

Engenharia de Software Seguro

Laboratório 4 — Vulnerable Apps (Mobile + Web)

Grupo 6

Elementos: Hugo Nunes, Miguel Teixeira

Repositório GitHub: c-academy-ess/api-todo-list-manager-grupo-6

Tag(s): lab4

Data: 14-02-2026

Objetivo. Identificar e explorar vulnerabilidades em aplicações móveis e web usando análise estática e dinâmica, conforme enunciado do Laboratório 4.

Ferramentas. Bytecode Viewer, Java/JCE, Docker, OWASP ZAP, Visual Studio Code + extensão CodeQL.

Conteúdo

1. Preparação de ferramentas.....	3
2. Ataques em aplicações móveis (Lab4-ess.apk).....	3
2.1 Artefactos encontrados (frase cifrada, chave e algoritmo).....	3
2.2 Resolução com DecodeAndroidKey.java	4
3. CodeQL — análise local (vulnapp-ess.zip + vulnapp_db.zip).....	5
3.1 CWE-79 (XSS).....	5
3.2 CWE-89 (SQLi)	5
3.3 CWE-327 (cripto fraca).....	6
4. Análise dinâmica — OWASP Juice Shop.....	6
4.1 Execução em Docker.....	6
4.2 Descobrir o Score Board (rota escondida)	7
4.3 SQL Injection no login (admin)	7
4.4 ZAP — Manual Explore	9
4.5 Fuzzing da password admin###	9
4.6 Alterar termo de pesquisa na barra de endereço.....	11
4.7 DOM XSS	11
4.8 Persisted XSS via API.....	12
4.9 Feedback em nome de outro utilizador	12
4.10 Roubo do cookie/token.....	13
5. Conclusão	13
Referências	14

1. Preparação de ferramentas

- Bytecode Viewer (v2.13.2) para descompilar/inspecionar o APK.
- VS Code + extensão CodeQL para executar interrogações QL sobre uma base de dados CodeQL.
- Docker para correr OWASP Juice Shop.
- OWASP ZAP para proxy/interceção e fuzzing.

2. Ataques em aplicações móveis (Lab4-ess.apk)

A aplicação pede uma frase ao utilizador e valida-a no método `MainActivity.verify(View v)`. Pela análise estática ao bytecode (strings do DEX), é possível recuperar os artefactos criptográficos usados e decifrar a frase correta.

2.1 Artefactos encontrados (frase cifrada, chave e algoritmo)

No APK foram observadas as seguintes constantes relevantes:

Artefacto	Valor
Frase cifrada (Base64)	5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2GOc=
Chave (hex, 16 bytes)	8d127684cbc37c17616d806cf50473cc
Algoritmo / modo / padding	AES / ECB / PKCS7Padding (equivalente a PKCS5Padding em Java)

Decifrando o ciphertext Base64 com AES-ECB e a chave acima, obtém-se a frase correta:

I want to believe

2.2 Resolução com DecodeAndroidKey.java

O programa foi completado preenchendo ciphertext, chave e transformação do Cipher.

Código java modificado:

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class DecodeAndroidKey_solved {
    private static final String ENCRYPTED_BASE64 =
        "5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2G0c=";
    private static final String KEY_HEX =
        "8d127684cbc37c17616d806cf50473cc";

    public static void main(String[] args) throws Exception {
        byte[] encryptedBytes = Base64.getDecoder().decode(ENCRYPTED_BASE64);
        byte[] keyBytes = hexStringToByteArray(KEY_HEX);

        SecretKeySpec secretKey = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);

        byte[] decrypted = cipher.doFinal(encryptedBytes);
        System.out.println("Descodificado: " + new String(decrypted));
    }

    private static byte[] hexStringToByteArray(String hex) {
        int len = hex.length();
        byte[] out = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            out[i / 2] = (byte) ((Character.digit(hex.charAt(i), 16) << 4)
                + Character.digit(hex.charAt(i + 1), 16));
        }
        return out;
    }
}
```

Saída ao executar:

Descodificado: I want to believe

3. CodeQL — análise local (**vulnapp-ess.zip + vulnapp_db.zip**)

Foi carregada a base de dados CodeQL (a partir do arquivo vulnapp_db.zip) no workspace do starter kit. Em seguida foram localizadas e executadas as queries correspondentes a CWE-79, CWE-89 e CWE-327.

CWE	Query CodeQL (ID)	O que deteta (resumo)
CWE-79	java/xss	Cross-Site Scripting (XSS) — dados não confiáveis refletidos/armazenados em HTML.
CWE-89	java/sql-injection	SQL Injection — queries construídas a partir de input controlado pelo utilizador.
CWE-327	java/weak-cryptographic-algorithm	Uso de algoritmos criptográficos fracos/quebrados (ex.: DES, MD5, SHA-1, ECB).

3.1 CWE-79 (XSS)

Local no código: pt.isel.vulnerableapp.controller.SearchController.

Source: input do utilizador vindo de HttpServletRequest.getParameter("query").

Sink: escrita direta no HTML gerado via PrintWriter.println sem encoding/escaping.

Exploração: pedido HTTP do tipo /search?query=<script>alert(1)</script> reflete o payload na resposta em ambiente controlado.

3.2 CWE-89 (SQLi)

Local no código: pt.isel.vulnerableapp.controller.UserController.

Source: valores de username vindos de parâmetros HTTP
(ex.:request.getParameter("username")).

Sink: construção de SQL por concatenação e execução por:

Statement.executeQuery()/executeUpdate().

Exploração: inserir caracteres de controlo SQL no username (ex.: ' OR '1'='1) pode permitir ler dados indevidos ou contornar validações, dependendo da query.

3.3 CWE-327 (cripto fraca)

Local no código: pt.isel.vulnerableapp.util.SecurityUtils.

Source: dados sensíveis (p.ex. passwords) passados como String/byte[] para funções utilitárias.

Sink: chamadas a APIs de criptografia com algoritmos fracos, como MessageDigest.getInstance("MD5"), MessageDigest.getInstance("SHA-1"), Cipher.getInstance("DES/ECB/PKCS5Padding").

Exploração: MD5/SHA-1 são vulneráveis a ataques de colisão; DES pode ser atacado com bruteforce; ECB expõe padrões.

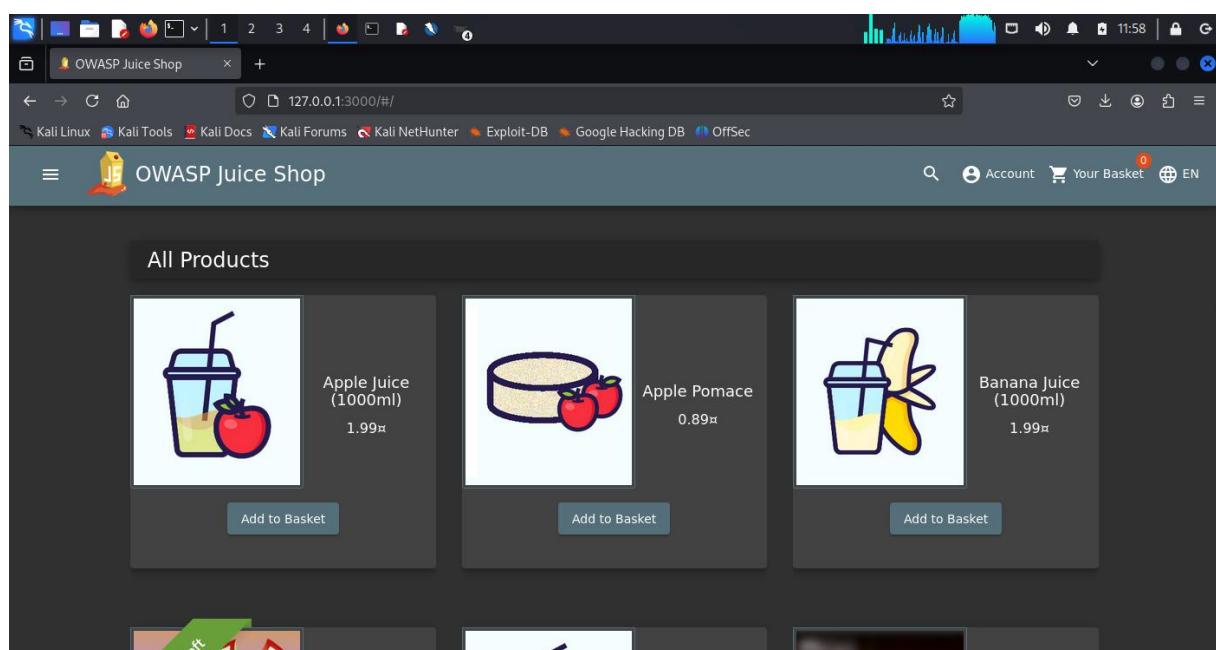
Isto reduz o custo de ataque e pode comprometer confidencialidade/integridade.

4. Análise dinâmica — OWASP Juice Shop

A aplicação foi executada localmente em Docker e analisada com DevTools e OWASP ZAP.

4.1 Execução em Docker

```
docker run -d -p 3000:3000 bkrimminich/juice-shop
```



4.2 Descobrir o Score Board (rota escondida)

Com DevTools (F12) → Sources → abrir main.js e pesquisar por "score". A rota encontrada permite aceder a:

<http://localhost:3000/#/score-board>

The screenshot shows the OWASP Juice Shop application interface. At the top, there's a navigation bar with links like Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. Below the navigation is the OWASP Juice Shop logo and a search bar. The main content area is titled 'All Products' and displays a progress bar for 'Hacking Challenges' at 7% and 'Coding Challenges' at 0%. It also shows a summary of 7/169 challenges solved. At the bottom, a note states '16 challenges are unavailable on Docker due to security concerns or technical incompatibility.'

4.3 SQL Injection no login (admin)

No formulário de login, injetou-se no campo Email um payload que force a query a devolver o primeiro registo (admin) ou que comente a validação.

Email: ' or 1=1--
Password: qualquer

ou

Email: admin@juice-sh.op' --
Password: qualquer

The image consists of two vertically stacked screenshots of the OWASP Juice Shop web application.

Top Screenshot (Login Page):

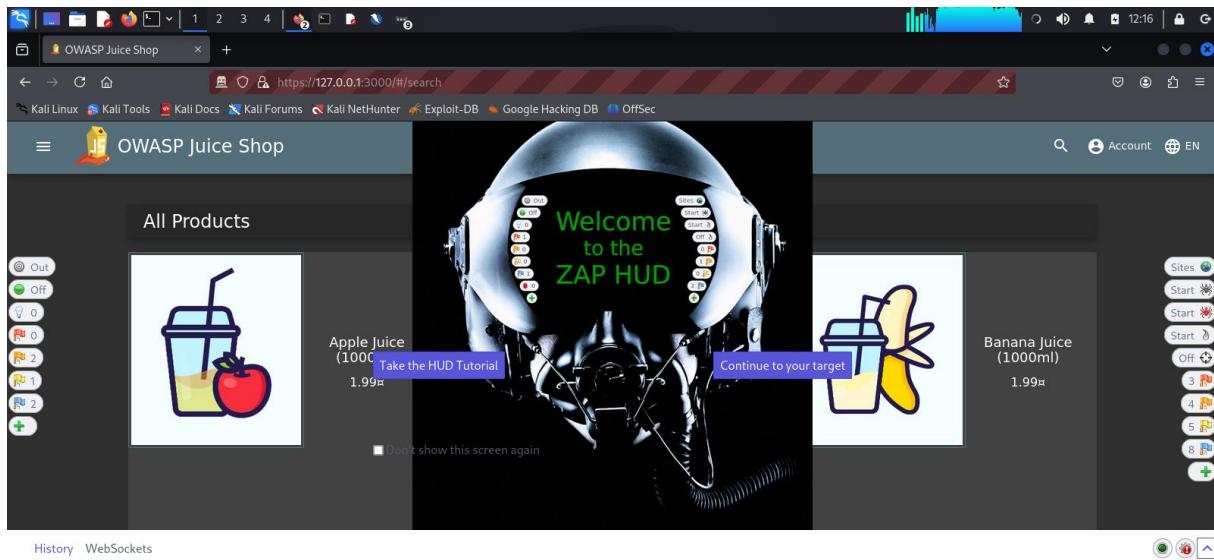
- The browser address bar shows `127.0.0.1:3000/#/login`.
- The page title is "Login".
- The "Email *" field contains the value "' or 1=1--".
- The "Password *" field contains the value "123".
- A "Log in" button is present.
- A "Remember me" checkbox is available.
- A "Forgot your password?" link is visible.
- A "G Log in with Google" button is present.
- A "Not yet a customer?" link is at the bottom.

Bottom Screenshot (Product List Page):

- The browser address bar shows `127.0.0.1:3000/#/search`.
- The page title is "OWASP Juice Shop".
- The main heading is "All Products".
- Three products are listed:
 - Apple Juice (1000ml)** - Price: 1.99€, Add to Basket button.
 - Apple Pomace** - Price: 0.89€, Add to Basket button.
 - Banana Juice (1000ml)** - Price: 1.99€, Add to Basket button.
- A user menu is open on the right, showing:
 - admin@juice-sh.op**
 - Orders & Payment**
 - Privacy & Security**
 - Logout**

4.4 ZAP — Manual Explore

Abrir ZAP → Quick Start → Manual Explore → lança browser (preferência Firefox) e navegar para o Juice Shop através desse browser, o tráfego aparece no History do ZAP.



4.5 Fuzzing da password admin###

Intercetar um pedido de login válido (POST /rest/user/login), enviou-se para o Fuzzer e aplicamos payloads no campo password com lista admin000...admin999. A credencial obtida foi:

```
admin@juice-sh.op / admin123
```

Engenharia de Software Seguro — Laboratório 4

Screenshot of ZAP 2.17.0 showing the Fuzzer module. A context menu is open over a POST request to '/rest/user/login' with the option 'Remove without confirmation?' selected.

Fuzzer Module Details:

- Header:** Text
- Body:** Text
- Request:**

```
POST http://localhost:3000/rest/user/login HTTP/1.1
host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0)
Gecko/20100101 Firefox/140.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json
Content-Length: 47
Origin: http://localhost:3000
Connection: keep-alive
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=g872mOLbrgjJwK7D09pB34o2nvd5btQkqYRtExW6z1PeaBMNyXV5ZM
WrX0
Priority: u=0
```
- Message:** {"email": "admin@juice-hp.op", "password": "123456"}
- Payloads Preview:** Shows a list of payloads including john.txt, abc123, password, computer, 123456, tigger, 1234, alib2c3, qwerty, 123, xxx, money, test, carmen, mickey, internet, summer, service, canada, hello, ranger, shadow, and postgres.
- Payloads:** A table showing payloads for task ID 11 Fuzzed, value 123456, type Body [41, 47].

Screenshot of ZAP 2.17.0 showing the Fuzzer module. A context menu is open over a POST request to '/rest/user/login' with the option 'Remove without confirmation?' selected.

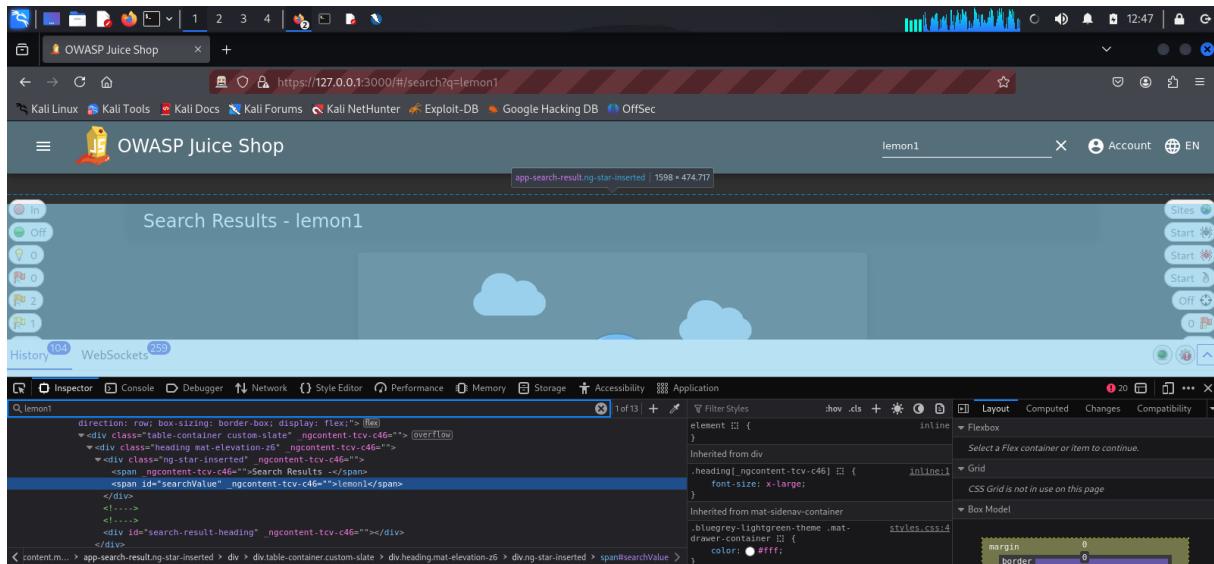
Fuzzer Module Details:

- Header:** Text
- Body:** Text
- Request:**

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Type: application/json
```
- Message:** {"email": "admin@juice-hp.op", "password": "123456"}
- Payloads Preview:** Shows a list of payloads including john.txt, abc123, password, computer, 123456, tigger, 1234, alib2c3, qwerty, 123, xxx, money, test, carmen, mickey, internet, summer, service, canada, hello, ranger, shadow, and postgres.
- Payloads:** A table showing payloads for task ID 8,088 Fuzzed, value admin123, type Body [386, 387].

4.6 Alterar termo de pesquisa na barra de endereço

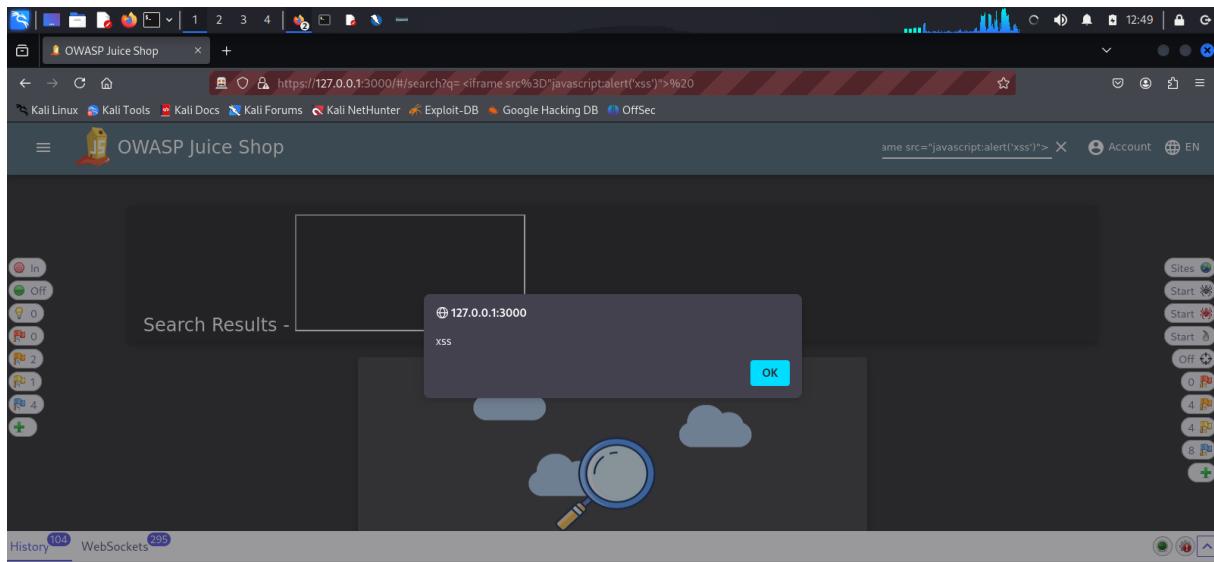
Após pesquisar por Lemon (/search?q=Lemon), alterou-se para Lemon1. O termo é apresentado num elemento com binding [innerHTML], permitindo manipulação via DOM/URL.



4.7 DOM XSS

Inseriu-se no campo de pesquisa um payload do tipo <iframe src="javascript:alert('xss')">.

O XSS ocorre porque o valor de pesquisa é injetado no DOM como HTML (via innerHTML) sem sanitização suficiente.



4.8 Persisted XSS via API

Correu-se o container com NODE_ENV=unsafe.

Exploração:

- Obteve-se um token (Authorization: Bearer)
- Enviou-se POST para /api/Products criando produto com description igual a um payload HTML
- Visitar /search para disparar o XSS.

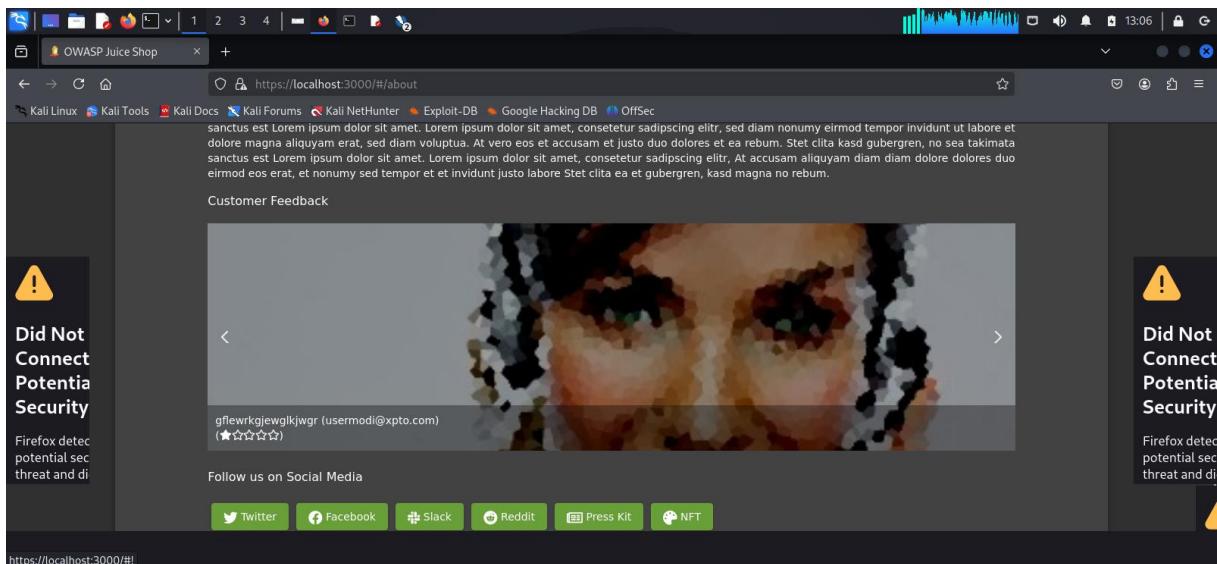
Mesmo com HTML injection, uma CSP estrita pode bloquear execução de scripts inline/event-handlers e URLs javascript, reduzindo impacto e dificultando exfiltração.

4.9 Feedback em nome de outro utilizador

Usando o ZAP como proxy, submeteu-se feedback normal no Contact Us, intercetou-se o pedido e modificou-se no requester o json o ID de outro utilizador antes de reenviar.

The screenshot shows the ZAP interface. The top bar displays 'Untitled Session - ZAP 2.16.0'. The left sidebar shows contexts and sites, with 'Default Context' selected. The main area has two tabs: 'Request' and 'Response'. The 'Request' tab shows a JSON payload being modified. The 'Response' tab shows a long URL with encoded parameters. Below the main area is a table of proxy history entries. The bottom right corner shows a status bar with network activity icons.

ID	Source	Req. Timestamp	Method	URI	Body
302	Proxy	2/14/26, 12:58:19 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8
303	Proxy	2/14/26, 12:58:20 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8
305	Proxy	2/14/26, 12:58:48 PM	POST	/api/Feedbacks	Content-Type: application/json; charset=utf-8
307	Proxy	2/14/26, 12:58:48 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8
308	Proxy	2/14/26, 12:58:48 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8
309	Manual	2/14/26, 1:00:39 PM	POST	/api/Feedbacks	Content-Type: application/json; charset=utf-8
310	Proxy	2/14/26, 1:00:39 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8
311	Proxy	2/14/26, 1:00:39 PM	GET	/api/Feedbacks/10	Content-Type: application/json; charset=utf-8



4.10 Roubo do cookie/token

Um atacante pode convencer a vítima a abrir um link/página dentro do domínio vulnerável que contenha um payload XSS.

Quando o script executa, ele pode ler o token de sessão caso esteja acessível via JavaScript, através da cookie sem HttpOnly ou token em localStorage, e, enviá-lo para um servidor controlado pelo atacante, exfiltração.

Com o token, o atacante pode autenticar-se como a vítima e invocar endpoints protegidos.

As mitigações incluem HttpOnly, SameSite, CSP, sanitização/encoding e validação no servidor.

5. Conclusão

O laboratório permitiu exercitar a análise estática e dinâmica em aplicações web como a deteção de XSS/SQLi/cripto fraca com CodeQL e exploração controlada com DevTools/ZAP.

As atividades reforçam a importância de validação e sanitização de entrada de dados, uso de queries parametrizadas e adoção de criptografia moderna.

Referências

- Enunciado do Laboratório 4 (ESS).
- OWASP Juice Shop — Appendix / Solutions (Score Board, SQLi login, XSS, API-only XSS, feedback spoofing).
- OWASP Juice Shop — Companion Guide (unsafe mode / NODE_ENV=unsafe).
- MDN Web Docs — Content-Security-Policy (CSP) e diretiva script-src.
- CodeQL Query Help (java/xss, java/sql-injection, java/weak-cryptographic-algorithm).