ZEDYL

Приватный мессенджер RuCQ

HackAtom 2024

Как поднять RuCQ в Docker

Для запуска RuCQ на Debian необходимы программные средства: система контроля версий Git и система контейнеризации Docker с плагином Compose.

Установка Git: https://git-scm.com/downloads/linux

Установка Docker с плагином Compose: https://docs.docker.com/engine/install/debian/

После установки Git и Docker, введите следующие команды для запуска мессенджера:

```
git clone https://github.com/Szent7/rucq
cd rucq
docker compose build
docker compose up -d
```

Контейнеры используют порты 5432 (стандартный порт Postgres) и 10015(основной порт), 10017 (для API запросов). Переназначать порты в docker-compose.yml настоятельно не рекомендуется.

В конечном итоге будет два контейнера: база данных и сам приватный чат. Автоматически будут созданы пользователи и чаты. В чатах будет отображаться история чата (легенда) и необходимый ФЛАГ для выполнения задачи.

Доступ к веб-интерфейсу осуществляется по пути http://{ip-адрес сервера с docker}:10015/.

Например, если адрес сервера 192.168.1.22, то после запуска контейнеров, доступ к веб-интерфейсу будет осуществляться по адресу http://192.168.1.22:10015/.

Эксплуатация уязвимостей

Для эксплуатации уязвимости с подменой Cookies необходимо получить Куки авторизированного пользователя (Логин: jr.Lieutenant; Пароль: Z7Pqh8v7). Авторизацию должен проводить не пентестер (так не интересно).

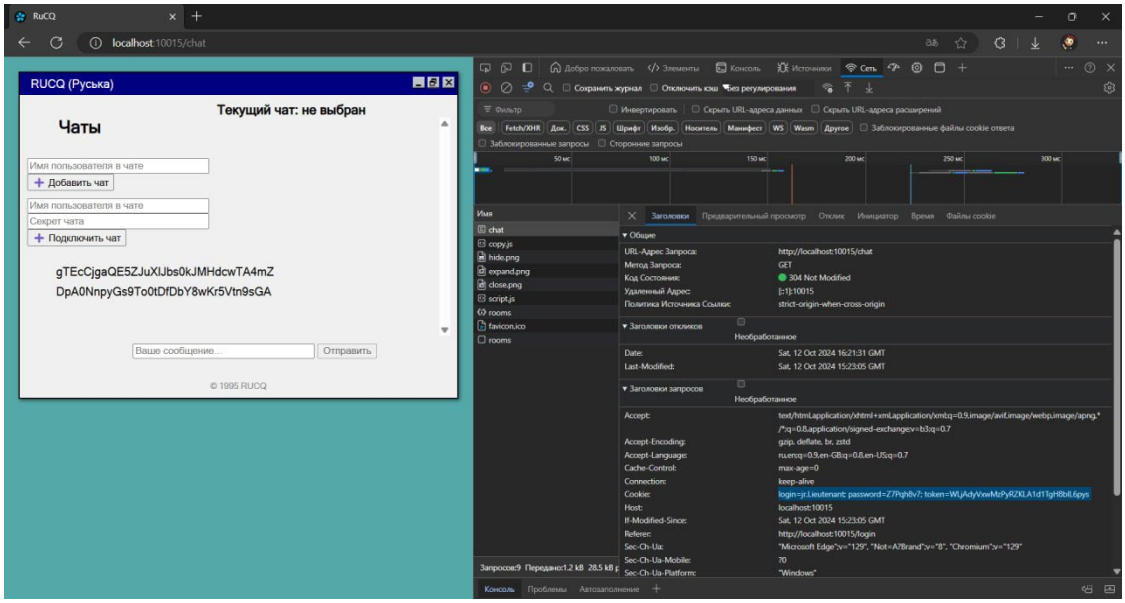

Рисунок 1. Получить куки любым возможным способом.

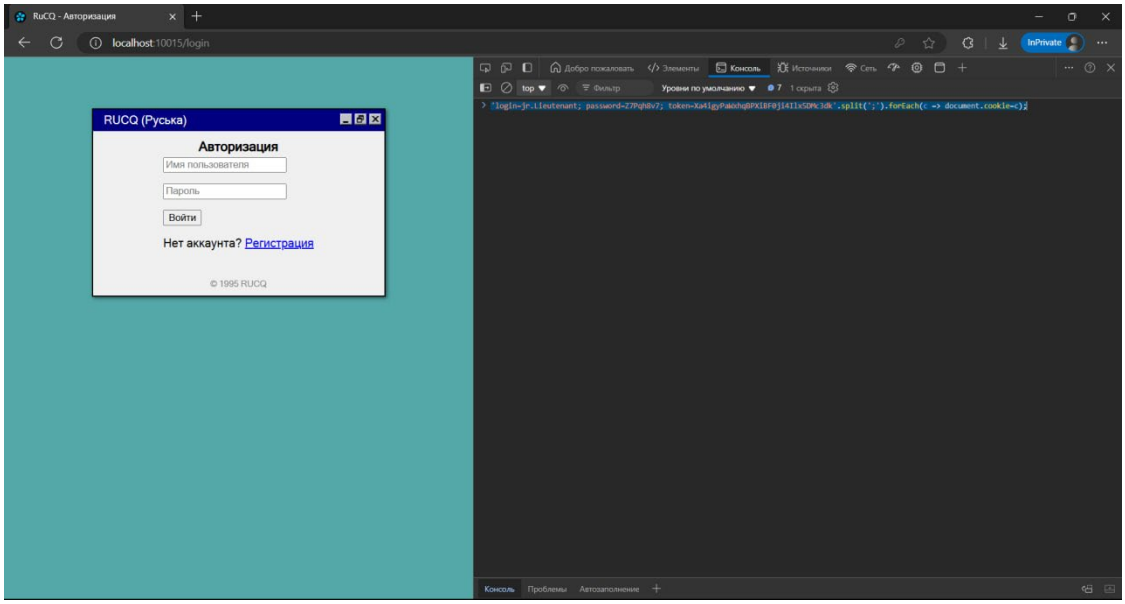

Рисунок 2. Вставить куки любым удобным способом.

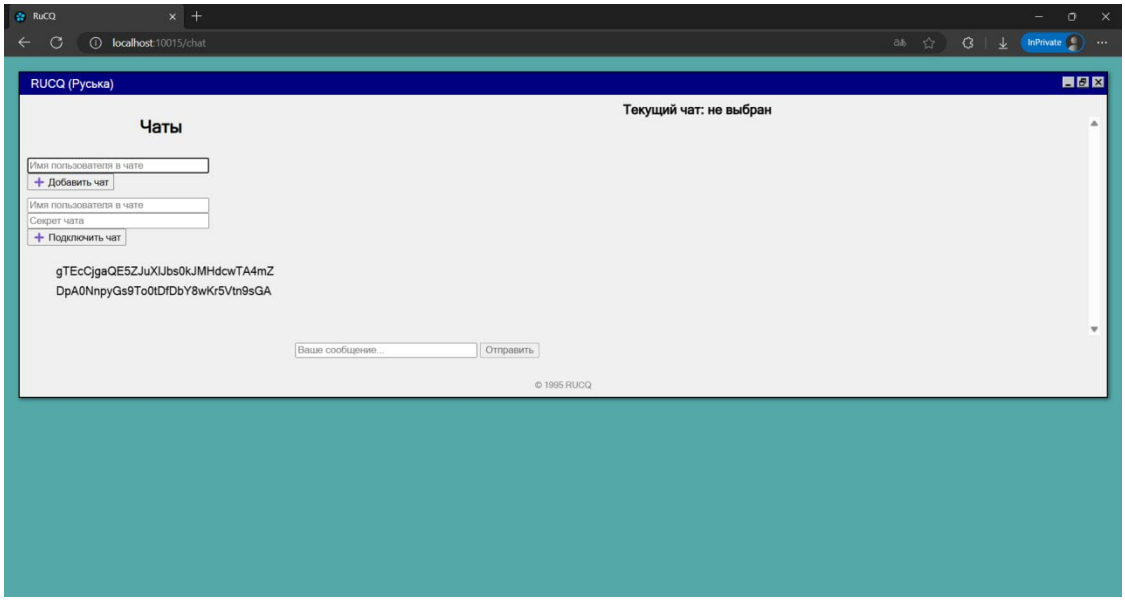

Рисунок 3. Доступ получен.

Для эксплуатации уязвимости с анализом трафика необходимо отследить трафик в момент авторизации пользователя, авторизацию должен проводить кто-то из жюри.

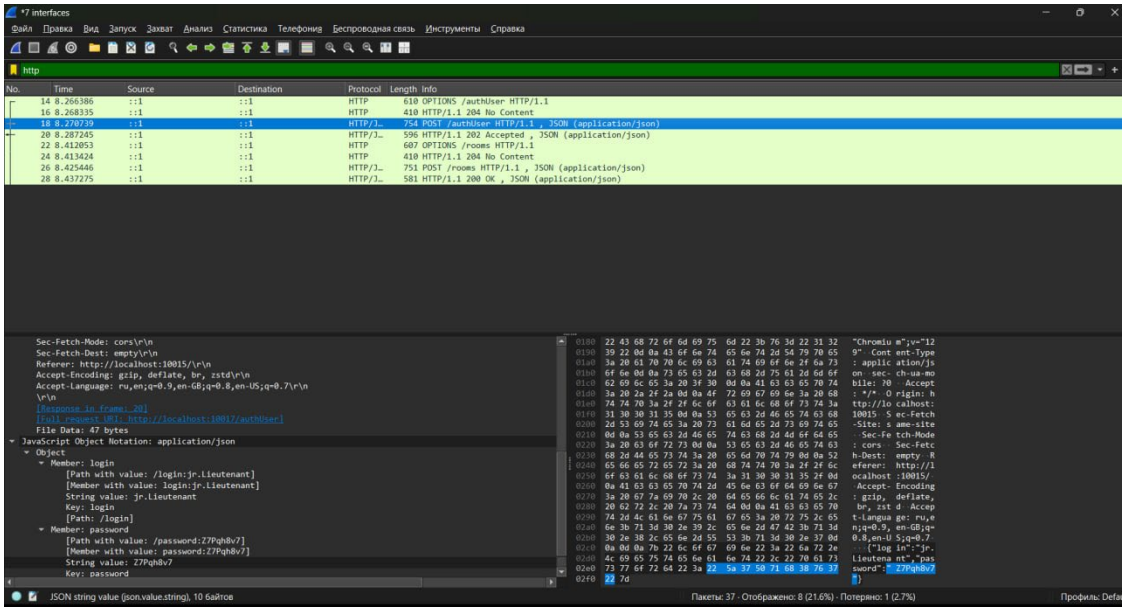

Рисунок 4. Отследить сетевые пакеты в любом анализаторе трафика.

Формат флага, который предстоит найти выглядит так: FLAG{Some_text}

Подробное описание уязвимостей можно прочитать в главе «Обнаруженные уязвимости».

Легенда

Добро пожаловать в историю RUCQ (Руська), приватного мессенджера, разработанного в 1995 году для секретных сотрудников спецслужб Российской Федерации.

RUCQ использует уникальную систему шифрования для каждого чат-канала, что позволяет гарантировать безопасность и конфиденциальность переписки.

Со временем, этот мессенджер перекочевал в знакомый всем ICQ (Аська), став доступным для обычных граждан, и с тех пор стал неотъемлемой частью общения в интернете.

Ты — пентестер, нанятый для проведения тестирования мессенджера RuCQ, используемого спецслужбами, с целью выявления уязвимостей. В ходе анализа приложения были обнаружены следующие уязвимости, которые оставили разработчики.

Обнаруженные уязвимости

1. Подмена Cookies для получения доступа к чужому аккаунту

Описание:

Уязвимость заключается в том, что приложение RuCQ недостаточно строго проверяет подлинность Cookies, что позволяет злоумышленнику подменить Cookie-файлы и получить доступ к учетной записи другого пользователя. Это может привести к компрометации личных данных и протекание конфиденциальной информации.

Риски:

- Неавторизованный доступ к учетным записям пользователей;

- Потеря конфиденциальности и целостности данных.

Рекомендации:

- Использовать механизм проверки подлинности Cookies на стороне сервера;

- Внедрить механизмы защиты от подмены Cookies, например, использование HttpOnly и Secure флагов.

Таблица, описывающая эксплуатацию уязвимости:

Первый шаг:	Получить любым удобным способом куки пользователя
Второй шаг:	Вставить эти куки любым удобным способом

2. Оставленный при сборке проекта протокол HTTP, не шифрующий трафик

Описание:

Приложение использует протокол HTTP вместо HTTPS, что делает все передаваемые данные уязвимыми для перехвата и модификации злоумышленниками. При отсутствии шифрования, данные, такие как сообщения и личная информация, могут быть легко просмотрены.

Риски:

- Позиция «человек посередине» (MITM), которая позволяет злоумышленникам перехватывать и изменять данные;

- Угроза компрометации личных данных пользователей.

Рекомендации:

- Перейти на использование HTTPS для всего трафика приложения;

- Настроить редиректы с HTTP на HTTPS и использовать сертификаты SSL для шифрования.

Таблица, описывающая эксплуатацию уязвимости:

Первый шаг:	Используя любой анализатор трафика (например, Wireshark), получить пакеты жертвы
Второй шаг:	Использовать полученную информацию (например, получить логин и пароль или прочитать сообщения)

3. Мелкие уязвимости, не дающие возможность привилегированного доступа

Описание:

В приложении отсутствуют ограничения на количество запросов к API, что может привести к перегрузке сервера и потенциальному отказу в обслуживании (DoS). Хотя эти уязвимости не дают возможности получить привилегированный доступ, они все же могут создать неудобства для пользователей.

Риски:

- Возможные проблемы с производительностью и доступностью сервиса.

- Увеличение нагрузки на сервер, что может повлечь за собой временные перебои в работе.

Рекомендации:

- Внедрить механизмы ограничения числа запросов (Rate Limiting) для защиты API;

- Разработать стратегии обработки нежелательных или вредоносных запросов.

Таблица, описывающая эксплуатацию уязвимости:

Первый шаг:	Записать действие к API (например, создание аккаунта)
Второй шаг:	Выполнить запрос к API используя необходимое ПО (Hoppscotch или др.)
Третий шаг:	Многократно отправлять запросы изменяя данные (можно автоматизировать)