

1 2



9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

Practical Assignment #3

Segurança em Tecnologias da Informação

Mestrado em Engenharia Informática

Dário Félix, 2018275530, dario@student.dei.uc.pt
Maria Dias, 2018274188, mddias@student.dei.uc.pt

Coimbra, 24 de maio de 2023



Figura 1: OWASP Juice Shop

Índice

Índice	I
Lista de Figuras	i
1 Introdução	1
2 Arquitetura	1
2.1 Estrutura de Rede e das VMs	1
2.2 Serviços & Software	1
2.3 Instalações & Configurações	2
2.3.1 Kali Linux	2
2.3.2 OWASP ZAP	2
2.3.3 OWASP Juice Shop	2
2.3.4 Apache	3
2.3.5 ModSecurity	3
3 OWSAP ZAP	4
3.1 Análise automática ao <i>website</i>	4
3.1.1 Páginas exploradas	4
3.1.2 Alertas	5
3.2 Análise ativa à aplicação <i>web</i>	5
3.2.1 Alertas	6
3.3 Gestão dos <i>add-ons</i> necessários para melhorar o teste e maximizar a identificação de ameaças	6
3.4 Ataque Fuzz ao formulário de início de sessão	7
3.5 Análise ativa e explorar a área autenticada	8
3.5.1 Páginas exploradas	9
3.5.2 Alertas	9
4 Teste de Segurança para Aplicações Web	10
4.1 WSTG-INFO Recolha de informações	10
4.1.1 WSTG-INFO-01 Realizar reconhecimento de descoberta de motores de pesquisa para detetar fugas de informações	10
4.1.2 WSTG-INFO-02 Identificação Servidor Web	10
4.1.3 WSTG-INFO-03 Analisar <i>metafiles</i> do servidor Web para detetar fugas de informação	10
4.1.4 WSTG-INFO-04 Enumerar aplicações no servidor Web	11
4.1.5 WSTG-INFO-05 Rever o conteúdo da página Web para detetar fugas de informação	12
4.1.6 WSTG-INFO-06 Identificar pontos de entrada da aplicação	12
4.1.7 WSTG-INFO-07 Mapear caminhos de execução na aplicação	12
4.1.8 WSTG-INFO-08 <i>Fingerprint</i> de <i>Framework</i> de Aplicações Web	12
4.1.9 WSTG-INFO-09 <i>Fingerprint</i> de Aplicações Web	13
4.1.10 WSTG-INFO-10 Mapear a Arquitetura da Aplicação	13
4.2 WSTG-CONF Testes de gestão da implementação e de configuração	13
4.2.1 WSTG-CONF-01 Testar a Configuração da Infraestrutura de Rede	13
4.2.2 WSTG-CONF-02 Testar Configuração da Plataforma da Aplicação	13
4.2.3 WSTG-CONF-03 Testar as extensões de ficheiros que lidam com informação sensível	13
4.2.4 WSTG-CONF-04 Rever <i>backups</i> antigos e arquivos não referenciados para detetar informações sensíveis	14
4.2.5 WSTG-CONF-05 Enumerar interfaces de administrador da infraestrutura e da aplicação	16
4.2.6 WSTG-CONF-06 Testar métodos <i>HTTP</i>	16
4.2.7 WSTG-CONF-07 Testar <i>HTTP Strict Transport Security</i>	16
4.2.8 WSTG-CONF-08 Testar política de domínio cruzado em <i>RIA</i>	16
4.2.9 WSTG-CONF-09 Testar permissões ficheiros	16
4.2.10 WSTG-CONF-10 Testar a tomada de controlo de subdomínios	16
4.2.11 WSTG-CONF-11 Testar armazenamento na <i>cloud</i>	17
4.3 WSTG-IDNT Testes de gestão de identidade	17
4.3.1 WSTG-IDNT-01 Testar definição de <i>roles</i>	17
4.3.2 WSTG-IDNT-02 Testar processo de registo de utilizadores	17

4.3.3	WSTG-IDNT-03 Testar processo de providenciamento de contas	18
4.3.4	WSTG-IDNT-04 Testar para contas de utilizador enumeráveis e previsíveis	19
4.3.5	WSTG-IDNT-05 Testar para política de nomes de utilizador fraca ou não forçada	21
4.4	WSTG-ATHN Testes de autenticação	21
4.4.1	WSTG-ATHN-01 Teste para credenciais transportadas através de um canal encriptado	21
4.4.2	WSTG-ATHN-02 Teste às credenciais predefinidas	21
4.4.3	WSTG-ATHN-03 Teste para detecção de um mecanismo de bloqueio fraco	22
4.4.4	WSTG-ATHN-04 Teste para contornar o esquema de autenticação	22
4.4.5	WSTG-ATHN-05 Teste para vulnerabilidade de lembrar <i>password</i>	23
4.4.6	WSTG-ATHN-06 Teste para detectar fragilidades da <i>cache</i> do navegador	23
4.4.7	WSTG-ATHN-07 Testar a política de palavras-passe fracas	24
4.4.8	WSTG-ATHN-08 Teste a fragilidades de perguntas e respostas de segurança	25
4.4.9	WSTG-ATHN-09 Teste à fragilidade de funcionalidades de alteração ou reposição de palavras-pases	26
4.4.10	WSTG-ATHN-10 Teste a autenticações mais fracas em canais alternativos	26
4.5	WSTG-ATHZ Testes de autorização	26
4.5.1	WSTG-ATHZ-01 Teste de inclusão da diretoria de travessia de ficheiros	26
4.5.2	WSTG-ATHZ-02 Teste para contornar o esquema de autorização	27
4.5.3	WSTG-ATHZ-03 Teste de escalonamento de privilégios	27
4.5.4	WSTG-ATHZ-04 Teste para referências inseguras diretas a objetos	27
4.6	WSTG-SESS Testes de gestão de sessão	27
4.6.1	WSTG-SESS-01 Testes ao esquema de gestão de sessão	27
4.6.2	WSTG-SESS-02 Testes aos atributos de <i>cookies</i>	27
4.6.3	WSTG-SESS-03 Testes para fixação de sessão	28
4.6.4	WSTG-SESS-04 Testes ás variáveis de sessão expostas	28
4.6.5	WSTG-SESS-05 Testes de <i>Cross-Site Request Forgery</i> (CSRF)	29
4.6.6	WSTG-SESS-06 Teste da funcionalidade de <i>logout</i>	29
4.6.7	WSTG-SESS-07 Testes ao tempo limite de sessão	29
4.6.8	WSTG-SESS-08 Testes a <i>Session Puzzling</i>	29
4.6.9	WSTG-SESS-09 Testes de sequestro da sessão	30
4.7	WSTG-INPV Testes de validação de <i>input</i>	30
4.7.1	WSTG-INPV-01 Testar <i>Reflected Cross Site Scripting</i>	30
4.7.2	WSTG-INPV-05 Testar <i>SQL Injection</i>	31
4.8	WSTG-ERRH Testes de tratamento de erros	32
4.8.1	WSTG-ERRH-01 Teste para tratamento inadequado de erros	32
4.8.2	WSTG-ERRH-02 Testes para stack traces	33
4.9	WSTG-CRYP Testes de criptografia fraca	34
4.10	WSTG-BUSL Testes de lógica de negócio	34
4.11	WSTG-CLNT Testes do lado do cliente	34
4.11.1	WSTG-CLNT-04 Testar para Redirecionamento de <i>URL</i> do lado do cliente	34
4.11.2	WSTG-CLNT-09 Testar <i>Clickjacking</i>	35
5	Firewall de Segurança para Aplicações Web (WAF)	35
5.1	OWASP ZAP	35
5.2	WSTG-INFO Recolha de informações	36
5.3	WSTG-CONF Testes de gestão da implementação e de configuração	36
5.4	WSTG-IDNT Testes de gestão de identidade	36
5.5	WSTG-ATHN Testes de autenticação	36
5.6	WSTG-ATHZ Testes de autorização	37
5.7	WSTG-SESS Testes de gestão de sessão	37
5.8	WSTG-INPV Testes de validação de <i>input</i>	37
5.9	WSTG-ERRH Testes de tratamento de erros	40
5.10	WSTG-CRYP Testes de criptografia fraca	41
5.11	WSTG-BUSL Testes de lógica de negócio	41
5.12	WSTG-CLNT Testes do lado do cliente	41
6	Conclusão	41
Referências		42

Lista de Figuras

1	OWASP Juice Shop	0
2	Cenários das fases do trabalho — testes de segurança e WAF	1
3	Execução da análise automática ao <i>website</i>	4
4	Vista em árvore das páginas exploradas	4
5	Contagem dos alertas encontrados durante o teste, divididos em categorias de risco	5
6	Exploração de políticas mais eficazes	5
7	Execução da análise ativa à aplicação <i>web</i>	5
8	Contagem dos alertas encontrados durante o teste, divididos em categorias de risco	6
9	Gestão dos <i>add-ons</i>	7
10	Exemplo de um ataque fuzzer ao formulário de <i>login</i>	7
11	Configuração do <i>browser</i>	8
12	Registo de um utilizador no JuiceShop	8
13	Conjunto de páginas acessíveis após efetuado o início de sessão	9
14	Exceto da vista em árvore das novas páginas exploradas	9
15	Cabeçalho de resposta	10
16	Conteúdo <i>robots.txt</i>	11
17	Conteúdo <i>security.txt</i>	11
18	Portos abertos	11
19	Alerta comentários suspeitos	12
20	Mapeamento da aplicação com <i>ZAP</i>	12
21	<i>Scan</i> para detetar versões	13
22	Acesso a diretoria <i>/ftp</i>	14
23	Conteúdo ficheiro <i>acquisitions.md</i>	14
24	Mensagem de erro	15
25	Conteúdo ficheiros de <i>backup</i>	15
26	Ficheiros identificados pelo <i>ZAP</i>	16
27	Métodos <i>HTTP</i> suportados	16
28	<i>Roles</i> da aplicação	17
29	Registo de utilizadores	18
30	Perfil de utilizador com email inventado	19
31	Login com utilizador inválido	20
32	Login com utilizador válido e palavra passe errada	20
33	<i>Review</i> com email de administrador	20
34	<i>Reviews</i> de utilizadores	21
35	Configuração de um ataque fuzz para encontrar contas predefinidas	22
36	Resultados do ataque fuzz em execução	22
37	<i>Token</i> enviado em <i>plaintext</i> após login	23
38	Utilização do <i>token</i> para aceder a páginas da área autenticada	23
39	Dados guardados no <i>local storage</i> pela aplicação <i>web</i>	23
40	Exemplo de um <i>header</i> de “Cache-Control” de uma página com conteúdo confidencial	24
41	Aceitação por parte da aplicação <i>web</i> de <i>passwords</i> como “12345”, apesar de não cumprir com as boas práticas que a própria aplicação recomenda (mas que não obriga)	24
42	Obrigatoriedade na definição de uma pergunta e resposta de segurança	25
43	Perguntas pré-geradas de natureza simplista que podem levar a respostas inseguras	25
44	Página para alterar a <i>password</i>	26
45	<i>Path Traversal</i>	26
46	Alerta do OWASP ZAP - Cookie No HttpOnly Flag	27
47	Alerta do OWASP ZAP - Cookie without SameSite Attribute	28
48	Alerta do OWASP ZAP - Loosely Scoped Cookie	28
49	Alerta do OWASP ZAP - Session ID in URL Rewrite	29
50	Alerta do OWASP ZAP - Absence of Anti-CSRF Tokens	29
51	Perfil antes do ataque de <i>XSS</i>	30
52	Perfil após possível ataque <i>XSS</i>	31
53	<i>SQL injection</i> na página de <i>login</i>	31
54	Resultado <i>SQL injection</i>	32
55	Ataque de <i>SQL Injection</i> no formulário de <i>login</i>	32
56	Mensagem de erro	33
57	Vulnerabilidade alertado pelo OWASP ZAP	33

58	Stack trace do erro provocado	34
59	Redirecionamento encontrado pelo <i>ZAP</i>	34
60	<i>Clickjacking</i> encontrado pelo <i>ZAP</i>	35
61	Alertas encontrado pelo <i>ZAP</i> após implementação da <i>WAF</i>	36
62	Erro ao tentar ataque <i>Reflected Cross Site Scripting</i>	37
63	Bloqueio do ataque <i>Reflected Cross Site Scripting</i>	37
64	Ataques <i>Reflected Cross Site Scripting</i> detetados pelo <i>ZAP</i>	38
65	<i>UnauthorizedError</i> ao aceder à diretoria	38
66	Diretoria Proibida	38
67	Ataque <i>SQL Injection</i>	39
68	<i>Logs</i> do <i>ModSecurity</i> de um bloqueio do ataque <i>SQL Injection</i>	40
69	Falso positivo no alerta “Information Disclosure - Debug Error Messages” detetado pelo OWASP <i>ZAP</i>	40
70	Proteção concebida pelo <i>WAF</i>	41

1 Introdução

Com o objetivo de compreender como identificar e explorar falhas de segurança numa aplicação *web*, bem como implementar medidas de proteção para mitigar essas vulnerabilidades, nomeadamente mediante um firewall de aplicação *web*, este relatório visa descrever o trabalho efetuado, isto é, uma auditoria de segurança [1].

A aplicação *web* utilizada neste trabalho é a *JuiceShop* [2] da OWASP [3]. Utilizaremos várias ferramentas, como o OWASP ZAP [4], para realizar as análises de vulnerabilidades ao *JuiceShop*. Seguiremos a estrutura da *Web Security Testing Guide* (WSTG) [5].

A tarefa está dividida em duas fases: a primeira fase é dedicada à exploração da segurança da *JuiceShop*, e a segunda fase visa monitorizar, filtrar e bloquear o tráfego HTTP para a *JuiceShop* por meio da implementação de um firewall, o *ModSecurity* [6], para resolver os problemas de segurança identificadas na primeira fase. A Figura 2 ilustra as duas fases do trabalho, representando o servidor *web* *JuiceShop*, o cliente de teste de intrusão e o *web application firewall* (WAF).

Scenario 1: Web security testing



Scenario 2: Web application firewall

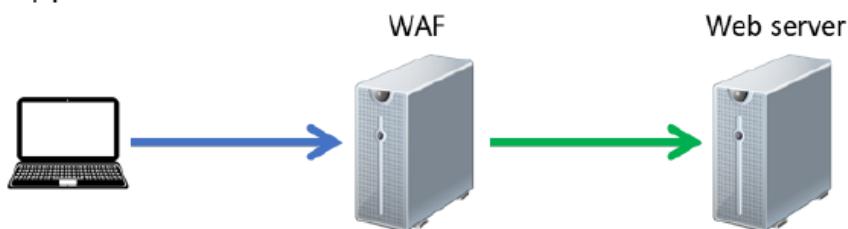


Figura 2: Cenários das fases do trabalho — testes de segurança e WAF

2 Arquitetura

2.1 Estrutura de Rede e das VMs

Segue-se na Tabela 1, a organização das VMs, distribuição de serviços e a atribuição de endereços IPs conforme o cenário da Figura 2.

ID VM	SO	Função	Serviços Instalados	Rede/CIDR	Endereço IP
VM1 Cliente	Kali Linux	Testes de Intrusão	ZAP; Nmap	10.20.30.0/24	10.20.30.1
VM2 Servidor	CentOS	Servidor Web + WAF	Apache; ModSecurity; Npm; JuiceShop		10.20.30.2

Tabela 1: Versões dos *softwares* utilizados.

2.2 Serviços & Software

Todo o trabalho foi realizado utilizando as versões dos *softwares* assinalados na Tabela 2.

Software	Versão
CentOS	CentOS Linux release 7.9.2009
OWASP ZAP	2.12.0
Kali Linux	Kali GNU/Linux Rolling 2023.1
ModSecurity	2.9.2
ModSecurity CRS (Core Rule Set)	2.2.9
Apache	Apache/2.4.6 (CentOS)
Nmap	v.6.40
JuiceShop	v14.5.1
npm (Node Package Manager)	8.19.2

Tabela 2: Versões dos *softwares* utilizados.

2.3 Instalações & Configurações

2.3.1 Kali Linux

Utilizou-se uma máquina virtual pré-construída de Kali Linux [7] por ser mais leve e pronta a ser importada para o VMWare [8].

Uma vez instalado, alterou-se o ficheiro `\etc\hosts`, de forma a mapear o domínio *juice-shop* para o IP da VM com o servidor web:

```
$ sudo nano \etc\hosts
```

```
(...)
10.20.30.2      juice-shop
(...)
```

2.3.2 OWASP ZAP

Instalou-se através do script transferido a partir da página oficial do Zed Attack Proxy (ZAP) [4].

```
$ sudo ./ZAP_2_12_0_unix.sh
```

2.3.3 OWASP Juice Shop

Instalação do npm e do git:

```
$ yum install nodejs epel-release npm git
```

Correção do erro “*ImportError : /lib64/libstdc+++.so.6 : version‘CXXABI_1.3.8’not found*” [9]:

```
# Follow below instructions to compile new version of gcc:

# Install dependencies
$ sudo yum check-update
$ sudo yum -y install wget make gcc-c++

# Download gcc new version
$ wget -O - 'https://ftpmirror.gnu.org/gcc/gcc-7.3.0/gcc-7.3.0.tar.xz' | tar -xJ

# This helps if you are behind a proxy
$ sed -i 's/ftp:/https:/' ./gcc-7.3.0/contrib/download_prerequisites

# Finally compile gcc
$ ( cd gcc-7.3.0 && ./contrib/download_prerequisites && mkdir build && cd build &&
  ./configure --enable-checking=release --enable-languages=c,c++
  --disable-multilib && make -j 8 && sudo make install ) && rm -fR gcc-7.3.0

# Unlink current version
```

```
$ sudo unlink /usr/lib64/libstdc++.so.6  
  
# Copy new version  
$ sudo cp /usr/local/lib64/libstdc++.so.6 /usr/lib64
```

Com o erro corrigido, fez-se o *download* do código fonte do *JuiceShop* disponível no GitHub [10]:

```
$ git clone https://github.com/juice-shop/juice-shop.git --depth 1
```

Aceda à pasta com:

```
$ cd juice-shop
```

Execute os seguintes comandos:

```
# Basta executar uma vez  
$ npm install  
  
$ npm start
```

A página está acessível em <http://localhost:3000>.

2.3.4 Apache

Instalação do Apache:

```
$ yum install httpd mod_ssl
```

Configuração do Apache e do Proxy [11]:

```
$ nano /etc/httpd/conf/httpd.conf
```

```
(...)  
<VirtualHost *:80>  
    SecRuleEngine On  
    ProxyPreserveHost On  
    ProxyPass / http://localhost:3000/  
    ProxyPassReverse / http://localhost:3000/  
</VirtualHost>  
(...)
```

Correção do erro “failed to make connection to backend” [12]:

```
$ sudo /usr/sbin/setsebool -P httpd_can_network_connect 1
```

Reiniciar o serviço do Apache:

```
$ systemctl restart httpd
```

2.3.5 ModSecurity

Instalação do *ModSecurity* [11], incluindo o *ModSecurity Core Rule Set* (CRS) [6]:

```
$ yum install httpd mod_security mod_security_crs
```

Verificar a instalação:

```
# Check loaded modules (Other modules may be installed, nonetheless for the  
→ purpose of this document the required one is security2_module)  
$ httpd -M
```

Reiniciar o serviço do Apache:

```
$ systemctl restart httpd
```

Testar:

```
# Using curl, now for an SQLi attack
$ curl -d \id=1 AND 1=1" http://localhost/index.php
```

3 OWSAP ZAP

3.1 Análise automática ao *website*

O OWASP ZAP, *Zed Attack Proxy*, desenvolvido pela OWASP, é uma ferramenta de teste de segurança utilizada para aplicações web, isto é, ajuda a identificar e corrigir vulnerabilidades em aplicações, mediante recursos automatizados e manuais [13].

A análise automática utiliza o Spider para mapear a aplicação web.

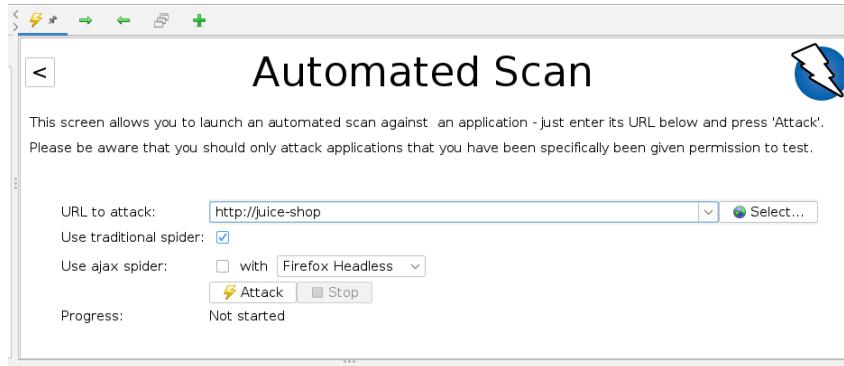


Figura 3: Execução da análise automática ao *website*

3.1.1 Páginas exploradas

O ZAP ao analizar a aplicação web, constrói um mapa das páginas da aplicação Web e dos recursos utilizados para renderizar essas páginas [13].

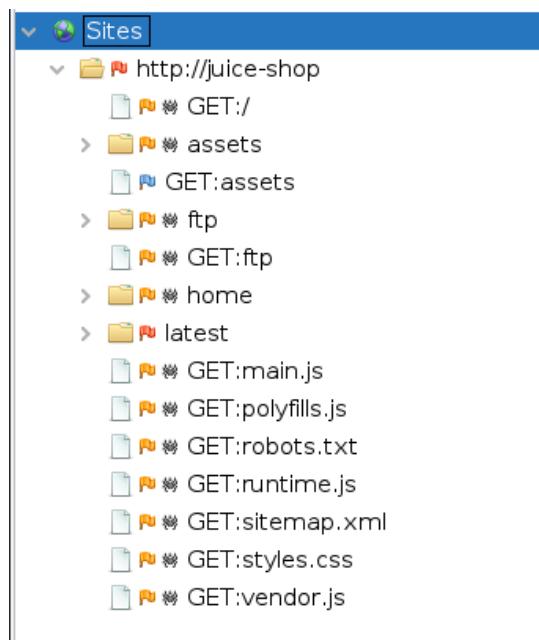


Figura 4: Vista em árvore das páginas exploradas

3.1.2 Alertas

Em seguida, regista os pedidos e respostas enviados para cada página e cria alertas se houver algo potencialmente errado com um pedido ou resposta [13].

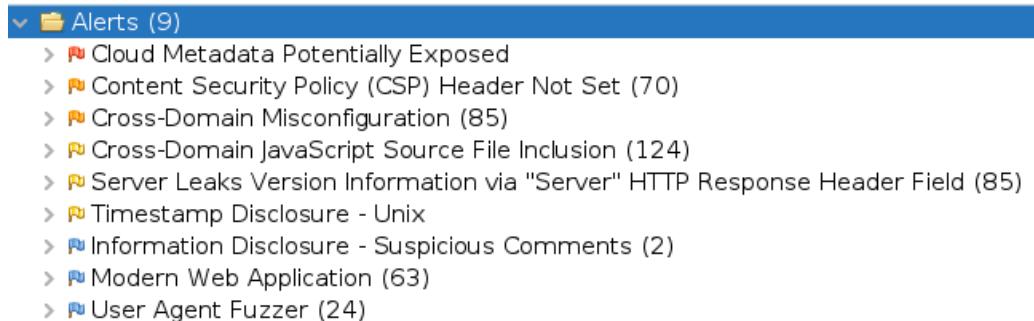


Figura 5: Contagem dos alertas encontrados durante o teste, divididos em categorias de risco

3.2 Análise ativa à aplicação web

A análise ativa tenta encontrar potenciais vulnerabilidades utilizando ataques conhecidos contra o alvo selecionado [14].

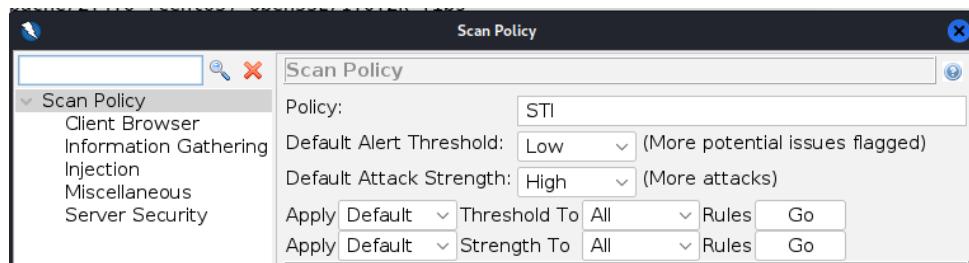


Figura 6: Exploração de políticas mais eficazes

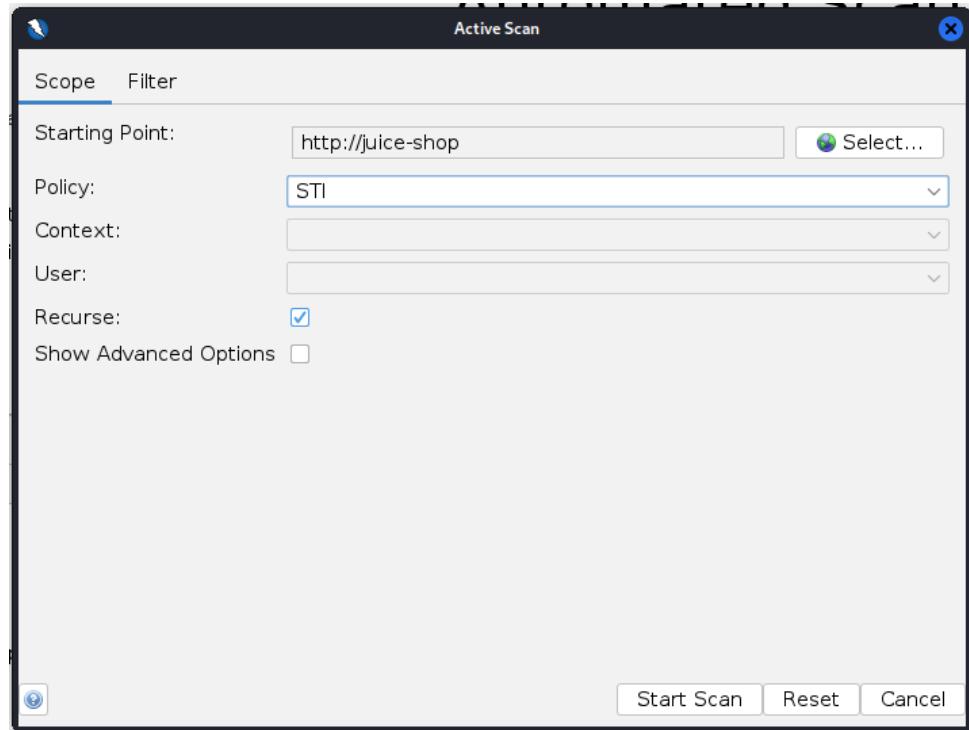


Figura 7: Execução da análise ativa à aplicação web

3.2.1 Alertas

Utilizando uma política com uma força de ataque alta, foram acrescentados novos alertas face a última análise, ver Figura 8.



Figura 8: Contagem dos alertas encontrados durante o teste, divididos em categorias de risco

3.3 Gestão dos *add-ons* necessários para melhorar o teste e maximizar a identificação de ameaças

Foram instalados os seguintes *add-ons*, ver Figura 9:

- Advanced SQLInjection Scanner - An advanced active injection bundle for SQLi (derived by SQLi-Map);
- FuzzDB Files - FuzzDB files which can be used with the ZAP fuzzer;

- FuzzDB Offensive - FuzzDB web backdoors and attack files which can be used with the ZAP fuzzer or for manual penetration testing;
- Wappalyzer Technology Detection - Technology detection using Wappalyzer: wappalyzer.com.

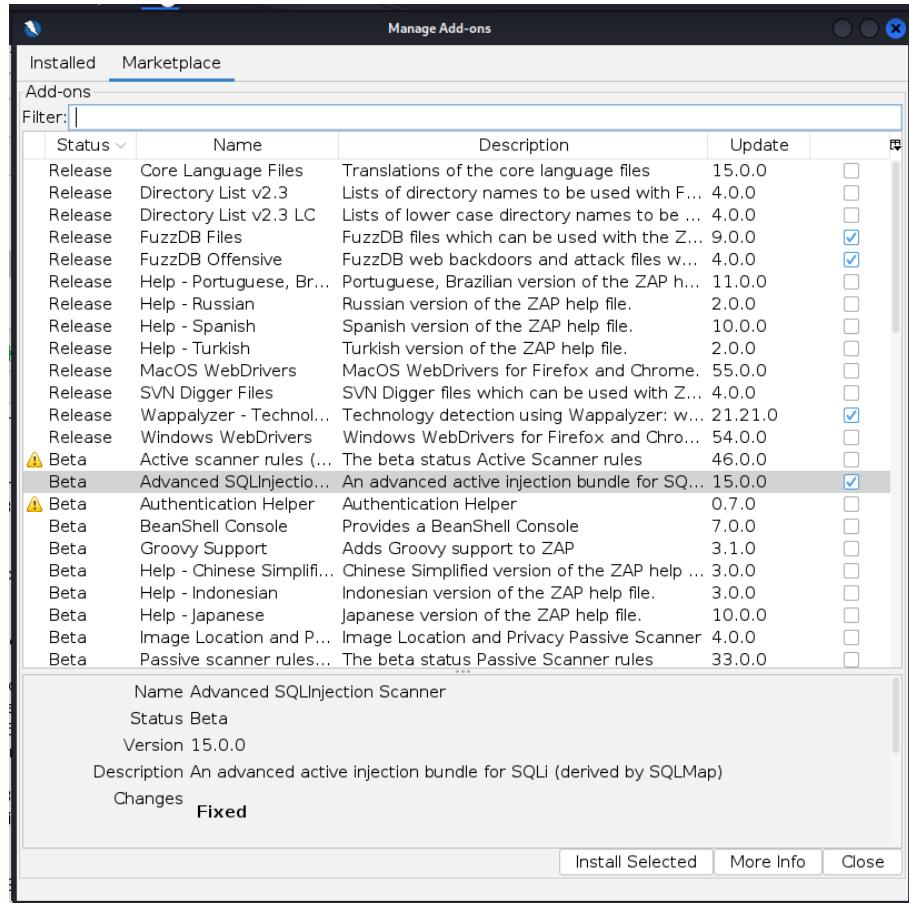


Figura 9: Gestão dos *add-ons*

3.4 Ataque Fuzz ao formulário de início de sessão

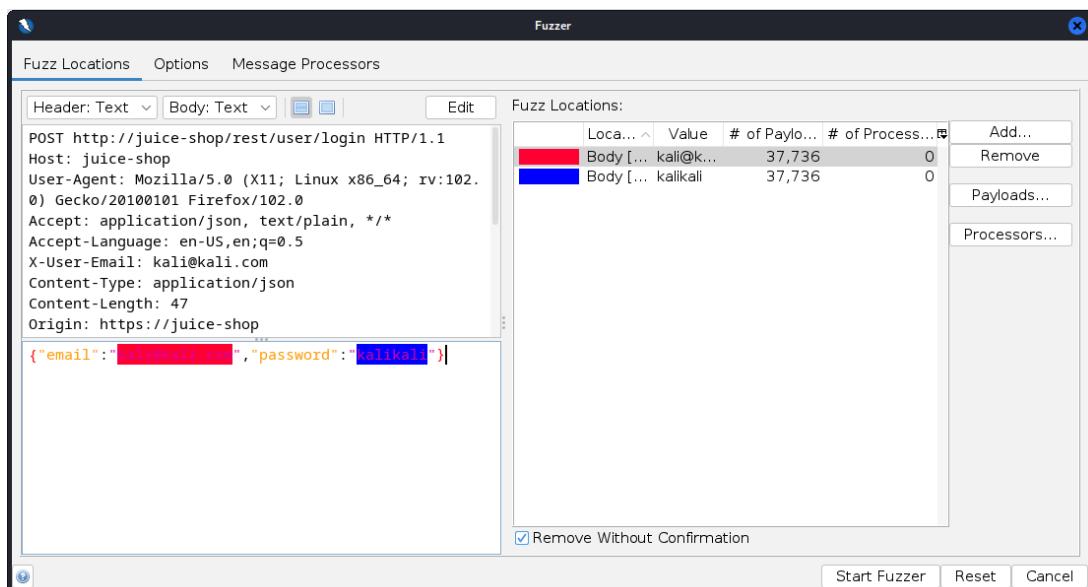


Figura 10: Exemplo de um ataque fuzzer ao formulário de *login*

3.5 Análise ativa e explorar a área autenticada

Configurou-se o navegador para utilizar o proxy do ZAP, ver Figura 11.

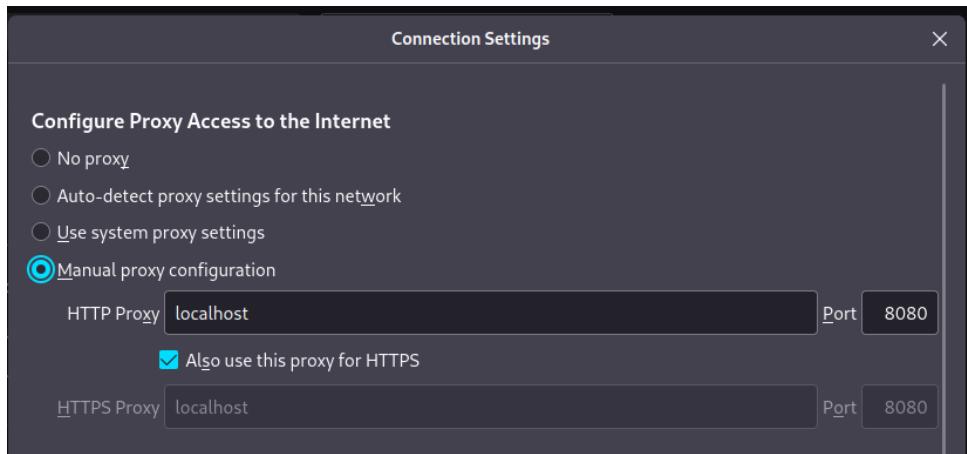


Figura 11: Configuração do *browser*

Criou-se um utilizador conforme a Figura 12, fez-se o *login*, e de seguida explorou-se as páginas da área autenticada, ver Figura 13.

A screenshot of a 'User Registration' form. The title is 'User Registration'. The form fields include: 'Email *' with the value 'kali@kali.com'; 'Password *' with a masked input showing six dots; a note below says 'Password must be 5-40 characters long.' and shows '8/20'; 'Repeat Password *' with a masked input showing six dots; a note below says '8/40'; a 'Show password advice' toggle switch is turned on; 'Security Question *' with a dropdown menu showing 'Your eldest sibling's middle name?'; a note below says 'This cannot be changed later!'; 'Answer *' with the value 'kali'; and a large blue 'Register' button at the bottom with a user icon.

Figura 12: Registo de um utilizador no JuiceShop

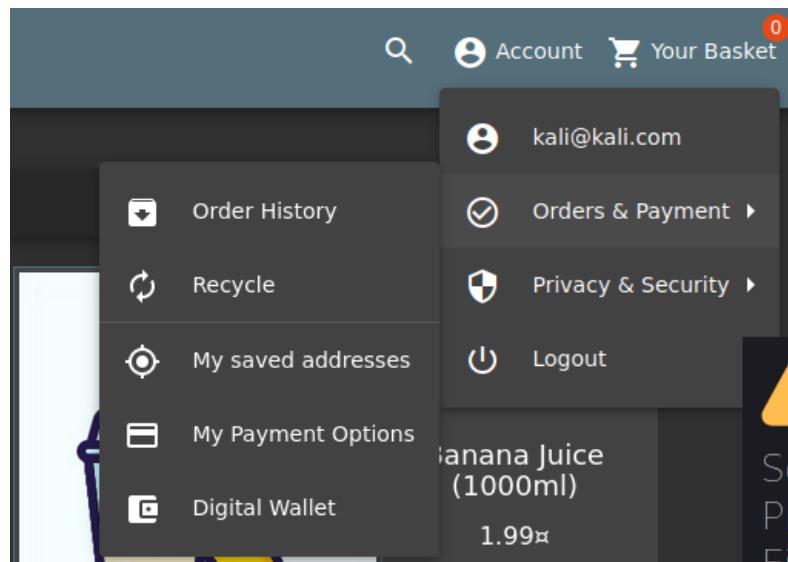


Figura 13: Conjunto de páginas acessíveis após efetuado o início de sessão

3.5.1 Páginas exploradas

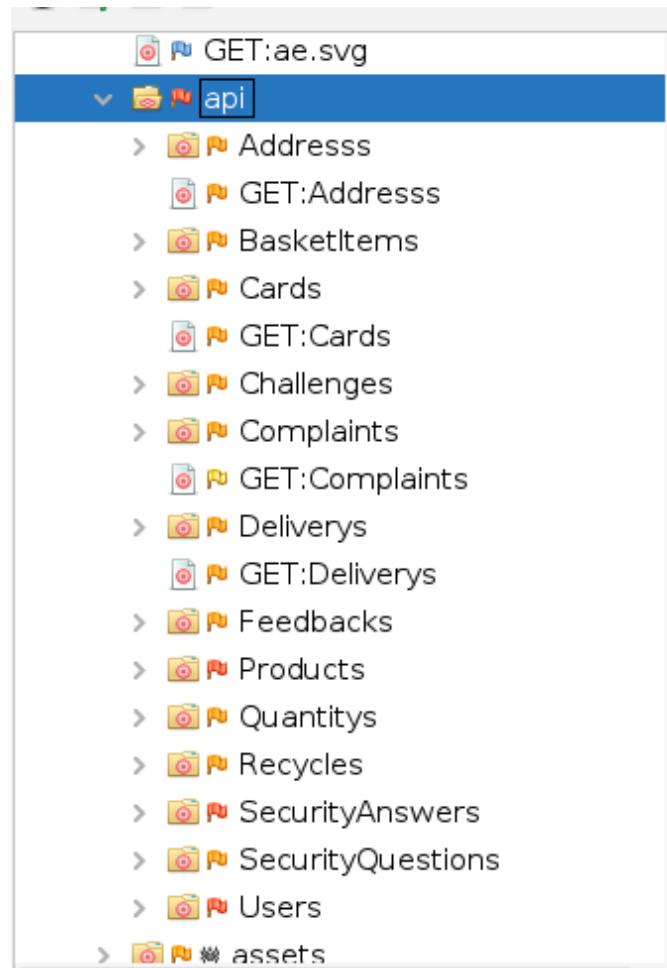


Figura 14: Excerto da vista em árvore das novas páginas exploradas

3.5.2 Alertas

Ver Subseção 3.2, uma vez que os alertas aí encontrados também incidem sobre a área autenticada.

4 Teste de Segurança para Aplicações Web

4.1 WSTG-INFO Recolha de informações

4.1.1 WSTG-INFO-01 Realizar reconhecimento de descoberta de motores de pesquisa para detetar fugas de informações

Os motores de busca recolhem regularmente informações de sites presentes na internet. Estas informações podem ser usadas para fazer reconhecimento de sites, uma vez que podem conter informações sensíveis de design ou configuração.

No âmbito do presente trabalho, **não é possível aplicar este ponto** uma vez que o site a analizar não está presente na internet com um ip público.

4.1.2 WSTG-INFO-02 Identificação Servidor Web

Através da identificação de informações (*fingerprint*) sobre o servidor web no qual um site está hospedado (nomeadamente tipo de servidor e versão de software), é possível encontrar vulnerabilidades e falhas que podem ser usados como pontos de ataque.

Uma das formas usadas para fazer *fingerprinting* de um servidor web é através de *banner grabbing*. Para o efeito, é efetuado um pedido HTTP ao servidor e é analisado o cabeçalho da resposta.

Neste caso foi usado o *curl* [15] com a flag *-I* para ser imprimida apenas a informação referente ao cabeçalho, como é possível verificar na Figura 15. Através da obtenção do cabeçalho ficamos a saber que o servidor usado é o *Apache versão 2.4.6* no sistema operativo CentOS (tal como descrito na secção anterior).

```
(kali㉿kali)-[~]
└─$ curl -s -I juice-shop
HTTP/1.1 200 OK
Date: Mon, 22 May 2023 15:30:52 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Mon, 22 May 2023 14:09:45 GMT
ETag: W/"7c3-18843cb6c44"
Content-Type: text/html; charset=UTF-8
Content-Length: 1987
Vary: Accept-Encoding
```

Figura 15: Cabeçalho de resposta

4.1.3 WSTG-INFO-03 Analisar *metafiles* do servidor Web para detetar fugas de informação

Através da análise de ficheiros *metadata* poderá ser possível encontrar caminhos ou funcionalidades escondidas. Pode também ser possível obter outras informações que permitam outro tipo de ataques, como é o caso de engenharia social.

O ficheiro *robots.txt* é incluído em páginas web e contém informação sobre que páginas do site não devem ser visitadas por *Web Spiders*, *Robots*, ou *Crawlers*.

Através do seguinte comando é possível obter as páginas às quais os motores de busca não conseguem aceder. Neste caso, como se pode verificar na Figura 16, os motores de busca não conseguem aceder à diretoria */ftp*. É de realçar que esta diretoria pode ser acedida de forma manual, alterando para o efeito a hiperligação no browser, pelo que as diretórias incluídas no ficheiro não se encontram livres de acessos.

```
$ curl -O -Ss juice-shop/robots.txt && head -n5 robots.txt
```

```
(kali㉿kali)-[~]
└─$ curl -O -sS juice-shop/robots.txt && head -n5 robots.txt
User-agent: *
Disallow: /ftp
```

Figura 16: Conteúdo *robots.txt*

Security.txt é um *standard* que permite definir políticas de segurança e detalhes de contacto caso seja encontradas falhas de segurança. Através da obtenção das informações deste ficheiro poderá também ser possível efetuar engenharia social e identificar algumas diretorias ou recursos da aplicação web que podem ser explorados. Como se pode verificar na Figura 17 através do uso do comando apresentado de seguida, é possível encontrar algumas informações que poderão ser úteis para efetuar engenharia social.

```
$ wget --no-verbose juice-shop/security.txt && cat security.txt
```

```
(kali㉿kali)-[~]
└─$ wget --no-verbose juice-shop/security.txt && cat security.txt
2023-05-22 12:03:59 URL:http://juice-shop/security.txt [403/403] → "security.txt" [1]
Contact: mailto:donotreply@owasp-juice.shop
Encryption: https://keybase.io/bkimminich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbdcda
Acknowledgements: #/score-board
Preferred-languages: en, ar, az, bg, ca, cs, da, de, ga, el, es, et, fi, fr, ka, he, hi, hu, id, it, ja, ko, lv, my, nl, no, pl, pt, ro, ru, si, sv, th, tr, uk, zh
Hiring: #/jobs
Expires: Wed, 22 May 2024 14:09:43 GMT
```

Figura 17: Conteúdo *security.txt*

4.1.4 WSTG-INFO-04 Enumerar aplicações no servidor Web

Através da informação de que aplicações estão instaladas num servidor web é possível explorar vulnerabilidades conhecidas dessas aplicações para obter acesso ao servidor e respetivos dados. Aplicações desatualizadas ou mal configuradas podem também constituir uma ameaça ao servidor.

Através do uso do nmap é possível identificar todos os portos abertos no *juice-shop* e os serviços a eles associados. Para o efeito podemos usar o comando apresentado de seguida e verificar na Figura 18 quais os portos abertos e os serviços associados.

```
$ nmap -Pn -sT -sV -p0-65535 juice-shop
```

```
(kali㉿kali)-[~]
└─$ nmap -Pn -sT -sV -p0-65535 juice-shop
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 13:04 EDT
Stats: 0:00:15 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 60.00% done; ETC: 13:04 (0:00:07 remaining)
Nmap scan report for juice-shop (192.168.38.242)
Host is up (0.00074s latency).
Not shown: 65531 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.4 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips)
111/tcp   open  rpcbind 2-4 (RPC #100000)
443/tcp   open  ssl/http Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k-fips)
3000/tcp  open  ppp?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port3000-TCP:V=7.93%I=7%D=5/22%Time=646BA0AB%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,979,"HTTP/1.\.1\x20200\x200K\r\nAccess-Control-Allow-Origin:\x2
SF:0\*\r\nX-Content-Type-Options:\x20nosniff\r\nX-Frame-Options:\x20SAMEOR
SF:IGIN\r\nFeature-Policy:\x20payment\x20'self'\r\nX-Recruiting:\x20/#/job
SF:s\r\nAccept-Ranges:\x20bytes\r\nCache-Control:\x20public,\x20max-age=0\
SF:r\nLast-Modified:\x20Mon,\x2022\x20May\x202023\x2014:09:45\x20GMT\r\nET
```

Figura 18: Portos abertos

4.1.5 WSTG-INFO-05 Rever o conteúdo da página Web para detetar fugas de informação

A inclusão de comentário e *metadata* no código é muito comum, contudo é necessário uma revisão do código de modo a garantir que não existem dados importantes nos comentários, como é o caso de credenciais, rotas sensíveis, chaves API privadas e endereços IP privados. Caso estas informações estejam presentes no código podem ser usadas como ponto de partida para efetuar ataques.

Através de uma análise efetuada com o *ZAP* foi possível encontrar alguns casos em que o código contém comentários que podem ajudar o atacante.

The screenshot shows the ZAP interface with a warning for 'Information Disclosure - Suspicious Comments' in a file named 'tutorial.js'. The URL is 'http://192.168.38.245/tutorial.js'. The warning details include:

- Risco: Informational
- Confiança: Low
- Parâmetro:
- Ataque:
- Evidência: query
- CWE ID: 200
- WASC ID: 13
- Fonte: Passivo (10027 - Information Disclosure - Suspicious Comments)
- Input Vector:
- Descrição: The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.
- Outra info: The following pattern was used: `lQUERYb` and was detected in the element starting with: `"use strict";(self.webpackChunkfrontend=self.webpackChunkfrontend||[]).push([{"id":648},{3907:{ie,j,S}=>{S.r(j),S.d(j,{hasInstruction", see evidence field for the suspicious comment/snippet.`

Figura 19: Alerta comentários suspeitos

4.1.6 WSTG-INFO-06 Identificar pontos de entrada da aplicação

Para que possamos identificar possíveis áreas de vulnerabilidade é importante identificar a sua superfície de ataque. Deste modo, devemos identificar pontos de entrada na aplicação e de *injection*, recorrendo para o efeito à análise da resposta a pedidos efetuados à aplicação.

4.1.7 WSTG-INFO-07 Mapear caminhos de execução na aplicação

Para testar uma aplicação de forma completa é necessário ter uma noção da estrutura da aplicação. Para o efeito é necessário fazer um mapeamento do site e identificar os *workflows* principais.

Recorrendo ao *ZAP*, é possível fazer o mapeamento de todo o site, como se pode observar na Figura 20.

The screenshot shows the ZAP interface with a 'Spider' tab active, displaying a list of processed URLs. The table includes columns for 'Processed', 'Método', 'URI', and 'Status'. The 'Processed' column shows a vertical list of green circular icons. The 'URI' column lists various URLs starting with 'http://192.168.38.245/'. The 'Status' column indicates all entries as 'Seed'.

Processed	Método	URI	Status
●	GET	http://192.168.38.245	Seed
●	GET	http://192.168.38.245/robots.txt	Seed
●	GET	http://192.168.38.245/sitemap.xml	Seed
●	GET	http://192.168.38.245/	Seed
●	GET	http://192.168.38.245/?zapHudReplaceReq=bce9a18e-8654-408e-8d5c-d30551d4404a	Seed
●	GET	http://192.168.38.245/103.js	Seed
●	GET	http://192.168.38.245/api	Seed
●	GET	http://192.168.38.245/api/Address	Seed
●	GET	http://192.168.38.245/api/Address/	Seed
●	GET	http://192.168.38.245/api/Address/7	Seed
●	GET	http://192.168.38.245/api/BasketItems	Seed
●	GET	http://192.168.38.245/api/BasketItems/	Seed
●	GET	http://192.168.38.245/api/BasketItems/9	Seed
●	GET	http://192.168.38.245/api/Cards	Seed
●	GET	http://192.168.38.245/api/Cards/	Seed
●	GET	http://192.168.38.245/api/Cards/7	Seed
●	GET	http://192.168.38.245/api/Challenges	Seed
●	GET	http://192.168.38.245/api/Challenges/?name=Score%20Board	Seed
●	GET	http://192.168.38.245/api/Challenges/?sort=name	Seed
●	GET	http://192.168.38.245/api/Deliverys	Seed
●	GET	http://192.168.38.245/ftp	Seed
●	GET	http://192.168.38.245/api/Deliverys/1	Seed
●	GET	http://192.168.38.245/api/Feedbacks	Seed
●	GET	http://192.168.38.245/api/Feedbacks/	Seed

Figura 20: Mapeamento da aplicação com *ZAP*

4.1.8 WSTG-INFO-08 Fingerprint de Framework de Aplicações Web

Atualmente são usadas diversas aplicações ou *frameworks* para a construção de sites ou de componentes de sites. Estas aplicações e *frameworks* por serem *OpenSource* têm uma estrutura conhecida, pelo

que se torna mais fácil a sua identificação.

Nos pontos WSTG-INFO-03 *Analisar metafiles do servidor Web para detetar fugas de informação* e WSTG-INFO-04 *Enumerar aplicações no servidor Web* já tínhamos descoberto algumas diretórias escondidas e serviços associadas à aplicação web a ser explorada (ver Figura 16, Figura 17 e Figura 18). No teste seguinte, recorrendo ao *whatweb* podemos identificar outros aspectos relevantes sobre a aplicação web, tais como a versão do *HTML*, o uso de *JQuery* e a sua versão e informações sobre os headers *HTTP* usados.

```
(kali㉿kali)-[~]
$ whatweb -a 4 juice-shop
http://juice-shop/ [200 OK] Apache[2.4.6], Country[RESERVED][ZZ], HTML5, HTTPServer[CentOS][Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips], IP[192.168.38.244], JQuery[2.2.4], Matomo, OpenSSL[1.0.2k-fips], Script[module], Title[OWASP Juice Shop], UncommonHeaders[access-control-allow-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Options[SAMEORIGIN]
```

Figura 21: Scan para detetar versões

4.1.9 WSTG-INFO-09 *Fingerprint* de Aplicações Web

O conteúdo deste ponto é atualmente tratado junto com o ponto WSTG-INFO-08 *Fingerprint* de *Framework* de Aplicações Web, abordado anteriormente.

4.1.10 WSTG-INFO-10 Mapear a Arquitetura da Aplicação

É necessário mapear a arquitetura e a rede da aplicação para poder identificar problemas de segurança e configuração que possam comprometer a segurança da aplicação.

No ponto WSTG-INFO-07 *Mapear caminhos de execução na aplicação* já foi feito o mapeamento da aplicação. De igual forma ao longo desta secções foram também identificados diversos problemas que podem comprometer a segurança da aplicação, como é o caso da identificação de versões de serviços usados pela aplicação e da identificação de diretórias e ficheiros com os quais o atacante pode obter informações importantes.

4.2 WSTG-CONF Testes de gestão da implementação e de configuração

4.2.1 WSTG-CONF-01 Testar a Configuração da Infraestrutura de Rede

Uma pequena falha de segurança numa aplicação pode comprometer toda a infraestrutura de um servidor web, pelo que é extremamente importante rever a configuração da aplicação e falhas de segurança conhecidas. Caso algum dos componentes da aplicação não for devidamente analisado e protegido a pode comprometer a aplicação.

Deste modo, é importante rever as configurações da aplicação e validar que não são vulneráveis e validar que as *frameworks* e sistemas usados não são suscetíveis a vulnerabilidades conhecidas (como credenciais e configurações default).

4.2.2 WSTG-CONF-02 Testar Configuração da Plataforma da Aplicação

A configuração e teste da configuração é extremamente importante na manutenção e criação de uma arquitetura uma vez que alguns sistemas podem conter configurações genéricas que podem não ser adequadas à aplicação na qual são instaladas. Deste modo, todas as funcionalidades instaladas que não sejam essenciais para a aplicação devem ser removidas.

4.2.3 WSTG-CONF-03 Testar as extensões de ficheiros que lidam com informação sensível

As extensões de ficheiros são bastante úteis para determinar as tecnologias, linguagens de programação e *plugins* que devem ser usados para responder a determinado pedido. Contudo, o uso da extensão de ficheiro *default* ajuda o atacante a obter informação sobre as tecnologias em uso. As extensões são também usadas para validar os ficheiros a ser carregados, o que pode constituir um problema caso o ficheiro não seja tratado da forma esperada ou caso contenha outro tipo de conteúdo que não o expectável.

Neste teste pretende-se procurar por ficheiros com extensões de ficheiros sensíveis e por extensões de ficheiros que possam conter dados em bruto.

Acedendo à diretoria */ftp* encontrada anteriormente, podemos observar na Figura 22 que conseguimos aceder a bastantes ficheiros que podem conter informação confidencial. Abrindo o ficheiro *acquisitions.md*, podemos ver na Figura 23 que este ficheiro contém conteúdo confidencial.

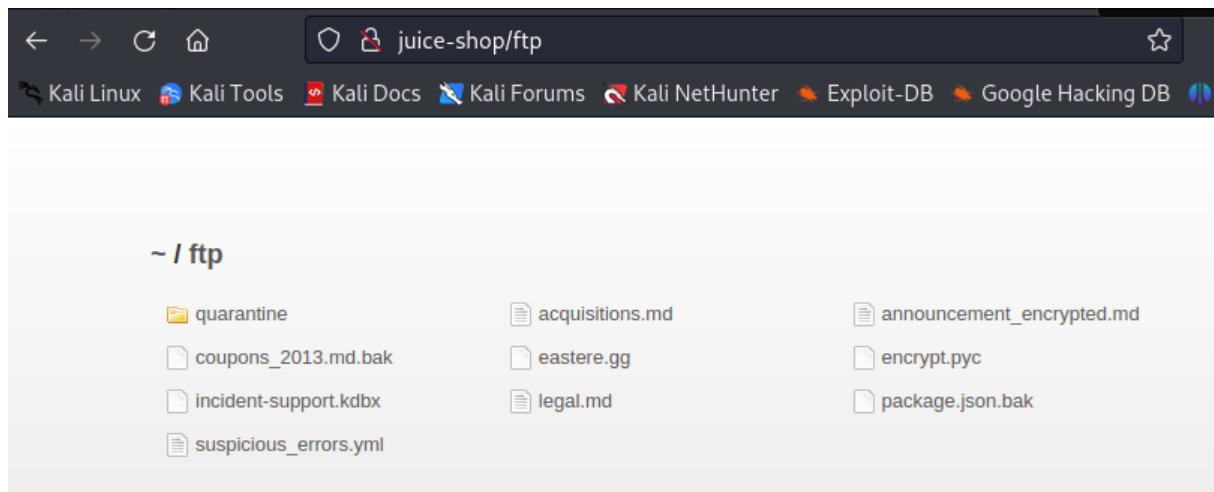


Figura 22: Acesso a diretoria */ftp*

A screenshot of a "Mousepad" text editor window. The title bar says "~/Downloads/acquisitions.md - Mousepad". The menu bar includes File, Edit, Search, View, Document, and Help. The toolbar below has icons for new, open, save, cut, copy, paste, find, and others. The main text area contains the following content:

```
1 # Planned Acquisitions
2
3 > This document is confidential! Do not distribute!
4
5 Our company plans to acquire several competitors within the next year.
6 This will have a significant stock market impact as we will elaborate in
7 detail in the following paragraph:
8
9 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
10 eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
11 voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
12 clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
13 amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
14 nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
15 sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
16 rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
17 ipsum dolor sit amet.
18
19 Our shareholders will be excited. It's true. No fake news.
20
```

Figura 23: Conteúdo ficheiro *acquisitions.md*

4.2.4 WSTG-CONF-04 Rever *backups* antigos e arquivos não referenciados para detetar informações sensíveis

É normal encontrar ficheiros esquecidos ou não referenciados que tenham sido esquecidos. Estes ficheiros podem conter informações importantes uma vez que podem dizer respeito a *backups* feitos de forma automática, ficheiros esquecidos ou até mesmo ficheiros antigos que se encontram na estrutura do projeto.

Como podemos observar na Figura 22, na diretoria */ftp* podemos encontrar dois ficheiros com a extensão *.bak* o que pode indicar que são ficheiros de *backup*. Ao tentar descarregar estes ficheiros,

aparece uma mensagem de erro, apresentada na Figura 24. Esta mensagem de erro pode facilmente ser contornada ao adicionar `%2500.md` ao url [16]

Na Figura 25 podemos analisar o conteúdo dos ficheiros. Podemos verificar que o ficheiro `coupons-2013.md` contém diversos cupões de desconto.

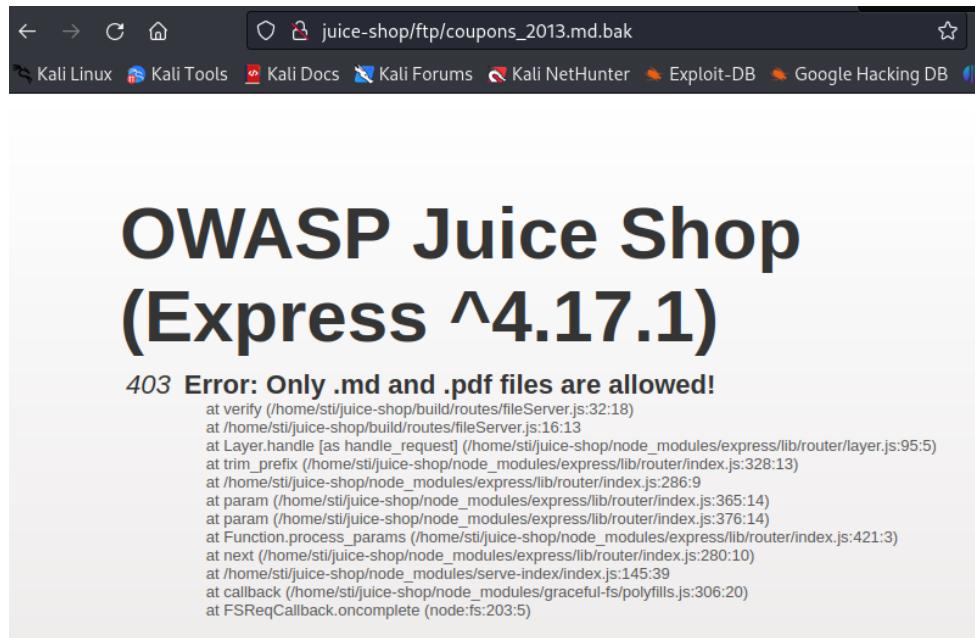


Figura 24: Mensagem de erro

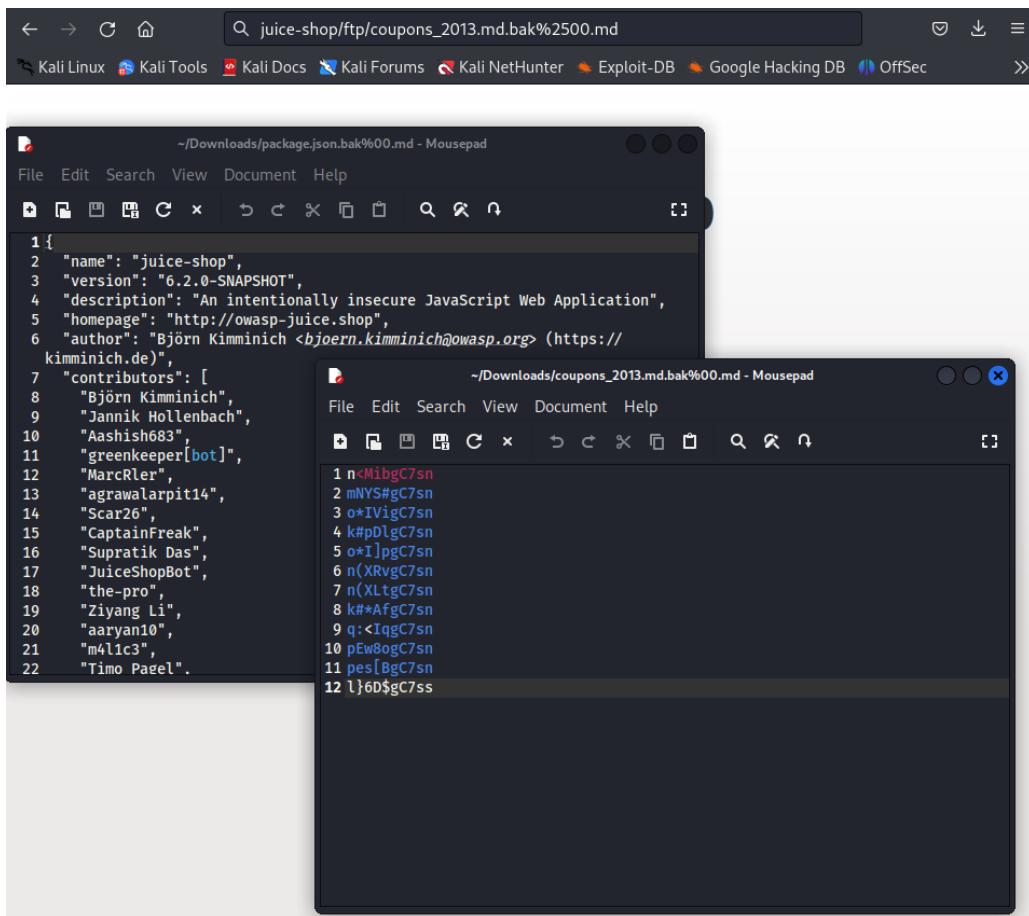


Figura 25: Conteúdo ficheiros de *backup*

4.2.5 WSTG-CONF-05 Enumerar interfaces de administrador da infraestrutura e da aplicação

Neste teste pretende-se identificar interfaces e funcionalidades de administrador escondidas de modo a verificar se um utilizador não autorizado consegue aceder às funcionalidades de administrador e de que forma.

Analizando o relatório do *ZAP* presente na Figura 26, podemos verificar que foram encontradas vulnerabilidades em alguns ficheiros que podem conter informações administrativas, credenciais ou configurações que podem ser usadas para atacar o sistema ou realizar ataques de engenharia social.

The screenshot shows the ZAP interface with the 'Alertas' tab selected. A specific alert is highlighted under the 'Hidden File Found' category. The details pane shows:

- URL: http://192.168.38.245/BitKeeper
- Risco: Medium
- Confiança: Low
- Parâmetro:
- Ataque:
- Evidência: HTTP/1.1 200 OK
- CWE ID: 538
- WASC ID: 13
- Fonte: Ativo (40035 - Hidden File Finder)
- Input Vector:
- Descrição: A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.

Figura 26: Ficheiros identificados pelo *ZAP*

4.2.6 WSTG-CONF-06 Testar métodos *HTTP*

Os métodos *HTTP* mais usados são o *GET* e o *POST*. Contudo, existem outros métodos pouco conhecidos que não são muito usados. Por esta razão, estes métodos podem ser usados para explorar as vulnerabilidades da aplicação.

Para verificar quais os métodos *HTTP* suportados pela aplicação podemos recorrer ao *nmap*. Na Figura 27 podemos verificar que o *nmap* identificou como métodos suportados *POST*, *OPTIONS*, *GET*, *HEAD* e *TRACE*, e identificou o *TRACE* como um comando potencialmente perigoso.

```
(kali㉿kali)-[~]
$ nmap -p 443 --script http-methods --script-args http-methods.url-path='/index.php' juice-shop
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-24 11:34 EDT
Nmap scan report for juice-shop (192.168.38.246)
Host is up (0.0016s latency).

PORT      STATE SERVICE
443/tcp    open  https
| http-methods:
|   Supported Methods: POST OPTIONS GET HEAD TRACE
|   Potentially risky methods: TRACE
|_  Path tested: /index.php

Nmap done: 1 IP address (1 host up) scanned in 1.15 seconds
```

Figura 27: Métodos *HTTP* suportados

4.2.7 WSTG-CONF-07 Testar *HTTP Strict Transport Security*

Este ponto não é aplicável no âmbito deste trabalho uma vez que a aplicação não foi desenvolvida sobre HTTPS.

4.2.8 WSTG-CONF-08 Testar política de domínio cruzado em *RIA*

Como neste caso não estamos a fazer uso de tecnologias como Adobe Flash, este ponto não se aplica.

4.2.9 WSTG-CONF-09 Testar permissões ficheiros

Este ponto não é aplicável no âmbito deste trabalho.

4.2.10 WSTG-CONF-10 Testar a tomada de controlo de subdomínios

Este ponto não é aplicável no âmbito deste trabalho.

4.2.11 WSTG-CONF-11 Testar armazenamento na *cloud*

Este ponto não é aplicável no âmbito deste trabalho.

4.3 WSTG-IDNT Testes de gestão de identidade

4.3.1 WSTG-IDNT-01 Testar definição de *roles*

Muitas aplicações possuem funcionalidades e serviços que só podem ser acedidos com as permissões correspondentes a um determinado *role*, sendo que numa aplicação geralmente existem diferentes tipos de *roles*.

Deste modo, é importante verificar que tipos de *roles* podemos encontrar na aplicação e se é possível a sua modificação.

Recorrendo ao *ZAP*, foi possível encontrar um ficheiro no qual constam 3 *roles*: *admin*, *costumer* e *deluxe*, tal como se pode observar na Figura 28.

The screenshot shows two parts of the ZAP interface. The top part displays a JSON response for a 'GET /role' request, listing three user objects with their roles and other details. The bottom part shows a table of requests for the 'role' endpoint, with several rows listed under the 'Method' column.

```

{
  "status": "success",
  "data": [
    {
      "UserId": "13",
      "Id": "1",
      "Caption": "#ratschi #honeedsfourlegs",
      "ImagePath": "assets/public/images/uploads/0/#ratschi1:#honeedsfourlegs-1572680969477.jpg",
      "createdAt": "2023-05-22T14:09:44.980Z",
      "updatedAt": "2023-05-22T14:09:44.980Z",
      "User": {"id": "13", "username": "", "email": "bjpeern@wpasp.org", "password": "92831b29e669749881963ba8463e466", "role": "deluxe", "deluxeToken": "e6fa2f15962e934406d5243af4fa513b70bf3ac971560df6fb95ddfcfe0e4", "lastLoginIp": "", "profileImage": "assets/public/images/uploads/13.jpg", "totpSecret": "", "isActive": true, "createdAt": "2023-05-22T14:09:43.523Z", "updatedAt": "2023-05-22T14:09:44.980Z", "deletedAt": null},
      "UserId": "4",
      "Id": "2",
      "Caption": "#Magen(e)tificent!",
      "ImagePath": "assets/public/images/uploads/magen(et).png",
      "createdAt": "2023-05-22T14:09:44.980Z",
      "updatedAt": "2023-05-22T14:09:44.980Z",
      "User": {"id": "4", "username": "Bkimenlich", "email": "bjoen.klaimlich@gmail.com", "password": "6edd9d726bcd873c59e41a8e75708c", "role": "admin", "deluxeToken": null, "lastLoginIp": "profileImage": "assets/public/images/uploads/defaultAdmin.png", "totpSecret": "", "isActive": true, "createdAt": "2023-05-22T14:09:43.520Z", "updatedAt": "2023-05-22T14:09:44.980Z", "deletedAt": null},
      "UserId": "14",
      "Id": "3",
      "Caption": "My rare collectors item! [REDACTED] * [REDACTED]",
      "ImagePath": "assets/public/images/uploads/my-rare-collectors-item-[REDACTED]-*[REDACTED]-1572680964534.jpg",
      "createdAt": "2023-05-22T14:09:44.980Z",
      "updatedAt": "2023-05-22T14:09:44.980Z",
      "User": {"id": "14", "username": "juice-shop", "email": "juice@juice-shop.com", "password": "6e076037760a11e09a0076c0596756c", "role": "customer", "deluxeToken": null, "lastLoginIp": "profileImage": "assets/public/images/uploads/defaultAdmin.png", "totpSecret": "", "isActive": true, "createdAt": "2023-05-22T14:09:43.520Z", "updatedAt": "2023-05-22T14:09:44.980Z", "deletedAt": null},
      "UserId": "18",
      "Id": "4",
      "Caption": "I love going hiking here...",
      "ImagePath": "assets/public/images/uploads/favorite-hiking-place.png",
      "createdAt": "2023-05-22T14:09:44.994Z",
      "updatedAt": "2023-05-22T14:09:44.994Z",
      "User": {"id": "18", "username": "Johnny", "email": "john@juice-shop.com", "password": "00479a957b6bd2459e6746478e44d5", "role": "customer", "deluxeToken": null, "lastLoginIp": "profileImage": "assets/public/images/uploads/default.svg", "totpSecret": "", "isActive": true, "createdAt": "2023-05-22T14:09:43.524Z", "updatedAt": "2023-05-22T14:09:44.998Z", "deletedAt": null},
      "UserId": "19",
      "Id": "5",
      "Caption": "My old workplace...",
      "ImagePath": "assets/public/images/uploads/IMG_4253.jpg",
      "createdAt": "2023-05-22T14:09:44.998Z",
      "updatedAt": "2023-05-22T14:09:44.998Z",
      "User": {"id": "19", "username": "Emma", "email": "emma@juice-shop.com", "password": "482f1c4a7e316afe5c56ea6314f7739", "role": "customer", "deluxeToken": null, "lastLoginIp": "profileImage": "assets/public/images/uploads/default.svg", "totpSecret": "", "isActive": true, "createdAt": "2023-05-22T14:09:43.524Z", "updatedAt": "2023-05-22T14:09:43.524Z", "deletedAt": null}
    ]
  ]
}

```

Método	URL	Match
GET	http://192.168.38.245/vendor.js	role
GET	http://192.168.38.245/vendorsmemories/	role

Figura 28: Roles da aplicação

4.3.2 WSTG-IDNT-02 Testar processo de registo de utilizadores

Dependendo dos requisitos de segurança dos sistemas, podem haver diferentes tipos de requisitos para acesso. Deste modo, devemos verificar se os requisitos para registo de utilizadores se alinham com os requisitos de segurança e do negócio.

Como podemos ver na Figura 29, no caso do *juice-shop* na página de registo apenas nos podemos registar como clientes, para o efeito devemos fornecer um email, uma password (com única condição de possuir mais de 5 caracteres e menos de 40) e escolher uma pergunta de segurança e fornecer a sua resposta. Através deste formulário de registo não existe nenhuma verificação da autenticidade das informações fornecidas e é possível introduzir palavras passe fracas (desde que dentro dos limites de caracteres). Um utilizador pode-se registar as vezes que quiser (desde que com um email diferente).

User Registration

Email *

Password *

① Password must be 5-40 characters long. 0/20

Repeat Password *

0/40

Show password advice

Security Question *

① This cannot be changed later!

Answer *

 Register

Already a customer?

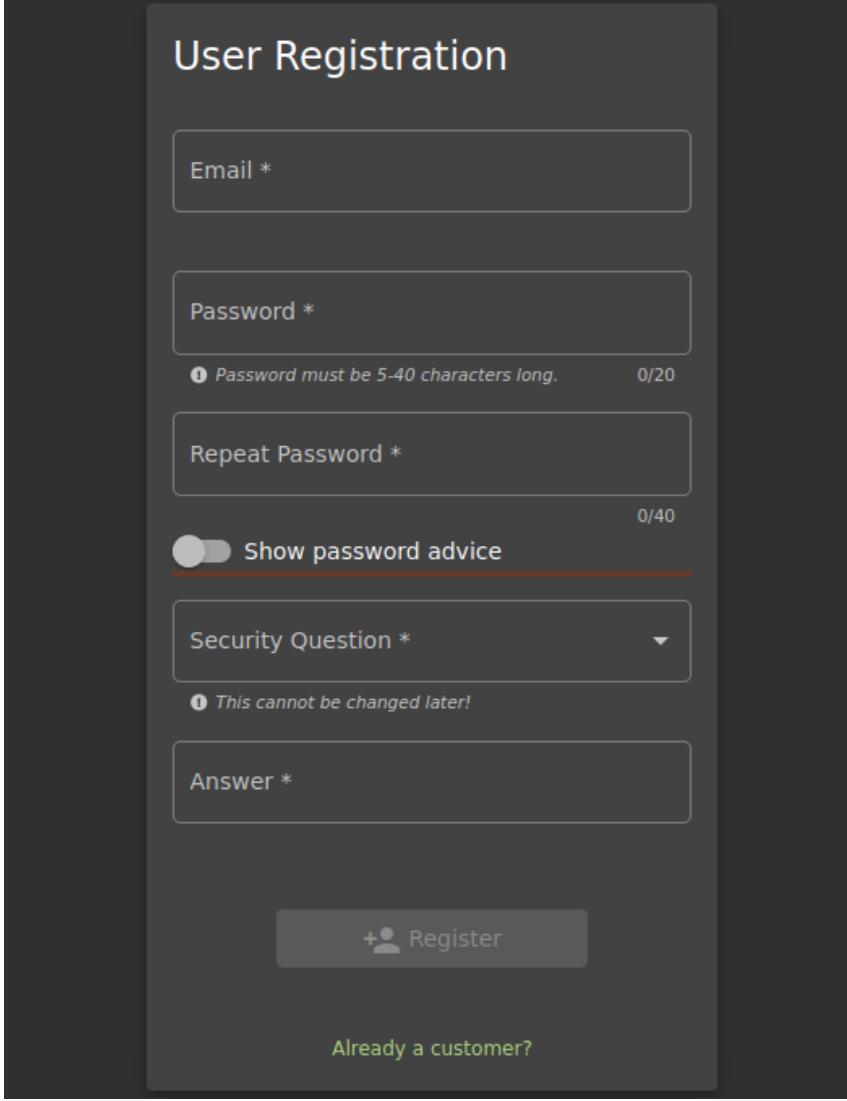


Figura 29: Registo de utilizadores

4.3.3 WSTG-IDNT-03 Testar processo de providenciamento de contas

O providenciamento de contas é uma oportunidade para os atacantes criarem uma conta válida ainda que sem autorização para tal ou sem fornecer dados que permitam a sua identificação, assim devemos ter em atenção que contas podem providenciar outras contas e de que tipo.

Na Figura 30, apresentada de seguida podemos observar uma conta criada com um email inventado, uma vez que o email fornecido no processo de registo não é verificado.

User Profile



Email:
ola@123

Username:
e.g. SuperUser

Set Username

\

File Upload:
Browse... No file selected.

Upload Picture

or

Image URL:
e.g. <https://www.gravatar.com/avatar/4b14523d1dcbb6229f7fa41a>

Link Image

Figura 30: Perfil de utilizador com email inventado

4.3.4 WSTG-IDNT-04 Testar para contas de utilizador enumeráveis e previsíveis

Neste teste pretende-se verificar se é possível recolher *usernames* válidos através da interação com o formulário de autenticação. Isto pode ser possível na medida em que por vezes as aplicações web reagem de maneira diferente a introduzir um utilizador inválido ou uma palavra passe errada.

Como podemos observar na Figura 32 e na Figura 31, ao inserir um utilizador inválido obtemos a mesma resposta que ao introduzir um utilizador válido com uma palavra passe errada, pelo que a aplicação se encontra protegida neste ponto.

Ao analisar algumas *reviews* no site podemos encontrar, tal como mostra a Figura 33, um email que parece ser o email do administrador do site pelo que pode constituir uma vulnerabilidade e ponto de entrada no site.

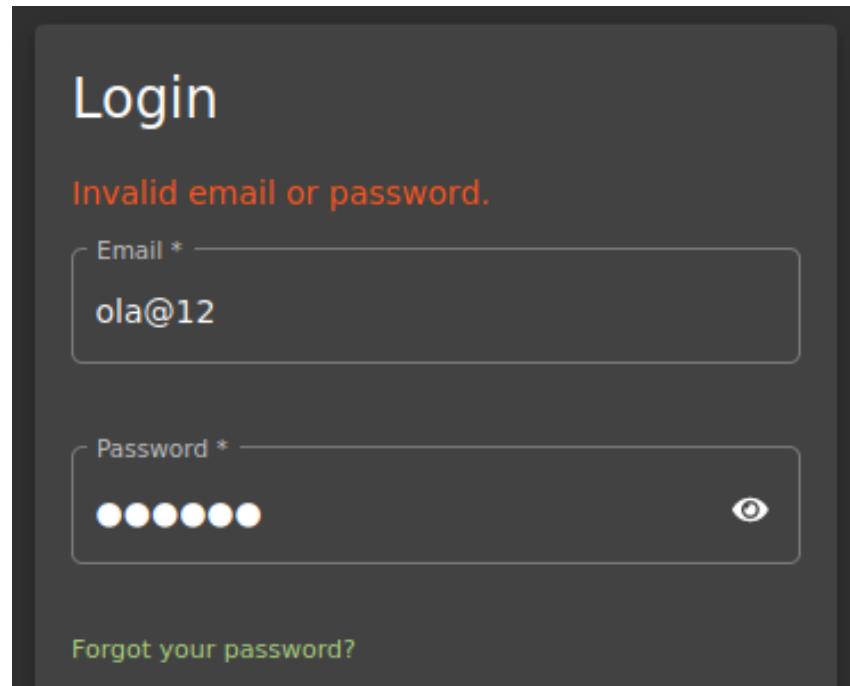


Figura 31: Login com utilizador inválido

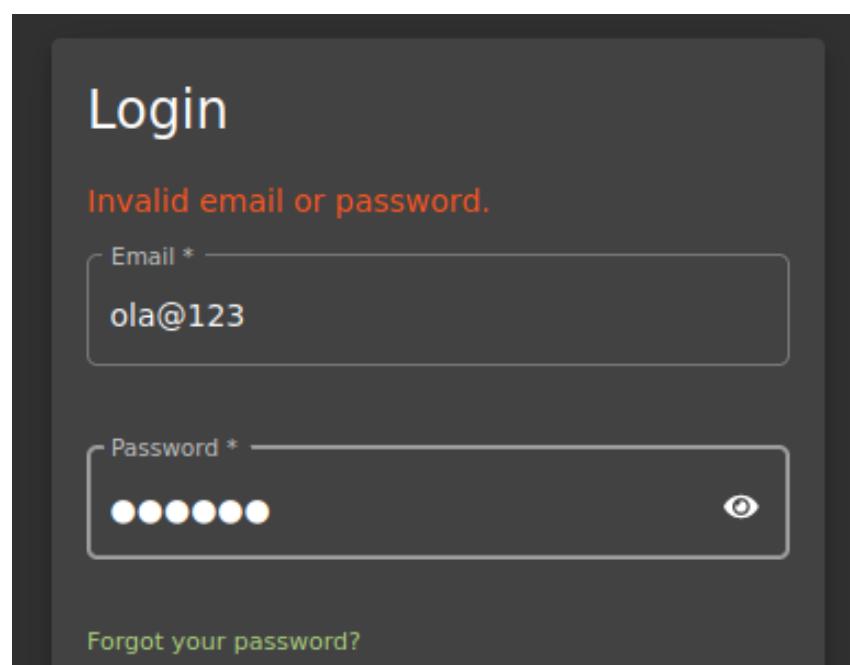


Figura 32: Login com utilizador válido e palavra passe errada

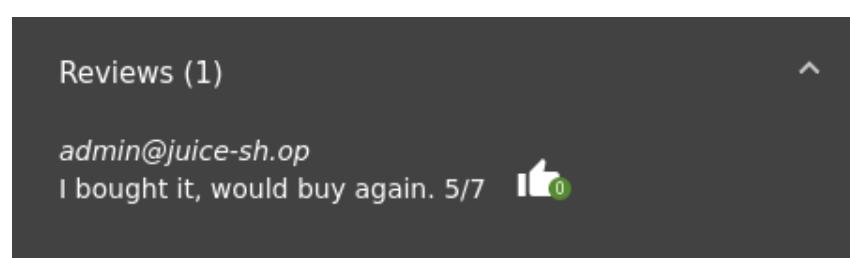


Figura 33: Review com email de administrador

4.3.5 WSTG-IDNT-05 Testar para política de nomes de utilizador fraca ou não forçada

Este teste tem por base o facto de os nomes de utilizador geralmente seguirem a estrutura nome, apelido e o facto de isso facilitar a enumeração de nomes de utilizador.

No caso do *juice-shop* podemos analisar os emails (que funcionam como nome de utilizador) presentes nas *reviews* apresentadas, como podemos observar na Figura 33 e na Figura 34. Apesar de não constituir um requisito, podemos observar que todos os emails das reviews seguem o formato *username@juice-sh.op*, pelo que a enumeração de contas se torna mais fácil. Através da análise das reviews conseguimos também identificar diversos emails de clientes.

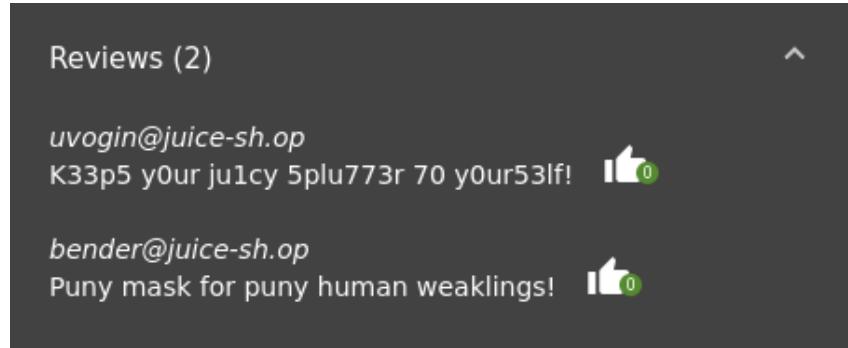


Figura 34: *Reviews* de utilizadores

4.4 WSTG-ATHN Testes de autenticação

Nesta secção, exploraremos diversos aspectos relacionados à autenticação em aplicações web, identificada como um dos principais pontos vulneráveis, como a qualidade das senhas utilizadas, a segurança nos canais de transporte, métodos de armazenamento, bem como os processos de alteração dessas senhas.

4.4.1 WSTG-ATHN-01 Teste para credenciais transportadas através de um canal encriptado

Verificar se os dados de autenticação são criptografados durante a transmissão é fundamental para impedir que invasores capturem informações de credenciais por meio de monitoramento do tráfego de rede. A falta de criptografia pode permitir que invasores visualizem e usem as credenciais para roubar contas de utilizadores. É importante garantir a criptografia adequada dos dados em trânsito, considerando não apenas o uso do protocolo HTTPS, mas também a robustez do algoritmo de criptografia e das chaves utilizadas.

Uma vez que o JuiceShop utiliza o protocolo HTTP, que já é uma vulnerabilidade *per se*, consideramos que os testes deste tópico não são aplicáveis no contexto do nosso trabalho.

4.4.2 WSTG-ATHN-02 Teste às credenciais predefinidas

Aplicações web frequentemente utilizam software de fácil instalação e configuração mínima. No entanto, essas aplicações muitas vezes são mal configuradas, com credenciais padrão amplamente conhecidas pelos atacantes. Não alterar essas senhas pode originar um acesso não autorizado por parte de atacantes.

Assim, realizamos um ataque fuzz utilizando o OWASP ZAP, recorrendo a dicionários de e-mails e/ou usernames e passwords mais comuns na sua configuração, ver Figura 35.

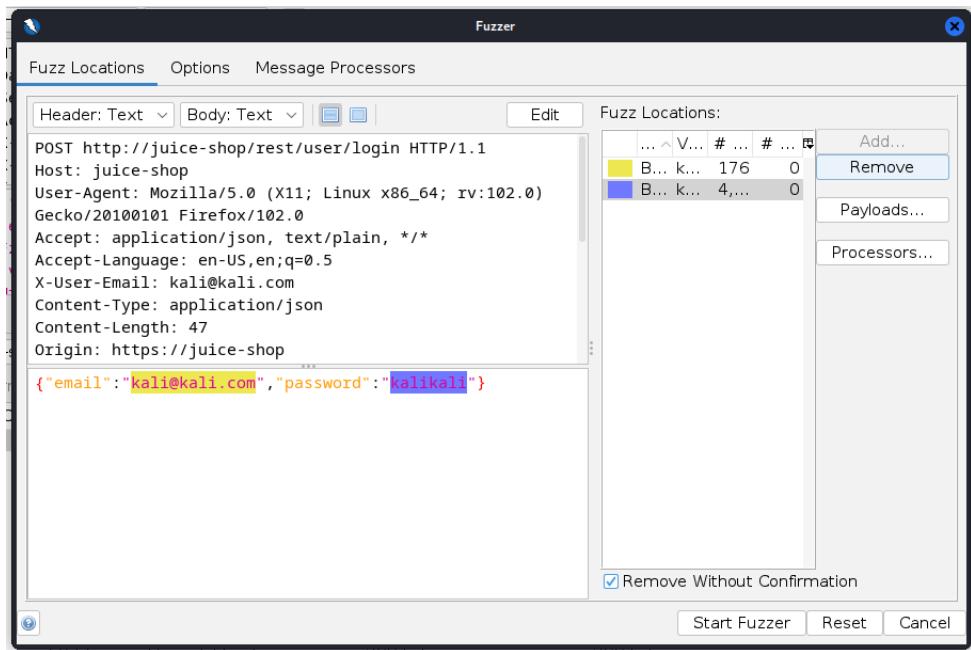


Figura 35: Configuração de um ataque fuzz para encontrar contas predefinidas

Na Figura 36 encontram-se os resultados do ataque fuzz ainda em andamento.

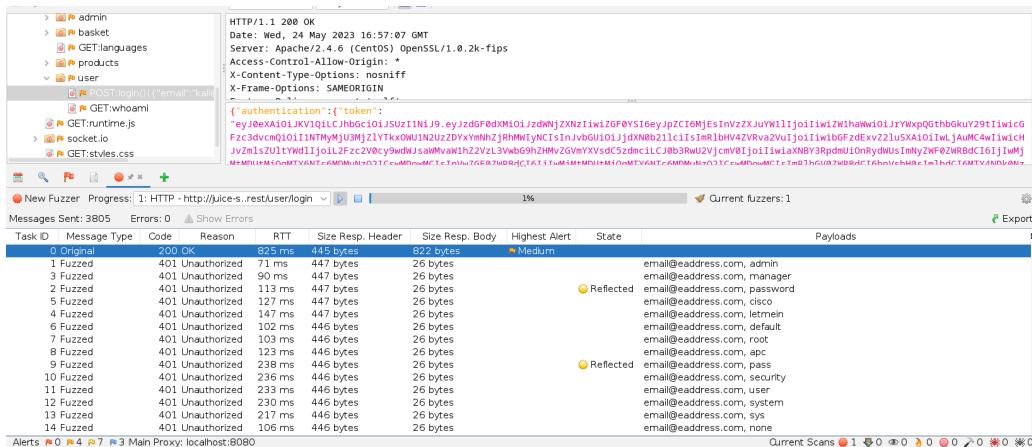


Figura 36: Resultados do ataque fuzz em execução

4.4.3 WSTG-ATHN-03 Teste para detecção de um mecanismo de bloqueio fraco

Mecanismos de bloqueio de conta são usados para mitigar ataques de força bruta. As contas normalmente são bloqueadas após 3 a 5 tentativas mal sucedidas e só podem ser desbloqueadas após um período de tempo predeterminado [5].

Como prova no teste anterior, Figura 36, efetivamente não há nenhuma proteção contra ataques de força bruta, considerando isto uma vulnerabilidade grave.

4.4.4 WSTG-ATHN-04 Teste para contornar o esquema de autenticação

Esta secção visa testar se o esquema de autenticação pode ser ignorado, isto é, contornando a página de *login*, e entrar numa página interna que só deve ser acessada apenas e após a autenticação.

Como as comunicações são feitas através do protocolo HTTP, ou seja, sem encriptação, os *tokens* de sessão são enviados em *plaintext*, ver Figura 37. Através da captação destes *tokens*, por meio de monitoramento do tráfego de rede, é possível aceder a páginas e a informações privadas de utilizadores sem passar pela página de login, ver Figura 38.

```

HTTP/1.1 200 OK
Date: Wed, 24 May 2023 16:57:07 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.1.0_2k-fips
{"authentication": {"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMlOiJzdWNjZXNzIiwiZW1haWwiOiJrYWxpQGtbGkuY29tIiwicGFzc3dvcmQiOiIiNTMyMjU3MjZlYTkxOWU1N2UzZDYxYmNhZjRhMWIyNCIsInJvbGUiOijjdXN0b21lcisImRlbHV4ZVRva2VuIjoiiwibGfdExvZ2luSXAiOiwLjAuMC4wiwicHjvZm1szU1lYwd1joiI2Fzc2V0cy9wdWjsaMmVwa1hZ2VzL3VwbG9hZHmvZGVmYXVsC5zdmciLCJ0b3RwU2VjcmV0IjoiIwiwicNBY3RpdmUiOnRydWUsimNyZWF0ZWRBdCI61j1wJMMEDUtMjQgMTy6NTc6MDMUznQ2ICswMDowMCIsInRlbGV0ZWRBdCI6bnVsbt0sImldCI6MTy4NDk0Nz0yOCwizXhwIjoxNjg0OTY1NDI4fQ.qzs_1_Ud25wRNuiAaa7Tnl8jsimzFjqqAY67d34z1uBejd1FsRiDA6f1HeDTSmmn-uE7U7G4B5NFM1L6P7xKbWduJNiwivJxi0dlPYi6ZtqXrSb5CXSKYLVazmqkjEz3FC646HtT8IhdSeN_a2HBe8r4cd0HmvGR7_jefZMtc", "bid": 6, "umail": "kali@kali.com"}}

```

Figura 37: Token enviado em *plaintext* após login

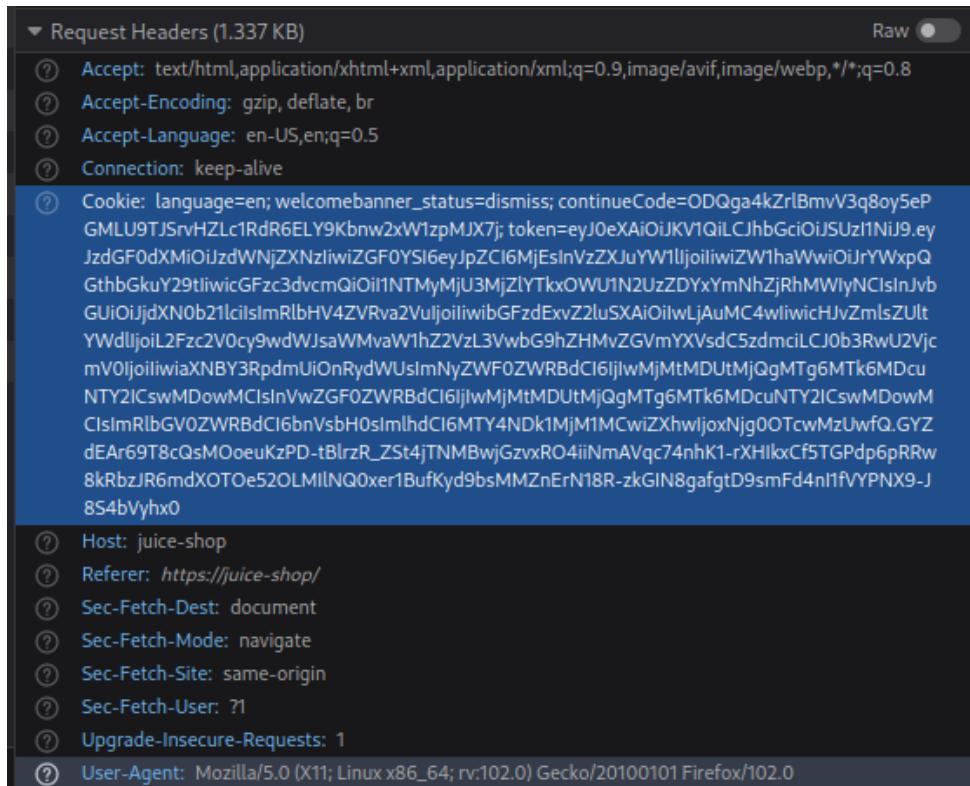


Figura 38: Utilização do token para aceder a páginas da área autenticada

4.4.5 WSTG-ATHN-05 Teste para vulnerabilidade de lembrar password

As aplicações web normalmente disponibilizam uma funcionalidade de lembrar da sessão, que permite que o utilizador permaneça autenticado por longos períodos, sem solicitar novamente ao utilizador as suas credenciais [5].

De facto, a aplicação web guarda no *local storage* tanto o *email* do utilizador como o *token* de sessão, ver Figura 39.

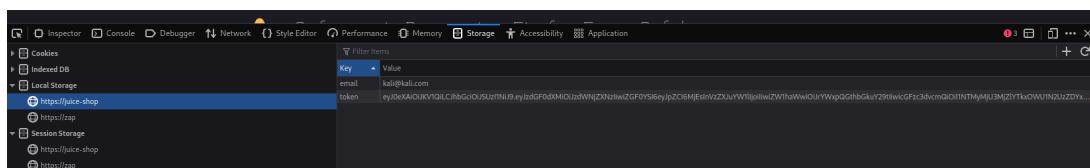


Figura 39: Dados guardados no local storage pela aplicação web

4.4.6 WSTG-ATHN-06 Teste para detectar fragilidades da cache do navegador

Nesta fase, deve-se verificar se a aplicação indica corretamente o navegador a não guardar dados confidenciais em *cache*.

Verificamos que o JuiceShop, especialmente nas páginas com informações confidenciais, contém explicitamente nos *headers* HTTP a não persistência do seu conteúdo em *cache*, ver Figura 40.

```

▼ Response Headers (527 B)
⑦ Accept-Ranges: bytes
⑦ Access-Control-Allow-Origin: *
⑦ Cache-Control: no-cache, no-store
⑦ Connection: Keep-Alive
⑦ Content-Length: 2076
⑦ Content-Type: text/html; charset=UTF-8
⑦ Date: Wed, 24 May 2023 18:56:14 GMT
⑦ ETag: W/"7c3-1884eb98703"
⑦ Feature-Policy: payment 'self'
⑦ Keep-Alive: timeout=5, max=100
⑦ Last-Modified: Wed, 24 May 2023 17:06:01 GMT
⑦ Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
⑦ Vary: Accept-Encoding
⑦ X-Content-Type-Options: nosniff
⑦ X-Frame-Options: SAMEORIGIN
X-Recruiting: /#/jobs

```

Figura 40: Exemplo de um *header* de “Cache-Control” de uma página com conteúdo confidencial

4.4.7 WSTG-ATHN-07 Testar a política de palavras-passe fracas

Este ponto tem como objetivo avaliar e testar a política de passwords utilizada pela aplicação *web*, isto é, se restringe passwords com pouca qualidade.

De facto, na página de registo, e apesar de apresentar perto do formulário conselhos para uma *password* mais segura, a verdade é que a aplicação *web* permite que o utilizador escolha e submeta uma *password* que não cumpre com esses mesmos conselhos de boas práticas para uma *password* efetivamente segura, como podemos ver na Figura 41.

The screenshot shows a registration form with two password input fields. The first field is labeled "Password *". It contains five dots and has a validation message below it: "Password must be 5-40 characters long." To its right is a character counter "5/20". The second field is labeled "Repeat Password *". It also contains five dots and has a character counter "5/40". Below these fields is a button labeled "Show password advice". A tooltip is displayed, listing six items: "contains at least one lower character" (red exclamation mark), "contains at least one upper character" (red exclamation mark), "contains at least one digit" (green checkmark), "contains at least one special character" (red exclamation mark), and "contains at least 8 characters" (red exclamation mark). At the bottom of the form is a "Security Question *" field with a dropdown arrow.

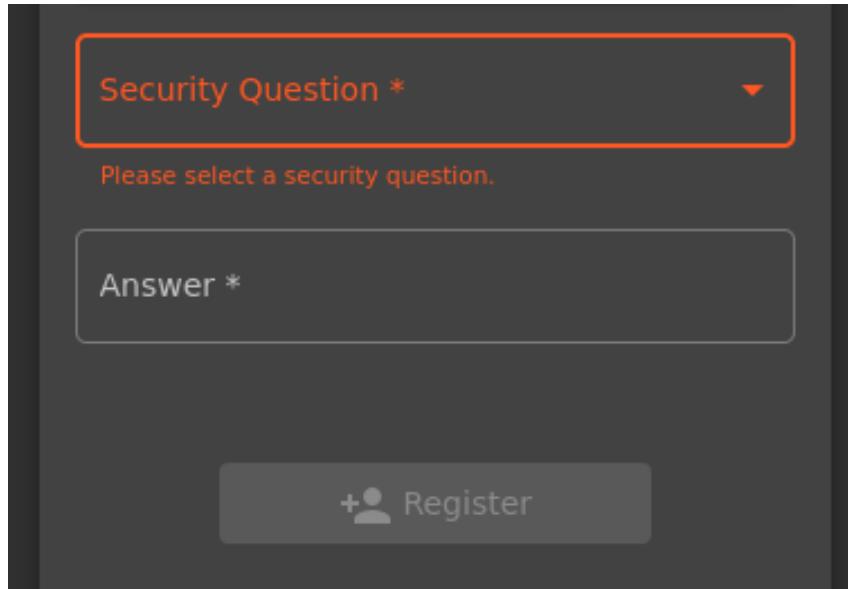
Figura 41: Aceitação por parte da aplicação *web* de *passwords* como “12345”, apesar de não cumprir com as boas práticas que a própria aplicação recomenda (mas que não obriga)

Além disso, e como já foi visto na Figura 36, através de ataques de força bruta com auxílio a dicionários de password comuns, foi possível encontrar para o email do administrador, admin@juice-sh.op, exibido nas próprias páginas da aplicação web, a respetiva password: “admin123”.

4.4.8 WSTG-ATHN-08 Teste a fragilidades de perguntas e respostas de segurança

As perguntas e respostas de segurança costumam ser utilizadas para recuperar passwords esquecidas, para redefinição de passwords ou como segurança extra sobre a password.

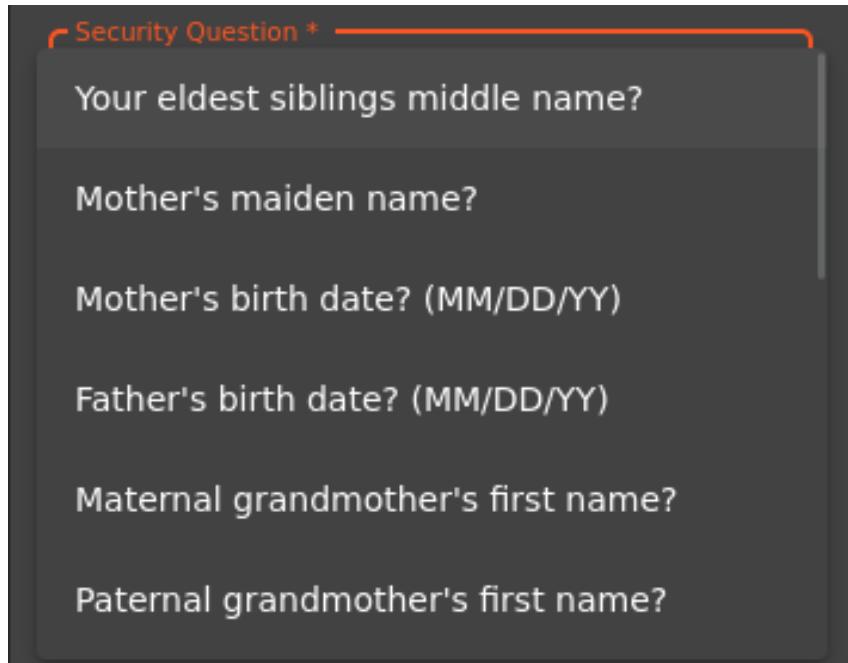
Nesta aplicação web é pedido no registo a definição de uma pergunta e resposta de segurança, ver Figura 42.



The screenshot shows a dark-themed web page with a registration form. At the top, there is a dropdown menu labeled "Security Question *". Below it, a message says "Please select a security question." A large input field is labeled "Answer *". At the bottom right is a button labeled "Register" with a user icon.

Figura 42: Obrigatoriedade na definição de uma pergunta e resposta de segurança

Este tipo de formulários são suscetíveis ao uso de ataques de *fuzzing* e de engenharia social, se conseguirmos informações do domínio semi-público, como o nome de familiares ou datas de nascimento, ver Figura 43.



The screenshot shows a dropdown menu with a red border and a red asterisk next to the label "Security Question *". The menu lists several pre-generated security questions:

- Your eldest sibling's middle name?
- Mother's maiden name?
- Mother's birth date? (MM/DD/YY)
- Father's birth date? (MM/DD/YY)
- Maternal grandmother's first name?
- Paternal grandmother's first name?

Figura 43: Perguntas pré-geradas de natureza simplista que podem levar a respostas inseguras

4.4.9 WSTG-ATHN-09 Teste à fragilidade de funcionalidades de alteração ou reposição de palavras-pases

A redefinição da *password* só é possível dentro da aplicação, e apenas é pedido a *password* antiga, sem qualquer confirmação com o utilizador, Figura 44.

The screenshot shows a 'Change Password' page with a success message: 'Your password was successfully changed.' Below it are three input fields: 'Current Password *', 'New Password *', and 'Repeat New Password *'. A validation message 'Password must be 5-40 characters long.' is displayed above the 'New Password' field, along with character count indicators '0/40' and '0/20' for each field. A 'Change' button is at the bottom.

Figura 44: Página para alterar a *password*

4.4.10 WSTG-ATHN-10 Teste a autenticações mais fracas em canais alternativos

Os canais alternativos podem representar riscos ao canal principal, pois podem ser utilizados para contorná-lo ou expor informações sensíveis. Esses canais podem incluir diferentes aplicações web, como versões para telemóvel, versões para outros países, além de sites para desenvolvimento e teste, mas relacionados ao site principal.

Uma vez que o JuiceShop não possuí um canal alternativo de autenticação, este ponto não é aplicável.

4.5 WSTG-ATHZ Testes de autorização

4.5.1 WSTG-ATHZ-01 Teste de inclusão da diretoria de travessia de ficheiros

As aplicações web implementam mecanismos de autenticação para controlar o acesso a ficheiros que não devem ser acedidos, para que atacantes não consigam ter acesso aos ficheiros nem fazer alterações nos mesmos.

Recorrendo ao *ZAP*, é possível encontrar a vulnerabilidade apresentada na Figura 45. Através deste alerta podemos verificar que o *ZAP* encontrou um ficheiro (neste caso *.jpg*), fora da diretoria de raiz.

The screenshot shows the ZAP interface with a 'Path Traversal' alert. The alert details are as follows:

URL:	http://192.168.38.245/assets/public/images/uploads/magn(/magn(thisshouldnotexistandhopefullyitwillnot)ifcient!-1571814229653.jpg)
Risco:	High
Confiança:	Low
Parâmetro:	_ID_magn
Ataque:	/magn(thisshouldnotexistandhopefullyitwillnot)ifcient!-1571814229653.jpg
Evidência:	CWE ID: 22 WASC ID: 33
Fonte:	Ativo (6 - Path Traversal)
Input Vector:	OData ID
Descrição:	The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP-based interface is

Figura 45: *Path Traversal*

4.5.2 WSTG-ATHZ-02 Teste para contornar o esquema de autorização

Neste teste pretende-se verificar se o esquema de autorização foi bem implementado. Deste modo devemos verificar se o utilizador sem estar autenticado consegue aceder a recursos reservados a utilizadores autenticados, ou se um utilizador autenticado consegue aceder a recursos de outros *roles*.

4.5.3 WSTG-ATHZ-03 Teste de escalonamento de privilégios

Neste teste pretende-se verificar que não existe escalonamento de privilégios através da modificação do *role* de um utilizador de modo a que este tenha acesso a privilégios que antes não tinha.

Um dos casos de escalonamento de privilégios ocorre quando o utilizador consegue adquirir uma conta administrativa. Através da *SQL Injection* apresentada na Figura 53 o utilizador passa a ter acesso a uma conta administrativa, pelo que é possível escalonamento de privilégios.

4.5.4 WSTG-ATHZ-04 Teste para referências inseguras diretas a objetos

Este teste pretende verificar se é possível aceder a recursos do sistema diretamente, sem autenticação. Este tipo de ataque é possível uma vez que a aplicação usa um input fornecido e usa-o para aceder a um objeto sem efetuar verificações de autenticação.

4.6 WSTG-SESS Testes de gestão de sessão

Esta secção aborda os mecanismos de gestão de sessões que as aplicações web utilizam para manter uma sessão ativa e salva no navegador.

4.6.1 WSTG-SESS-01 Testes ao esquema de gestão de sessão

Qualquer aplicação *web* possuí mecanismos pelo qual controla e mantém o estado de um utilizador. Para evitar a autenticação contínua a cada página do serviço, as aplicações *web* implementam vários mecanismos para armazenar e validar credenciais por um período de tempo limitado. Estes mecanismos são conhecidos como gestão de sessões [5].

Deve-se verificar se os *cookies* e outros *tokens* de sessão são criados de forma segura. Um invasor pode sequestrar facilmente as sessões de outros utilizadores legítimos [5].

O OWASP ZAP identificou várias situações que serão expostas nos tópicos seguintes.

4.6.2 WSTG-SESS-02 Testes aos atributos de *cookies*

Os *cookies* costumam ser um vetor de ataque importante, normalmente direcionados a outros utilizadores. O HTTP é um protocolo sem estado, e para corrigir esse problema, as sessões foram criadas e anexadas às solicitações HTTP, por exemplo, através do “Set-Cookie”. Os *cookies* podem ser utilizados para gestão de sessão, personalização e monitorização. Os *cookies* podem ser protegidos através da definição dos atributos e dos prefixos de *cookies* [5].

Na Figura 46 identificamos que o JuiceShop não utiliza a flag *HttpOnly*, o que significa que o *cookie* pode ser acedido por JavaScript.

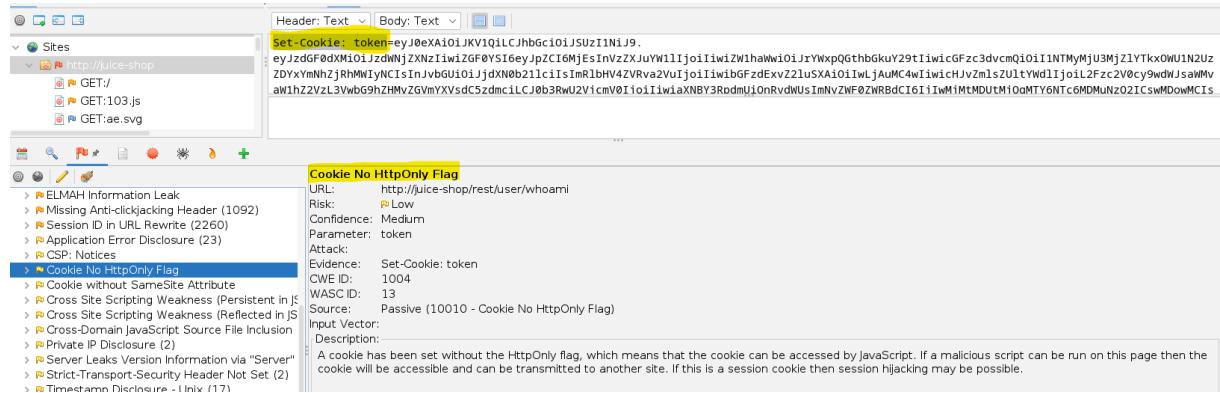


Figura 46: Alerta do OWASP ZAP - Cookie No HttpOnly Flag

Na Figura 47 verificamos que o JuiceShop não utiliza o atributo “SameSite”, o que significa que o *cookie* pode ser enviado como resultado de um pedido *cross-site*.



Figura 47: Alerta do OWASP ZAP - Cookie without SameSite Attribute

Na Figura 48 apresenta ainda outro alerta reportado sobre a delimitação por domínio ou caminho dos *cookies*.



Figura 48: Alerta do OWASP ZAP - Loosely Scoped Cookie

4.6.3 WSTG-SESS-03 Testes para fixação de sessão

A vulnerabilidade de fixação de sessão ocorre quando os *cookies* da sessão são mantidos com o mesmo valor antes e depois do processo de autenticação. Ocorre quando os *cookies* de sessão são utilizados para armazenar informações de estado, como adicionar itens a um carrinho de compras antes do *login* [5].

Assim, um invasor começa por obter um conjunto de *cookies* de sessão antes de fazer *login*. Depois, o invasor pode então enviar esses *cookies* para o navegador da vítima. Se a vítima fizer *login* e se os *cookies* não forem atualizados, ela passará a ser identificada pelos *cookies* de sessão fornecidos pelo invasor. O invasor pode então passar-se pela vítima utilizando esses *cookies* já conhecidos por ele [5].

Não foram encontradas vulnerabilidades que se enquadram neste tópico.

4.6.4 WSTG-SESS-04 Testes ás variáveis de sessão expostas

Quando os *tokens* de sessão são expostos, isso permite que um invasor se faça passar pela vítima e obtenha acesso não autorizado a aplicação. É crucial garantir que esses *tokens* estejam constantemente protegidos contra interceptação, especialmente durante a comunicação entre o navegador do cliente e os servidores da aplicação web [5].

No JuiceShop, e como já foi referido, a aplicação utiliza HTTP, e por isso, os *tokens* ou *cookies* de sessão não estão encriptados na comunicação entre o cliente e o servidor.

Na Figura 49, verificamos que o ID da sessão está visível no URL. Além disso, o ID da sessão pode ficar armazenado no histórico do navegador ou nos registos do servidor.



Figura 49: Alerta do OWASP ZAP - Session ID in URL Rewrite

4.6.5 WSTG-SESS-05 Testes de *Cross-Site Request Forgery (CSRF)*

O *Cross-Site Request Forgery (CSRF)* ocorre quando um utilizador é induzido a realizar ações indesejadas numa aplicação web ao qual ele está autenticado. Com o uso de técnicas de engenharia social, como enviar um *link* por *e-mail* ou *chat*, um invasor pode manipular os utilizadores a executar ações. Um ataque CSRF bem-sucedido pode resultar no comprometimento de dados e de operações. Administradores da aplicação web também podem ser vítimas de um ataque CSRF [5].

A Figura 50 documenta que o JuiceShop não previne ataques do tipo CSRF.

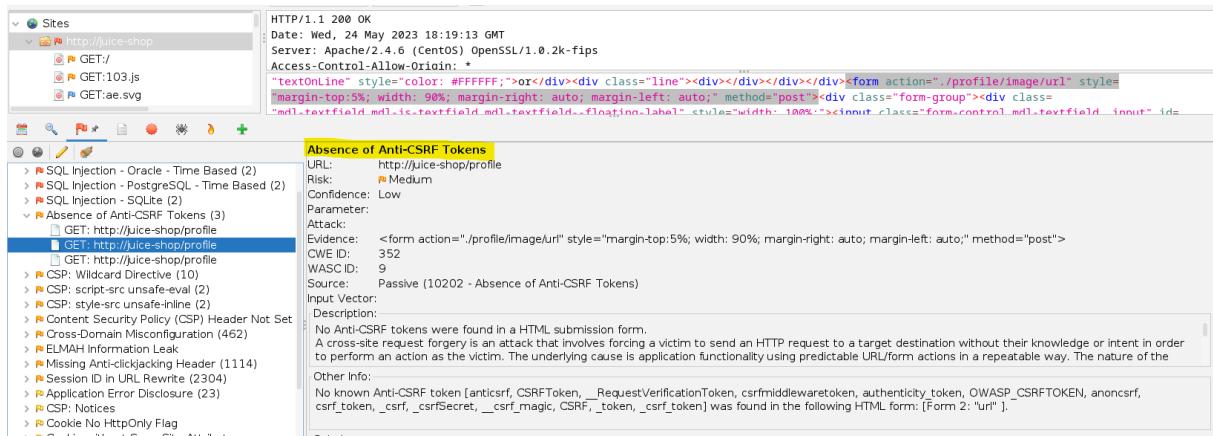


Figura 50: Alerta do OWASP ZAP - Absence of Anti-CSRF Tokens

4.6.6 WSTG-SESS-06 Teste da funcionalidade de *logout*

Garantir um encerramento adequado da sessão é muito importante na segurança de aplicações web. Ao reduzir o tempo de vida dos *tokens* de sessão, diminuímos a probabilidade de um ataque bem-sucedido, uma vez que atua como uma medida de prevenção contra ataques, como *Cross-Site Scripting (XSS)* e *Cross-Site Request Forgery (CSRF)*. Esses ataques dependem da presença de uma sessão autenticada pelo utilizador [5].

A funcionalidade de *logout* está a funcionar corretamente, uma vez que remove as informações de sessão do utilizador guardadas no *local storage* do navegador.

4.6.7 WSTG-SESS-07 Testes ao tempo limite de sessão

Deve-se verificar se a aplicação encerra automaticamente a sessão de um utilizador, quando fica inativo por um determinado período de tempo. Desta forma, garante que a mesma sessão não possa ser reutilizada e que nenhum dado sensível fique armazenado em *cache* no navegador. O tempo de limite mais adequado deve ser equilibrado, dependendo do nível de sensibilidade dos dados manipulados pela aplicação [5].

Note-se que a aplicação web em análise não tem nenhum tempo limite de sessão.

4.6.8 WSTG-SESS-08 Testes a *Session Puzzling*

Esta falha de segurança ocorre quando uma aplicação utiliza a mesma variável de sessão para diferentes propósitos. Isto permite que um invasor accesse páginas numa sequência não prevista pelos desenvolvedo-

res, fazendo com que a variável de sessão seja definida num contexto e posteriormente utilizada noutra contexto [5].

Como a maneira mais eficaz de detectar estas vulnerabilidades é por meio de uma revisão do código-fonte [5], consideramos que este tópico está fora do âmbito do trabalho.

4.6.9 WSTG-SESS-09 Testes de sequestro da sessão

Um atacante que obtém acesso aos *cookies* de sessão de outro utilizador pode se fazer passar por esse utilizador ao apresentar esses *cookies*. Esse tipo de ataque é chamado de sequestro de sessão. Quando os invasores controlam a rede utilizada pela vítima, os *cookies* de sessão podem ser indevidamente expostos ao invasor por meio de uma conexão HTTP. Para evitar isso, é importante marcar os *cookies* de sessão com o atributo “Secure”, garantindo que sejam transmitidos apenas via conexão HTTPS [5].

Mais uma vez, e restringindo ao contexto do JuiceShop, as vulnerabilidades provocadas pelo uso do HTTP já foram amplamente abordadas, sendo este tópico mais uma dessas vulnerabilidades, consequência do uso do HTTP.

4.7 WSTG-INPV Testes de validação de *input*

Nesta subsecção iremos abordar alguns testes relacionados com a validação de input. Devido à extensão da presente secção iremos abordar apenas alguns dos testes.

4.7.1 WSTG-INPV-01 Testar *Reflected Cross Site Scripting*

Este tipo de ataques ocorre quando o atacante é capaz de introduzir código executável num resposta HTTP. Este tipo de ataque apenas afeta utilizadores que acedem ao conteúdo malicioso e não afeta a aplicação propriamente dita.

No contexto da aplicação a ser testada foi encontrado um caso em que esta situação pode ocorrer. Acedendo ao menu de perfil, o utilizador pode optar por adicionar uma foto de perfil através de uma imagem no computador ou pode optar por indicar o *url* da imagem. Caso a opção escolhida seja a de indicar a imagem através de um *url*, é possível adicionar um *url* não correspondente a uma imagem, isto pode ser verificado na Figura 51 e na Figura 52.

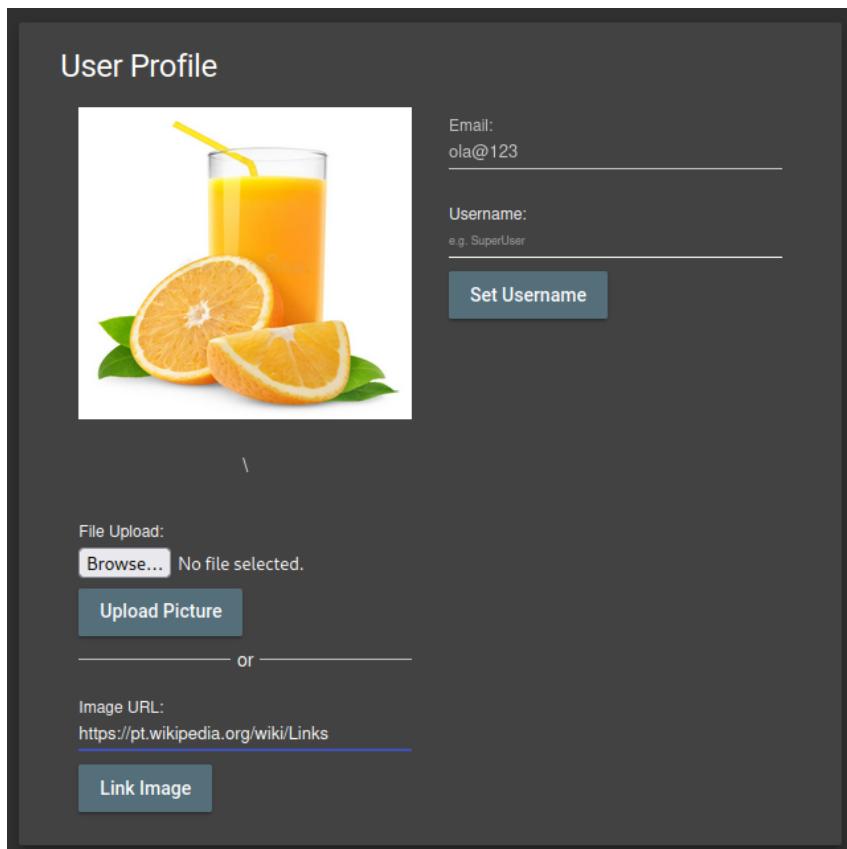


Figura 51: Perfil antes do ataque de XSS

The screenshot shows a 'User Profile' page with a dark background. At the top, it says 'User Profile'. Below that, there's a 'profile picture' section with a placeholder image and a 'File Upload:' field containing the message 'Browse... No file selected.' A blue button labeled 'Upload Picture' is below it. To the right, there are fields for 'Email:' (containing 'ola@123') and 'Username:' (containing 'e.g. SuperUser'), with a 'Set Username' button. A horizontal line with the word 'or' separates this from an 'Image URL:' field containing 'e.g. https://www.gravatar.com/avatar/4b14523d1dcbb6229f7fa41a' and a 'Link Image' button.

Figura 52: Perfil após possível ataque *XSS*

4.7.2 WSTG-INPV-05 Testar *SQL Injection*

Um ataque de *SQL Injection* ocorre quando é possível usar *queries SQL* para injetar ou obter informação da aplicação web. Este tipo de ataques geralmente ocorre quando não existe validação dos inputs do utilizador.

No *juice-shop* foram encontrados alguns locais nos quais é possível fazer *SQL Injection*, nomeadamente na página de *login*. Na Figura 53 e Figura 54 podemos analisar o ataque de *SQL Injection* e o seu resultado.

The screenshot shows a 'Login' page with a dark background. It has two input fields: 'Email *' containing "' OR 1=1 --" and 'Password *' containing several dots. Below the inputs is a 'Forgot your password?' link. At the bottom is a 'Log in' button with a user icon and a 'Remember me' checkbox. A link 'Not yet a customer?' is at the very bottom.

Figura 53: *SQL injection* na página de *login*

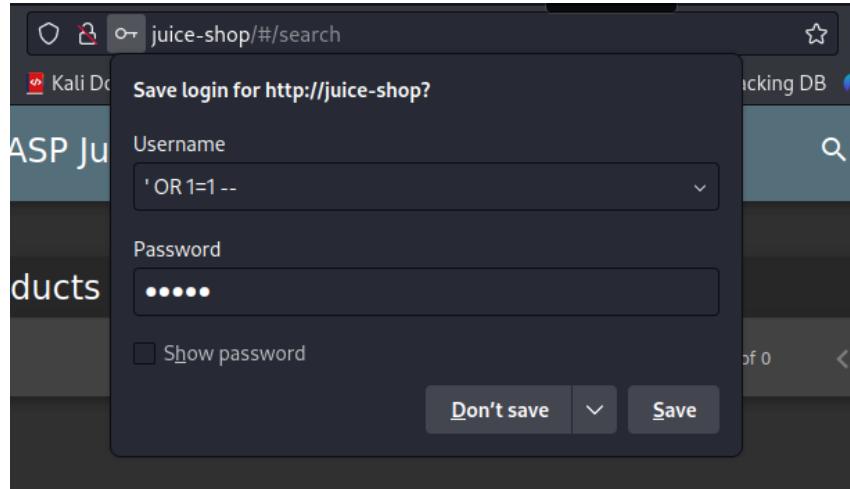


Figura 54: Resultado *SQL injection*

4.8 WSTG-ERRH Testes de tratamento de erros

Erros gerados pelas aplicações podem permitir que invasores obtenham informações sobre as APIs internas, mapeiem serviços, identifiquem versões e tipos de aplicações ou contornem exceções de lógica de segurança. O tratamento adequado de erros é fundamental para prevenir essas vulnerabilidades e garantir a segurança dos sistemas.

4.8.1 WSTG-ERRH-01 Teste para tratamento inadequado de erros

De facto, no JuiceShop, é fácil encontrar este tipo de vulnerabilidades, por exemplo, se inserirmos SQL no formulário de *login*, como demonstrado na Figura 55, obtemos bastante informação sobre o que provocou o erro, nomeadamente sobre *query*, ver Figura 56, que pode ser um meio para encontrar outras vulnerabilidades mais graves.

Figura 55: Ataque de *SQL Injection* no formulário de *login*

```

HTTP/1.1 500 Internal Server Error

{
  "error": {
    "message": "SQLITE_ERROR: near '\"' AND password = '\"': syntax error",
    "stack": "Error\n    at Database.<anonymous> (/home/dariofelix/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n    at Database.serialize (<anonymous>)\n    at /home/dariofelix/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n    at new Promise (<anonymous>)\n    at Query.run (/home/dariofelix/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n    at /home/dariofelix/juice-shop/node_modules/sequelize/lib/sequelize.js:315:28\n    at processTicksAndRejections (node:internal/process/task_queues:96:5)",
    "name": "SequelizeDatabaseError",
    "parent": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Users WHERE email = 'John' or 1=1' AND password = '4868e239f165ab6c7d47b86ba389be91' AND deletedAt IS NULL"
    },
    "original": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Users WHERE email = 'John' or 1=1' AND password = '4868e239f165ab6c7d47b86ba389be91' AND deletedAt IS NULL"
    },
    "sql": "SELECT * FROM Users WHERE email = 'John' or 1=1' AND password = '4868e239f165ab6c7d47b86ba389be91' AND deletedAt IS NULL",
    "parameters": {}
  }
}

```

Figura 56: Mensagem de erro

4.8.2 WSTG-ERRH-02 Testes para stack traces

Facilmente e ao explorar a aplicação web, encontra-se vulnerabilidades que se enquadraram neste tópico, como é o caso da Figura 57, que foi reportado pelo OWASP ZAP via *proxy*, enquanto se navegava pela aplicação *web* no *browser*.

The screenshot shows the OWASP ZAP interface with the following details:

- Left Panel (Alerts):** Shows various security issues found during the scan, including:
 - GET:favicon.ico
 - GET:main.js
 - GET:MaterialIcons-Regular.woff2
 - GET:polyfills.js
 - GET:profile
 - rest
- Center Panel (Details):**
 - Title:** HTTP/1.1 500 Internal Server Error
 - Description:** This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
 - Solution:** Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
 - Reference:**
 - Alert Tags:**

Key	Value
WSTG-v42-ERRH-02	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/4.2-Testing_for_Vulnerabilities/4.2.2-Common_Vulnerabilities/4.2.2.1-Information_Exposure/4.2.2.1.1-Application_Error_Disclosure
WSTG-v42-ERRH-01	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/4.2-Testing_for_Vulnerabilities/4.2.2-Common_Vulnerabilities/4.2.2.1-Information_Exposure/4.2.2.1.1-Application_Error_Disclosure
- Bottom Panel (Summary):** Shows current scans and proxy status.

Figura 57: Vulnerabilidade alertado pelo OWASP ZAP

Além disso, provocou-se um erro ao adicionar uma *path* incorreta ao URL da aplicação *web*, cuja a resposta do sistema está ilustrada na Figura 58.

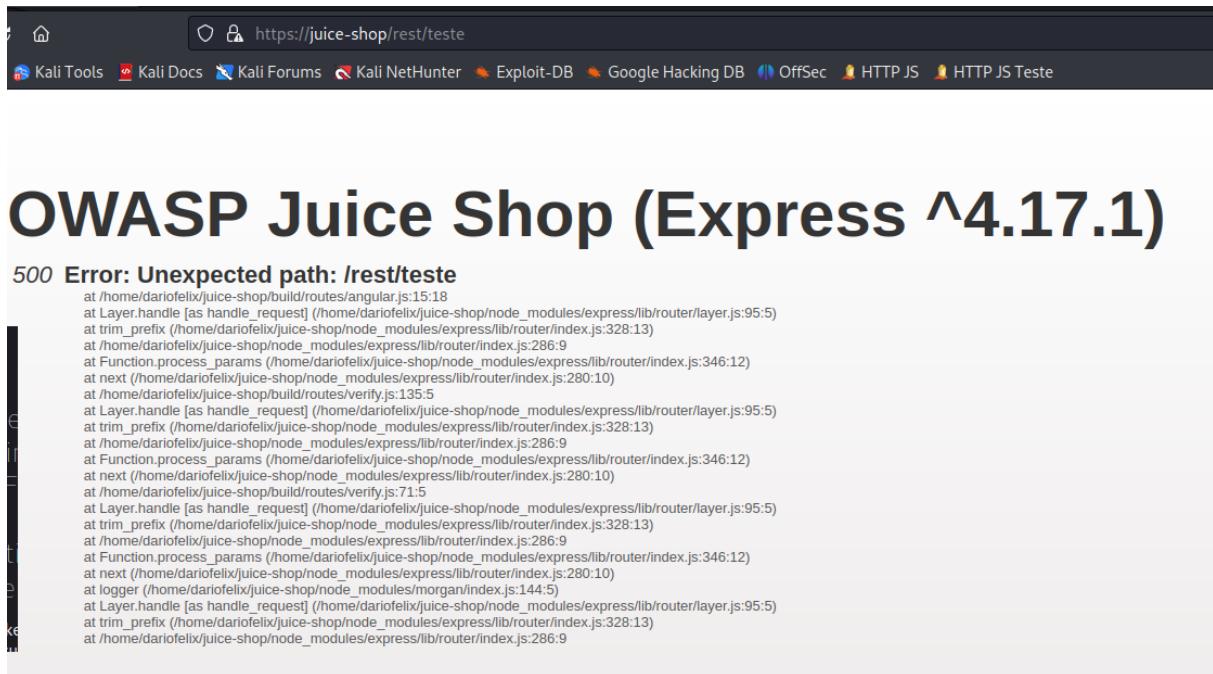


Figura 58: Stack trace do erro provocado

4.9 WSTG-CRYP Testes de criptografia fraca

Esta secção não é aplicável no âmbito deste trabalho uma vez que o *juice-shop* não é desenvolvido sobre *HTTPS*.

4.10 WSTG-BUSL Testes de lógica de negócio

Esta secção não é aplicável no âmbito deste trabalho.

4.11 WSTG-CLNT Testes do lado do cliente

Nesta subsecção iremos abordar alguns testes relacionados com testes do lado do cliente. Devido à extensão da presente secção iremos abordar apenas alguns dos testes.

4.11.1 WSTG-CLNT-04 Testar para Redirecionamento de *URL* do lado do cliente

Este tipo de vulnerabilidade ocorre quando uma aplicação aceita um *url* como input que pode levar outros utilizadores a serem redirecionados para uma página maliciosa.

Este tipo de vulnerabilidade foi encontrada recorrendo ao *ZAP*, como se pode ver na Figura 59.

Figura 59: Redirecionamento encontrado pelo *ZAP*

4.11.2 WSTG-CLNT-09 Testar *Clickjacking*

Este ataque consiste em enganar o utilizador para que este interaja com conteúdo malicioso. Com a ajuda do *ZAP* foi possível detetar alguns casos de *clickjacking*.

The screenshot shows the ZAP interface with a sidebar on the left containing icons for session management, URL rewriting, vulnerable libraries, application errors, CSP notices, cookies, cross-site scripting, cross-domain JavaScript inclusion, private IP disclosure, server version leaks, strict transport security headers, timestamp disclosures, and X-Content-Type-Options headers. A specific alert is selected: "Missing Anti-clickjacking Header (247)". The details panel on the right provides the following information:

- Missing Anti-clickjacking Header**
- URL:** http://192.168.38.245/socket.io/?EIO=4&transport=polling&t=OXB2ZkB&sid=Bv0X2-AzVUbX5RB9AHwq
- Risco:** Medium
- Confiança:** Medium
- Parâmetro:** X-Frame-Options
- Ataque:**
- Evidência:**
- CWE ID:** 1021
- WASC ID:** 15
- Fonte:** Passivo (10020 - Anti-clickjacking Header)
- Input Vector:**
- Descrição:** The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Figura 60: *Clickjacking* encontrado pelo *ZAP*

5 Firewall de Segurança para Aplicações Web (*WAF*)

Terminada a auditoria de segurança efetuada anteriormente, passou-se à ativação da *WAF* já configurada de modo a verificar se esta é capaz de monitorizar, detetar e prevenir alguns dos ataques e problemas já identificados na secção anterior.

Para poder ter maior controlo sobre o funcionamento da *WAF* podemos usar o comando seguinte para verificar as mensagens de erro do servidor *Apache*.

```
$ tail -f /var/log/httpd/error_log
```

O comando seguinte permite a monitorização do funcionamento da *WAF*, uma vez que são apresentados todos os ataques e possíveis ataques.

```
$ tail -f /var/log/httpd/modsec_audit.log
```

5.1 OWASP ZAP

Nesta fase do trabalho optámos por começar por repetir os testes efetuados anteriormente com o *ZAP* para podermos fazer uma comparação direta sobre o número de alertas gerados antes e depois da ativação da *WAF*.

Comparando os resultados apresentados na Figura 8 e na Figura 61 podemos concluir que grande parte dos alertas identificados na primeira fase deixaram de existir, assim, podemos verificar que a *WAF* é capaz de bloquear bastantes ataques. Ainda assim, foram identificados alguns alertas que são importantes analisar, uma vez que podem surgir falsos positivos.

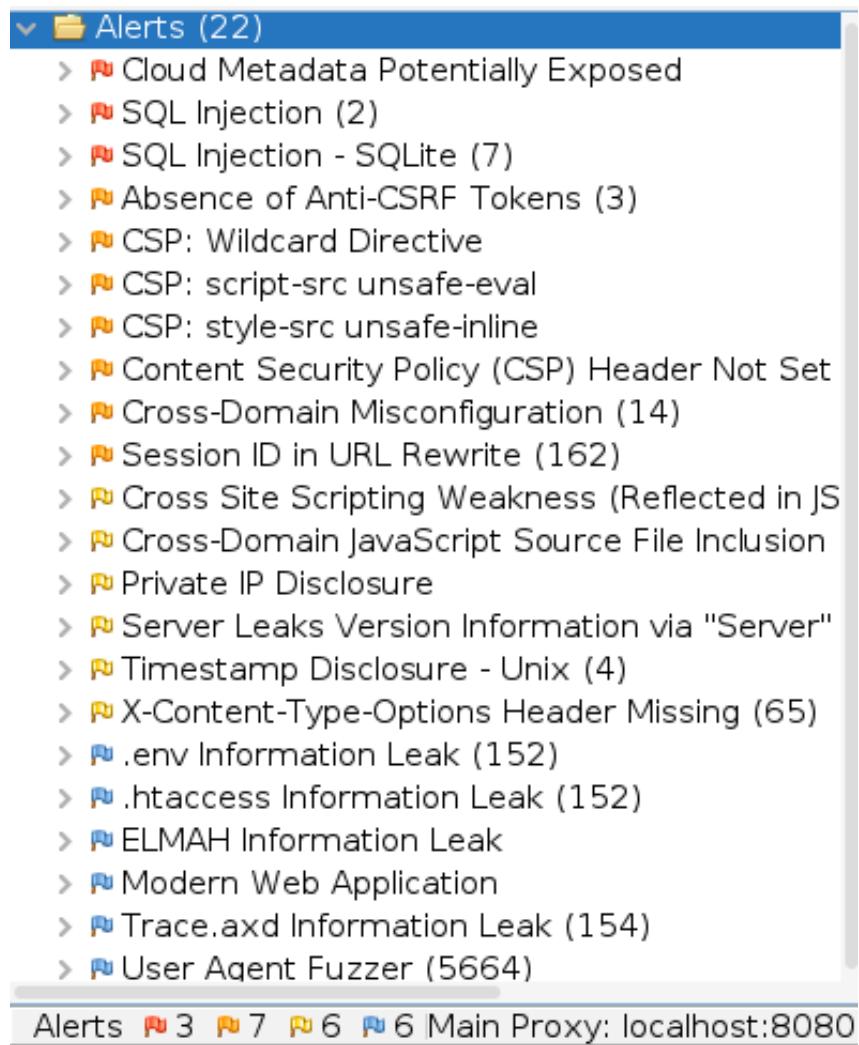


Figura 61: Alertas encontrado pelo *ZAP* após implementação da *WAF*

5.2 WSTG-INFO Recolha de informações

Nesta secção o objetivo passa, tal como o nome indica, pela recolha de informação acerca da aplicação web de modo a poder obter informações que nos serão úteis na exploração de vulnerabilidades. Como esta fase visa a recolha de informação e não faz nenhum ataque à aplicação, todos os testes efetuados na fase anterior podem ser replicados e são obtidos os mesmos resultados. O teste efetuado no ponto *WSTG-INFO-05* é a única exceção à regra, uma vez que na primeira fase o *ZAP* tinha detetado um padrão num dos ficheiros fonte da aplicação e após a aplicação da *WAF* o mesmo não se verificou (tal como se pode verificar na Figura 61).

5.3 WSTG-CONF Testes de gestão da implementação e de configuração

Nesta secção pretendia-se analisar aspetos relacionados com a configuração da aplicação e de que forma isso tem impacto na segurança da aplicação. À semelhança do ponto anterior, neste caso as informações obtidas sobre a aplicação e os testes efetuadas não são possíveis de proteger com recurso à *WAF*, pelo que os testes efetuados obtêm o mesmo resultado que foi apresentado anteriormente.

5.4 WSTG-IDNT Testes de gestão de identidade

Esta secção não é aplicável no âmbito deste trabalho.

5.5 WSTG-ATHN Testes de autenticação

Muitos dos problemas identificados na fase anterior continuam prevalecentes, como a utilização do HTTP ao invés do HTTPS no transporte das credenciais, também continuamos a poder utilizar ferramentas de ataque de *brute force* para forçar a autenticação. Além disso, ainda existem outros problemas

detetados na fase anterior como a política de palavra-pases fracas, as fragilidades nas perguntas de segurança e as vulnerabilidades em lembrar a *password* (ou a sessão) no navegador, que só estão passíveis de serem resolvidos através da modificação do código-fonte.

5.6 WSTG-ATHZ Testes de autorização

Esta secção diz respeito ao acesso a conteúdo privilegiado sem permissões para aceder ao mesmo. Na fase anterior tinha sido encontrada uma vulnerabilidade de *Path Traversal* (*WSTG-ATHZ-01*), no qual foi detetado um ficheiro fora da diretoria de raiz da aplicação. Como podemos ver na Figura 61, nesta nova iteração o mesmo não se verificou.

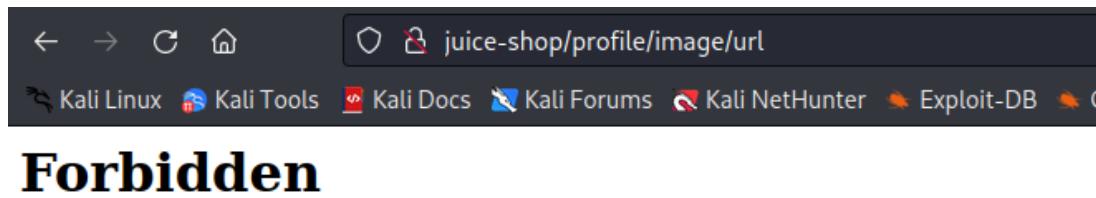
5.7 WSTG-SESS Testes de gestão de sessão

Os alertas incidentes neste tópico diminuíram, ainda assim é normal aparecerem alguns alertas, uma vez que muitas das correções necessárias são ao nível do código-fonte e outras ao nível da tecnologia utilizada nas comunicações entre cliente e servidor (HTTP vs. HTTPS), ver Figura 61 — *Session ID in URL Rewrite; Absence of Anti-CSRF Tokens*.

5.8 WSTG-INPV Testes de validação de *input*

Esta secção pretendia testar as validações de input da *juice-shop*. As principais vulnerabilidades relativas a esta secção encontradas na primeira fase deste trabalho diziam respeito a *Cross Site Scripting* e a *SQL Injection*.

Nesta fase foi repetido o teste efetuado no ponto *WSTG-INPV-01*. Como podemos ver na Figura 62 e na Figura 63, este ataque foi detetado e bloqueado pela *WAF*.



Forbidden

You don't have permission to access /profile/image/url on this server.

Figura 62: Erro ao tentar ataque *Reflected Cross Site Scripting*

```
--ebaf3c49-C-
imageUrl
--ebaf3c49-F-
HTTP/1.1 403 Forbidden
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
--ebaf3c49-H-
Message: Access denied with code 403 (phase 3). Pattern match "'$1\\d{2}1'" at RESPONSE STATUS. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_50_outbound.conf"] [line "53" [id "970901"] [rev "2"] [msg "The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "ERROR"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP CRS/2.2.6"] [tag "PCI 6.5.6"]"]
Message: Warning. Operator GE matched 4 at TX:outbound_anomaly_score. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_60_correlation.conf"] [line "40" [id "981205"] [msg "Outbound Anomaly Score Exceeded (score: 4). The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "INFO"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP CRS/2.2.9"] [tag "PCI 6.5.6"]]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.38.241] ModSecurity: Access denied with code 403 (phase 3). Pattern match "'$1\\\\\\\\\\\\\\\\d{2}1'" at RESPONSE STATUS. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_50_outbound.conf"] [line "53" [id "970901"] [rev "2"] [msg "The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "ERROR"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP CRS/2.2.9"] [tag "PCI 6.5.6"] [hostname "juice-shop"] [uri "/profile/image/url"] [unique_id "ZG-01327cruTPtqA7eckQAAo="]]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 192.168.38.241] ModSecurity: Warning. Operator GE matched 4 at TX:outbound_anomaly_score. [file "/etc/httpd/modsecurity.d/activated_rules/modsec_unity_crs_60_correlation.conf"] [line "40" [id "981205"] [msg "Outbound Anomaly Score Exceeded (score: 4). The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "INFO"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP CRS/2.2.9"] [tag "PCI 6.5.6"] [hostname "juice-shop"] [uri "/profile/image/url"] [unique_id "ZG-01327cruTPtqA7eckQAAo="]]
Action: Intercepted (phase 3)
Apache-Handler: proxy-server
Stopwatch: 1685048971042918 19233 (- -)
Stopwatch: 1685048971042918 19233 combined=86888 p1=265, p2=8189, p3=56, p4=0, p5=97, sr=70, sw=1, l=0, gc=0
Processor: ModSecurity for Apache/2.9.2 (http://www.modsecurity.org/); OWASP CRS/2.2.9.
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
Engine: "ENABLED"
--ebaf3c49-Z-
```

Figura 63: Bloqueio do ataque *Reflected Cross Site Scripting*

Ainda assim, o ZAP detetou 2 ataques de *Reflected Cross Site Scripting*. Observando a Figura 64 podemos verificar as diretórias nas quais foi detetada a possibilidade de ocorrência deste ataque. Contudo, ao tentarmos aceder a cada uma delas podemos constatar que a *WAF* bloqueia o seu acesso, pelo que não é possível efetuar o ataque descrito.

The screenshot shows the ZAP interface. On the left, there's a tree view of 'Alerts' under 'Contexts'. One alert is selected, highlighted in blue, which corresponds to the details shown on the right. The details panel for this alert is titled 'Cross Site Scripting Weakness (Reflected in JSON Response)'. It provides information such as the URL (`http://juice-shop/api/Users/`), Risk (Low), Confidence (Low), Parameter (email), Attack (script), Evidence, CWE ID (79), WASC ID (8), Source (Active), Input Vector (JSON), and Description (A XSS attack was reflected in a JSON response, this might leave content consumers vulnerable to attack if they don't appropriately handle the data (response)). Below the description, there's an 'Other Info:' section stating 'Raised with LOW confidence as the Content-Type is not HTML'. Under 'Solution:', it says 'Phase: Architecture and Design' and provides a note: 'Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding'.

Figura 64: Ataques *Reflected Cross Site Scripting* detetados pelo ZAP

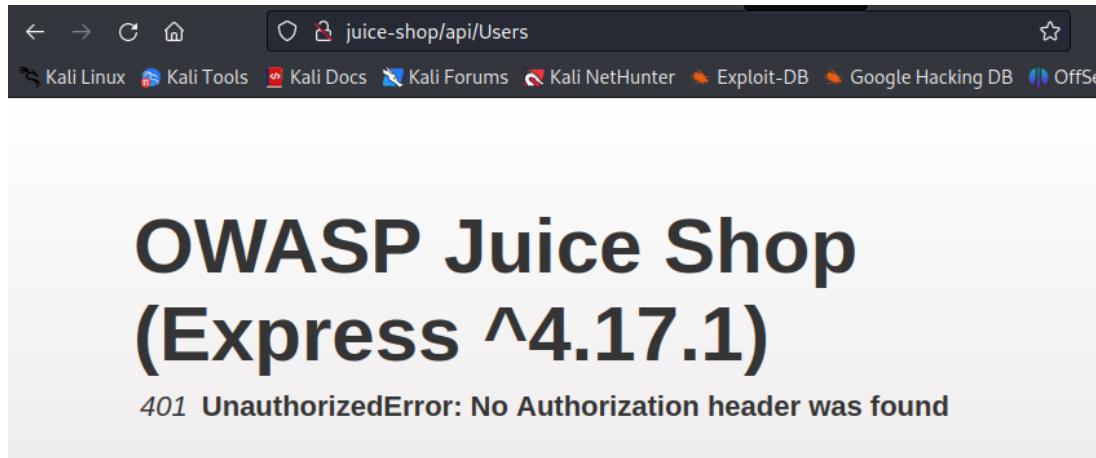


Figura 65: *UnauthorizedError* ao aceder à diretoria

The screenshot shows a web browser window with the address bar set to `http://juice-shop/rest/user/login`. The page content displays the title 'Forbidden' and a message: 'You don't have permission to access /rest/user/login on this server. Additionally, a 500 Internal Server Error error was encountered while trying to use an ErrorDocument to handle the request.'

Figura 66: Diretoria Proibida

Relativamente ao ponto *WSTG-INPV-02*, sobre *SQL Injection*, podemos ver na Figura 67 e na Figura 68, que a *WAF* é capaz de detetar e bloquear este tipo de ataques.

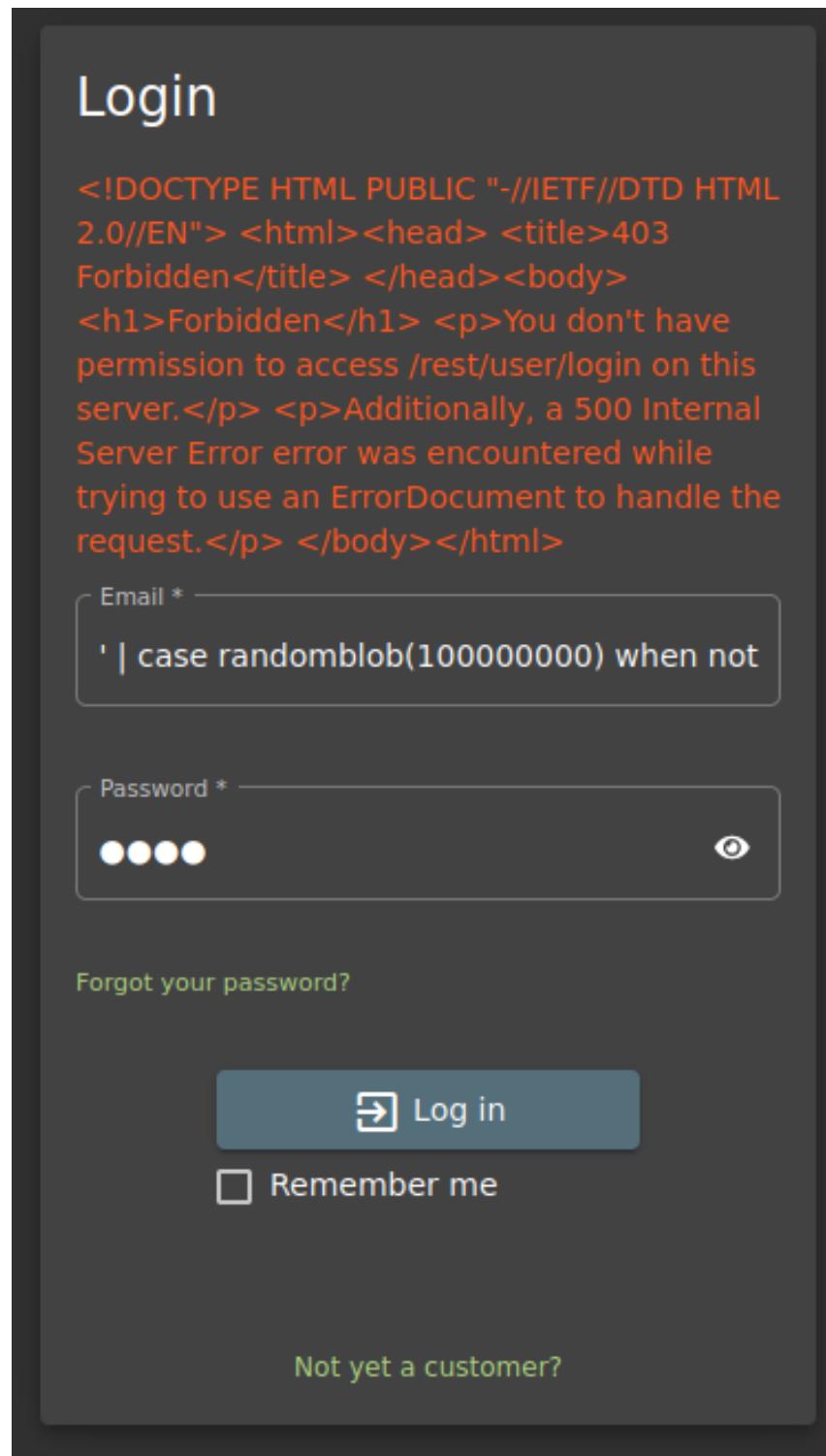


Figura 67: Ataque *SQL Injection*

```
--57155956-C--
{"email":''} | case randomblob(100000000) when not null then \\"\\\"\\\" else \\"\\\"\\\" end | '","password":"kalikalai"
--57155956-F-- Service
HTTP/1.1 403 Forbidden
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

--57155956-H--
Message: Access denied with code 403 (phase 3). Pattern match ""5\|d{2}$" at RESPONSE_STATUS. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_50_outbound.conf"] [line "53"] [id "970901"] [rev "2"] [msg "The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "ERROR"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP_TOP_10/A6"] [tag "PCI/6.5.6"]
Message: Warning. Operator GE matched 4 at TX:outbound_anomaly_score. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_60_correlation.conf"] [line "40"] [id "981205"] [msg "Outbound Anomaly Score Exceeded (score: 4): The application is not available"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 10.20.30.1] ModSecurity: Access denied with code 403 (phase 3). Pattern match ""5\\\\\\\\\\\\\\\\d{2}$" at RESPONSE_STATUS. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_50_outbound.conf"] [line "53"] [id "970901"] [rev "2"] [msg "The application is not available"] [data "Matched Data: 500 found within RESPONSE_STATUS: 500" [severity "ERROR"] [ver "OWASP CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "WASCTC/WASC-13"] [tag "OWASP_TOP_10/A6"] [tag "PCI/6.5.6"] [hostname "juice-shop"] [uri "/rest/user/login"] [unique_id "ZG-BaNwi-j0czrKKEzRzdgAAAAAs"]
Apache-Error: [file "apache2_util.c"] [line 271] [level 3] [client 10.20.30.1] ModSecurity: Warning. Operator GE matched 4 at TX:outbound_anomaly_score. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_60_correlation.conf"] [line "40"] [id "981205"] [msg "Outbound Anomaly Score Exceeded (score: 4): The application is not available"] [hostname "juice-shop"] [uri "/rest/user/login"] [unique_id "ZG-BaNwi-j0czrKKEzRzdgAAAAAs"]
Action: Intercepted (phase 3)
Apache-Handler: proxy-server
Stopwatch: 1685045608580638 23364 (- - -)
Stopwatch2: 1685045608580638 23364; combined=1910, p1=194, p2=1632, p3=35, p4=0, p5=49, sr=63, sw=0, l=0, gc=0
Producer: ModSecurity for Apache/2.9.2 (http://www.modsecurity.org/); OWASP CRS/2.2.9.
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
Engine-Mode: "ENABLED"
```

Figura 68: Logs do *ModSecurity* de um bloqueio do ataque *SQL Injection*

Contudo, ao replicar o teste de *SQL Injection* efetuado na primeira fase pudemos constatar que a *WAF* não foi capaz de detetar o ataque, pelo que foi possível efetuar login na aplicação. Acreditamos que isto possa ser um problema com o *ModSecurity*, uma vez que foi testado também um dos payloads (' or 1=1 --) que segundo o ficheiro de configuração das regras de *SQL* do *ModSecurity* é detetado pelo mesmo e também foi possível efetuar login.

5.9 WSTG-ERRH Testes de tratamento de erros

De facto, e no que diz respeito a este tópico, o OWASP ZAP reportou falsos positivos, ver Figura 69 e Figura 70, aliás o próprio *software* classifica este alerta como grau de confiança baixo. Ao confrontar com o mesmo teste realizado na fase anterior e que estão documentados na Figura 57 e Figura 58, verificamos que o *WAF* corrigiu esta vulnerabilidade.

The screenshot shows the OWASP ZAP interface. On the left, the 'Contexts' panel shows 'Default Context' and 'Sites'. Under 'Sites', 'http://juice-shop' is selected. The main pane displays an 'HTTP/1.1 403 Forbidden' response with the following content:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /rest/teste<br>on this server.</p>
<p>Additionally, a 500 Internal Server Error<br>error was encountered while trying to use an ErrorDocument to handle the request.</p>
```

Below this, the 'Information Disclosure - Debug Error Messages' alert is detailed:

- URL:** http://juice-shop/rest/teste
- Risk:** Low
- Confidence:** Medium
- Parameter:**
- Attack:**
- Evidence:** Internal Server Error
- CWE ID:** 200
- WASC ID:** 13
- Source:** Passive (10023 - Information Disclosure - Debug Error Messages)
- Input Vector:**
- Description:** The response appeared to contain common error messages returned by platforms such as ASP.NET, and Web-servers such as IIS and Apache. You can configure the list of common debug messages.
- Other Info:**

The bottom of the alert shows a list of URLs related to the error message, all of which are GET requests to various socket endpoints.

Figura 69: Falso positivo no alerta “Information Disclosure - Debug Error Messages” detetado pelo OWASP ZAP

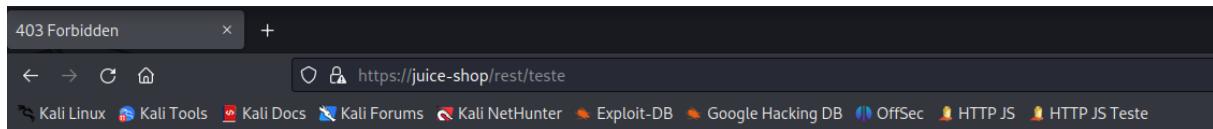


Figura 70: Proteção concebida pelo WAF

5.10 WSTG-CRYP Testes de criptografia fraca

Esta secção não é aplicável no âmbito deste trabalho.

5.11 WSTG-BUSL Testes de lógica de negócio

Esta secção não é aplicável no âmbito deste trabalho.

5.12 WSTG-CLNT Testes do lado do cliente

Neste ponto pretendia-se efetuar testes do lado do cliente. Na fase anterior foram abordados os problemas de redirecionamento de *URL* e de *clickjacking*, como podemos ver na Figura 61 nenhum destes ataques foi detetado após a implementação da WAF.

6 Conclusão

Ao longo deste trabalho, seguimos as diretrizes do WSTG para realizar testes de segurança à aplicação *web* Juice Shop. Esta experiência permitiu-nos compreender melhor os pontos vulneráveis mais comuns em aplicações *web* e como testá-los, além de nos fornecer uma base sólida para continuarmos a aprimorar as nossas habilidades na área da segurança da informação.

Referências

- [1] J. Granjal, *Segurança Prática em Sistemas e Redes com Linux*. FCA, 2017.
- [2] OWASP, “Juice shop,” 2023. [Online]. Available: <https://owasp.org/www-project-juice-shop/>
- [3] ——, “Open worldwide application security project,” 2023. [Online]. Available: <https://owasp.org/>
- [4] ——, “Zed attack proxy,” 2023. [Online]. Available: <https://www.zaproxy.org/>
- [5] ——, “Web security testing guide,” 2023. [Online]. Available: <https://owasp.org/www-project-web-security-testing-guide/stable/>
- [6] ——, “Modsecurity core rule set,” 2023. [Online]. Available: <https://owasp.org/www-project-modsecurity-core-rule-set/>
- [7] Kali, “Prebuilt virtual machines,” 2023. [Online]. Available: <https://www.kali.org/get-kali/#kali-virtual-machines>
- [8] ——, “Import pre-made kali vmware vm,” 2023. [Online]. Available: <https://www.kali.org/docs/virtualization/import-premade-vmware/>
- [9] B. Wego, “Install a version of gcc with supports for cxxabi_1.3.8,” 2019. [Online]. Available: <https://superuser.com/questions/1432321/importerror-lib64-libstdc-so-6-version-cxxabi-1-3-8-not-found>
- [10] OWASP, “Github juice shop,” 2023. [Online]. Available: <https://github.com/juice-shop/juice-shop>
- [11] M. Papiernik, “How to use apache as a reverse proxy with mod_proxy on centos 7,” 2017. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod_proxy-on-centos-7
- [12] T. NGUYEN, “Httpd apache2 issue failed to make connection to backend on centos,” 2019. [Online]. Available: <https://blog.duchinh.net/fix-httdp-issue-failed-make-connection-backend/>
- [13] OWASP, “Owasp zap - getting started,” 2023. [Online]. Available: <https://www.zaproxy.org/getting-started/>
- [14] ——, “Owasp zap - active scan,” 2023. [Online]. Available: <https://www.zaproxy.org/docs/desktop/start/features/ascan/>
- [15] R. Chandel, “Banner grabbing,” 2020. [Online]. Available: <https://www.hackingarticles.in/multiple-ways-to-banner-grabbing/>
- [16] “Download bypass: Poison null byte,” 2021. [Online]. Available: <https://wiki.zacheller.dev/web-app-pentest/upload-download/error-only-.md-and-.pdf-files-are-allowed>