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 Vulhub

Группа: P3416

Выполнил:

Сиразетдинов Азат Ниязович

г. Санкт-Петербург

2025

Выполнение

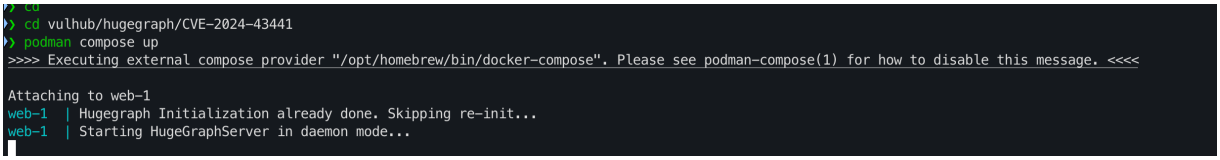
Выбранная уязвимость

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

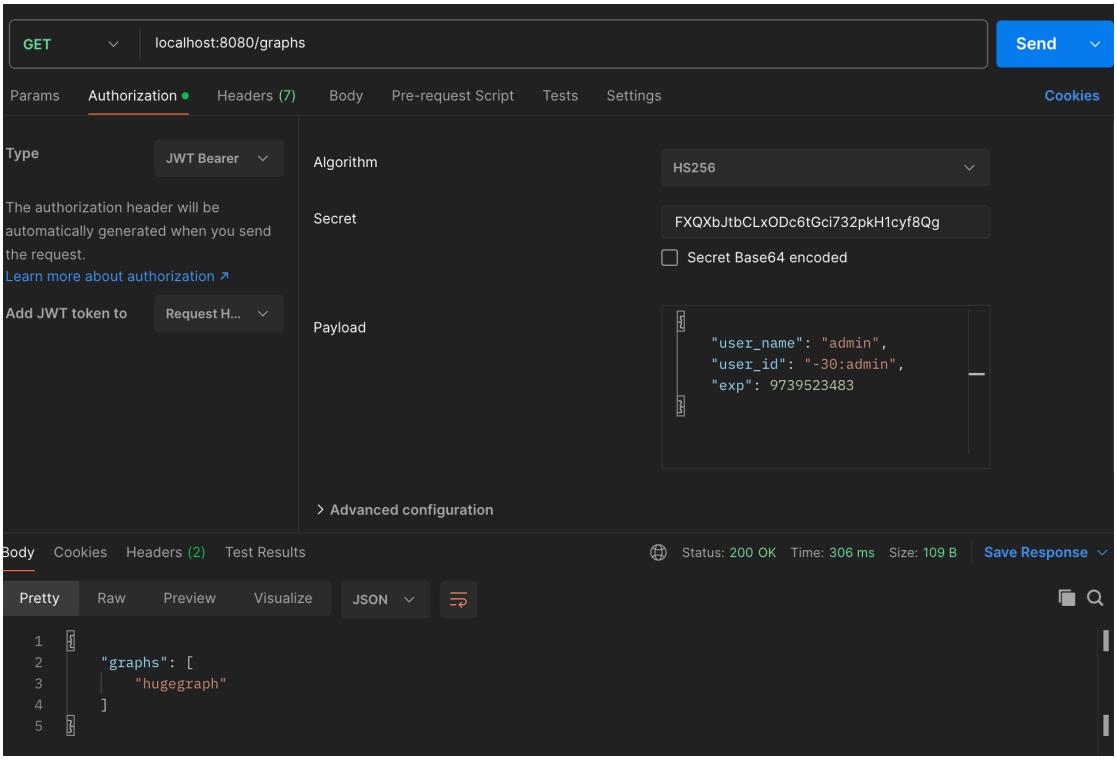
Последовательность действий для воспроизведения уязвимости

1) Клонируем репозиторий vulhub и запускаем контейнеры

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



2) Воспроизводим уязвимость



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 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 10000:10000000
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

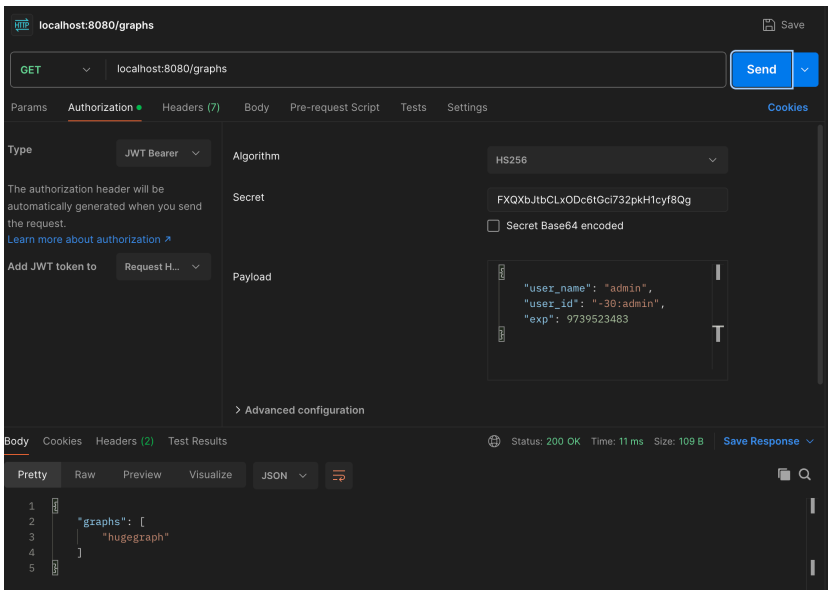
Сделаем новый файл docker-compose.yml:

```
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`



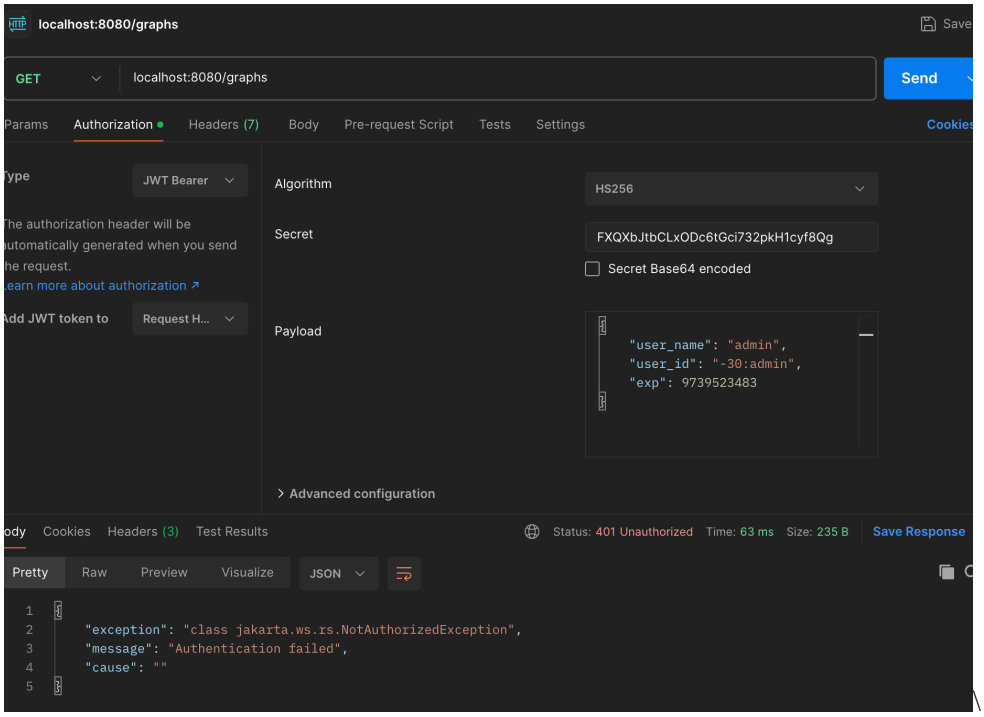
Как и ожидалось, уязвимость повторяется

Обновление HugeGraph до версии 1.5

В описании уязвимости это указано как надлежащее решение. Проверим:

Обновим docker-compose файл и воспользуемся версией 1.5

Запустим контейнер командой `podman compose up`



Теперь запрос падает с ошибкой Unauthorized

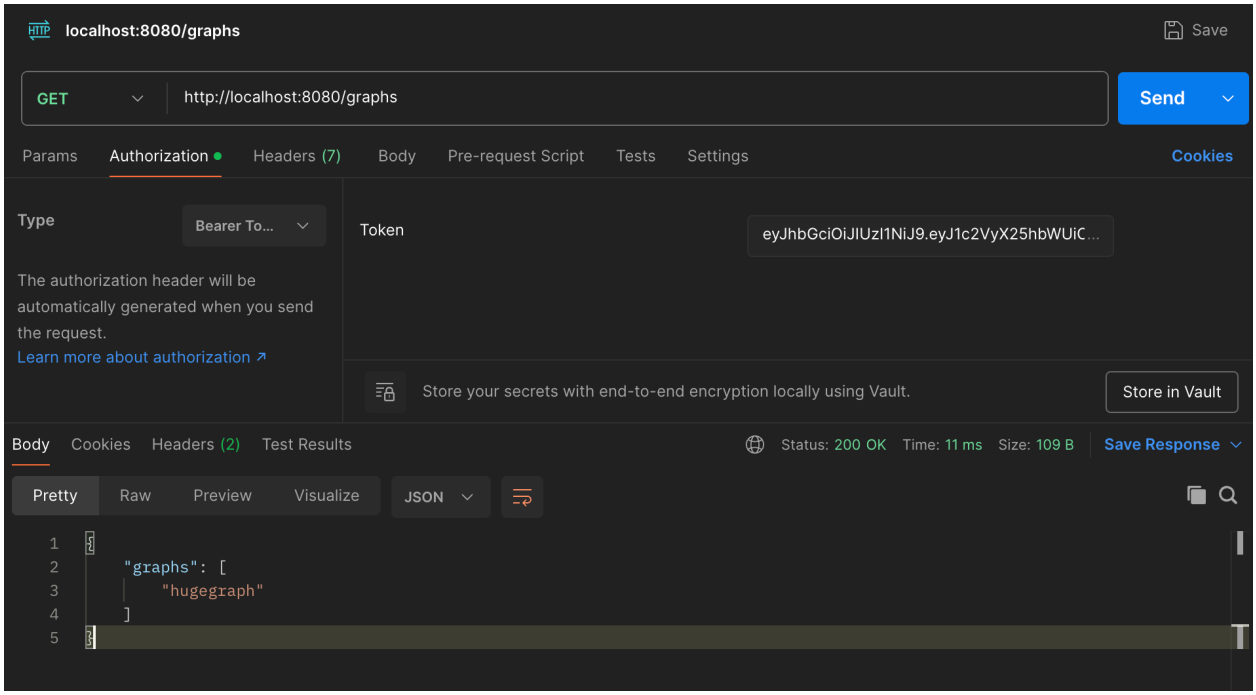
Для получения JWT токена выполним запрос

```
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

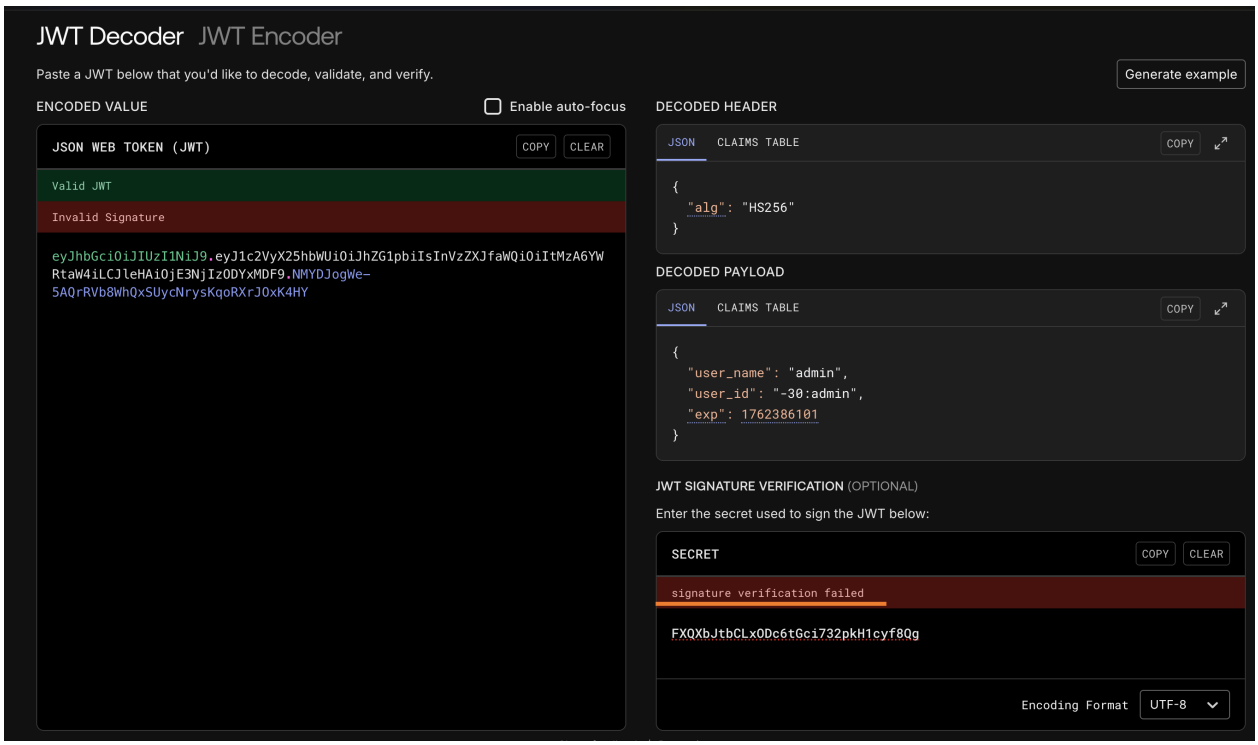
```
{
    "token":
"eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyX25hbWUiOiJhZG1pbiIsInVzZXJfaWQiOiItMzA6YWRtaW4iLCJleHAi
OjE3NjIzODU5ODR9.wHrMsan129unXzjgz2I0_Tw6z89gdYCkT6Xn5n5RIp0"
}
```

Воспользуемся токеном для получения информации о графах



Аутентификация пройдена успешно

Заодно проверим в JWT debugger что ключ использует другой секрет:



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