



FACULDADE DE  
CIÊNCIAS E TECNOLOGIA  
UNIVERSIDADE D  
**COIMBRA**

Departamento Engenharia Informática

STI 2021/2022

Relatório Trabalho Prático N°3:

Trabalho realizado por:  
João Calhau 2016255704  
Tatiana Simões 2018285812

## 1. Introdução

Os principais objetivos propostos para este trabalho são explorar as metodologias de Web Application Security Testing e Web Application Security Firewall.

No primeiro cenário começamos por realizamos os 6 testes pedidos com o OWASP ZAP no “JuiceShop” website:

- a. Executar um automated scan ao website.
- b. Executar um active scan ao website (explorar as políticas mais eficazes).
- c. Gerenciar add-on necessários para melhorar o teste e maximizar a identificação de ameaças.
- d. Executar um Fuzz attack ao formulário de login.
- e. Executar um teste de manual penetration para explorar as ameaças que estão logged in.
- f. Configurar o active scan do OWASP ZAP para explorar a área de autenticação.

De seguida apresentamos os resultados seguindo as **WSTG** guidelines.

## 2. Estrutura cenário 1



Figura 1 - cenário 1: cliente têm comunicações diretas com o web server

Para este cenário usamos duas VMs, uma o web server com a JuiceShop e outra o kali linux.

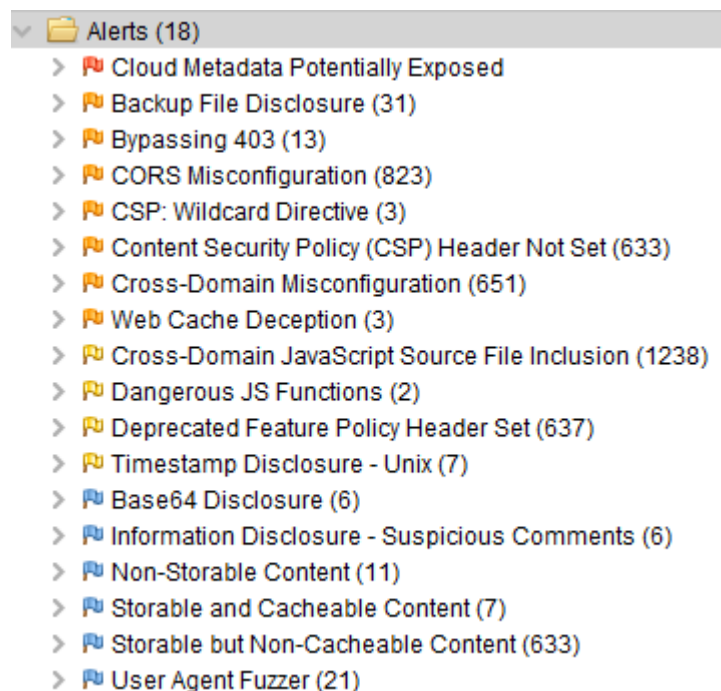
Começamos por realizar o automated scan ao website, seguido por um active scan onde fomos alterando as policies e adicionando add-ons de modo a encontrar os mais eficazes.

Add-ons
Advanced SQLInjection Scanner
Image Location and Privacy Scanner
Port Scanner
Active scanner rules (alpha)
Passive scanner rules (alpha)

Procedemos com um Fuzz attack ao formulário de login com payloads sql e uma manual penetration a várias funcionalidade.

Por último configuramos o OWASP ZAP active scan para explorar a área de autenticação criando um utilizador com uma sql injection no username.

Automated scan sem e com addons:



### 3. Web application security testing

#### 3.1. Information Gathering

### 3.1.1. Fingerprint Web Server

Consiste em determinar a versão e o tipo do servidor web a correr possibilitando a descoberta de possíveis vulnerabilidades.

```
(kali㉿kali)-[~]  
$ curl -s -I 192.168.1.80:3000  
HTTP/1.1 200 OK  
Access-Control-Allow-Origin: *  
X-Content-Type-Options: nosniff  
X-Frame-Options: SAMEORIGIN  
Feature-Policy: payment 'self'  
Accept-Ranges: bytes  
Cache-Control: public, max-age=0  
Last-Modified: Mon, 30 May 2022 03:41:14 GMT  
ETag: W/"7c3-181130db535"  
Content-Type: text/html; charset=UTF-8  
Content-Length: 1987  
Vary: Accept-Encoding  
Date: Mon, 30 May 2022 12:45:03 GMT  
Connection: keep-alive  
Keep-Alive: timeout=5
```

Figura 2- Tentativa de banner grabbing

Na resposta não conseguimos determinar a versão e o tipo do servidor web.

```
(kali㉿kali)-[~]  
$ nc 192.168.1.80 3000  
GET / SANTA CLAUS/1.1  
HTTP/1.1 400 Bad Request  
Connection: close
```

Figura 3- malformed request

Ao realizar um malformed request apenas obtemos a resposta Bad Request e que não ocorreu a ligação.

### 3.1.2. Review Webserver Metafiles for Information

```
(kali㉿kali)-[~]
$ wget http://192.168.1.80:3000/robots.txt
--2022-05-30 00:09:04-- http://192.168.1.80:3000/robots.txt
Connecting to 192.168.1.80:3000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28 [text/plain]
Saving to: 'robots.txt'

robots.txt      100%[=====>]      28 --.-KB/s   in 0s

2022-05-30 00:09:04 (1.11 MB/s) - 'robots.txt' saved [28/28]

(kali㉿kali)-[~]
$ cat robots.txt
User-agent: *
Disallow: /ftp
```

#### Leakage

Figura 4- get ao ficheiro robots.txt(existente na diretoria root do servidor web JuiceShop) e o seu conteúdo

Neste teste podemos observar que spiders/robots/crawlers(User-agent: \*) são fortemente encorajados a evitar aceder aos recursos da directoria /ftp.

### 3.1.3. Enumerate Applications on Webserver

```
(kali㉿kali)-[~]
$ nmap -Pn -sT -sV -p0-65535 192.168.1.80
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-30 09:17 EDT
Stats: 0:00:20 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 95.35% done; ETC: 09:17 (0:00:00 remaining)
Nmap scan report for debian.lan (192.168.1.80)
Host is up (0.0041s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.53 ((Debian))
443/tcp   open  http     Apache httpd 2.4.53
3000/tcp   open  ppp?
1 service unrecognized despite returning data. If you know the service/version
mit.cgi?new-service :
SF-Port3000-TCP:V=7.92%I=7%D=5/30%Time=6294C3E1%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,962,"HTTP/1.1\x20200\x200K\r\nAccess-Control-Allow-Origin:\x2
```

Figura 5- Teste à existência de web applications em non-standard ports

Este comando procura todos os ports abertos no ip 192.168.1.80 e tenta determinar quais serviços estão ligados a eles .

No port 22 observamos um serviço ssh, e nos ports 80 e 443 serviços http.

No port 3000 esperávamos ver também um serviço http(uma vez que é onde o servidor se encontra em execução) mas o serviço não é reconhecido.

```
(kali@kali)-[~]
$ host -t ns 192.168.1.80:3000
Host 192.168.1.80:3000 not found: 3(NXDOMAIN)
```

Figura 6- Teste à existência de virtual hosts

Não encontramos virtual hosts associados ao ip 192.168.1.80

### 3.1.4. Review Webpage Content for Information Leakage

Usando o ZAP obtivemos os seguintes dados nos quais podemos observar que o site tem comentários que expõem informação, obtendo o alerta “Information Disclosure - Suspicious Comments”:

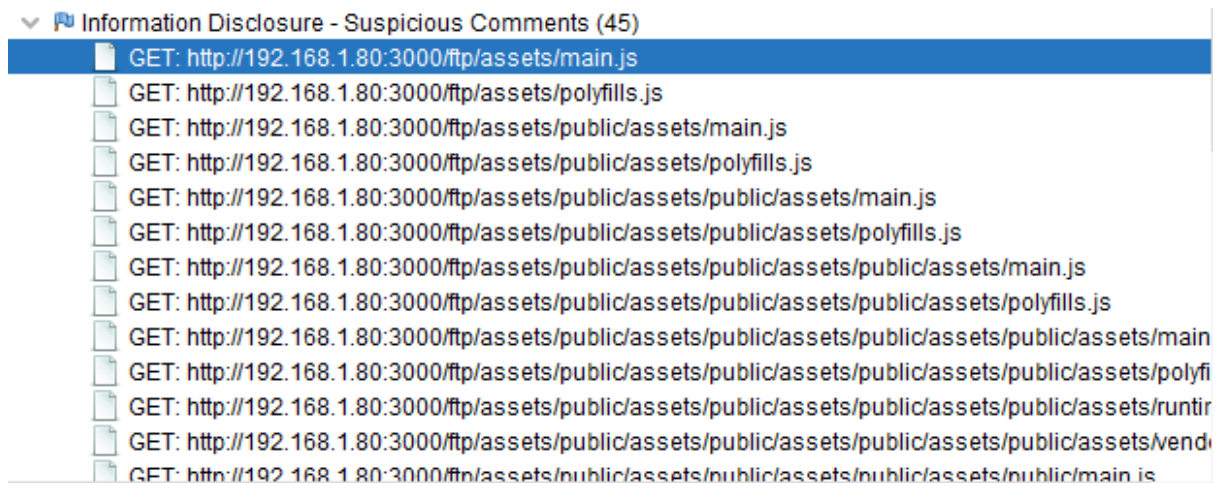


Figura 7 - Alertas de comentários suspeitos

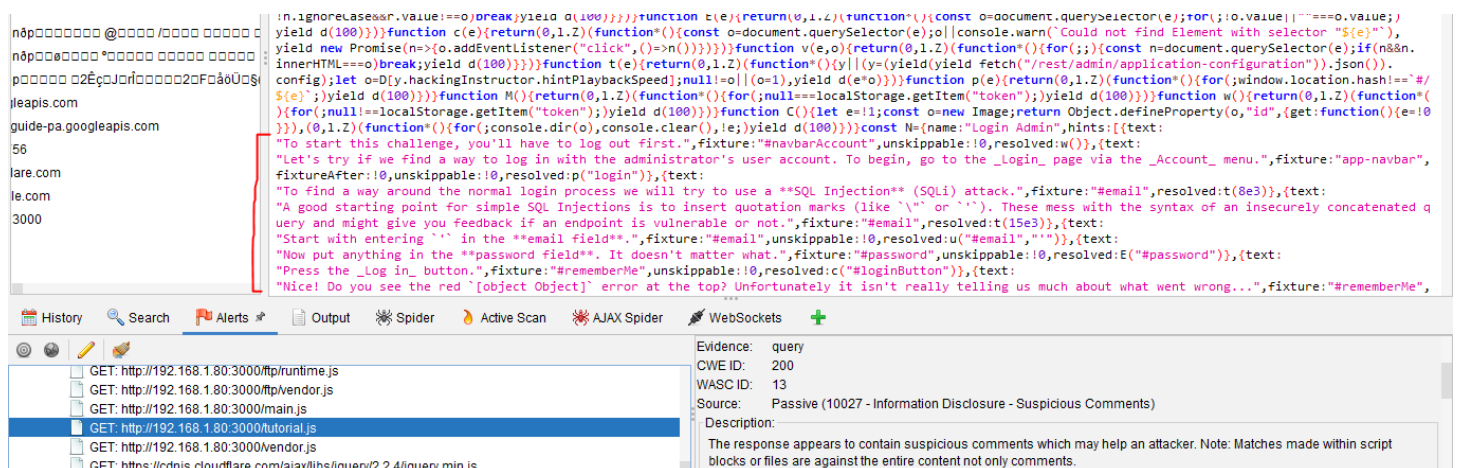
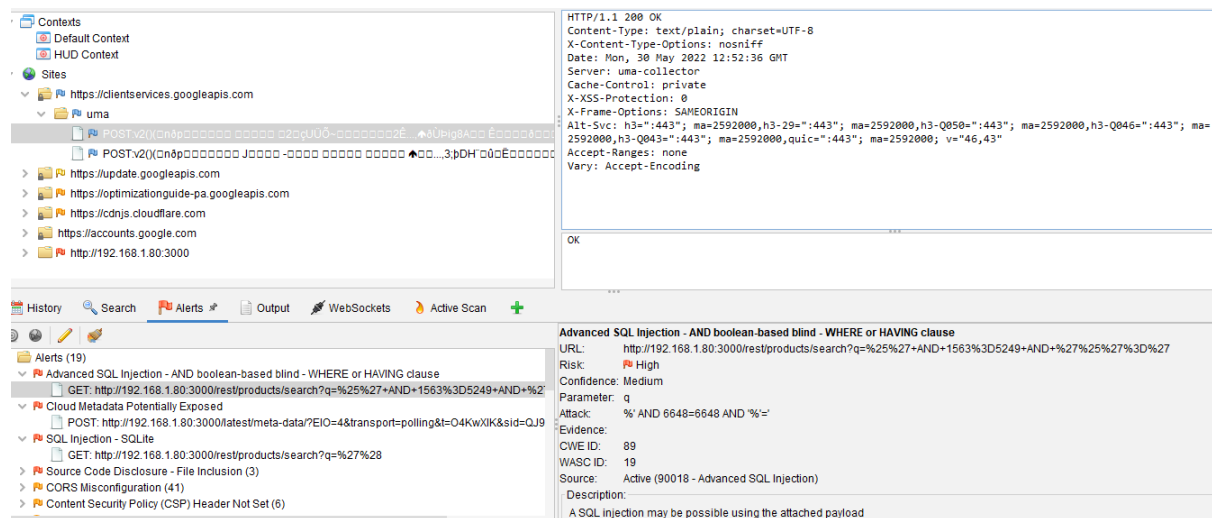


Figura 8 - Exemplo de comentário com information leakage

### 3.1.5. Identify Application Entry Points

Ao inserir `%' AND 6648=6648 AND '%='` na barra de pesquisa através do ZAP com active scan, o comando foi aceite dado que logical AND é aceite pelo site.



É possível também ver o sessionID no URL

```
GET http://192.168.1.80:3000/socket.io/?EIO=4&transport=polling&t=O4K-6pE&sid=TXNid1HEM4AJhyD6AB9T
HTTP/1.1
Host: 192.168.1.80:3000
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"
Accept: */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.61 Safari/537.36
sec-ch-ua-platform: "Windows"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://192.168.1.80:3000/
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: language=en; welcomebanner_status=dismiss
```

Figura 9 - sessionID no URL

### 3.1.6. Map Execution Paths Through Application

Para descobrir os paths possíveis de uma forma automática, usamos o spider e guardamos os resultados no ficheiro com o nome `site_tree.txt`.

### 3.1.7. Fingerprint Web Application

Na maior parte dos nossos requests observamos que existe uma cookie `io`, definida automaticamente, chamada `socket.io` que é uma biblioteca de javascript para web applications em real time.



```
POST http://192.168.1.80:3000/socket.io/?EIO=4&transport=polling&t=04KwXIK&sid=QJ9A_49WBr8_wNSNABwi
HTTP/1.1
Host: 192.168.1.80:3000
Connection: keep-alive
Content-Length: 2
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"
Accept: */*
Content-type: text/plain;charset=UTF-8
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
sec-ch-ua-platform: "Windows"
Origin: https://192.168.1.80:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://192.168.1.80:3000/
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Figura 10 - cookie "socket.io"

## 3.2. Configuration and Deployment Management Testing

### 3.2.1. Test Network Infrastructure Configuration

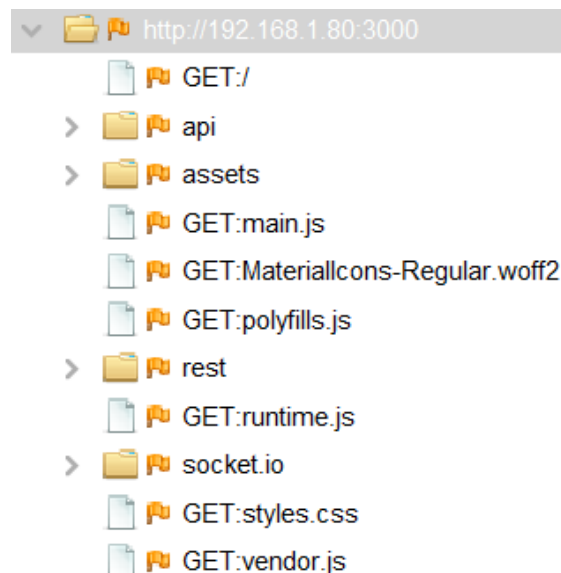


Figura 11- Configuração da infraestrutura

### 3.2.2. Test File Extensions Handling for Sensitive Information



## Site Alerts



### Image Exposes Location or Privacy Data (2)

http://192.168.1.80:3000/assets/public/images/uploads/magn(et)ificent!-1571

http://192.168.1.80:3000/assets/public/images/uploads/my-rare-collectors-item!-

%5B%CC%B2%CC%85\$%CC%B2%CC%85(%CC%B2%CC%85-

%CD%A1%C2%B0-%CD%9C%CA%96-

%CD%A1%C2%B0%CC%B2%CC%85)%CC%B2%CC%85\$%CC%B2%CC%85%5

1572603645543.jpg

Figura 12 - Vulnerabilidades nas imagens

### 3.2.3. Enumerate Infrastructure and Application Admin Interfaces

Um dos testes que realizamos foi um Fuzz Attack ao formulário de login, onde selecionamos a opção SQL injection como payload. Os resultados estão guardados no report *Fuzz\_Attack-ZAP-Report.pdf* e no ficheiro *Fuzz\_attack.csv*.

Nos nossos resultados conseguimos observar que com o username “a' or 1=1;--” e com qualquer password era possível entrar na conta do administrador e obter informações privadas como por exemplo o seu email(o código 200 ok significa que o ataque foi bem sucedido).

301	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	193	714	85	363	838	[a' or 1=1; --, ' ; exec master..xp_cmdshell 'ping 10.10.1.2'--]	
302	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	184	714	79	363	838	[a' or 1=1; --, create user name identified by 'pass123']	
303	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	164	715	130	363	838	[a' or 1=1; --, create user name identified by pass123 temporary tablespa	
304	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	227	714	61	363	838	[a' or 1=1; --, ' ; drop table temp --]	
305	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	245	714	75	363	838	[a' or 1=1; --, exec sp_addlogin 'name', 'password']	
306	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	224	714	83	363	838	[a' or 1=1; --, exec sp_addsrvrolemember 'name', 'sysadmin']	
307	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	238	715	134	363	838	[a' or 1=1; --, insert into mysql.user (user, host, password) values ('name	
308	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	241	714	85	363	838	[a' or 1=1; --, grant connect to name; grant resource to name;]	
309	Fuzzed	Mon May	POST	http://192.168.1.80:3000/rest/user/login	200 OK	184	715	227	363	838	[a' or 1=1; --, insert into users(login, password, level) values( char(0x70)	

### 3.2.4. Test HTTP Methods

- > 📁 Insecure HTTP Method - PATCH (104)
- > 📁 Insecure HTTP Method - PUT (104)
- > 📁 Insecure HTTP Method - COPY (104)
- > 📁 Insecure HTTP Method - LOCK (104)
- > 📁 Insecure HTTP Method - MKCOL (104)
- > 📁 Insecure HTTP Method - MOVE (104)
- > 📁 Insecure HTTP Method - PROPFIND (104)
- > 📁 Insecure HTTP Method - PROPPATCH (104)
- > 📁 Insecure HTTP Method - UNLOCK (104)

Figura 13 - métodos HTTP

### 3.2.5. Test RIA Cross Domain Policy

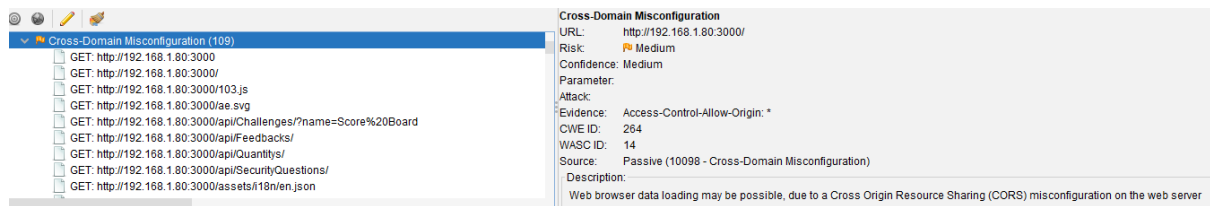


Figura 14 - Cross-domain Misconfiguration

### 3.2.6. Test File Permission

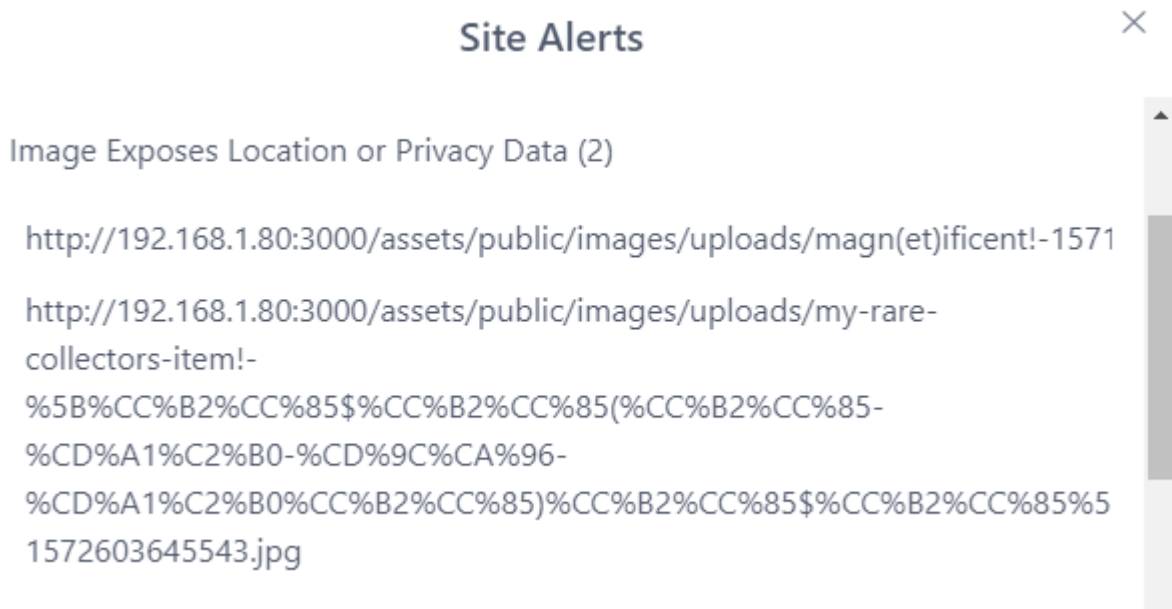


Figura 15 - Vulnerabilidades nas imagens

## 3.3. Identity Management Testing

### 3.3.1. Test Role Definitions

Os utilizadores podem ser clientes da JuiceShop ao registrarem-se na página de login. Podem também aceder à página do admin, mesmo não o sendo nem estando na sua conta.

### 3.3.2. Test User Registration Process

No registo é pedido um email, uma password e uma questão de segurança. Não ocorre qualquer verificação do email para entrar na conta, podendo as informações de identidade serem facilmente falsificadas. Qualquer password desde que tenha um tamanho entre 5-40 é aceite.

### 3.3.3. Testing for Account Enumeration and Guessable User Account

Tentamos manualmente entrar em contas de clientes e do admin com os usernames e passwords mais comuns mas tivemos sempre o resultado “invalid email or password”.

## 3.4. Authentication Testing

### 3.4.1. Testing for Default Credentials

Testando as palavras mais usuais para o username: "admin", "administrator", "root", "system", "guest", "operator", or "super" e para a password: "password", "pass123", "12345" e "asd12" não foi possível entrar na conta

### **3.4.2. Testing for Weak Lock Out Mechanism**

Após inúmeras tentativas de login falhadas, a submissão não estava bloqueada, o que quer dizer que está vulnerável a ataques de brute force.

### **3.4.3. Testing for Bypassing Authentication Schema**

Como é possível fazer login como administrador através de uma SQL injection, é possível aceder à página de utilizador sem saber qualquer credencial.

### **3.4.4. Testing for Weak Password Policy**

Requer no mínimo 5 caracteres mas aceita 12345, asd12, etc

### **3.4.5. Testing for Weak Security Question Answer**

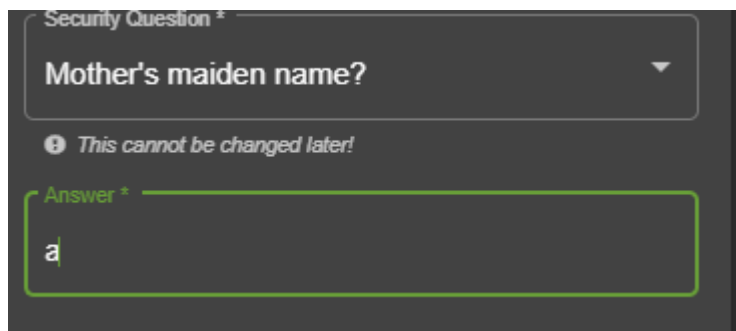


Figura 16 - Respostas fracas



### **3.4.6. Testing for Weak Password Change or Reset Functionalities**

Numa testagem manual foi possível mudar a palavra pass repetindo a mesma.

## **3.5. Authorization Testing**

### **3.5.1. Testing Directory Traversal File Include**

Informação sensível nos URLs e Cookie Slack Detections.

- >  Cookie Slack Detector (131)
- >  Image Exposes Location or Privacy Data (2)

### **3.5.2. Testing for Bypassing Authorization Schema**

Como visto anteriormente é possível aceder à conta de administrador sem qualquer credencial

### **3.5.3. Testing for Privilege Escalation**

Não encontramos evidências de um utilizador poder obter privilégios de administrador através da sua conta.

### **3.5.4. Testing for Insecure Direct Object References**

Ao correr o ZAP em modo de ataque obtivemos as seguintes mensagens no site da juiceshop:

- "View another user's shopping basket";
- "Post a product review as another user or edit any user's existing review".

Confirmamos assim a possibilidade de ataques de cross site scripting, que permitem aos utilizadores aceder a informações que não são suas, como reviews e o shopping basket e a edição deste.








### 3.6. Session Management Testing

Como referido anteriormente é possível obter o sessionID através do URL.








### 3.7. Input Validation Testing

Para detectar falhas no website começamos por realizar vários active scans mudando as policies.

Desta forma encontrámos vários erros nomeadamente de risco alto:

- >  Advanced SQL Injection - AND boolean-based blind - WHERE or HAVING clause (6)
- >  Advanced SQL Injection - Microsoft SQL Server/Sybase time-based blind
- >  Cloud Metadata Potentially Exposed (12)
- >  SQL Injection (3)
- >  SQL Injection - MySQL
- >  SQL Injection - SQLite (41)
- >  Source Code Disclosure - File Inclusion (7)

risco médio:

- >  CORS Misconfiguration (133)
- >  Content Security Policy (CSP) Header Not Set (205)
- >  Cross-Domain Misconfiguration (109)
- >  HTTP Only Site (52)
- >  Insecure HTTP Method - PATCH (337)
- >  Insecure HTTP Method - PUT (336)
- >  Missing Anti-clickjacking Header (201)
- >  Session ID in URL Rewrite (797)
- >  Vulnerable JS Library
- >  Web Cache Deception (7)

risco baixo:

- >  CORS Misconfiguration (133)
- >  Content Security Policy (CSP) Header Not Set (205)
- >  Cross-Domain Misconfiguration (109)
- >  HTTP Only Site (52)
- >  Insecure HTTP Method - PATCH (343)
- >  Insecure HTTP Method - PUT (343)
- >  Missing Anti-clickjacking Header (201)
- >  Session ID in URL Rewrite (797)
- >  Vulnerable JS Library
- >  Web Cache Deception (7)
- >  Cross-Domain JavaScript Source File Inclusion (6)
- >  Private IP Disclosure
- >  Timestamp Disclosure - Unix (22)
- >  X-Content-Type-Options Header Missing (798)

informativos:

- > 📄 Cookie Slack Detector (145)
- > 📄 Image Exposes Location or Privacy Data (2)
- > 📄 Information Disclosure - Suspicious Comments (5)
- > 📄 Insecure HTTP Method - COPY (104)
- > 📄 Insecure HTTP Method - LOCK (104)
- > 📄 Insecure HTTP Method - MKCOL (104)
- > 📄 Insecure HTTP Method - MOVE (104)
- > 📄 Insecure HTTP Method - PROPFIND (104)
- > 📄 Insecure HTTP Method - PROPPATCH (104)
- > 📄 Insecure HTTP Method - UNLOCK (104)
- > 📄 Re-examine Cache-control Directives (2)
- > 📄 User Agent Fuzzer (203)

Detalhes no ficheiro active\_scan.html.

### 3.8. Business Logic Testing

Dado que não temos informações concretas sobre o negócio/empresa, não é aplicável ao nosso projeto.

### 3.9. Client Side Testing

É possível fazer data loading no browser devido ao CORS.

**CORS Misconfiguration**

URL: <http://192.168.1.80:3000/rest/admin/application-configuration>

Risk: 📄 Medium

Confidence: High

Parameter:

Attack: Origin: <http://9FAWNud.com>

Evidence: Access-Control-Allow-Origin: \*

CWE ID: 942

WASC ID: 14

Source: Active (40040 - CORS Header)

Description:

This CORS misconfiguration could allow an attacker to perform AJAX queries to the vulnerable website from a malicious page loaded by the victim's user agent.

In order to perform authenticated AJAX queries, the server must specify the header "Access-Control-Allow-Credentials: true" and the

Figura 16 - Exemplo de vulnerabilidade devido ao CORS

## 4. Referências:

<https://github.com/OWASP/wstg/tree/master/document>