

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Информационная безопасность

Работа 2

Анализ и устранение уязвимости на примере реального CVE с
использованием Vulnhub

Группа: Р3416

Выполнил:

Сиразетдинов Азат Ниязович

г. Санкт-Петербург

2025

Выполнение

Выбранная уязвимость

В рамках выполнения лабораторной работы была выбрана уязвимость CVE-2024-43441 графовой базы данных hugegraph. Суть уязвимости заключается в том, что если в базе данных включен режим аутентификации JWT но не сконфигурирован параметр auth.token_secret, то будет использоваться захардкоженный секрет «FXQXbJtbCLxODc6tGci732pkH1cyf8Qg». Злоумышленник может использовать его чтобы сгенерировать JWT ключ и получить доступ к базе данных

Последовательность действий для воспроизведения уязвимости

- Клонируем репозиторий vulhub и запускаем контейнеры

```
git clone https://github.com/vulhub/vulhub.git
cd vulhub/hugegraph/CVE-2024-43441
podman compose up
```

```
/> cu
>>> cd vulhub/hugegraph/CVE-2024-43441
>>> podman compose up
>>> Executing external compose provider "/opt/homebrew/bin/docker-compose". Please see podman-compose(1) for how to disable this message. <<<
Attaching to web-1
web-1 | Hugegraph Initialization already done. Skipping re-init...
web-1 | Starting HugeGraphServer in daemon mode...
```

- Воспроизводим уязвимость

The screenshot shows the Postman interface with a GET request to `localhost:8080/graphs`. The Authorization tab is active, displaying the configuration for generating a JWT token:

- Type: JWT Bearer
- Algorithm: HS256
- Secret: FXQXbJtbCLxODc6tGci732pkH1cyf8Qg
- Secret Base64 encoded

The payload field contains the following JSON object:

```
{"user_name": "admin", "user_id": "-30:admin", "exp": 9739523483}
```

The response pane shows the following JSON output:

```
1 [ { "graphs": [ 2 | "hugegraph" 3 ] 4 ] 5 }
```

Мы можем самостоятельно создать JWT токен используя захардкоженный стандартный секрет и payload с юзером admin. Запрос возвращает граф в базе данных

Злоумышленник может получить несанкционированный доступ к базе данных зная лишь юзернейм пользователя

Анализ корневой причины

Корневая причина - CWE-302 (Authentication Bypass by Assumed-Immutable Data). Если администратор базы данных сконфигурировал JWT авторизацию но забыл создать секретный ключ, то используется стандартный.

По-хорошему база данных должна требовать ручного указания секретного ключа, либо генерировать секретный ключ случайно при первичной конфигурации сервера. Использование стандартных значений в таких вещах недопустимо (как минимум нужно писать warning в консоль при запуске).

```
web-1 | 2025-11-04 23:00:34 [main] [INFO] o.a.h.u.ConfigUtil - Scanning option 'graphs' directory './conf/graphs'
web-1 | 2025-11-04 23:00:34 [main] [INFO] o.a.h.c.InitStore - Init graph with config file: ./conf/graphs/hugegraph.properties
web-1 | 2025-11-04 23:00:34 [db-open-1] [INFO] o.a.h.b.s.r.RocksDBStore - Opening RocksDB with data path: rocksdb-data/data/m
web-1 | 2025-11-04 23:00:34 [db-open-1] [INFO] o.a.h.b.s.r.RocksDBStore - Failed to open RocksDB 'rocksdb-data/data/m' with database 'hugegraph', try to init CF later
web-1 | 2025-11-04 23:00:34 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'schema-id-hugegraph' with capacity 10000
web-1 | 2025-11-04 23:00:34 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'schema-name-hugegraph' with capacity 10000
web-1 | 2025-11-04 23:00:34 [db-open-1] [INFO] o.a.h.b.s.r.RocksDBStore - Opening RocksDB with data path: rocksdb-data/data/s
web-1 | main dict load finished, time elapsed 585 ms
web-1 | model dict load finished, time elapsed 34 ms.
web-1 | 2025-11-04 23:00:35 [db-open-1] [INFO] o.a.h.b.s.r.RocksDBStore - Opening RocksDB with data path: rocksdb-data/data/g
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.c.o.l.Uns - OHC using JNA OS native malloc/free
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init LevelCache for 'vertex-hugegraph' with capacity 10000:1000000
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init LevelCache for 'edge-hugegraph' with capacity 1000:1000000
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'users-hugegraph' with capacity 10240
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'users_pwd-hugegraph' with capacity 10240
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'token-hugegraph' with capacity 10240
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'audit-log-limiter-hugegraph' with capacity 10240
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.c.CacheManager - Init RamCache for 'users-role-hugegraph' with capacity 10240
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.b.s.r.RocksDBStore - Write down the backend version: 1.11
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.StandardHugeGraph - Graph 'hugegraph' has been initialized
web-1 | 2025-11-04 23:00:35 [main] [WARN] o.a.h.c.HugeConfig - The config option 'initing_store' is redundant, please ensure it has been registered
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.u.ConfigUtil - Scanning option 'graphs' directory './conf/graphs'
web-1 | Please input the admin password:
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.StandardHugeGraph - Close graph standardhugegraph[hugegraph]
web-1 | 2025-11-04 23:00:35 [main] [INFO] o.a.h.HugeFactory - HugeFactory shutdown
web-1 | 2025-11-04 23:00:35 [hugegraph-shutdown] [INFO] o.a.h.HugeFactory - HugeGraph is shutting down
web-1 | Initialization finished.
web-1 | Starting HugeGraphServer in daemon mode...
web-1 | Connecting to HugeGraphServer (http://0.0.0.0:8080/graphs)....OK
web-1 | Started [pid 282]
```

Логи пустые и не содержат ни единого предупреждения о небезопасной конфигурации

Разработка мер защиты

Для разработки мер защиты нам для начала требуется самостоятельно воспользоваться базой данных hugegraph

Сделаем новый файл docker-compose.yml:

```
services:
  hugegraph:
    image: hugegraph/hugegraph:1.3.0
    environment:
      - PASSWORD=123456
    ports:
      - "8080:8080"
~
```

Сначала проверим что на версии 1.3 все повторяется

Переменная среды PASSWORD требуется для включения режима аутентификации

Запустим контейнер командой `podman compose up`

The screenshot shows the Postman interface with a successful API call to `localhost:8080/graphs`. The response body is a JSON object:

```

1
2   "graphs": [
3     "hugegraph"
4   ]
5

```

Как и ожидалось, уязвимость повторяется

Обновление HugeGraph до версии 1.5

В описании уязвимости это указано как надлежащее решение. Проверим:

Обновим docker-compose файл и воспользуемся версией 1.5

```

services:
  hugegraph:
    image: hugegraph/hugegraph:1.5.0
    ports:
      - "8080:8080"
~
```

Запустим контейнер командой `podman compose up`

The screenshot shows the Postman interface with an unauthorized response to `localhost:8080/graphs`. The response body is a JSON object:

```

1
2   "exception": "class jakarta.ws.rs.NotAuthorizedException",
3   "message": "Authentication failed",
4   "cause": ""
5

```

Теперь запрос падает с ошибкой Unauthorized

Для получения JWT токена выполним запрос

```
curl --location 'localhost:8080/graphs/hugegraph/auth/login' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic YWRtaW46MTIzNDU2' \
--data '{
  "user_name": "admin",
  "user_password": "123456"
}'
```

В ответ получаем

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyX25hbWUiOiJhZG1pbmIsInVzZXJfaWQiOiItMzA6YWRtaW4iLCJleHAi
  OjE3NjIxODU5ODR9.wHrMsan129unXzjgz2I0_Tw6z89gdYCkT6Xn5n5RIp0"
```

Воспользуемся токеном для получения информации о графах

The screenshot shows the Postman interface with the following details:

- HTTP:** localhost:8080/graphs
- Method:** GET
- URL:** http://localhost:8080/graphs
- Authorization:** Bearer Token (selected)
- Token:** eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyX25hbWUiOiJhZG1pbmIsInVzZXJfaWQiOiItMzA6YWRtaW4iLCJleHAiOjE3NjIxODU5ODR9.wHrMsan129unXzjgz2I0_Tw6z89gdYCkT6Xn5n5RIp0
- Body:** Pretty (selected), Raw, Preview, Visualize, JSON
- Headers:** (2)
- Test Results:** Status: 200 OK, Time: 11 ms, Size: 109 B, Save Response

The response body is displayed as:

```
1  [
2   "graphs": [
3     "hugegraph"
4   ]
5 ]
```

Аутентификация пройдена успешно

Заодно проверим в JWT debugger что ключ использует другой секрет:

JWT Decoder [JWT Encoder](#)

Paste a JWT below that you'd like to decode, validate, and verify.

Enable auto-focus [Generate example](#)

ENCODED VALUE

JSON WEB TOKEN (JWT)

```
eyJhbGciOiJIUzI1NiJ9eyJc2VyX25hbWUiOiJhZG1pbisInVzZXJfaWQiOitMzA6YW
RtaW4iLCJleHAiOiJE3NjIzODYxMDF9.NMYDjogWe-
SAQrRVb8WhQxSuycNrysKqpRXrJ0xK4HY
```

[COPY](#) [CLEAR](#)

DECODED HEADER

JSON CLAIMS TABLE

```
{  
  "alg": "HS256"  
}
```

[COPY](#) [CLEAR](#)

DECODED PAYLOAD

JSON CLAIMS TABLE

```
{  
  "user_name": "admin",  
  "user_id": "-30:admin",  
  "exp": 1762386101  
}
```

[COPY](#) [CLEAR](#)

JWT SIGNATURE VERIFICATION (OPTIONAL)

Enter the secret used to sign the JWT below:

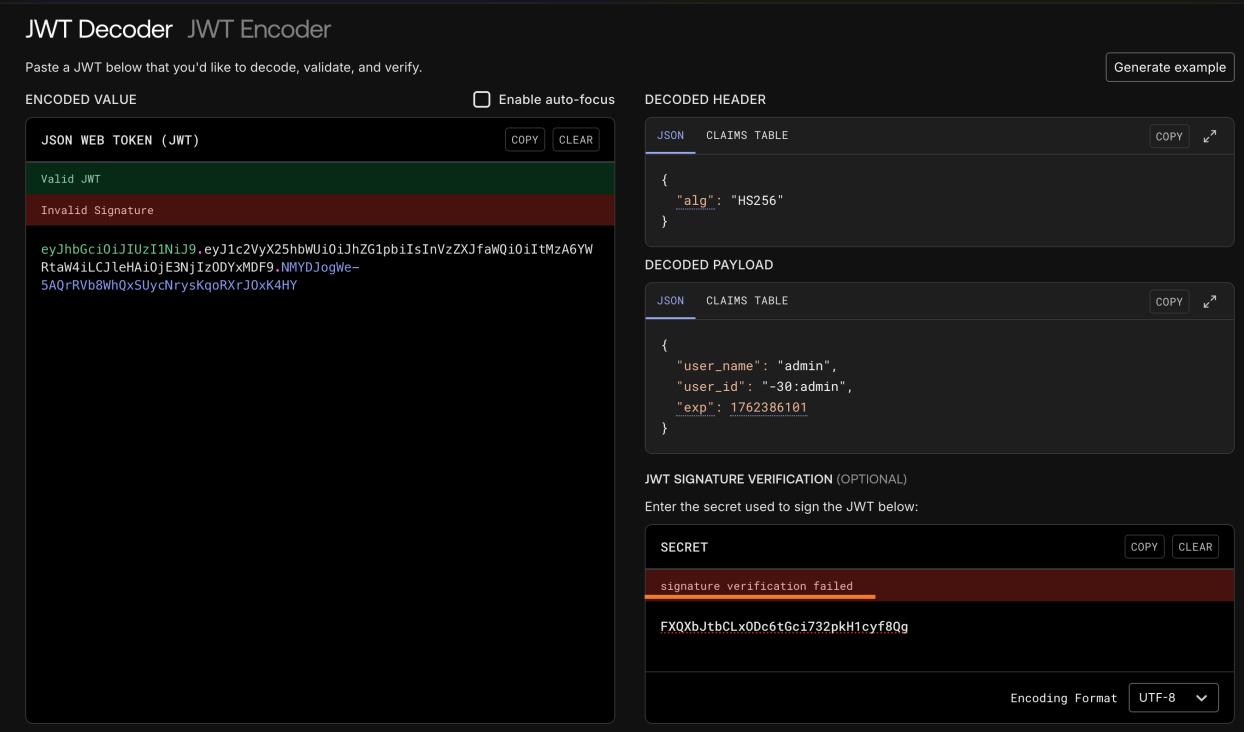
SECRET

```
signature verification failed
```

FXQXbJtbCLx0Dc6tGci732pkH1cyf8Qg

[COPY](#) [CLEAR](#)

Encoding Format [UTF-8](#)



Что и требовалось доказать, в версии 1.5 используется генерация секрета для JWT токена