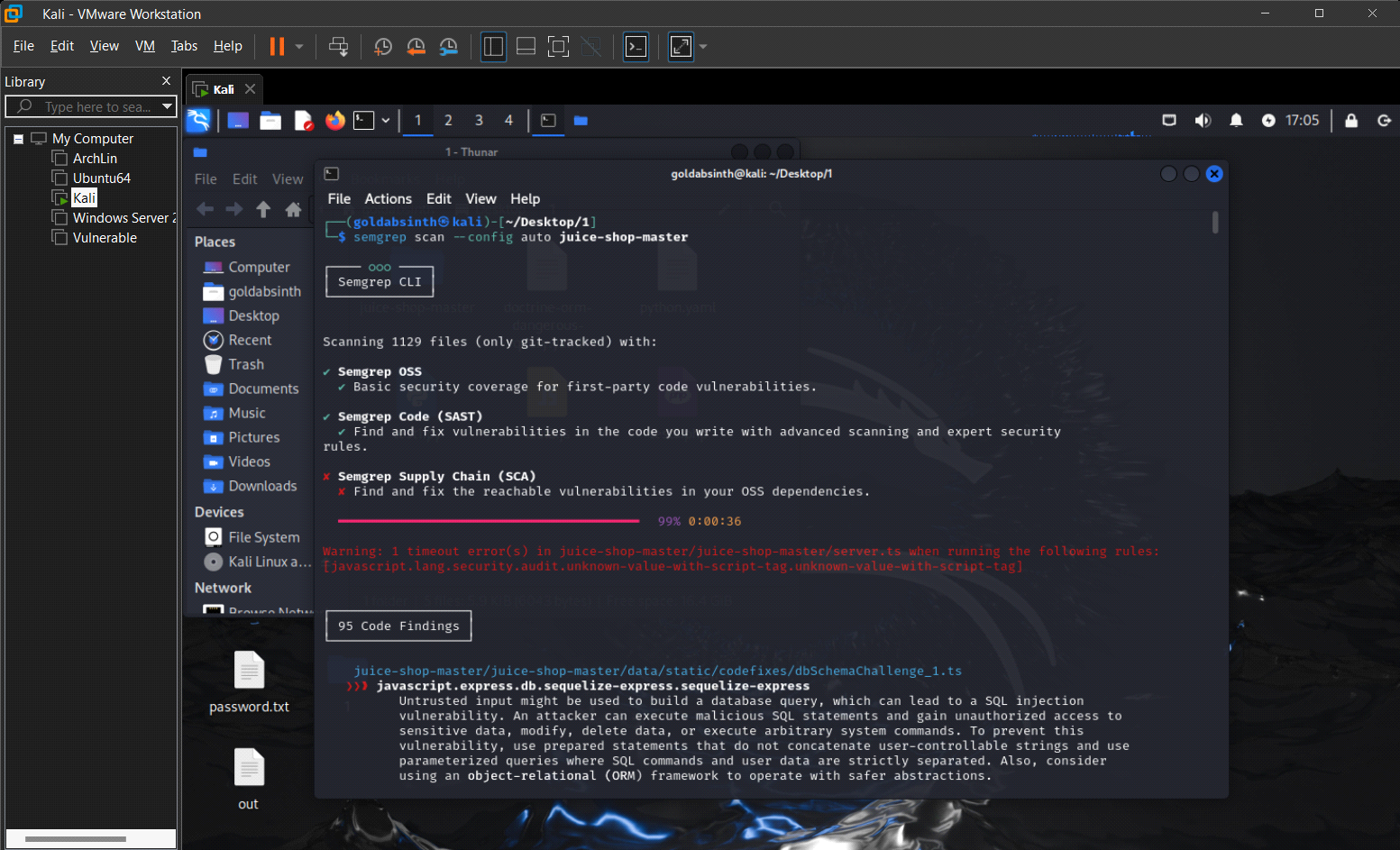
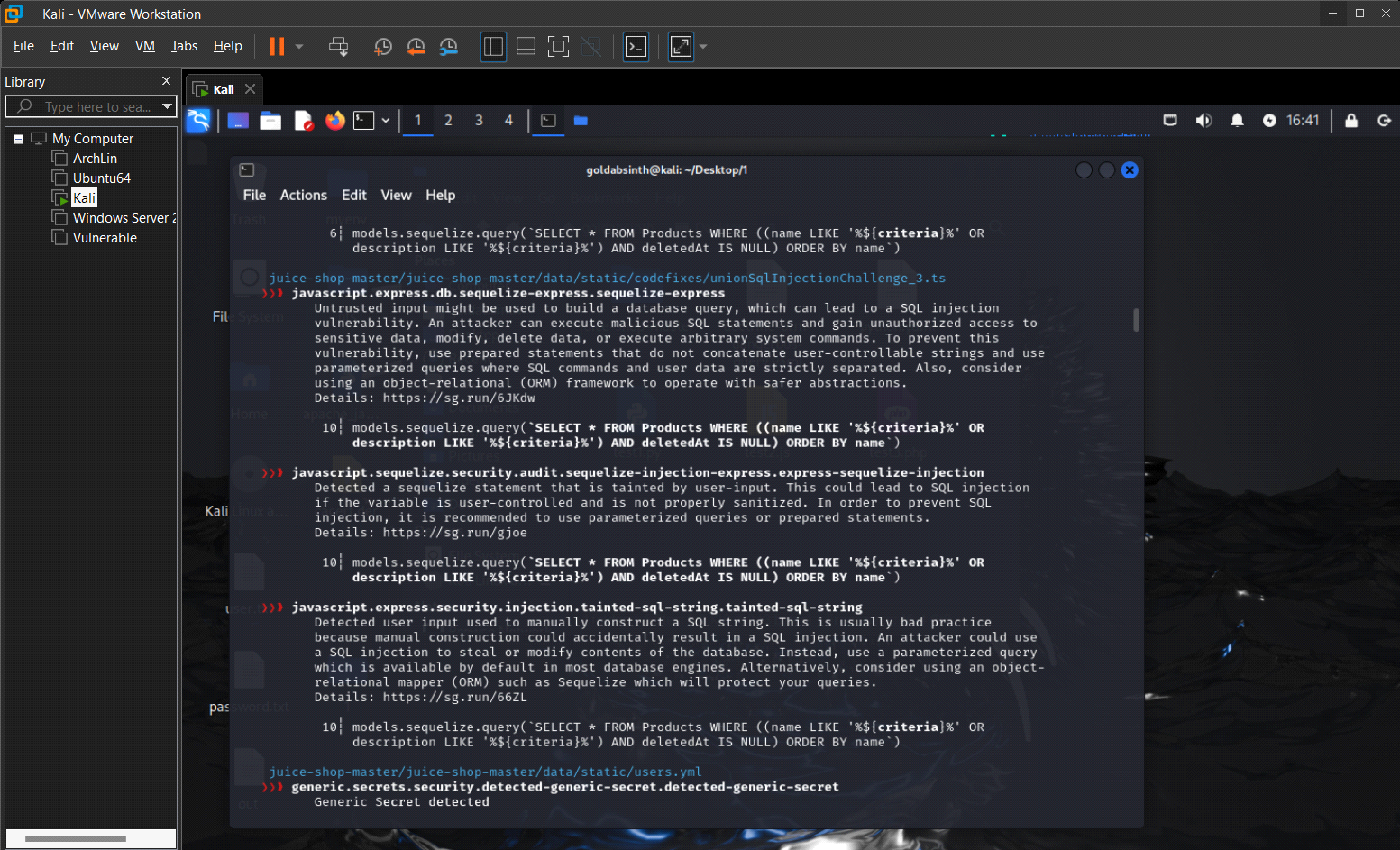
**Практическое задание. Анализ защищенности веб-приложения.**

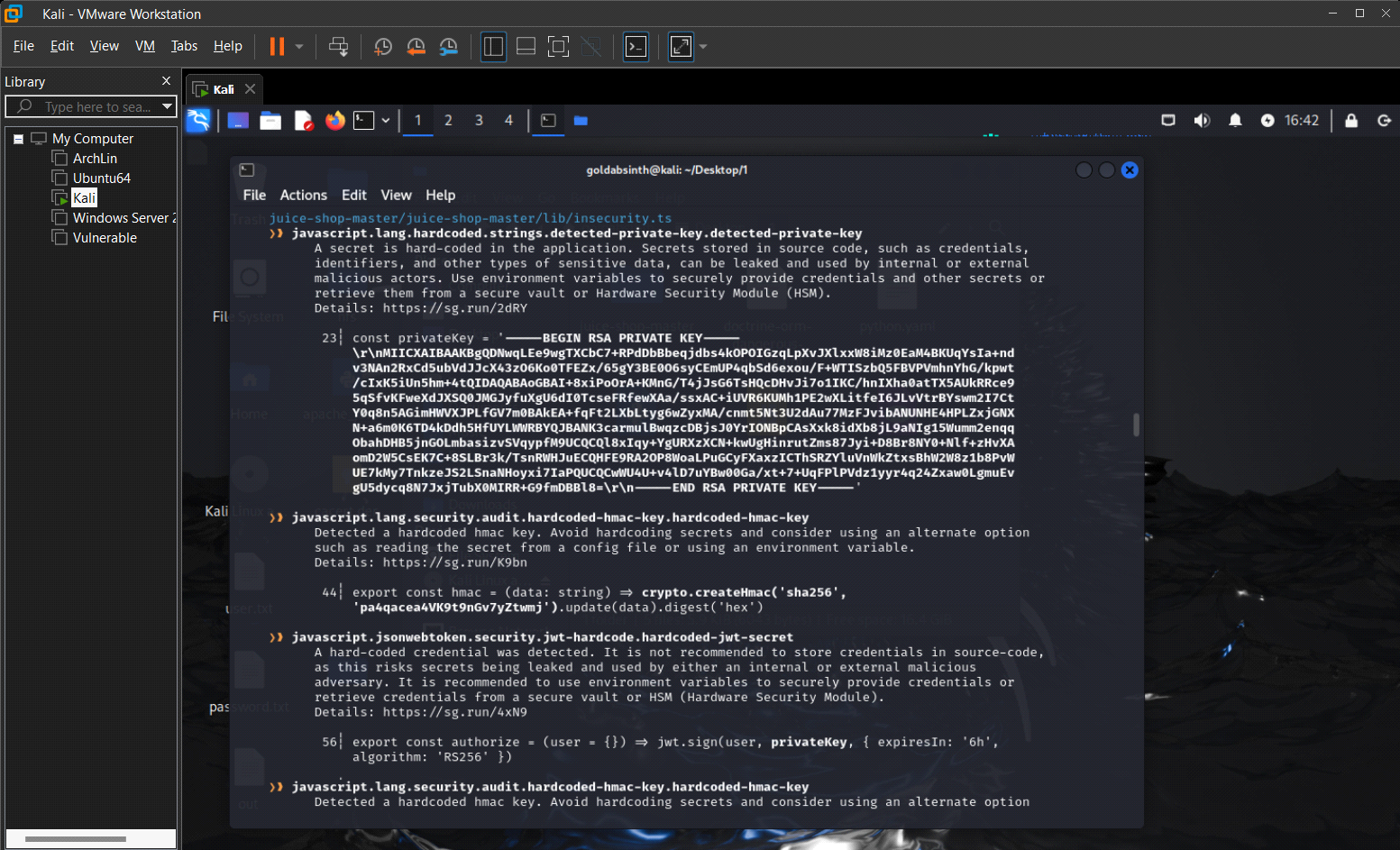
**Введение.**

OWASP Juice Shop — это проект с открытым исходным кодом, размещенный на некоммерческой платформе [Open Web Application Security Project HYPERLINK "https://owasp.org/"® (OWASP)](https://owasp.org/). Он разрабатывается и поддерживается волонтерами. Это уязвимое веб-приложение, предназначенное для обучения и практики в области безопасности веб-приложений. Оно предоставляет практикующим специалистам реалистичную и интерактивную среду, в которой они могут практиковать и улучшать свои навыки в выявлении и использовании уязвимостей веб-приложений.

**Статический анализ с использованием Semgrep.**







**Уязвимости, найденные в веб-приложении Juice Shop из OWASP TOP 10.**

A03:2021-Injection - Database Schema

A03:2021-Injection - Login Admin

A03:2021-Injection - Ephemeral Accountant

A01\_2021-Broken\_Access\_Control - Manipulate Basket

A01\_2021-Broken\_Access\_Control - View Basket

A03:2021-Injection - CWE-79: Cross-site Scripting, CWE-89: SQL Injection, and CWE-73: External Control of File Name or Path

A01\_2021-Broken\_Access\_Control - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, CWE-201: Insertion of Sensitive Information Into Sent Data, and CWE-352: Cross-Site Request Forgery.

**Эксплуатация уязвимостей в OWASP Juice Shop**:

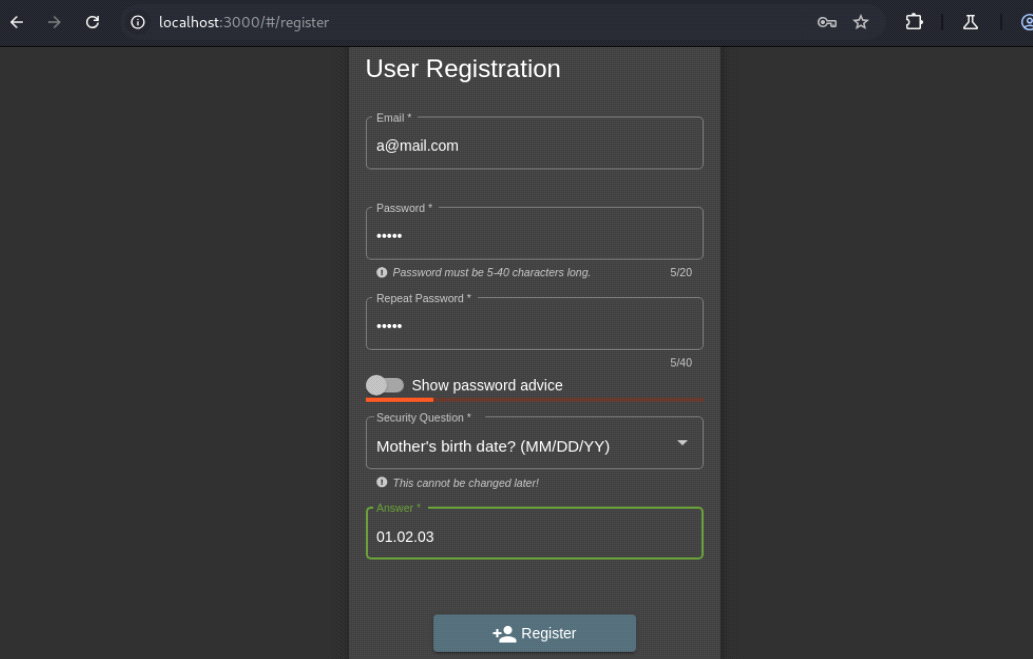
* Broken Access Control

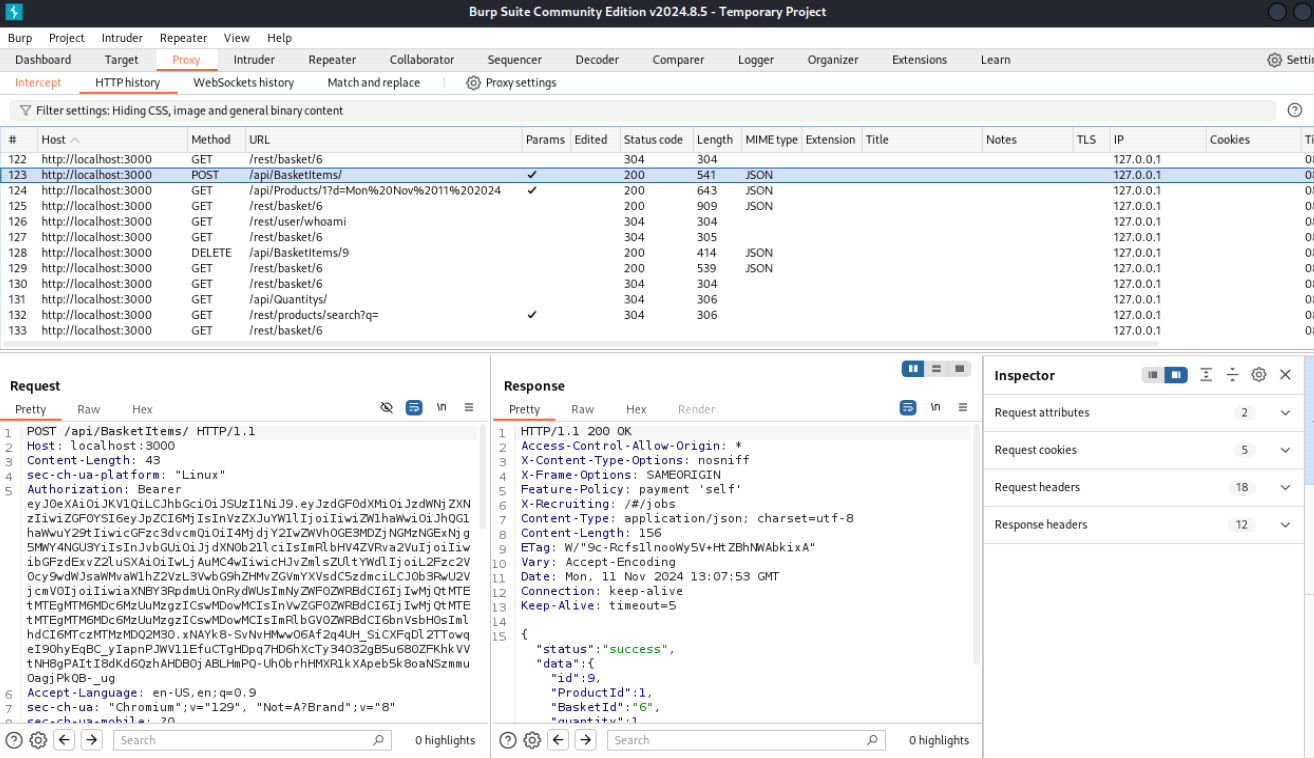
Задача: Поместить товар в корзину другого пользователя.  
 - создаем нового пользователя и добавляем в корзину товар

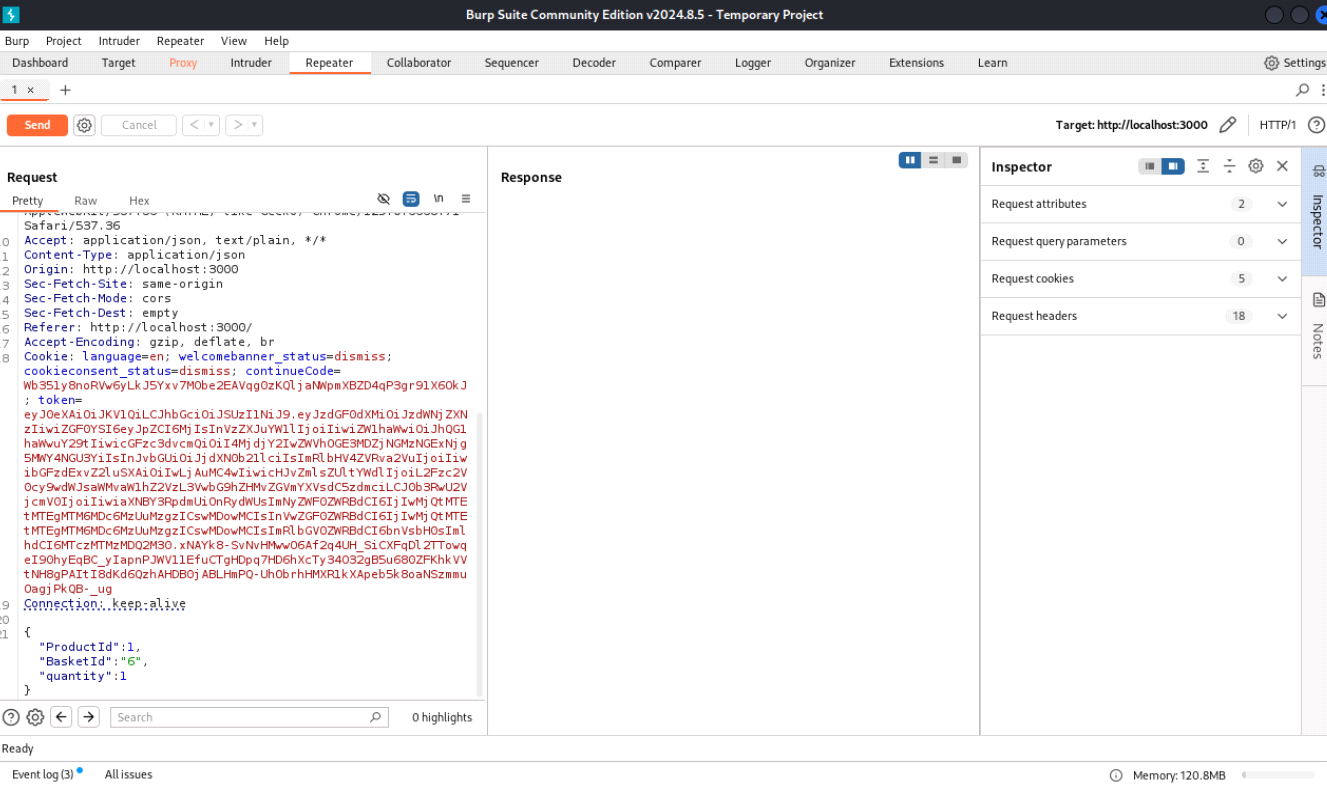
- в Burpsuite находим пакет POST /api/BasketItems/ HTTP/1.1

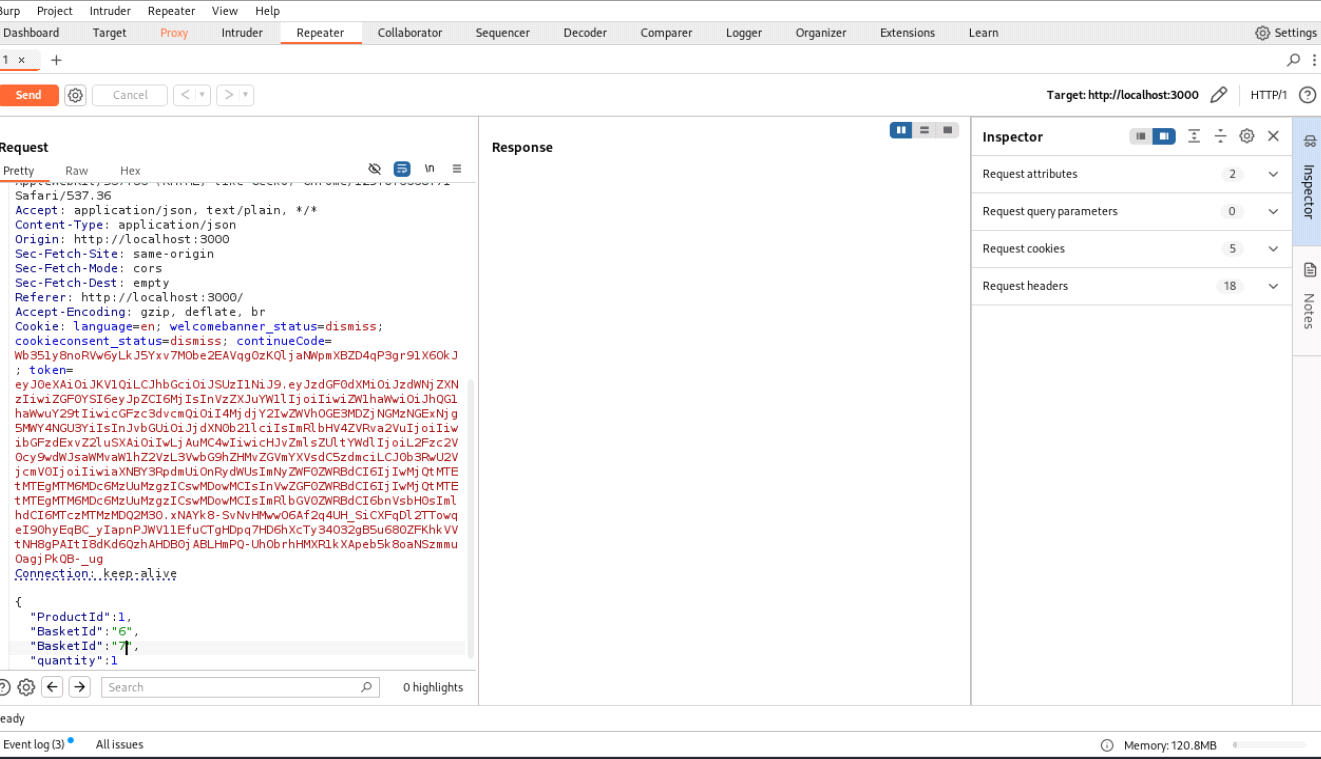
- отправляем его в репитер

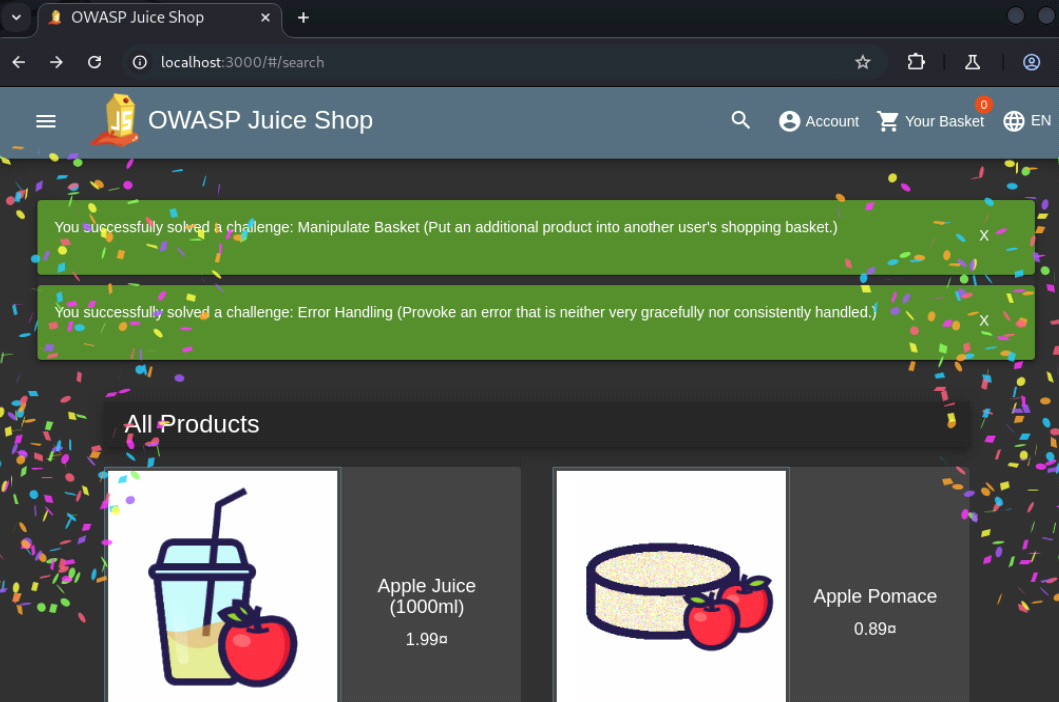
- добавляем в запросе строку с корзиной под другим номером и отправляем











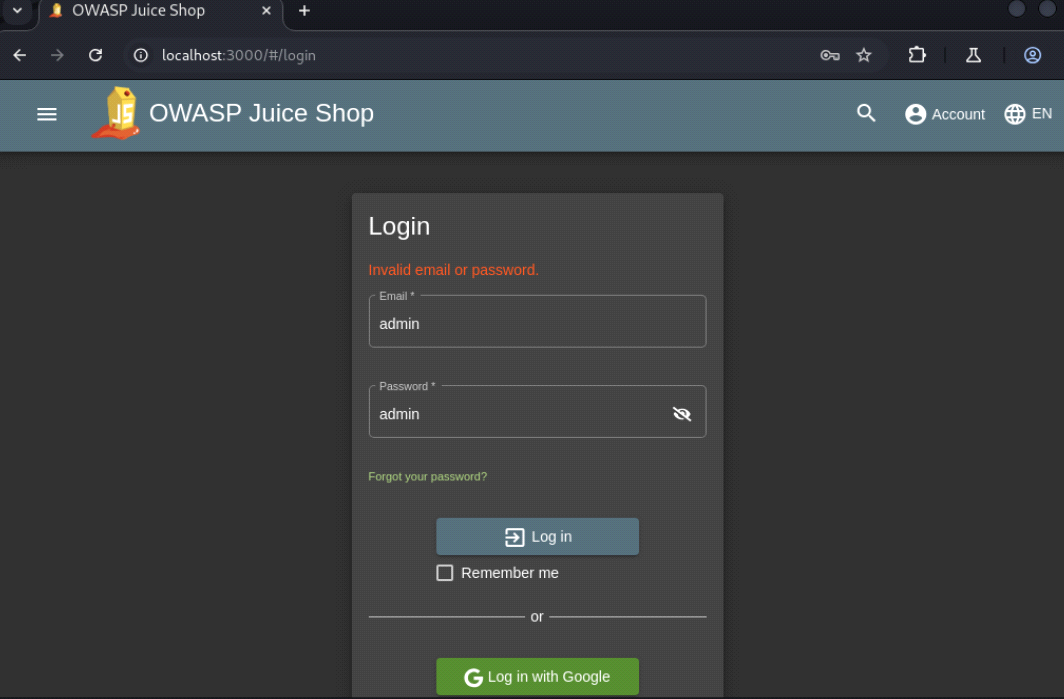
* Injection

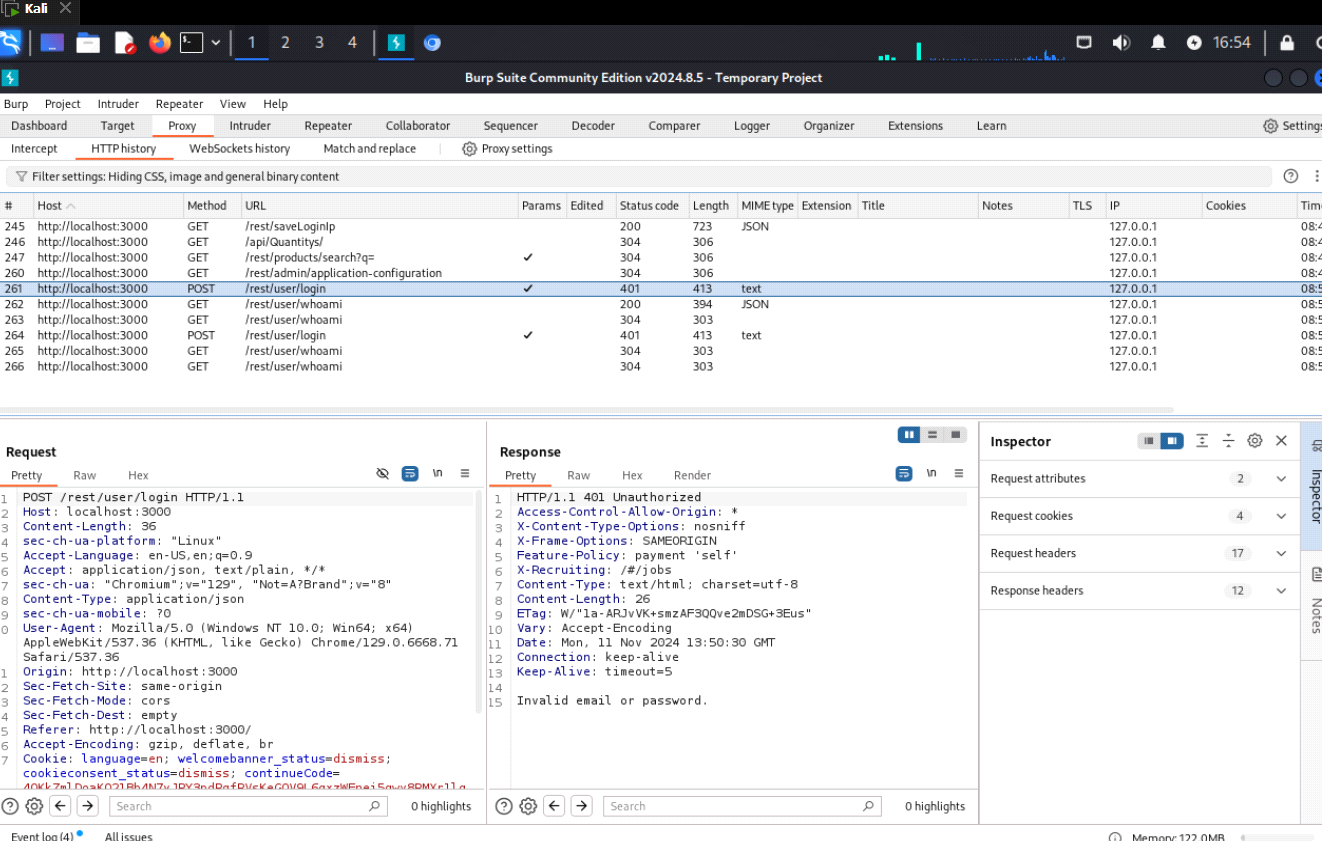
- пробуем залогиниться введя логин/пароль: admin - admin

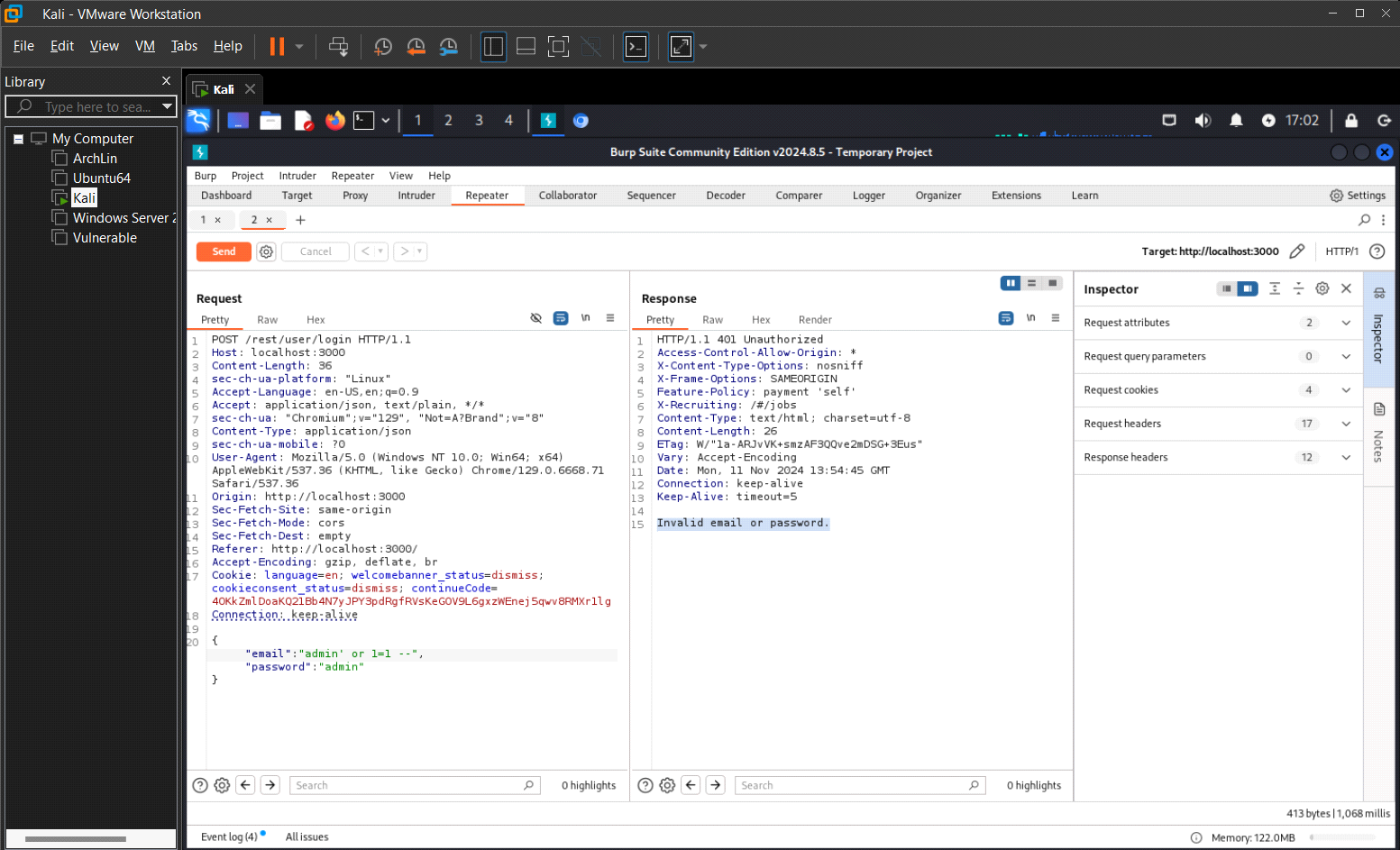
- находим запрос и отправляем в репитер

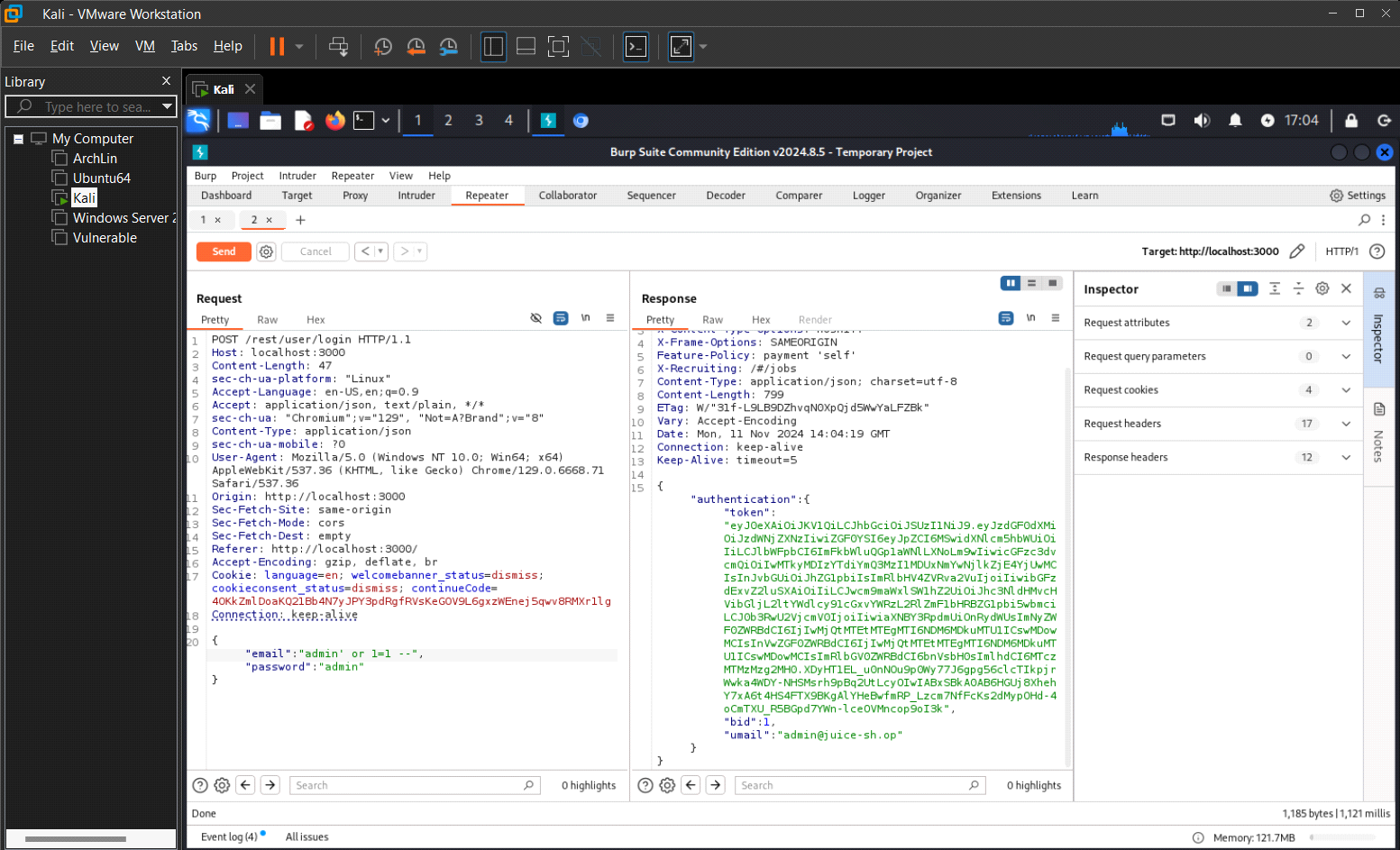
- вставляем простейшую инъекцию <<’ or 1=1 -->>

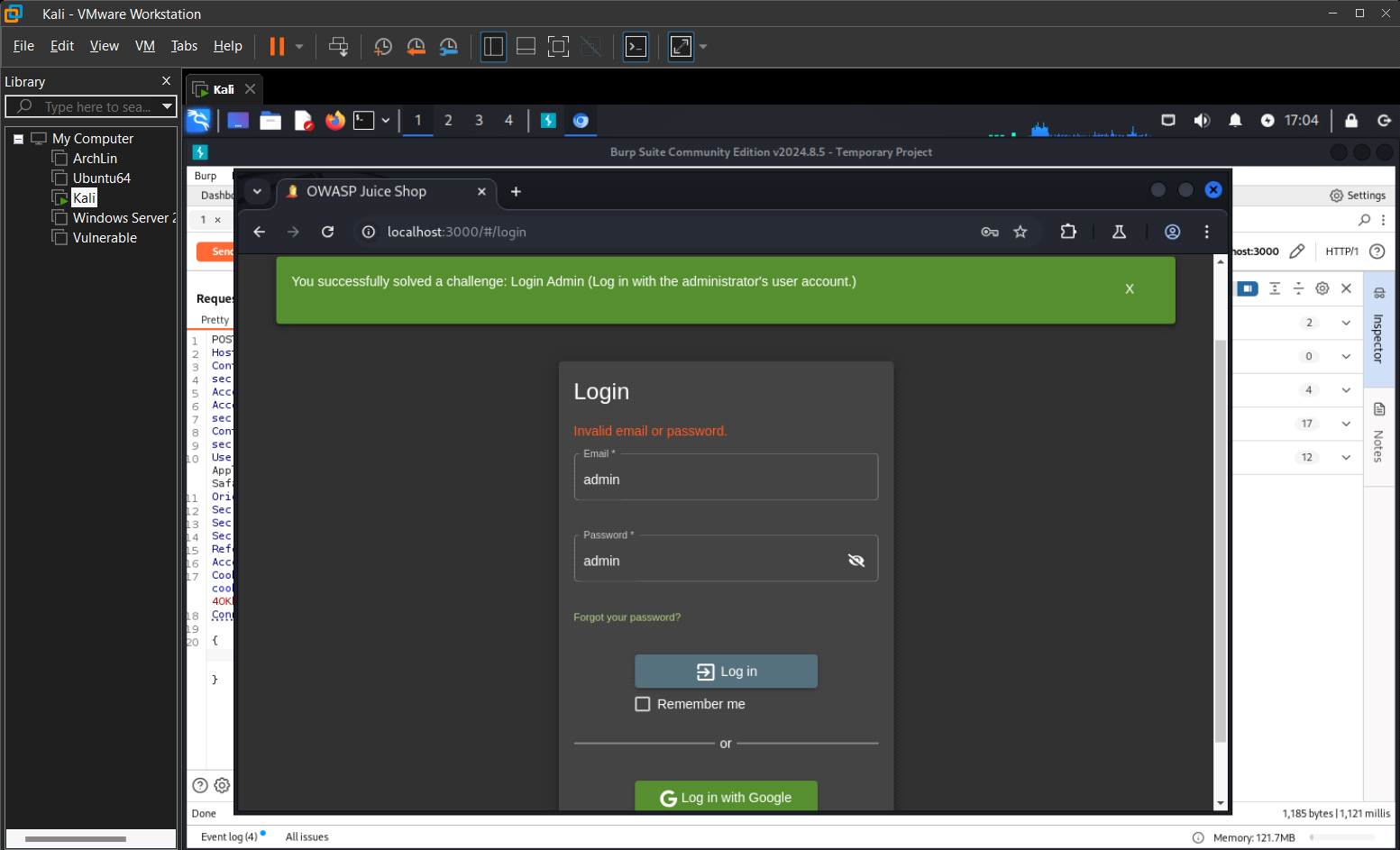
- повторяем запрос и получаем логин администратора и его токен









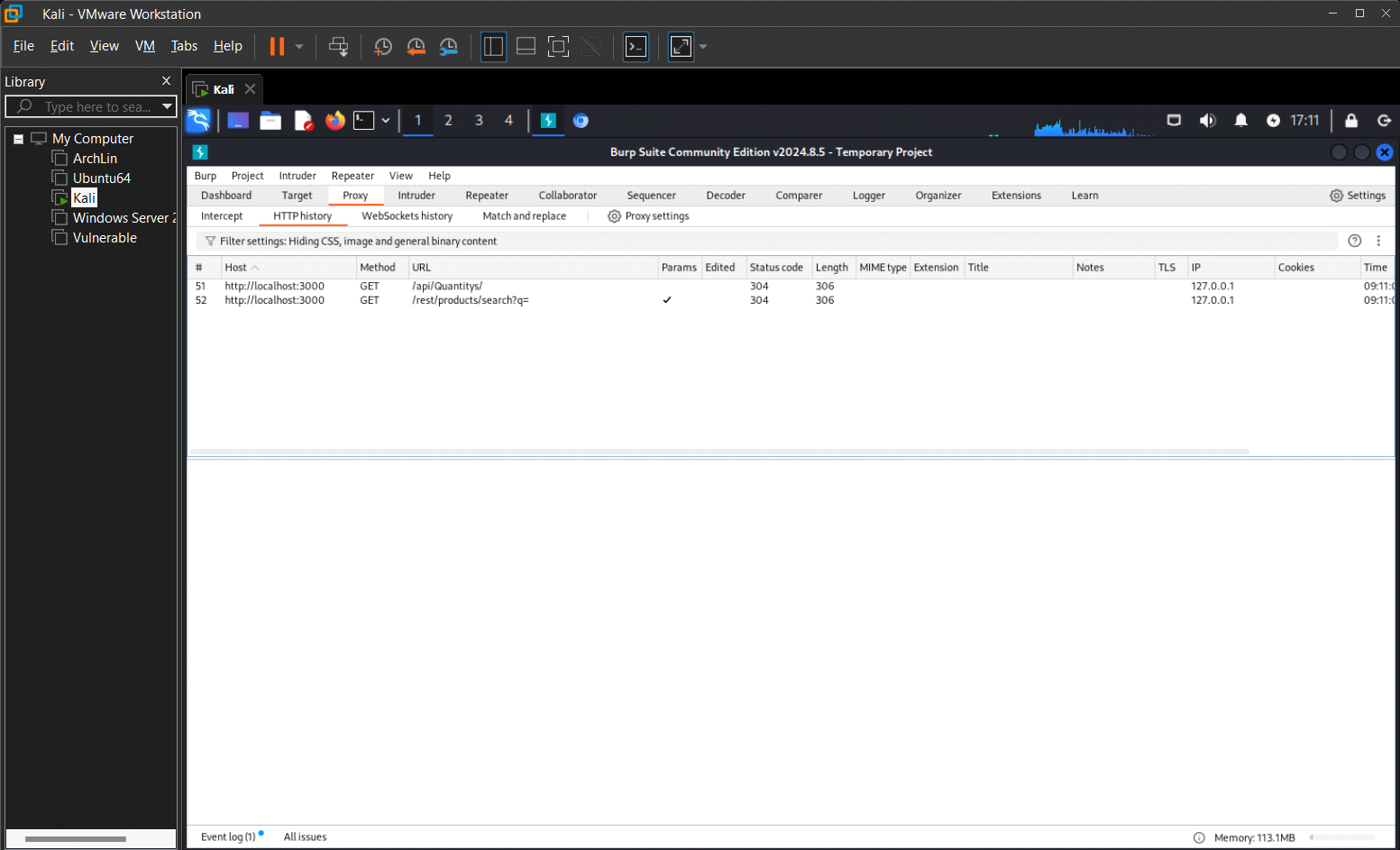


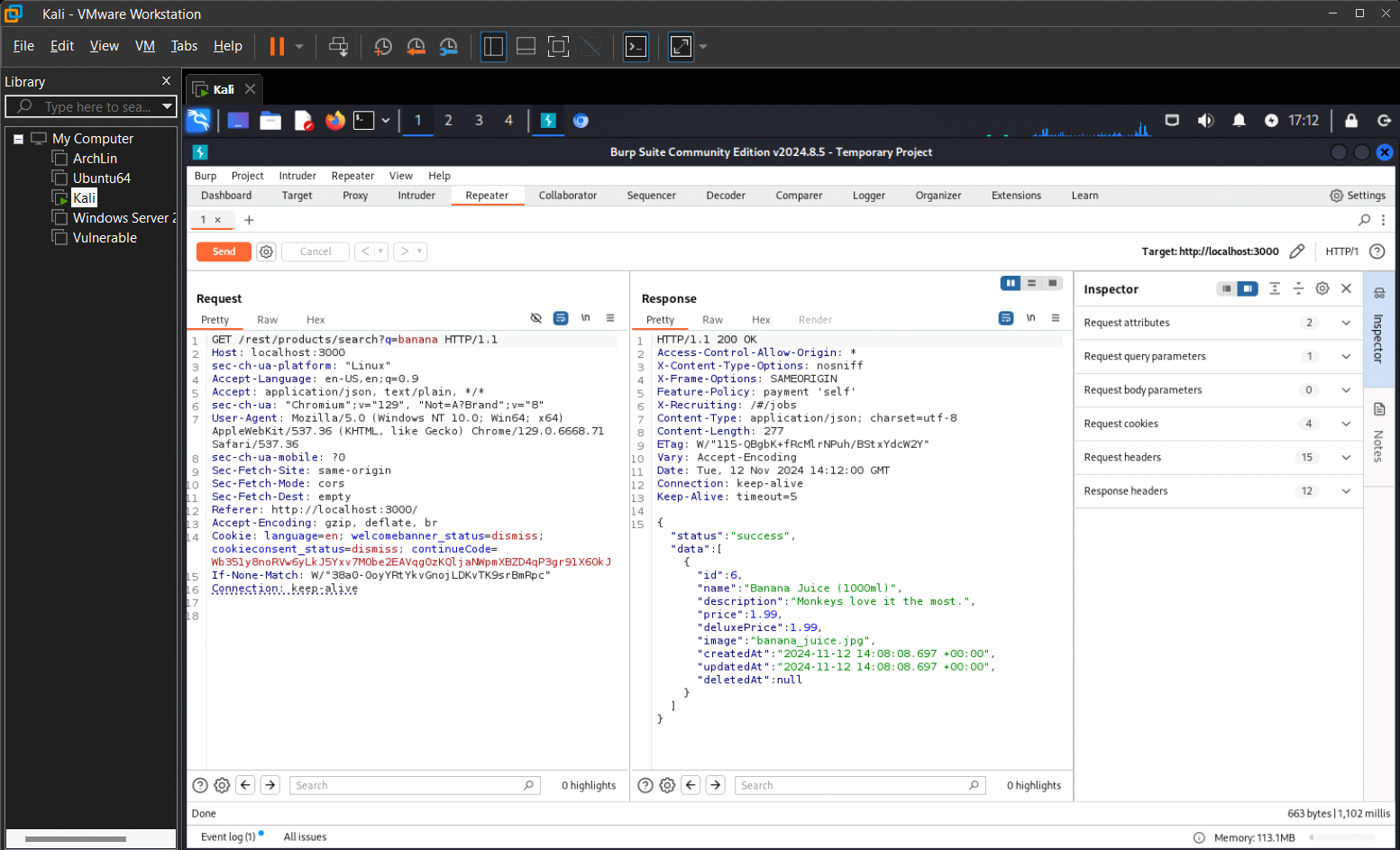
* Получение структуры базы данных. Injection.

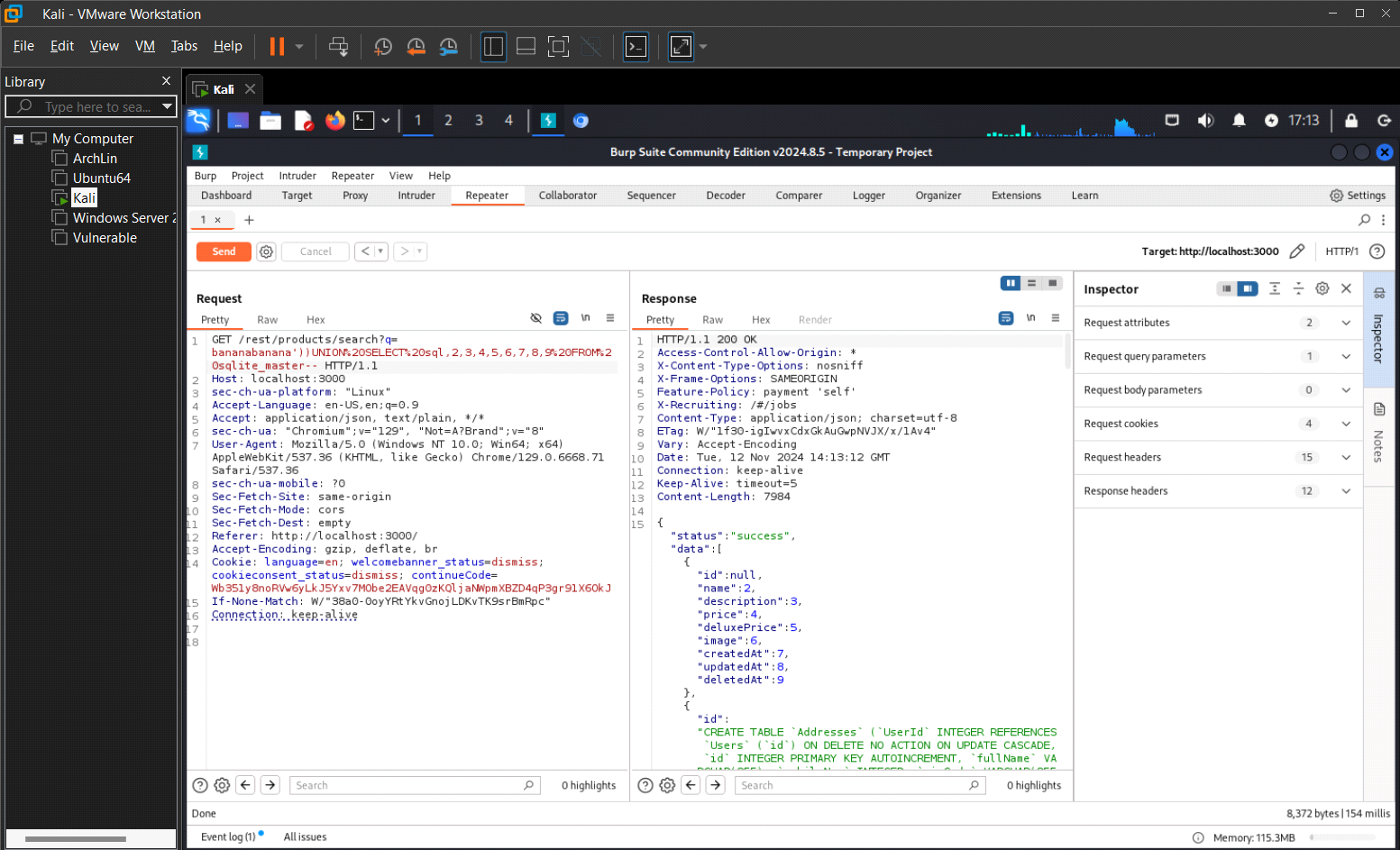
- выполняем поисковый запрос и находим его в POST - /rest/products/search?q=

- отправляем на репитер, дополняем запрос аргументом banana, получим корректный ответ; отправив ‘ получим сообщение об ошибке и понимание, что используется sqlite

-используем инъекцию - '))UNION%20SELECT%20sql,2,3,4,5,6,7,8,9%20FROM%20sqlite\_master-- и получим ответ со структурой БД







4.Password Strenght. Перебор пароля. A07:2021 – Identification and Authentication Failures

CWE-307: Improper Restriction of Excessive Authentication Attempts

- пробуем зайти в учетную запись администратора используя его email и случайный пароль

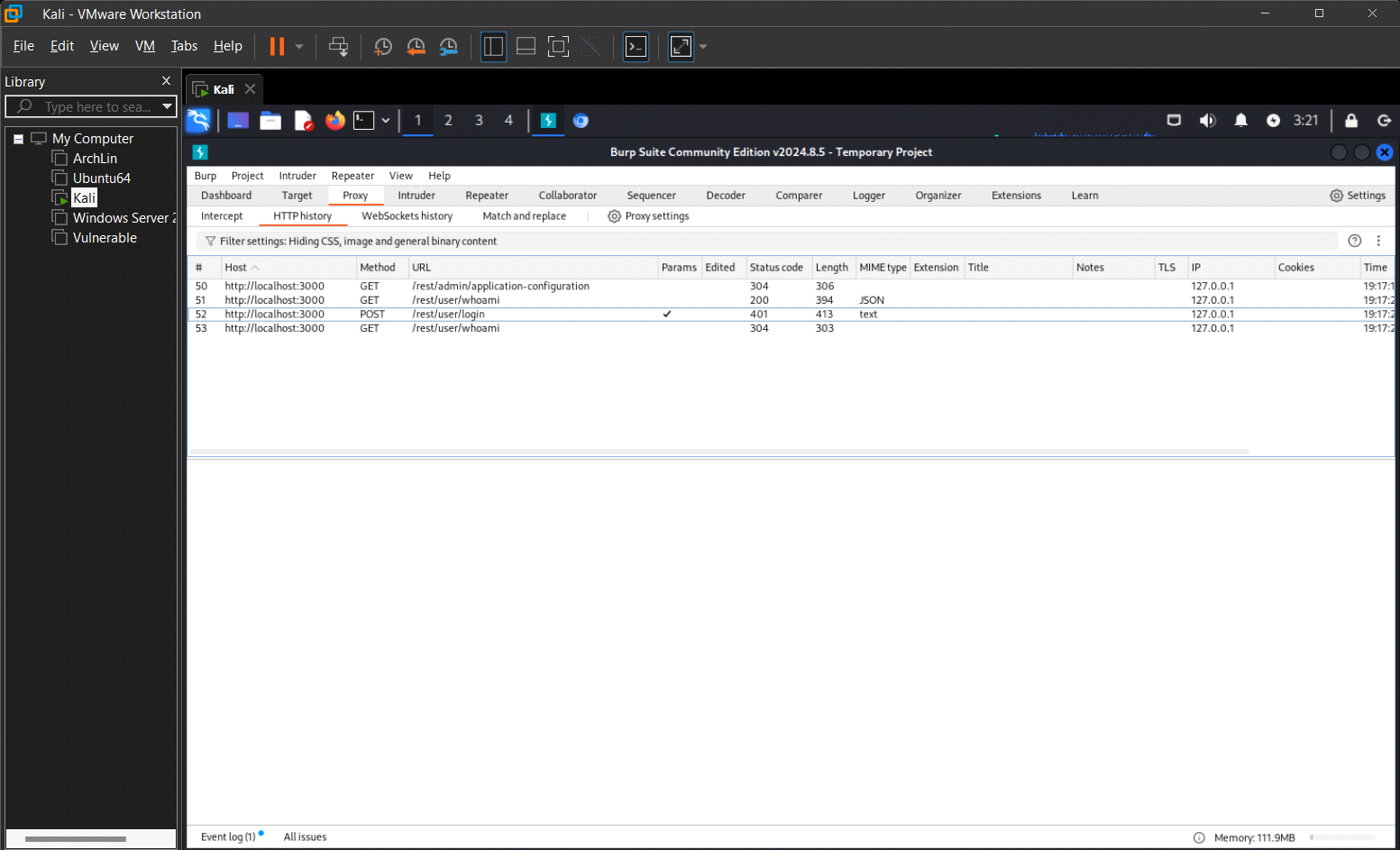
- находим POST запрос и отправляем его в интрудер

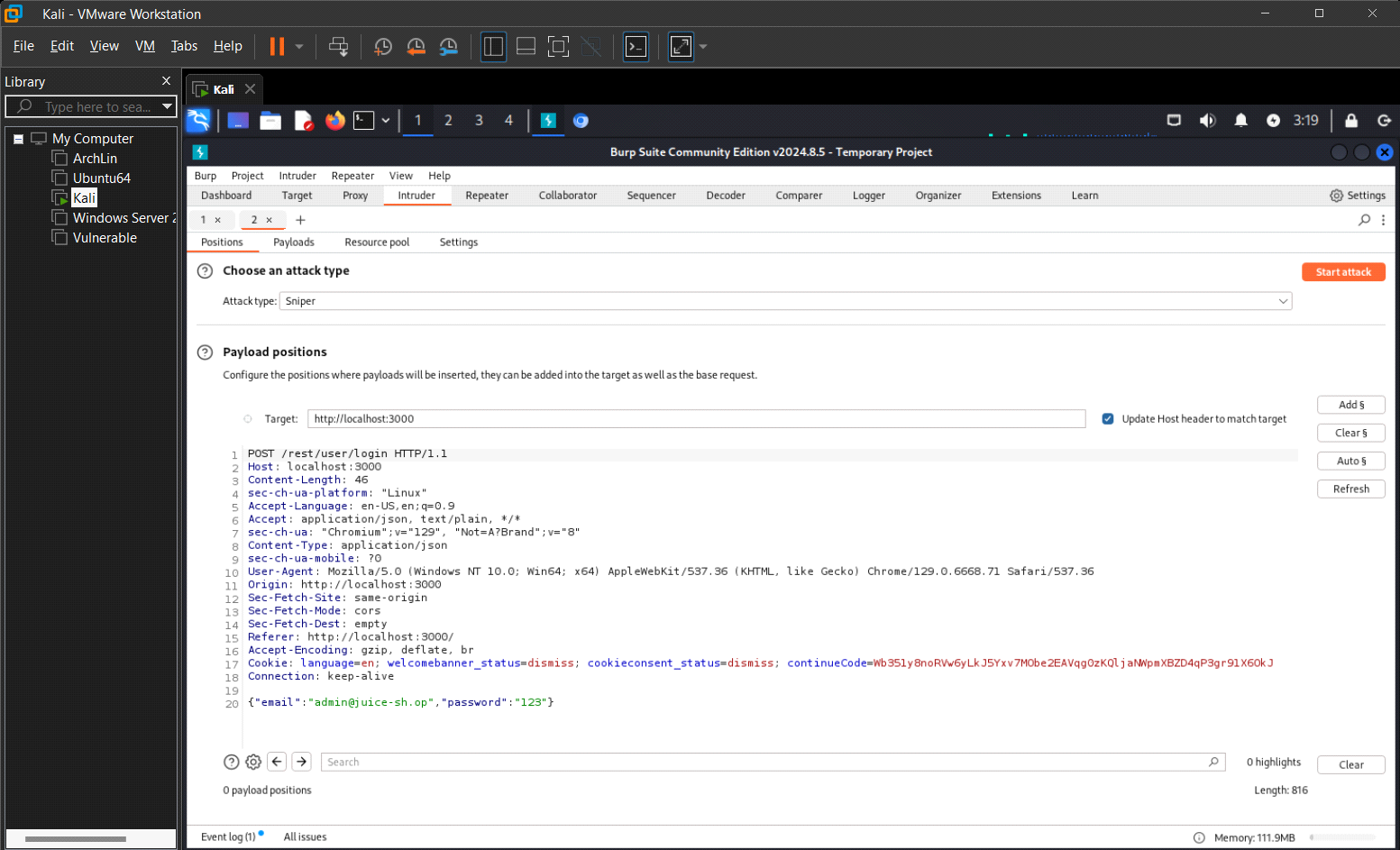
- находим позицию для загрузки

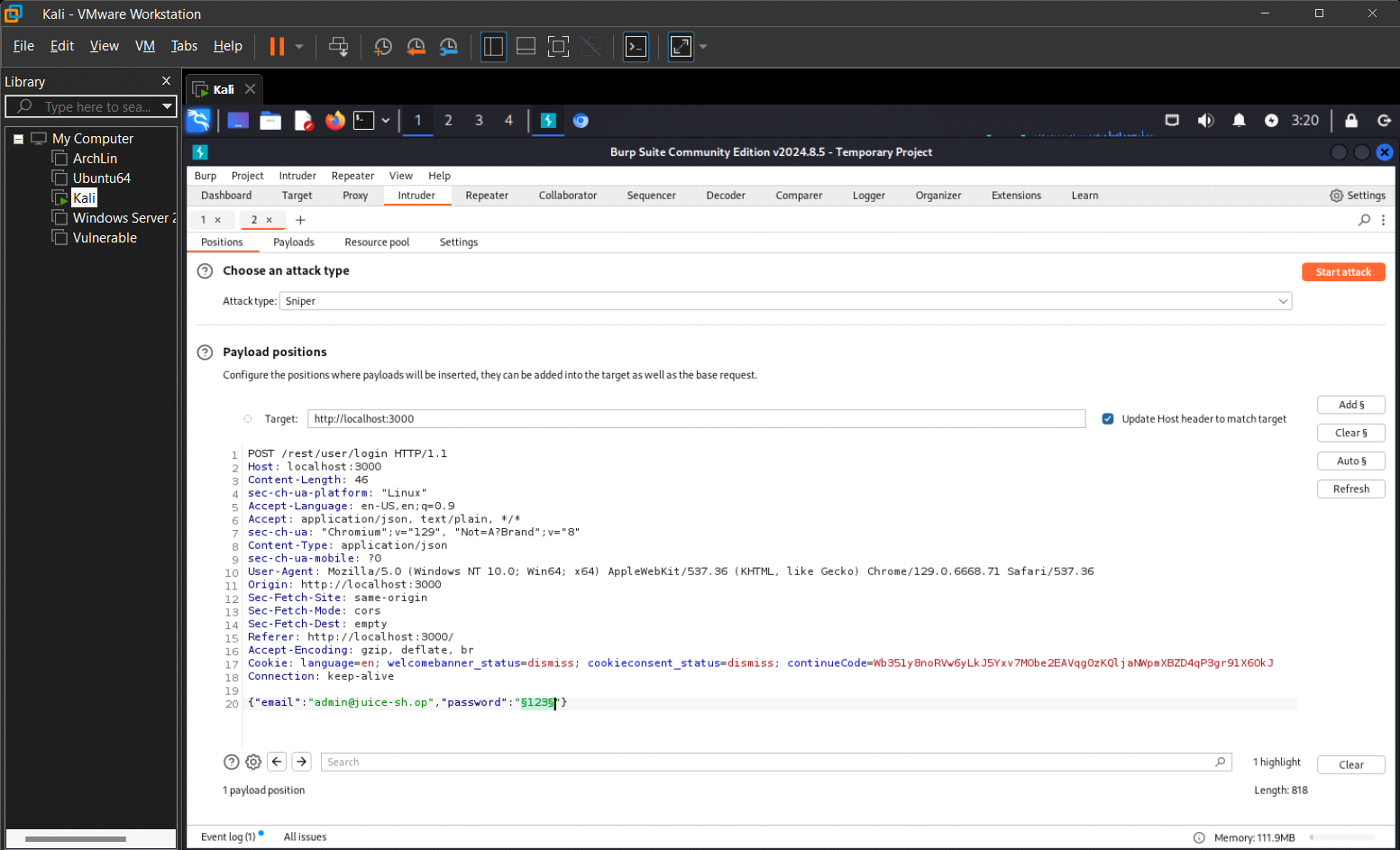
- добавляем словарь самых распространенных паролей

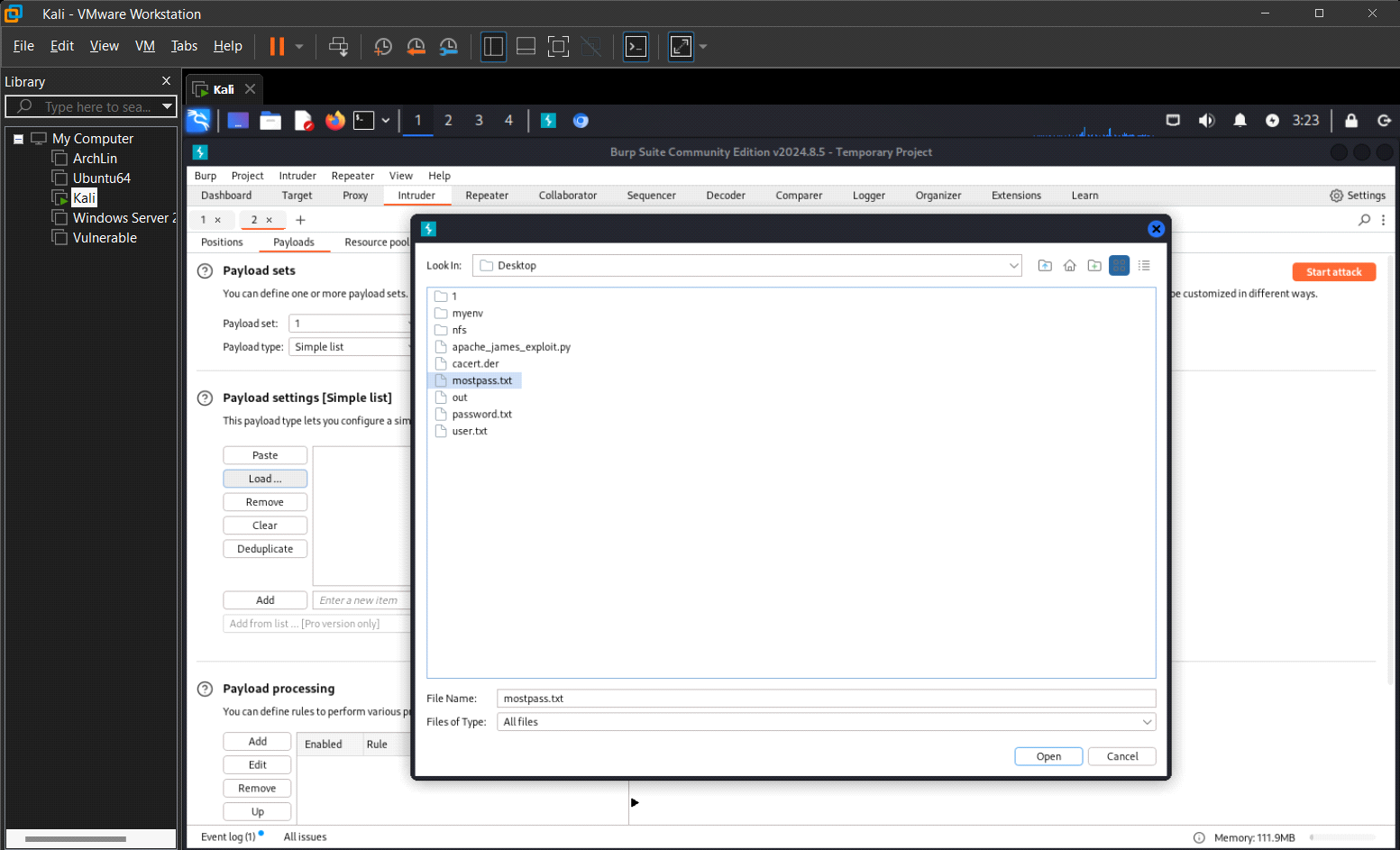
- указываем регулярное выражение для автоматизации процесса и получения данных от сервера

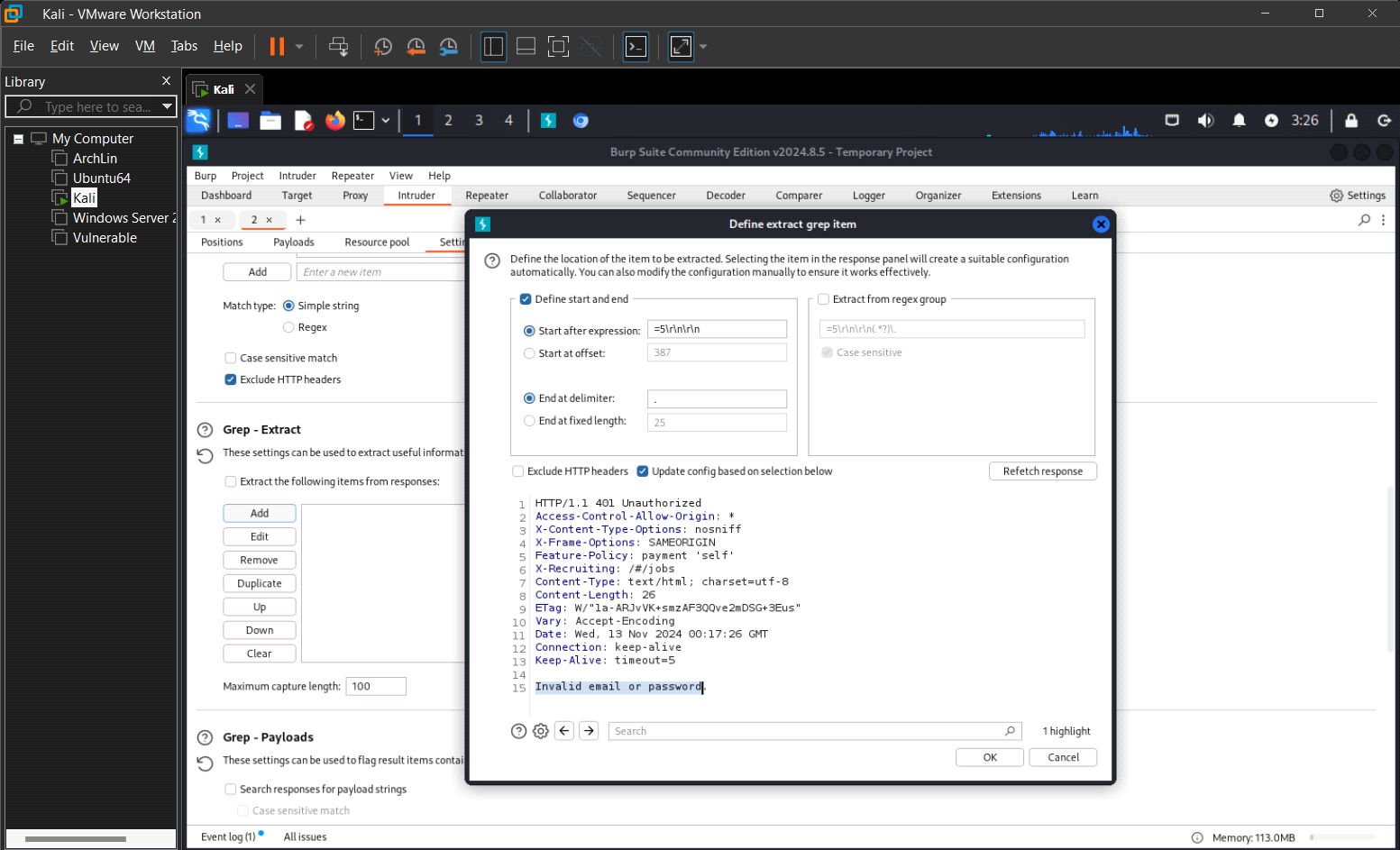
- запускаем атаку, получаем пароль и токен аутентификации

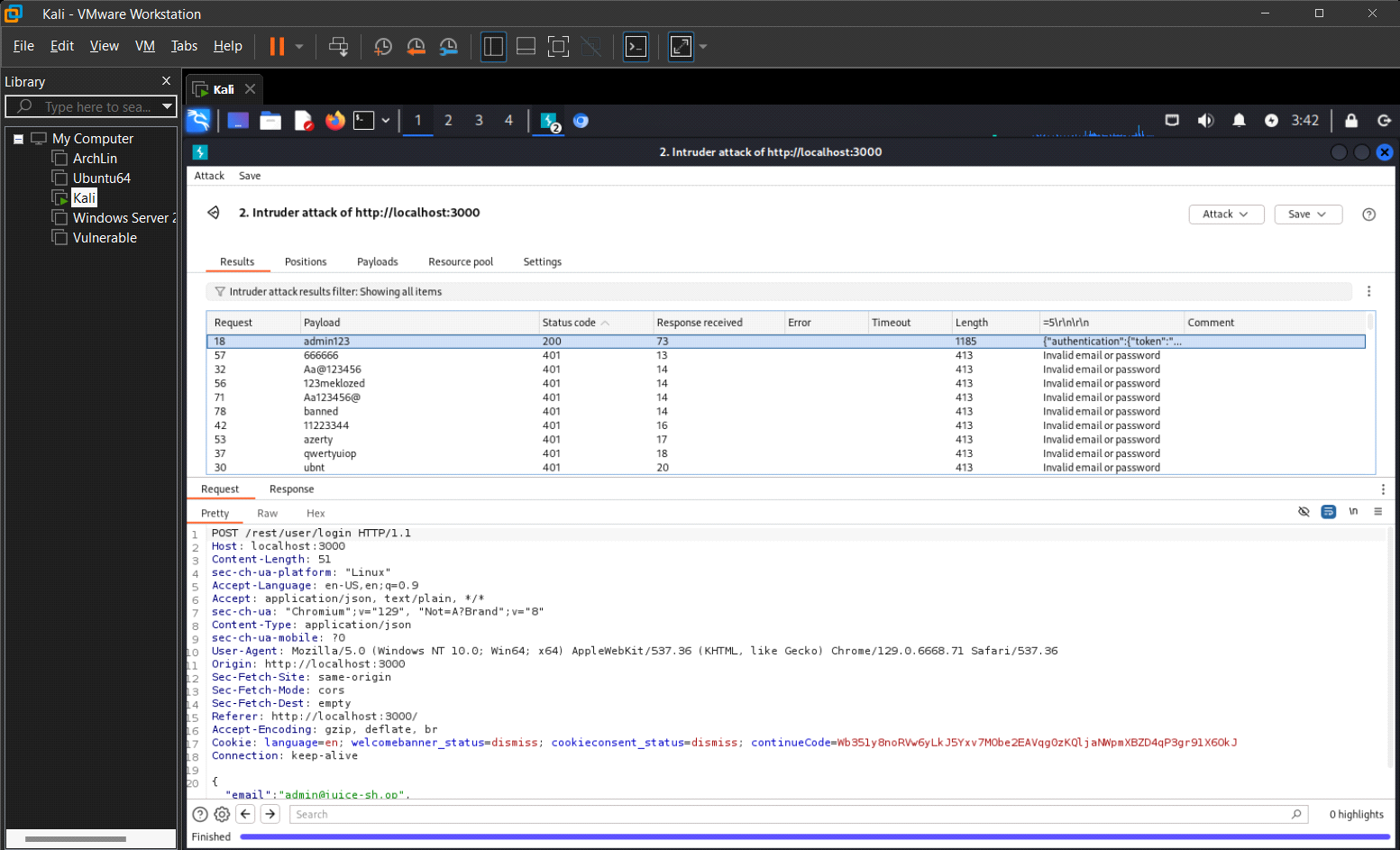


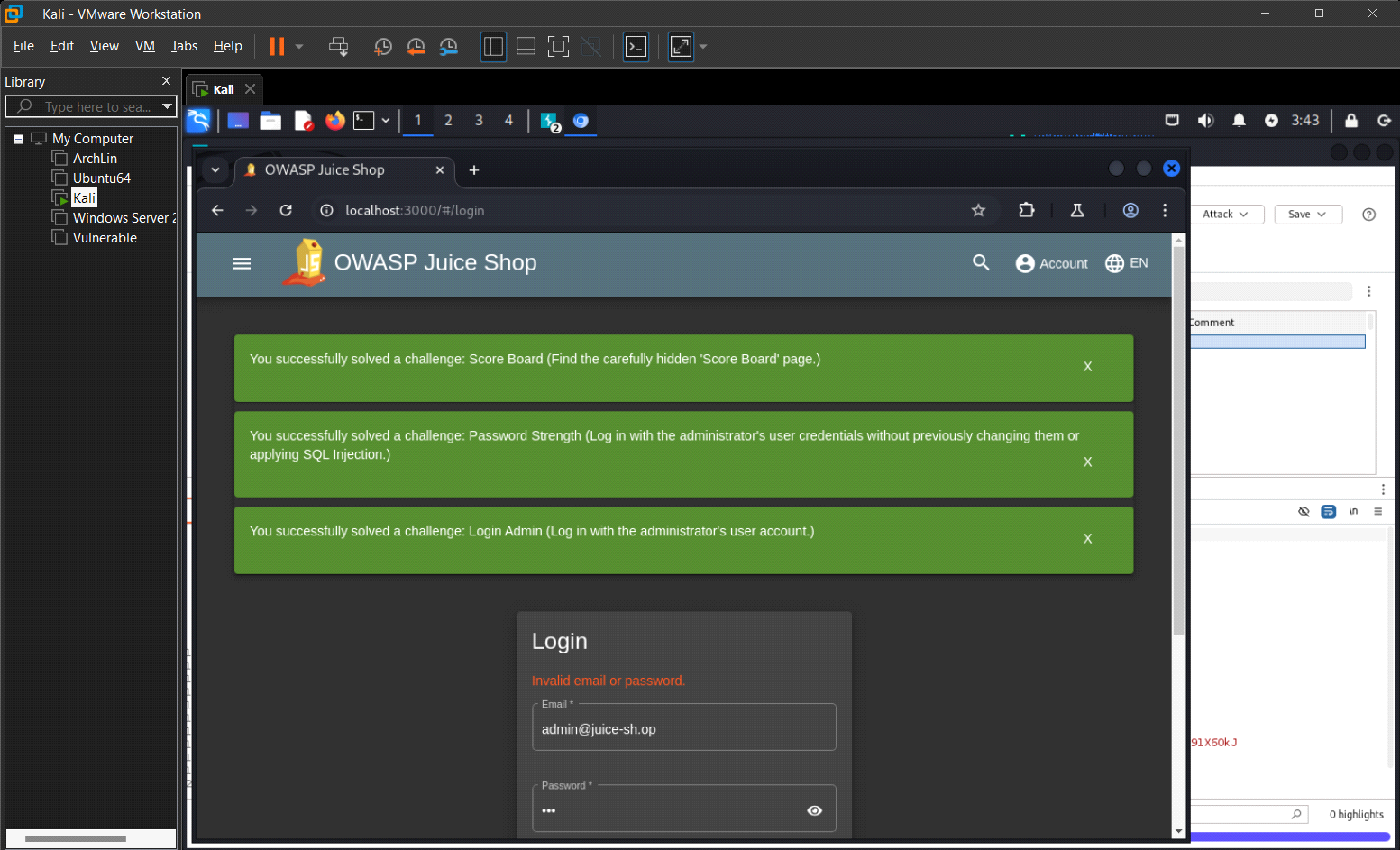












**Рекомендации по устранению к продемонстрированным уязвимостям.**

**Injection (инъекция)** — это уязвимость, при которой злоумышленник может ввести вредоносный код в приложение через входные данные. Наиболее распространенные виды инъекций включают SQL-инъекцию, NoSQL-инъекцию, OS-инъекцию и LDAP-инъекцию.

Рекомендации по устранению:

Использовать параметризованные запросы и подготовленные выражения:

SQL: параметризованные запросы или подготовленные выражения для предотвращения SQL-инъекций:

SELECT \* FROM users WHERE username = ? AND password = ?

NoSQL: параметризованные запросы или библиотеки, которые автоматически экранируют входные данные.

Использование ORM (Object-Relational Mapping), таких как Hibernate для Java или Sequelize для Node.js, они автоматически экранируют входные данные и предотвращают инъекции.

Валидация и экранирование входных данных:

Валидация и экранирование входных данные перед их использованием в запросах.

Использование библиотек для валидации и экранирования, такие как OWASP ESAPI.

Использование безопасных API и библиотек:

Использование безопасных API и библиотек для выполнения запросов к базе данных.

Исключение использования устаревших или небезопасных функций.

Использование механизмов контроля доступа:

Ограничение доступа к базе данных только тем пользователям и приложениям, которым это необходимо.

Использование принципа наименьших привилегий для управления доступом к базе данных.

А так же:

Регулярные обновления и патчи

Мониторинг и логирование

Обучение и осведомленность

**Уязвимость A01:2021-Broken Access Control** занимает первое место в списке OWASP Top Ten 2021. Она возникает, когда приложение не ограничивает доступ к функциям и данным на основе ролей и привилегий пользователей.

Рекомендации по устранению уязвимости:

Принцип наименьших привилегий:

Предоставление пользователям только тех привилегий, которые им действительно необходимы для выполнения их задач.

Ограничение доступа к функциям и данным на основе ролей и привилегий.

Проверка доступа на серверной стороне:

Проверка доступа на серверной стороне, а не только на клиентской.

Нельзя полагаться на клиентскую сторону для проверки доступа, так как клиентские данные могут быть подделаны.

Централизованные механизмы контроля доступа:

Использование централизованных систем управления доступом, такие как OAuth, OpenID Connect или другие стандарты аутентификации и авторизации.

Централизованные системы облегчают управление доступом и обеспечивают единообразную политику безопасности.

А так же:

Регулярные обновления и патчи

Мониторинг и логирование

Обучение и осведомленность

Тестирование безопасности OWASP ZAP или Burp Suite

**A07:2021 – Identification and Authentication Failures (Ошибки идентификации и аутентификации)** — это одна из категорий уязвимостей, перечисленных в списке OWASP Top Ten 2021. Эта категория включает в себя различные проблемы, связанные с неправильной реализацией механизмов идентификации и аутентификации пользователей.

Рекомендации по защите:

Использование сильных паролей: Требовать от пользователей использования сложных паролей и регулярно обновлять их.

Безопасное хранение паролей: Использовать сильные алгоритмы хеширования, такие как bcrypt, scrypt или Argon2, для хранения паролей.

Внедрение многофакторной аутентификации (MFA): Использовать дополнительные методы аутентификации для повышения безопасности.

Управление сессиями: Правильно управлять сессиями пользователей, используя безопасные методы и протоколы.

Защита от атак перебора паролей: Внедрять механизмы защиты от атак перебора паролей, такие как ограничение количества попыток входа или использование CAPTCHA.

Обновление протоколов аутентификации: Использовать современные и безопасные протоколы аутентификации, такие как OAuth 2.0 или OpenID Connect