen der aktuellen Verbreitung von Phishing-Angriffen und der Möglichkeit des Session-Hijacking bewerten wir XSS-Schwachstellen in der Regel mit dem Gefahrenpotenzial [hoch]. Wenn die Ausnutzung jedoch nur unter bestimmten Umständen möglich ist, wird das Gefahrenpotenzial reduziert. Eine Behebung ist trotzdem dringend anzuraten, da nicht ausgeschlossen werden kann, dass es doch Möglichkeiten für eine einfache Ausnutzung gibt.

E3.1 HTML-Injection möglich [mittel]

Beobachtung

mgm security partners GmbH – 11.06.2024

Vaultwarden wird mit einem Admin-Dashboard ausgeliefert. Diese Funktionalität kann nach der hier beschriebenen Anleitung aktiviert werden:

https://github.com/dani-garcia/vaultwarden/wiki/Enabling-admin-page

Nach der Aktivierung ist das Vaultwarden Admin Dashboard unter dem Pfad `/admin` verfügbar. Das Dashboard enthält eine Seite zur Benutzerverwaltung. Darüber kann ein Administrator zum Beispiel die Rollenzuweisung für Benutzer, die Mitglieder einer Organisation sind, über einen Dialog ändern.

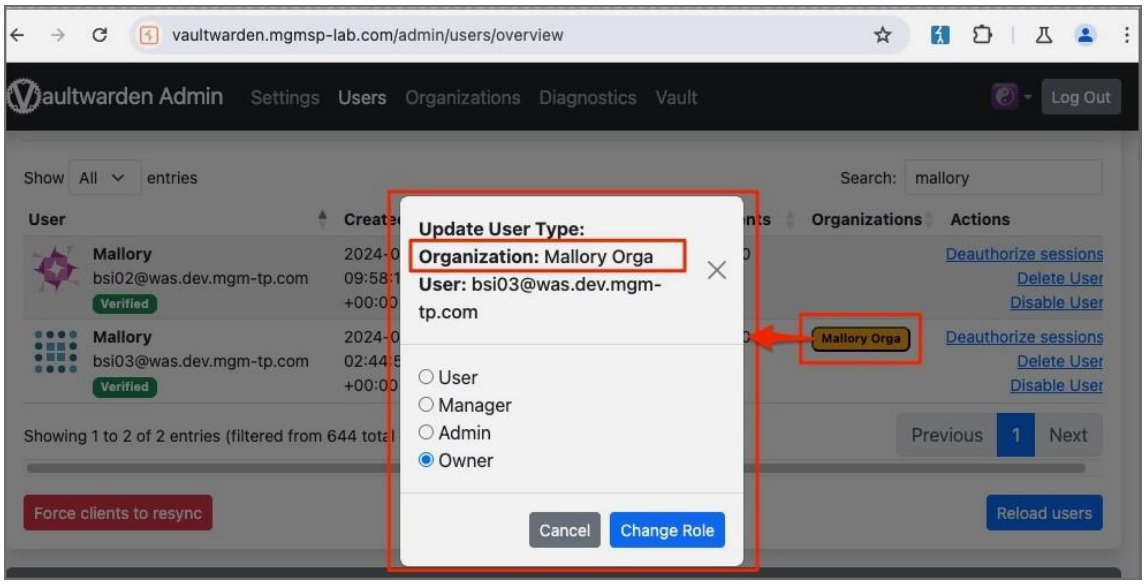


Abbildung 22: Benutzerverwaltung - Dialog zur Rollenzuweisung

Dieser Dialog ist anfällig für HTML-Injection. Payloads können in das Feld Organisationsname und E-Mail-Adresse injiziert werden. Als Proof-of-Concept, wurde die folgende Payload im Organisationsname gespeichert. Da es hierbei Längenbeschränkungen im Frontend gibt, muss der Request z.B. über den Burp Proxy abgefangen und entsprechend verändert werden.

```
<div style=position:absolute;width:100%;height:100%;background:red;z-
index:9>Foobar</div>
```

Die Payload im Organisationsnamen wird anschließend als HTML-Code im Dialog der Benutzerverwaltung geparst und entsprechend gerendert.

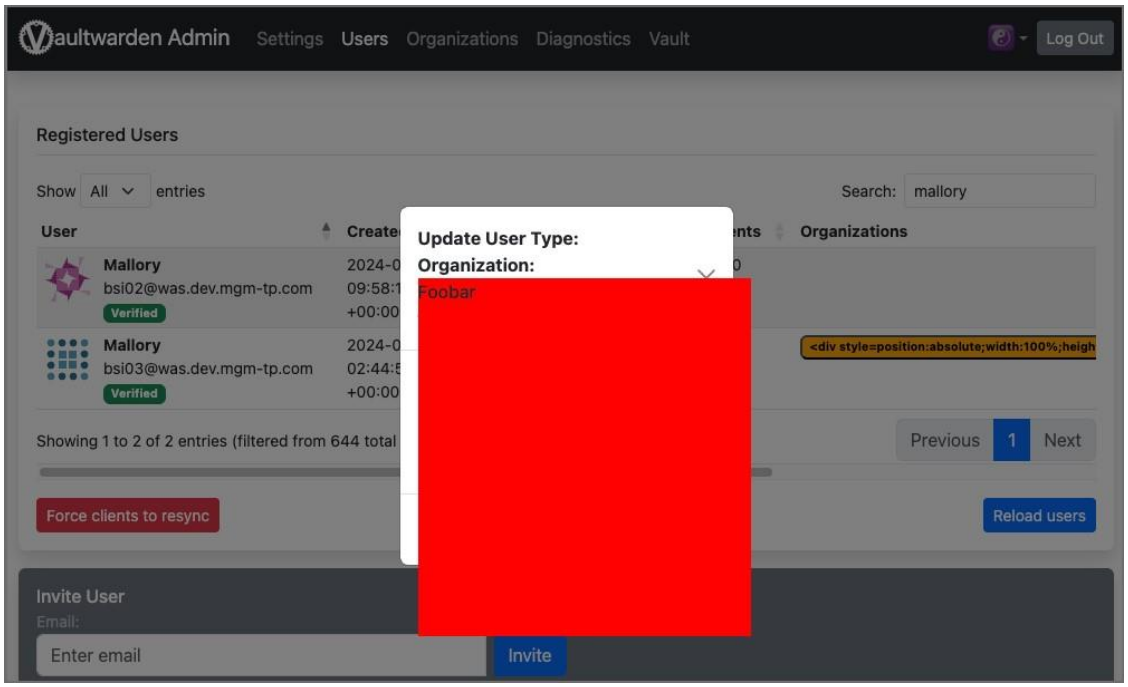


Abbildung 23: Eingeschleuster HTML-Code

Vaultwarden verwendet standardmäßig eine strenge *Content-Security-Policy* (CSP). Diese CSP verhinderte erfolgreich Versuche, JavaScript über die Schwachstelle auszuführen. Eine Eskalation der Ausnutzung dieser Schwachstelle von HTML-Injection zu Cross-Site Scripting (XSS) war dadurch während des Penetrationstests nicht möglich. Es kann jedoch folgende Payload dazu verwendet werden, um betroffene Nutzer beim Aufruf des Dialogs auf beliebige, andere Seiten weiterzuleiten.

```
<meta name=language content=1;https://mgm-sp.com HTTP-EQUIV=refresh />
```

Die Standard-CSP der Anwendung ist wie folgt definiert.

```
content-security-policy:
default-src 'self';
base-uri 'self';
form-action 'self';
object-src 'self' blob:;
script-src 'self' 'wasm-unsafe-eval';
style-src 'self' 'unsafe-inline';
child-src 'self' https://*.duosecurity.com https://*.duofederal.com;
frame-src 'self' https://*.duosecurity.com https://*.duofederal.com;
frame-ancestors 'self' chrome-
extension://nngceckbapebfimnlniiiahkandclblb chrome-
extension://jbkfoedolllekgbhcbcoahefnbanhhlh moz-extension://* ;
img-src 'self' data: https://haveibeenpwned.com ;
connect-src 'self' https://api.pwnedpasswords.com
https://api.2fa.directory https://app.simplelogin.io/api/
https://app.addy.io/api/ https://api.fastmail.com/
https://api.forwardemail.net ;
```

Anmerkung: Das Finding wurde durch die Semgrep Pro SAST-Analyse identifiziert (s. D1.2) und durch die dynamische Analyse bestätigt.

Ausnutzbarkeit

HTML-Injection-Schwachstellen sind ein Zeichen fehlender Input-Validierung. Durch das Einfügen von HTML-Tags ist es möglich die Optik und die Inhalte der Seite zuändern und Links zu bösartigen Seiten einzubetten, oder unter Umständen Skripte auszuführen.

Da das Admin-Dashboard Requests mit Hilfe von Cookies authentifiziert (siehe Finding E8.1), könnte ein eingebettetes HTML-Formular mit zusätzlicher Benutzerinteraktion auch dazu verwendet werden, andere zustandsändernde API-Endpunkte aufzurufen, die im Admin-Dashboard verfügbar sind.

Anmerkung: Die Ausnutzbarkeit dieses Findings ist auf HTML-Injektion beschränkt. Da jedoch trotzdem ein *Open-Redirect,* von dem Administratoren der Anwendung betroffen sein könnten, möglich ist, wird das Finding mit der Kritikalität [mittel] bewertet.

Schwachstelle im Code

— Quelle: https://github.com/dani-garcia/vaultwarden/blob/1.30.3/src/static/scripts/admin_users.js#L201

src > static > scripts > JS admin_users.js > updateUserOrgType

```
189   const userOrgTypeDialog = document.getElementById("userOrgTypeDialog");
190   // Fill the form and title
191   userOrgTypeDialog.addEventListener("show.bs.modal", function(event) {
192       // Get shared values
193       const userEmail = event.relatedTarget.parentNode.dataset.vwUserEmail;
194       const userUuid = event.relatedTarget.parentNode.dataset.vwUserUuid;
195       // Get org specific values
196       const userOrgType = event.relatedTarget.dataset.vwOrgType;
197       const userOrgTypeName = ORG_TYPES[userOrgType]["name"];
198       const orgName = event.relatedTarget.dataset.vwOrgName;
199       const orgUuid = event.relatedTarget.dataset.vwOrgUuid;
200
201       document.getElementById("userOrgTypeDialogTitle").innerHTML = `<b>Update User Type:</
          b><br><b>Organization:</b> ${orgName}<br><b>User:</b> ${userEmail}`;
202       document.getElementById("userOrgTypeUserUuid").value = userUuid;
203       document.getElementById("userOrgTypeOrgUuid").value = orgUuid;
204       document.getElementById(`userOrgType${userOrgTypeName}`).checked = true;
205   }, false);
206
```

Abbildung 24: Quellcode der für die HTML-Injection verantwortlichen Funktion.

Maßnahme

Benutzereingaben müssen vor der Eingabe serverseitig validiert und vor der Ausgabe im jeweiligen Kontext enkodiert werden. Im vorliegenden Fall müssen nicht vertrauenswürdige Benutzereingaben (Name der Organisation, E-Mail des Benutzers) HTML-kodiert werden, bevor sie in den finalen HTML-Code eingebettet werden.

E4 Server-Side Request Forgery (SSRF)

Server-Side Request Forgery kann immer dann auftreten, wenn eine Webabwendung selbst als Client agiert, d.h. ihrerseits eine weitere Anwendung oder einen Dienst aufruft und dabei Parameter nutzt, die sie vom äußeren Client (Browser, App, Fat-Client etc) bekommt und diese nicht ausreichend validiert.

Der äußere Client kann in diesem Fall die Kommunikation des Servers mit dem aufgerufenen Dienst manipulieren oder gar andere Dienste ansprechen. Das können externe Dienste (im Internet), interne Dienste (im Intranet) oder lokale Dienste (auf dem Server selbst) sein.

Beispiel:

Eine Anwendung wird wie folgt aufgerufen: http://example.com/action.do?update=backend/check. Die Anwendung verwendet den Wert des Parameters update, um daraus einen Dienstaufruf zu erzeugen, z.B. nach diesem Muster: http://backend/check.

Da ein Angreifer den Parameter mit beliebigen Werten belegen kann, muss an der Verwendungsstelle eine Validierung erfolgen. Ist dies nicht der Fall, würde beispielsweise mit dem vom Angreifer ausgetauschten String http://example.com/action.do?update=10.1.2.3:8123/get_account_balance der serverseitige Aufruf http://10.1.2.3:8123/get_account_balance abgesetzt werden.

Hinweis

SSRF-Schwachstellen lassen sich mit den Mitteln eines Penetrationstests in vielen Fällen nicht nachweisen. D.h. aus der Tatsache, dass keine Schwachstelle gefunden worden ist, lässt sich nicht auf das Nichtvorhandensein dieser Schwachstelle schließen. Um eine zuverlässige Negativaussage treffen zu können, ist in der Regel eine Analyse der betreffenden Codestellen erforderlich.

E4.1 Server-Side Request Forgery möglich [niedrig]

Beobachtung

Im Vaultwarden-Server konnten zwei *Server-Side-Request-Forgery*-Schwachstellen identifiziert werden.

In der Standardinstallation verwendet Vaultwarden einen internen *Icon-Service*, um Favicons von externen Websites auf den Vaultwarden-Server herunterzuladen. Diese Icons werden anschließend in der Benutzeroberfläche von Vaultwarden angezeigt. Ein Icon-Download kann durch eine nicht-authentifizierte Anfrage an den API-Endpunkt `/icons/<domain>/icon.png` ausgelöst werden.

Durch Setzen des Pfadparameters `<domain>` auf eine vom Angreifer kontrollierte Domäne, …

```
GET /icons/suqgizc607wzjk9ke3w2kpwp6gc70xom.mgmsp-lab.com/icon.png HTTP/1.1
Host: vaultwarden.mgmsp-lab.com
```

… wird ein Request vom Server an das vom Angreifer bestimmte Ziel abgesetzt.

#	Time	Type	Payload	Source IP address
1	2024-Apr-16 03:19:07.792 UTC	HTTP	suqgizc607wzjk9ke3w2kpwp6gc70xom	185.40.248.10
2	2024-Apr-16 03:19:07.607 UTC	DNS	suqgizc607wzjk9ke3w2kpwp6gc70xom	172.253.193.194
3	2024-Apr-16 03:19:07.608 UTC	DNS	suqgizc607wzjk9ke3w2kpwp6gc70xom	172.217.44.133
4	2024-Apr-16 03:19:07.871 UTC	HTTP	suqgizc607wzjk9ke3w2kpwp6gc70xom	185.40.248.10

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | Hex | Hackvertor

```
1  GET /favicon.ico HTTP/1.1
2  referer: https://suqgizc607wzjk9ke3w2kpwp6gc70xom.mgmsp-lab.com/
3  accept: text/html, text/*;q=0.5, image/*, */*;q=0.1
4  user-agent: Links (2.22; Linux X86_64; GNU C; text)
5  accept-language: en,*;q=0.1
6  cache-control: no-cache
7  pragma: no-cache
8  accept-encoding: gzip, br
9  host: suqgizc607wzjk9ke3w2kpwp6gc70xom.mgmsp-lab.com
10
11
```

Vaultwarden überprüft den übergebenen Parameter `<Domain>` und lehnt den Request ab, wenn die Domain zu einer internen IP aufgelöst wird. Die Validierung kann über die Vaultwarden-Log-Datei bestätigt werden. Im folgenden Beispiel wird der Request an den Host `localhost` korrekt abgelehnt.

```
      @BSI-CAOS3:/var/www/vw-data$ tail -n 10 vaultwarden.log
[2024-05-14 04:55:42.849][request][INFO] GET /alive
[2024-05-14 04:55:42.851][response][INFO] (alive) GET /alive => 200 OK
[2024-05-14 04:56:20.306][request][INFO] GET /icons/localhost/icon.png
[2024-05-14 04:56:20.309][vaultwarden::api::icons][DEBUG] IP 127.0.0.1 for domain 'localhost'
 is not a global IP!
[2024-05-14 04:56:20.309][vaultwarden::api::icons][WARN] Unable to download icon: Host resolv
es to a non-global IP. localhost
[2024-05-14 04:56:20.310][response][INFO] (icon_internal) GET /icons/<domain>/icon.png => 200
 OK
```

Abbildung 25: Abgelehnter Request an localhost

Die Validierung kann jedoch umgangen werden, wenn eine zugelassene (externe) Domain übergeben wird, die auf eine blockierte interne Domain weiterleitet. Dabei ist es auch möglich direkt auf eine interne IP weiterzuleiten. Vaultwarden folgt dann der Weiterleitung. Im folgenden Proof-of-Concept, wird beim Aufruf von der externen Domain `alcatraz.mgm-sp.team/favicon.icon` auf `http://localhost/admin` weitergeleitet.

```
→  CAOS3 curl https://alcatraz.mgm-sp.team/favicon.ico -i
HTTP/1.1 302 Found
Date: Tue, 14 May 2024 06:44:08 GMT
Server: Apache
Location: http://localhost/admin
Content-Length: 206
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://localhost/admin">here</a>.</p>
</body></html>
```

Abbildung 26: Weiterleitung auf eine interne Domain

Wird der Favicon-Download von der vorbereiteten externen Domain ausgelöst, folgte der Vaultwarden-Server dem Redirect und sendet anschließend eine Anfrage an den angegeben internen Host. Das Verhalten kann über die Vaultwarden-Logdatei bestätigt werden:

```
[redacted]@BSI-CAOS3:/var/www/vw-data$ tail -n 22 vaultwarden.log

[2024-05-14 06:55:04.241][reqwest::async_impl::client][DEBUG] redirecting 'https://alcatraz.mgm-sp
.team/favicon.ico' to 'http://localhost/admin'
[2024-05-14 06:55:04.243][reqwest::connect][DEBUG] starting new connection: http://localhost/
[2024-05-14 06:55:04.244][request][INFO] GET /admin
[2024-05-14 06:55:04.244][vaultwarden::api::admin::_][WARN] Request guard `AdminToken` is forwardi
ng.
[2024-05-14 06:55:04.245][response][INFO] (admin_page_login) GET /admin/ [2] => 200 OK
[2024-05-14 06:55:04.246][vaultwarden::api::icons][DEBUG] Icon from https://alcatraz.mgm-sp.team/f
avicon.ico, is not a valid image type
```

Abbildung 27: Folgenden des Redirects an einen internen Host

Vaultwarden prüft die Antwort des Server-Side-Request. Nur wenn die Antwort auf einen Bildinhalt hinweist (beginnend mit der Dateisignatur bzw. den *Magic Bytes* eines Bildes), wird der Inhalt an den Benutzer zurückgegeben.

Zudem ist API-Endpunkt zur Einrichtung der Duo 2FA-Authentifizierung (`/api/two-factor/duo`) anfällig für *Server-Side-Request-Forgery*. Der Zielhost kann, wie im Screenshot zu sehen, über den Parameter `host` angegeben werden.

Request

Pretty Raw Hex Hackvertor

```
1 PUT /api/two-factor/duo HTTP/1.1
2 Host: vaultwarden.mgmsp-lab.com
3 Authorization: Bearer
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJuYmYiOjE3MTU2NzU4MTQsImV4cCI6MTcxNTY4M
4 Connection: close
5 Content-Length: 183
6
7 {
    "masterPasswordHash":"[redacted]",
    "integrationKey":"integrationkey",
    "secretKey":"secretkey",
    "host":"suqgizc607wzjk9ke3w2kpwp6gc70xom.mgmsp-lab.com"
  }
```

Abbildung 28: SSRF über die Duo 2FA-Authentifizierung

Der obige Request führt anschließend zum folgenden, vom Vaultwarden-Server abgesetzten Request.

#	Time	Type	Payload	Source IP address
19	2024-May-14 08:58:27.119 UTC	HTTP	suqgizc607wzjk9ke3w2kpwp6gc70xom	185.40.248.10
18	2024-May-14 08:37:09.880 UTC	DNS	suqgizc607wzjk9ke3w2kpwp6gc70xom	172.253.225.37

Description Request to Collaborator Response from Collaborator

Pretty Raw Hex Hackvertor

```
1 GET /auth/v2/check HTTP/1.1
2 authorization: Basic aW50ZWdyYXRpb25rZXk6MDQwNjYxZDI4MTI5NTU1N2FkZWRlZDE3ZmRiN
3 user-agent: vaultwarden:Duo/1.0 (Rust)
4 date: Tue, 14 May 2024 08:58:27 +0000
5 accept: */*
6 accept-encoding: gzip, br
7 host: suqgizc607wzjk9ke3w2kpwp6gc70xom.mgmsp-lab.com
8
```

Abbildung 29: Empfangener SSRF-Request

Ausnutzbarkeit

Server-Side-Request-Forgery (SSRF) ermöglicht einem Angreifer das Auslösen bestimmter HTTP-Anfragen durch den Server. Die weitere Ausnutzbarkeit hängt stark von den anderen internen Diensten ab, die im internen Netzwerk bzw. auf dem Server der getesteten Anwendung laufen.

Ein Angreifer könnte über diese Schwachstelle HTTP-GET-Anfragen an beliebige interne Endpunkte auslösen. Die Anwendung gibt jedoch nur dann den Inhalt der Antwort des SSRF-Angriffs an den Benutzer weiter, wenn der Inhalt der Antwort ein Bild ist.

Anmerkung: Durch dies Einschränkung handelt es sich um eine (fast) blinde SSRF-Schwachstelle. Daher wird dieses Finding nur mit der Kritikalität [niedrig] eingestuft.

Maßnahme

Im Falle einer Redirect-Antwort von einem externen Server sollte Vaultwarden vor dem Redirect den neuen Ziel-Host noch einmal überprüfen, um sicherzustellen, dass er nicht zum internen Netzwerk gehört.

Alternativ könnte Vaultwarden Redirects auch generell ablehnen, wenn der Redirect zu einem neuen Host führt.

E5 Session-ID im URL oder Referrer (Referrer-Leck)

Wird die Session-ID per Cookie transportiert, so sollte diese zu keinem Zeitpunkt auch über den URL transportiert werden. Denn hierdurch ergeben sich zusätzliche Ansatzpunkte, um sie auszuspähen:

- Im Browser des Benutzers (History, ggf. Bookmarks, "über die Schulter Schauen") für denjenigen, der physikalischen Zugang zum Browser des Benutzers hat
- Kopieren und Versenden von Seitenausschnitten, die den Link mit der Session-ID enthalten
- Server-Logs/Proxy-Logs, auf die möglicherweise Dritte Zugriff haben
- Über den Referrer-Header, der den URL der Seite enthält, auf welcher der Klick erfolgt ist oder von welcher das Laden von Bildern etc. erfolgt ist. Wenn es sich dabei um externe Websites handelt, gelangt die Session-ID über den Referrer in die Hände Dritter.

E5.1 Übertragung sensibler Daten in der URL [niedrig]

Beobachtung

Nach einer erfolgreichen Anmeldung baut die Webanwendung zusätzlich eine Websocket-Verbindung zum Vaultwarden-Server auf. Die entsprechende Anfrage wird mit dem in der URL transportierten JWT des Nutzers authentifiziert.

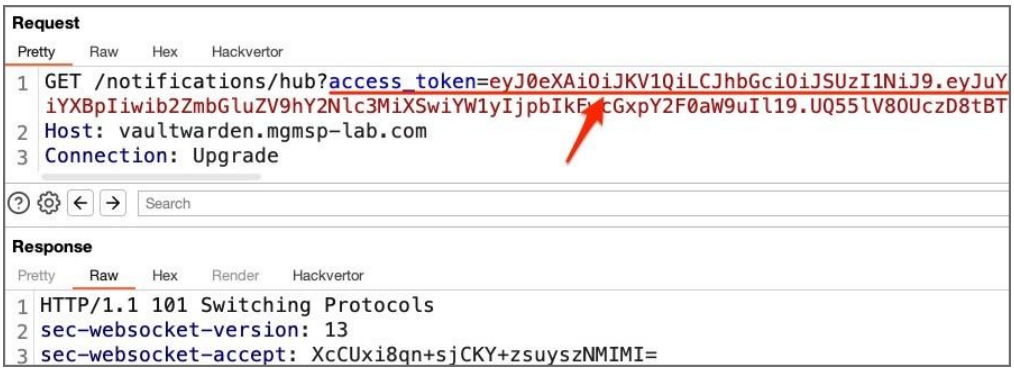


Abbildung 30: Transport des JWT in der URL beim Protokollwechsel

Folglich wird der vollständige JWT des Benutzers zum Beispiel im Log des verwendeten Apache-Servers protokolliert.

```
[redacted]@BSI-CAOS3:~$ sudo cat /var/log/apache2/access.log | grep access_token | tail
10.54.20.25 - - [14/May/2024:04:11:17 +0000] "GET /notifications/hub?access_token=eyJ0eXAiOiJKV1QiLC
.eyJuYmYiOjE3MTU2NTU5NDQsImV4cCI6MTcxNTY2MzE0NCwiaXNzIjoiaHR0cHM6Ly92YXVsdHdhcmRlbi5tZ21zcC1sYWIuY2
joiYTA4MDczYWItMzA5Mi00NWZkLTliNDEtMWMyZmNmMzZmOTQ2IiwicHJlbWl1bSI6dHJ1ZSwibmFtZSI6Ik1hbGxvcnkiLCJlb
hcy5kZXYubWdtLXRwLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJzc3RhbXAiOiI4ODgwM2Y0NC1jNzBiLTQzZDctYjI1YS1
CJkZXZpY2UiOiJjNDJkYjU5Ny03YWMwLTQ0NGUtYWViNS00NmUzNmMyYjEzNWEiLCJzY29wZSI6WyJhcGkiLCJvZmZsaW5lX2Fj
iQXBwbGljYXRpb24iXX0.QqcqlIrUbF2tS7rc9FfBap26Eh6T3QjaY13DH--6pbGd0cd7_U_QEG1UJAscL2iX8aZt0tcV72SWmx
VYDBoiA3wvD7-hoXzxjQ1OFUrogEG3yLz4LR3jq7mOP6ZmeRvIb52eeo5JZmntsRHbhvjxqolWS6XOOki_7aS1kK9rAK5efwHcA
LX0cL4SryW1lxuSAvqFzCANAbxbIFbPvHcY3kRZlXVRs8OiU1bycHa_fq0iuYA81mPc3-gQc8dEUhjMP9wyeKGd_1jVpMCExu_X
iGB_X9kZA HTTP/1.1" 101 6009 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.3
o) Chrome/123.0.0.0 Safari/537.36"
10.54.20.25 - - [14/May/2024:04:13:11 +0000] "GET /notifications/hub?access_token=eyJ0eXAiOiJKV1QiLC
```

Abbildung 31: Speicherung des vollständigen JWT in einer Log-Datei

Ausnutzbarkeit

Ein in der URL transportiertes JWT könnte an verschiedenen Stellen protokolliert werden, z. B. im Browserverlauf des Benutzers oder in Log-Dateien zwischengeschalteter Proxy-Server bzw. des Webserver.

Gelangt ein JWT über die exponierten Stellen an einen Angreifer, könnte es dazu verwendet werden, sich gegenüber der Anwendung als rechtmäßiger Benutzer auszugeben. Dies könnte schließlich zur Kompromittierung des Nutzerkontos führen.

Im Zusammenhang mit Vaultwarden könnte ein Angreifer, der im Besitz des JWT eines Nutzers ist, bestimmte Änderungen an den persönlichen und organisatorischen Einstellungen des Nutzers vornehmen. Dazu gehören Aktionen, die keine erneute Authentifizierung mit dem Master-Passwort-Hash erfordern:

- Löschen des *Emergency-Access*
- Ändern der Rollen von Organisationsmitgliedern

Anmerkung: Bei Besitz eines JWT könnte ein Angreifer zwar auch verschlüsselte Daten des entsprechenden Nutzers abrufen, aber er wäre nicht in der Lage, die Daten auch zu entschlüsseln. Die Entschlüsselung erfordert Kenntnis über das Master-Passworts des Benutzers. Deswegen wird das Finding nur mit einem geringen Risiko bewertet.

Maßnahme

Sensible Informationen sollten niemals innerhalb der URL transportiert werden. Der Websocket-Endpunkt sollte die Berechtigung des anfragenden Benutzers auf der Grundlage des in einem benutzerdefinierten HTTP-Header transportierten Zugriffstokens überprüfen, anstatt zu diesem Zweck URL-Abfrageparameter zu verwenden.

E6 Logik

Während viele Sicherheitsprobleme auf fehlerhafte oder fehlende Sicherheitskontrollen wie Authentifizierung oder Eingabevalidierung zurückzuführen sind, missbrauchen Schwachstellen in der Geschäftslogik legitime Verarbeitungsabläufe einer Anwendung auf unvorhergesehene Weise. Die Auswirkungen hängen stark von der konkreten Anwendung ab, umfassen jedoch häufig Timing-Angriffe, die Unterbrechung der beabsichtigten Schrittfolge oder den wiederholten Aufruf einer Funktion.

Da automatisierte Scans in der Regel nicht in der Lage sind, die Geschäftslogik einer Anwendung zu berücksichtigen, muss auf diese Kategorie von Schwachstellen manuell getestet werden.

E6.1 Denial-of-Service: IP-basierte Benutzersperrung möglich [niedrig]

Beobachtung

Vaultwarden verhindert das Erraten von Anmeldedaten, indem die anfragende IP nach mehreren fehlgeschlagenen Anmeldeversuchen für eine gewisse Zeit für den Zugriff auf den Anmeldeendpunkt gesperrt wird. Wenn sich anschließend ein Nutzer von einer blockierten IP aus bei einem beliebigen Konto anmelden möchte, antwortet die Anwendung immer mit einer 429-Statuscode.

Abbildung 32: Fehlermeldung bei IP-basierter Benutzersperrung

Ausnutzbarkeit

Die IP-basierte Zugriffsbeschränkung funktioniert möglicherweise nicht wie erwartet, wenn viele Benutzer von derselben IP-Adresse auf die Anwendung zugreifen. Dies ist ein häufiges Szenario für Benutzer, die öffentliche Proxys verwenden, oder für Benutzer in einem Unternehmen, in dem der ausgehende Internetverkehr über ein einziges Netzwerk-Gateway läuft. In einem solchen Fall kann ein Brute-Force-Angriff dazu führen, dass die gemeinsame IP vieler Benutzer blockiert wird, wodurch die Anwendung für die legitimen Benutzer nicht mehr verfügbar ist (Denial-of-Service).

Schwachstelle im Code

— Quelle: https://github.com/dani-garcia/vaultwarden/blob/1.30.3/src/ratelimit.rs#L22-L29

Die Überprüfung der Zugriffsbeschränkung ist in der Datei `src/ratelimit.rs` implementiert. Die entsprechende Funktion ist `check_limit_login`. Wie im Quellcode zu sehen ist, berücksichtigt die Funktion nur die Quell-IP der Anfrage. Die Parameter des Schwellenwerts werden durch die Umgebungsvariablen *LOGIN_RATELIMIT_SECONDS* und *LOGIN_RATELIMIT_MAX_BURST* definiert. Standardmäßig wird eine Sperre ausgelöst, wenn innerhalb von 60 Sekunden 10 geflaggte Anfragen von einer IP-Adresse eingehen.

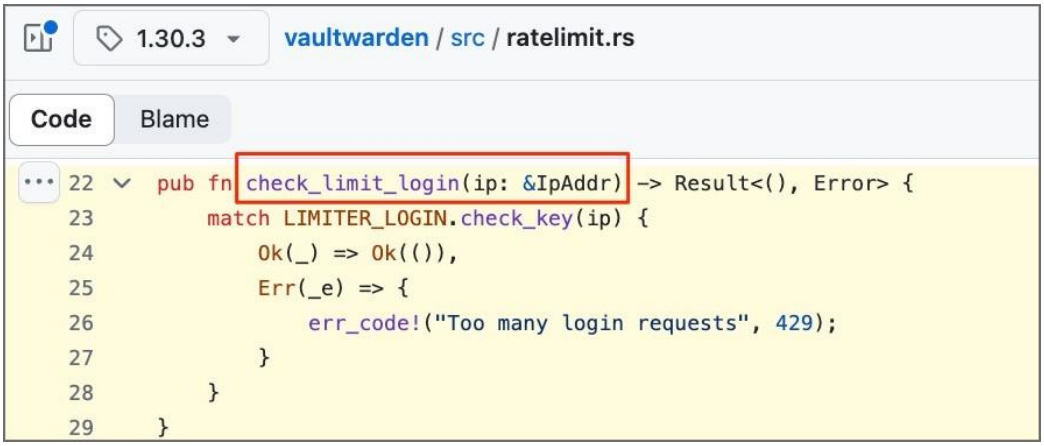


Abbildung 33: Denial-of-Service - Schwachstelle im Quellcode

Maßnahme

Die Lösung besteht in der Regel nicht darin, den Benutzer oder seine verwendete IP nach einer bestimmten Anzahl an Fehlversuchen auszusperren. Vor allem das Sperren von IP-Adressen kann unter Umständen mehr als einen Benutzer aussperren, z. B. wenn die IP-Adresse eines öffentlichen genutzten Proxys oder ein Proxy einer großen Firma geblockt wird.

Es gibt verschiedene Maßnahmen, um DoS gegen Benutzerkonten für einen Angreifer zu erschweren:

- Hinzufügen einer Zeitverzögerung während des Logins, nach mehreren fehlgeschlagenen Anmeldeversuchen ("Teergrube").
- Verwendung von klassischen CAPTCHAs oder CAPTCHA-Alternativen wie "Fun CAPTCHAs" oder Google s reCAPTCHA, die nach einer bestimmten Anzahl von Fehlversuchen gelöst werden müssen und den Benutzer dazu auffordern, Zuordnungen basierend auf Form, Farben oder logischen Zusammenhängen zu treffen.
  Erkennen von Password-Cracking-Versuchen durch Monitoring des Verkehrs mit Hilfe einer Web-Application-Firewall oder eines IDS/IPS, das nach definierten Regeln reagiert und automatisiert Gegenmaßnahmen einleitet.

E6.2 Unzureichende Anti-Automatisierung [niedrig]

Beobachtung

Vaultwarden schützt einige seiner API-Endpunkte mit einem strengen Rate-Limiting, um automatisierte Angriffe zu verhindern. Der Endpunkt für die Abfrage des Passworthinweises ist jedoch nicht durch diesen Schutz abgedeckt.

Das ermöglicht es jedem nicht authentifizierte Benutzer, ein automatisiertes Skript zu verwenden, um eine große Anzahl von Passwort-Hinweis-Anfragen zu senden. Dadurch würde die Anwendung die entsprechende Zahl an E-Mails an die angegebene E-Mail-Adresse senden. Wie im folgenden Screenshot zu sehen ist, wurden innerhalb eines kurzen Zeitraums über hundert E-Mails über die Anwendung an ein Testpostfach gesendet.

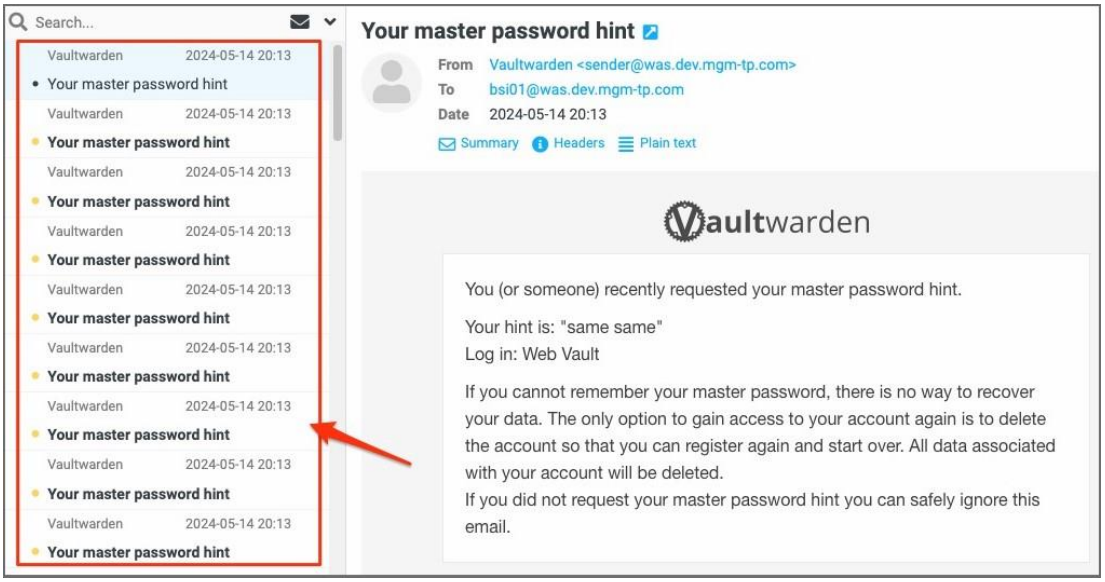


Abbildung 34: E-Mail-Flooding via Passworthinweis

Zusätzlich muss der Nutzer bei der Änderung der E-Mail-Adresse des zugehörigen Kontos bestätigen, dass er tatsächlich der Eigentümer der neuen E-Mail-Adresse ist. Dies geschieht durch die Übermittlung eines 6-stelligen Verifizierungscodes, der an die neue E-Mail-Adresse gesendet wird. Vaultwarden hat ebenfalls kein Rate-Limiting am Endpunkt für die Übermittlung des Verifizierungscodes implementiert. Folglich könnte es möglich sein, den Code zu erraten.

Während des Tests wurden über 100 Anfragen mit ungültigem E-Mail-Verifizierungscode gesendet. Danach war eine Anfrage mit dem gültigen Code immer noch erfolgreich. Die erfolgreiche Anfrage ist im folgenden Screenshot mit dem HTTP-Statuscode 200 für den korrekten Verifizierungscode in Anfrage #102 zu sehen.

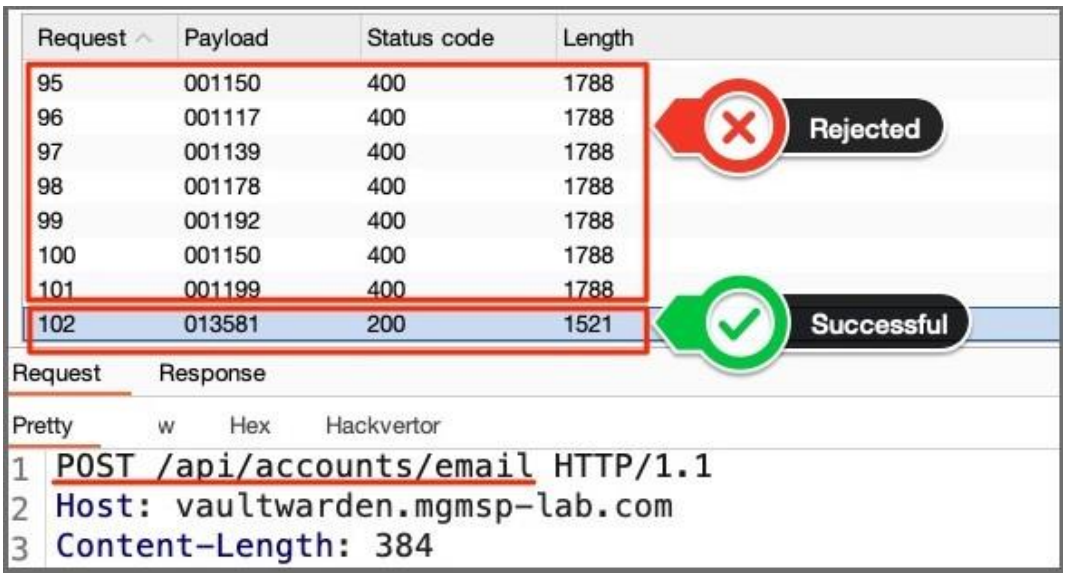


Abbildung 35: Erfolgreiches Erraten des Verifizierungscodes

Ausnutzbarkeit

Angreifer könnten eine große Anzahl von Anfragen an die Anwendung senden, die auf Funktionen abzielen, die umfangreiche Systemressourcen beanspruchen, um einen Denial-of-Service-Angriff durchzuführen. Infolgedessen könnte das System überlastet werden und für andere Benutzer nicht mehr verfügbar sein.

Ein weiterer Angriffsvektor ist E-Mail-Flooding. Ein umfassender Angriff auf die Funktion zum Erhalten des Passworthinweises könnte dazu führen, dass der verwendete E-Mail-Server von anderen E-Mail-Servern auf eine schwarze Liste gesetzt wird. Folglich könnten von der Anwendung gesendete E-Mails bei einigen E-Mail-Diensten als Spam markiert werden. Da der Angriff in diesem Fall nur gegen bestehende Benutzer (registrierte E-Mail-Adressen) durchgeführt werden kann ist, der Kreis der potenziellen betroffenen Nutzer begrenzt

Zudem kann ein Angreifer bei einem erfolgreichen Brute-Force-Angriff auf den Verifizierungscode, den Schritt der E-Mail-Verifizierung umgehen und eine E-Mail-Adresse, die ihm nicht gehört, seinem eigenen Konto zuweisen. Dies könnte zum Beispiel bei Social Engineering-Angriffen nützlich sein.

Anmerkung: Die beiden betroffenen API-Endpunkte erfordern keine Authentifizierung.

Maßnahme

Für ein bestimmtes Benutzerkonto (E-Mail-Adresse) sollte die Anwendung den Passwort-Hinweis nur einmal innerhalb eines bestimmten Zeitfensters, z. B. eines Tages, senden. Wiederholte Anfragen für dasselbe Benutzerkonto sollten ignoriert werden.

Auf den Endpunkt zur Überprüfung der E-Mail-Adresse (`/api/accounts/email`) sollte, wie für andere Endpunkte bereits existierend, ein strenges Rate-Limiting angewendet werden.

E6.3 Upload beliebiger Dateiformate möglich [info]

Beobachtung

Vaultwarden *Send* ist eine Funktion, mit der Benutzer Informationen auf sichere Weise mit anderen teilen können. *Send*-Elemente, entweder Text oder eine Datei, werden auf dem Vaultwarden-Server in verschlüsselter Form gespeichert und können nur mit dem Wissen des Verschlüsselungsschlüssels entschlüsselt werden.

Beim Erstellen eines *Send*-Objekts kann der Benutzer eine beliebige Datei auf den Server hochladen. Bei normaler Nutzung der *Send*-Funktion wird der Inhalt der Datei vom Client verschlüsselt, bevor er auf dem Vaultwarden-Server gespeichert wird. Dabei erlaubt Vaultwarden dem Benutzer, einen beliebigen Dateinamen zu wählen. Folglich könnten Dateien mit potenziell gefährlichen, ausführbaren Endungen wie z.B. `.sh` oder `.exe` hochgeladen werden.

Im folgenden Beispiel wurde eine Shell-Datei (`.sh`) auf den Server hochgeladen.

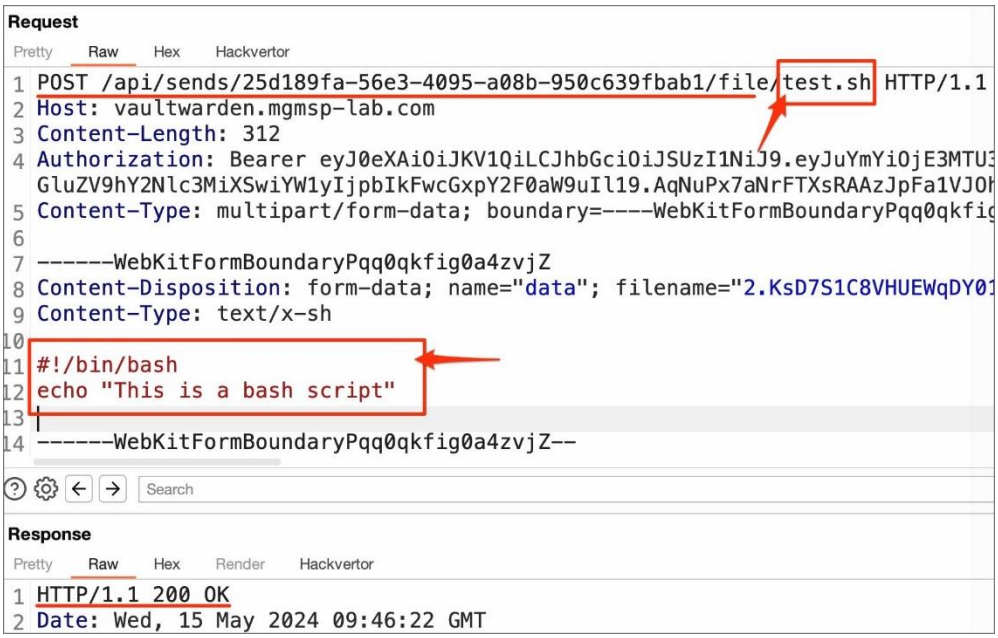


Abbildung 36: Upload von Dateien mit beliebiger Endung.

Der obige Request führt dazu, dass die Testdatei `test.sh` auf den Server hochgeladen wird.

Abbildung 37: Hochgeladene Datei 'test.sh' auf dem Dateisystem des Vaultwarden-Server.

Obwohl das Hochladen einer Datei mit beliebiger Datei-Endung möglich ist, gibt es keine Möglichkeit, die Datei wieder herunterzuladen. Wenn der Dateiname einen Punkt (`.`) enthält, gibt ein Download-Request einen Fehler zurück. Der Grund für dieses Verhalten ist, dass die Download-Methode die Anfrageparameter (den `file_id`-Parameter im untenstehenden Screenshot) in einen Vaultwarden-eigenen Typ `SafeString` umwandelt. Der Konstruktor dieses `SafeString`-Typs löst einen Fehler aus, wenn der Parameter bestimmte Zeichen enthält (nur alphanumerische Zeichen und Bindestriche sind erlaubt).

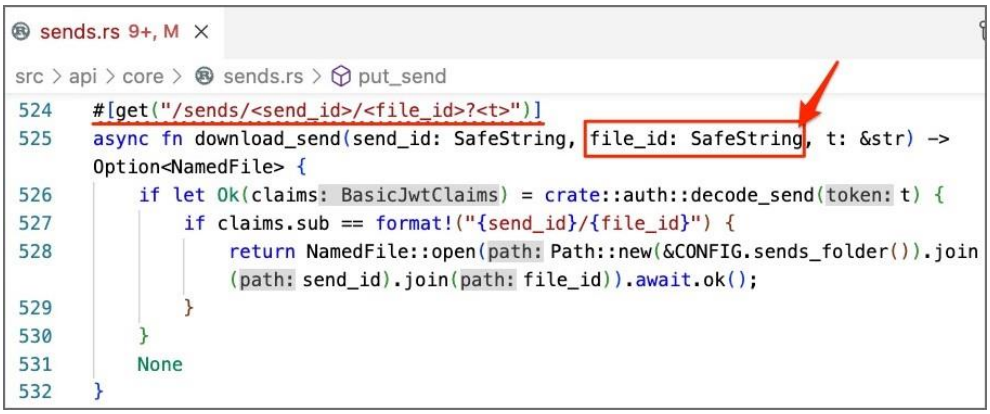


Abbildung 38: Dateien mit Dateiendung können nicht heruntergeladen werden.

Ausnutzbarkeit

Ein Angreifer könnte eine Datei mit beliebigem Dateiformat und Inhalt auf den Vaultwarden-Server hochladen.

Wenn es dem Angreifer gelingt, Dateien auf dem Betriebssystem auszuführen (z.B. durch Ausnutzung einer *Local-File-Inclusion*-Schwachstelle) oder ein autorisierter Backend-Benutzer versehentlich die vom Angreifer hochgeladene Datei ausführt, dann erlaubt diese Schwachstelle die Ausführung beliebigen Codes auf dem Zielsystem. Im Kontext von Vaultwarden ist dieser Angriffsvektor aber eher unwahrscheinlich, da hochgeladenen Dateien nicht als ausführbare markiert sind.

Zudem könnte die hochgeladene Datei zur Umgehung der *Content-Security-Policy* verwendet werden (siehe Finding E3.1). Der Angreifer könnte eine JavaScript-Datei hochladen und die Datei über eine *Cross-Site-Scripting*-Schwachstelle (siehe Finding E3.1) und ein HTML `script`-Tag laden, um die CSP *script-src*-Direktive zu umgehen. Im Kontext von Vaultwarden funktioniert dieser Angriffsvektor aber nicht, da der Download-Endpunkt die Datei ohne Angabe des Inhaltstyps zurückgibt und den *X-Content-Type-Options*-Header auf den Wert `nosniff` setzt. In diesem Zusammenhang verweigern alle modernen Browser die Ausführung von JavaScript aus der im `script`-Tag geladenen Datei.

Anmerkung: Da im Rahmen des Penetrationstests keine direkte Ausnutzbarkeit nachgewiesen werden konnte, dient das Finding rein der Information.

Maßnahme

Der Dateiupload-Endpunkt sollte den Parameter `file_id` am Upload-Endpunkt validieren. Der Parameter sollte in den benutzerdefinierten *SafeString*-Typ umgewandelt werden (ähnlich wie beim Download-Endpunkt), um sicherzustellen, dass der Parameter keine eingeschränkten und potenziell schädlichen Zeichen enthält.

E7 Offenlegung von Informationen

Informationsabfluss bezeichnet die unbeabsichtigte Preisgabe vertraulicher Informationen durch ein System bzw. eine Applikation. Die so offen gelegten Daten können Informationen über die Benutzer, Geschäftsdaten oder technische Informationen beinhalten.

Oft kann das Abfließen technischer Informationen von Angreifern für die Durchführung weiterer Angriffe missbraucht werden. In der ersten Phase eines Angriffs werden in der Regel so viele Informationen wie möglich über das Zielsystem gesammelt. Je umfangreicher und je detailliertere Informationen beschafft werden können, desto präziser können Angriffe durchgeführt werden. Dies erhöht die Erfolgswahrscheinlichkeit eines Angriffs signifikant. Systeme und Anwendungen sollten dementsprechend so konfiguriert werden, dass nur im notwendigen Maß Informationen bereitgestellt werden. Jede überflüssige Information sollte nach Möglichkeit zurückgehalten werden.

E7.1 Offenlegung von Informationen [niedrig]

Beobachtung

Die Antwort des API-Endpunkts `/api/config` gibt die genaue Version des verwendeten Vaultwarden-Servers preis.

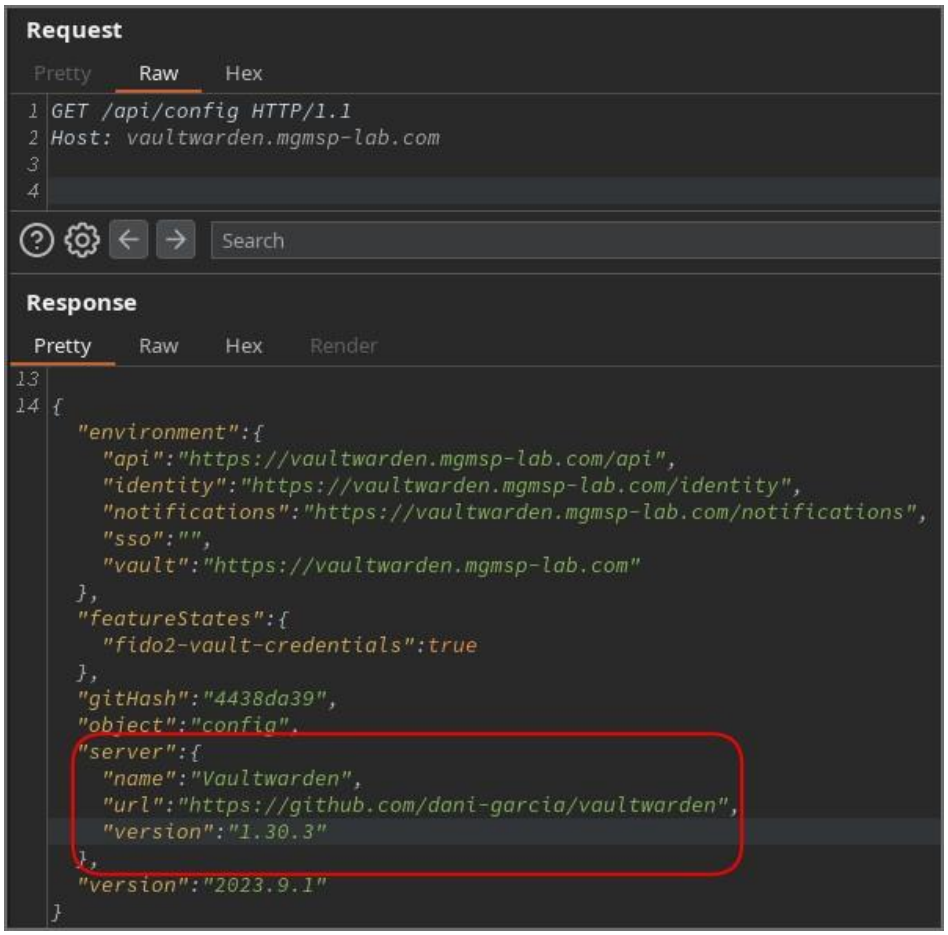


Abbildung 39: Offenlegung der Versionsnummer

Ausnutzbarkeit

Ein Angreifer könnte die gewonnenen, internen Informationen dazu verwenden, um weitere Angriffe auf den Server oder die Anwendung zu planen. Er könnte auch nach bekannten Schwachstellen und Exploits suchen, die unter Umständen öffentlich verfügbar sind.

Maßnahme

In der Serverantwort sollten keine internen Informationen offengelegt werden. Insbesondere sollten keine zusätzlichen sensiblen Informationen wie Versionsstrings oder die verwendete Software übermittelt werden.

E8 Session-Management

Session-Management beschreibt allgemein die technischen und logischen Voraussetzungen, die notwendig sind, um Sitzungen, Aktionen und Transaktionen in einer Umgebung mit zustandslosen Verbindungen zu betreiben.

E8.1 Unzureichende Cookie-Konfiguration im Admin-Dashboard [niedrig]

Beobachtung

Vaultwarden wird mit einem Admin-Dashboard ausgeliefert. Diese Funktionalität kann wie in der folgend verlinkten Anleitung aktiviert werden.

https://github.com/dani-garcia/vaultwarden/wiki/Enabling-admin-page

Nach der Aktivierung ist das Vaultwarden Admin Dashboard unter dem Endpunkt `/admin` verfügbar. Im Gegensatz zur eigentlichen Anwendung, bei der sich angemeldete Benutzer mit einem im `Authorization`-Header übertragenen JWT authentifizieren, verwendet das Admin-Dashboard *Cookies* als Transportmedium für das JWT.

Wie im folgenden Screenshot zu sehen ist, wird das Cookie ohne das `Secure`-Flag gesetzt.

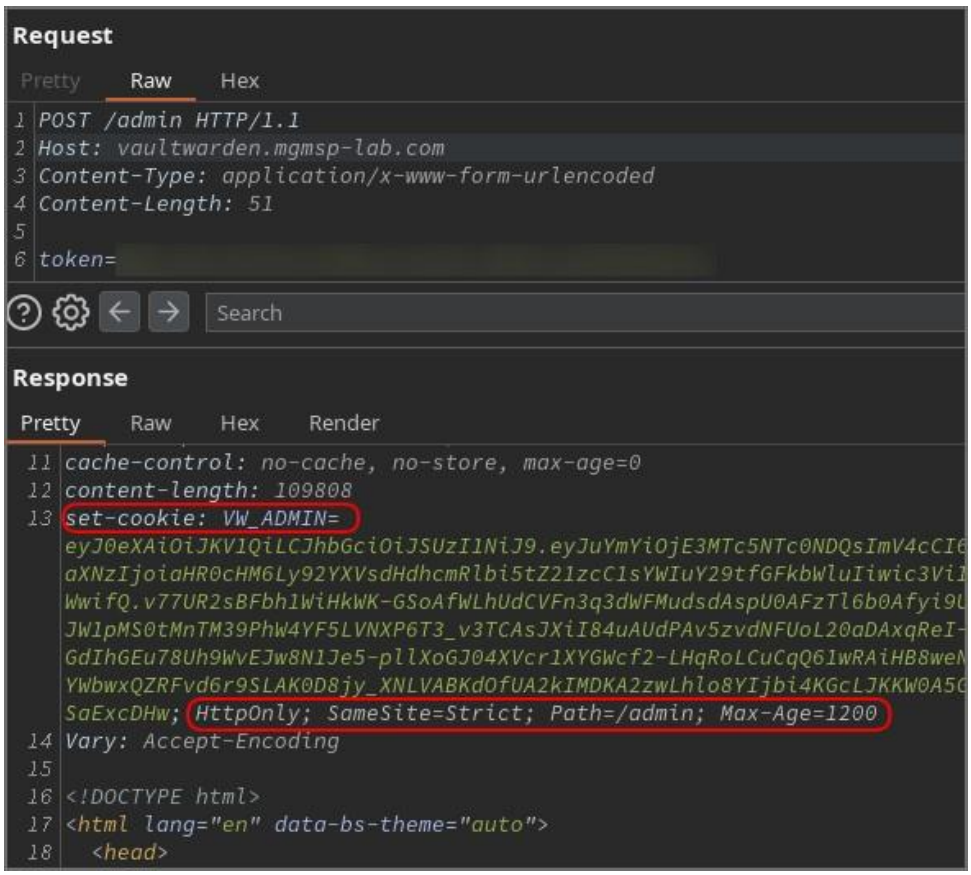


Abbildung 40: VW_ADMIN-Session-Cookie ist ohne das Secure-Flag gesetzt

Es sollte beachtet werden, dass der Vaultwarden-Server selbst nicht weiß, ob er über HTTP oder HTTPS angesprochen wurde, um das `Secure`-Cookie-Attribut

entsprechend zu aktivieren. Typischerweise lauscht die Anwendung in einem gewöhnlichen produktiven Einsatz immer auf einem Klartext-HTTP-Port. HTTPS könnte durch einen Reverse-Proxy, der vor dem Vaultwarden-Server steht, hinzugefügt werden. In einem solchen Kontext verlässt sich der Vaultwarden-Server auf den Reverse-Proxy, um das Verbindungsprotokoll zu bestimmen. Die Beispiele für die Webserverkonfiguration auf Vaultwarden Proxy Beispiele enthalten bereits den Mechanismus zur Übergabe des Verbindungsprotokolls an den Vaultwarden Server (unter Verwendung des benutzerdefinierten Headers `X-Forwarded-Proto`).

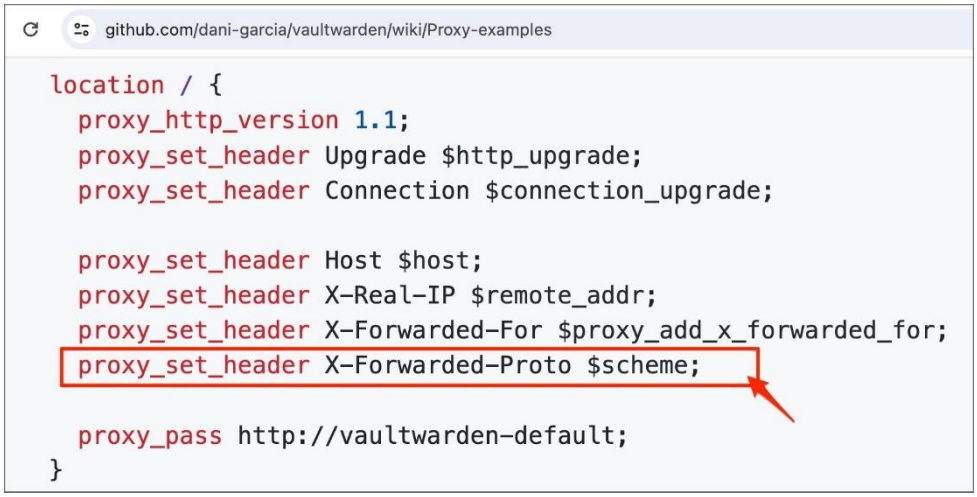


Abbildung 41: Beispiel nginx-Konfiguration

Ausnutzbarkeit

Eine unsichere Cookie-Konfiguration begünstigt die Kompromittierung der Sitzung eines Benutzers.

Das Admin-Cookie `VW_ADMIN` wird ohne das Attribut `Secure` übertragen. Somit wird eine unverschlüsselte Rückübertragung der Cookies an den Server zugelassen. Durch die Verwendung des Attributs `Secure` wird der Browser des Clients angewiesen, die Cookies ausschließlich über verschlüsselte Verbindungen an den Server zu übermitteln. Dadurch wird das Auslesen der Cookies im Rahmen von Man-in-the-Middle-Angriffen erheblich erschwert.

Maßnahme

Um sicherzustellen, dass das Secure-Flag für den Session-Cookie immer aktiviert ist, wenn sich der Benutzer über HTTPS verbindet, sollte der Vaultwarden-Server den `X-Forwarded-Proto`-Header in der Anfrage überprüfen. Falls dieser Header auf `https` gesetzt ist, sollte das Secure-Flag hinzugefügt werden.

E9 Datenvalidierung

Die meisten Schwachstellen in Webanwendungen haben eine ungenügende Datenvalidierung (oft als Input-Datenvalidierung bezeichnet) als Ursache.

Unter Datenvalidierung sind alle Daten und zugehörigen Prüfungen zu verstehen, die die Daten bearbeiten, welche von außen in ein System kommen. Dazu gehören nicht nur die unmittelbaren Benutzereingaben, sondern auch die Daten, die von Drittsystemen (z. B. aus Datenbanken) kommen. Weiter muss jedes System sicherstellen, dass die Daten, die an Drittsysteme zur Verarbeitung weitergegeben werden, keinen schadhaften Code enthalten.

Korrekterweise muss daher zwischen Input-Validierung- und Output-Enkodierung unterschieden werden. Bei den folgenden einzelnen Schwachstellen und Bedrohungen wird ggf. auf diesen Unterschied aufmerksam gemacht.

E9.1 Log-Injektion [niedrig]

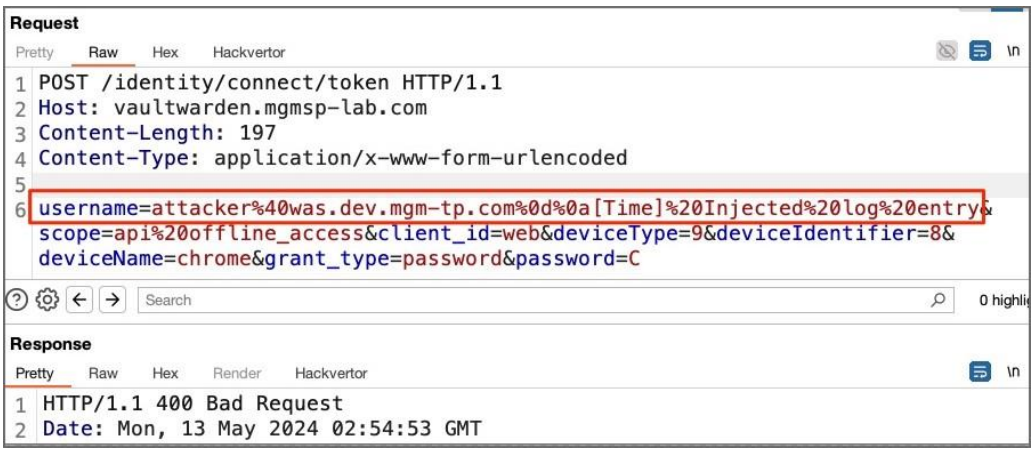
Beobachtung

Vaultwarden protokolliert standardmäßig bestimmte vom Benutzer ausgelöste Ereignisse in einer auf dem Server liegenden Log-Datei. Bei Verwendung des offiziellen Docker-Image wird die entsprechende Log-Datei unter `/data/vaultwarden.log` abgelegt und über ein Docker-Volume in den Host-Rechner eingebunden. Weitere Informationen zur Vaultwarden-Protokollierung können unter folgendem Link eingesehen werden:

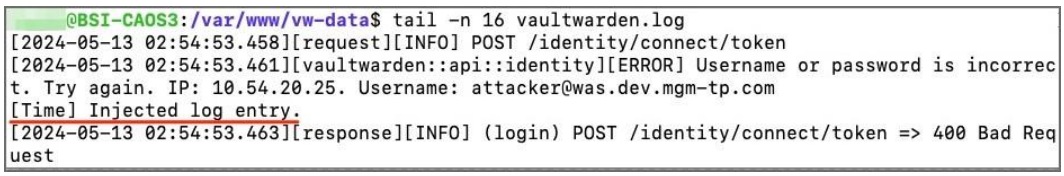
https://github.com/dani-garcia/vaultwarden/wiki/Logging

Zeilenumbrüche werden bei der Protokollierung von nutzerkontrolliertem Input nicht korrekt enkodiert. Ein Zeilenumbruch (`%0d%0a`) im Nutzerinput führt damit in der Log-Datei zu einer neuen Zeile. Dies ermöglicht es einem Angreifer, Log-Einträge zu fälschen oder bösartige Inhalte in die Datei einzuschleusen.

Im folgenden Beispiel wurde eine ungültige Login-Anfrage an den Vaultwarden-Server gesendet. Diese Anfrage enthält einen Zeilenumbruch im Feld `username`.



In der Log-Datei wird der obige Request wie folgt protokolliert. Die Zeile nach dem protokollierten Nutzernamen erweckt durch den Zeilenumbruch den Eindruck ein weiterer Eintrag in der Log-Datei zu sein. Der gefälschte Log-Eintrag wäre nicht von einem echten zu unterscheiden.



Ausnutzbarkeit

Diese Schwachstelle ermöglicht es einem Angreifer, beliebigen Inhalt in die Log-Datei des Vaultwarden-Servers einzufügen. Dies könnte dazu genutzt werden, um zum Beispiel zu versuchen die Spuren eines Angriffs zu verwischen oder sogar den Angriff einer anderen Partei zuzuordnen.

Maßnahme

Sonderzeichen, die in benutzerkontrollierten Eingaben enthalten sind und in Log-Dateien mit aufgenommen werden, sollten vor der Speicherung korrekt kodiert werden. Im Falle von Vaultwarden werden Log-Einträge durch ein Zeilenumbruchzeichen getrennt. Daher sollten vor allem *Newline*-Zeichen in protokollierten Eingaben kodiert werden. Zum Beispiel mit URL-Kodierung zu `%0d%0a`.

F. Anhänge

Folgende Anhänge wurden als Teil dieses Berichtes mitgeliefert:

- Ordner: burp
    - Arbeitsprotokolle Burp-Analysen
- vaultwarden-server-audited.xlsx, vaultwarden-browser-extension-audited.xlsx
    - Bewertet Ergebnisse aller verwendeter SAST-/SECRETS-Scanner in einem Excel-Dokument (s. Abschnitt D1)
- vaultwarden-server-excluded.xlsx, vaultwarden-browser-extension-excluded.xlsx
    - Excel-Datei mit Auflistung aller von der Bewertung ausgenommener SAST-/SECRETS-Findings (s. Abschnitt D1)
- vaultwarden-server-sca.xlsx
    - Deduplizierte Ergebnisse aller verwendeter SCA-Scanner in einem Excel-Dokument (s. Abschnitt D2)
- vaultwarden-server-sca-excluded.xlsx
    - Als Duplikat ausgeschlossene Findings aller SCA-Scanner in einem Excel-Dokument (s. Abschnitt D2)
- Ordner: raw
    - Unbewertete Rohdaten aller verwendeten Scanner (s. Abschnitt D)

G. Referenzen

/1/ Input- und Output-Datenvalidierung in Webanwendungen
https://cwe.mitre.org/data/definitions/79.html

/2/ Übersicht zu Datenvalidierung
https://www.owasp.org/index.php/Data_Validation

/3/ Cheat Sheet zu Datenvalidierung
https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet

/4/ Übersicht zur XSS-Prävention
https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

/5/ Unvalidierte Redirects und Forwards
https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

/6/ Übersicht zu SQL-Injection
https://www.owasp.org/index.php/SQL_Injection

/7/ Übersicht zu Blind-SQL-Injection
https://www.owasp.org/index.php/Blind_SQL_Injection

/8/ SQL-Injection-Prävention
https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

/9/ Best Practice Maßnahmen zur Auditierung und Härtung von Datenbanken
https://benchmarks.cisecurity.org/downloads/browse/index.cfm?category=benchmarks.servers.database

/10/ System-Kommandos aufrufen
https://cwe.mitre.org/data/definitions/78.html

/11/ Path-Traversal
https://www.owasp.org/index.php/Path_Traversal

/12/ Buffer-Overflow-Angriff
https://www.owasp.org/index.php/Buffer_overflow_attack

/13/ Format-String-Angriff
https://www.owasp.org/index.php/Format_string_attack

/14/ Benutzer-Enumeration
https://www.owasp.org/index.php/Testing_for_user_enumeration_(OWASP-AT-002)

/15/ OWASP - Blocking Brute Force Attacks
https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

/16/ Länge und Komplexität von Passwörtern
https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity

/17/ Übersicht zu Authentication
https://www.owasp.org/index.php/Authentication_Cheat_Sheet

/18/ Übersicht zu Session-Management
https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

/19/ Session-Fixation-Schwachstelle
http://www.acros.si/papers/session_fixation.pdf

/20/ Einführung in CSRF bzw. Session Riding
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

/21/ CSRF-Prävention
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet

/22/ Privilegienerweiterung
https://cwe.mitre.org/data/definitions/269.html

/23/ Zugriffskontrolle
https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

/24/ Sichere Passwort-Verwaltung
https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

/25/ Sichere Datenspeicherung
https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

/26/ Best Practices for a Secure “Forgot Password” Feature
https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet

/27/ Fehlerhafte Benutzer-Authentisierung und Session-Verwaltung
https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

/28/ Denial of Service
https://www.owasp.org/index.php/Denial_of_Service

/29/ CAPTCHA: Telling Humans and Computers Apart Automatically
http://www.captcha.net/

/30/ SSL Best Practices
https://www.ssllabs.com/projects/best-practices/index.html

/31/ Übersicht von öffentlich bekannten Schwachstellen
https://nvd.nist.gov/

/32/ Secure Coding of CORS
http://www.andlabs.org/html5/rejectCOR.php

/33/ Informationen über HSTS
http://blog.securenet.de/2012/11/02/ssl-stripping-die-ignorierte-gefahr/

/34/ CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')
https://cwe.mitre.org/data/definitions/77.html

/35/ OWASP Content Security Policy
https://www.owasp.org/index.php/Content_Security_Policy

/36/ OWASP Clickjacking
https://www.owasp.org/index.php/Clickjacking

/37/ OWASP HTML5 Security Cheat Sheet
https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet

/38/ OWASP HTTP Headers
https://www.owasp.org/index.php/List_of_useful_HTTP_headers

/39/ Unrestricted Upload of File with Dangerous Type
https://cwe.mitre.org/data/definitions/434.html

/40/ Unrestricted File Upload
https://www.owasp.org/index.php/Unrestricted_File_Upload

/41/ OWASP TLS Cheat Sheet
https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

/42/ Sichere Passwortablage
https://paragonie.com/blog/2016/02/how-safely-store-password-in-2016