FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Trabalho Prático 3

Segurança em Tecnologias da Informação

Mestrado em Engenharia Informática

2022/2023

David Caetano - uc2018283431@student.uc.pt
Tomás Ventura - uc2018279147@student.uc.pt

1) Introdução

Com a finalidade de localizar e mitigar falhas de segurança na Web-App Juicy-Shop, usamos o guia WSTG versão 4.2 para todos os testes.

O OWASP ZAP foi usado como Proxy entre o firefox e o apache, para a captura de pacotes HTTP e a realização de testes manuais/automatizados.

Os testes realizados foram:

- Scan Automatizado
- Scan Ativo (Força: Alta e Threshhold de alerta: Baixo)
- Fuzz Attack ao login Form
- Teste Manuais por envio de pacotes manipulados e interação com a Web Page do Juicy-shop

Os add-ons relevantes para os testes que foram usados no OWASP ZAP foram:

- Directory List v2.3
- Directory List v2.3 LC
- FuzzDB Files
- FuzzDB Offensive
- SVN Digger Files
- Wappalyzer • Technology
- Windows WebDrivers
- Active scanner rues (beta)
- Advanced SQLInjection Scanner
- BeanShell Console
- Image Location and Privacy Scan

2) Arquitetura

- Estrutura da Rede

Todos os elementos foram instalados e testados na mesma máquina virtual com o CentOS7.

Temos o Juicy Shop a receber os pacotes vindos do Apache que funciona um reverse proxy que segue as regras ativas do modsecurity.

A comunicação do cliente com apache pode ser feita de forma direta no browser ou com o OWASP ZAP, podendo este também estar entre o apache e o browser interceptando todos os pacotes enviados/recebidos.

- Servidores

Juicy Shop está a correr num docker container, acessível no porto 3000 (https://localhost:3000).

- Serviços

Apache com modsecurity está a correr nativamente, acessível no porto 80 (http://localhost/) e está a funcionar como um reverse proxy, todos os pedidos direcionados ao apache são enviados de volta para o Juicy Shop.

Este é o ficheiro de configuração do apache:

```
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost

<Directory />
	AllowOverride none
	Require all granted
</Directory>

<IfModule dir_module>
	DirectoryIndex index.html
</IfModule>

<Files ".ht*">
	Require all denied
</Files>

ErrorLog "logs/error_log"
LogLevel warn

<IfModule log_config_module>

	LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
	LogFormat "%h %l %u %t \"%r\" %>s %b" common

	<IfModule logio_module>
	# You need to enable mod_logio.c to use %I and %O
	LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
	</IfModule>

	CustomLog "logs/access_log" combined
</IfModule>

<IfModule alias_module>
	ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
</IfModule>

<Directory "/var/www/cgi-bin">
	AllowOverride None
	Options None
	Require all granted
</Directory>

<IfModule mime_module>
	TypesConfig /etc/mime.types
	AddType application/x-compress .Z
	AddType application/x-gzip .gz .tgz
```

```
        AddType text/html .shtml
        AddOutputFilter INCLUDES .shtml
</IfModule>

AddDefaultCharset UTF-8

<IfModule mime_magic_module>

        MIMEMagicFile conf/magic
</IfModule>

EnableSendfile on
IncludeOptional conf.d/*.conf

ProxyPass "/" "http://localhost:3000/"
ProxyPassReverse "/" "http://localhost:3000/"
```

3) Teste de Segurança de Aplicações Web

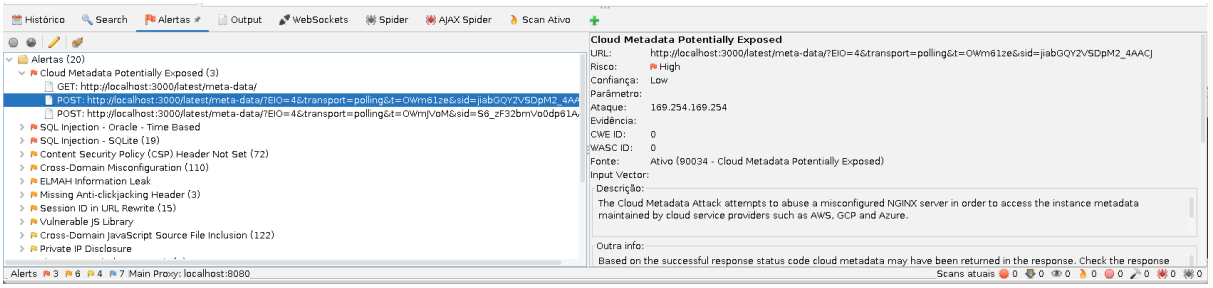
1 Information Gathering

WSTG-v4.2-INFO-03 Review Webserver Metafiles for Information Leakage
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/03-Review_Webserver_Metafiles_for_Information_Leakage

Detectado pelo teste automatizado do OWASP ZAP.

O NGINX está mal configurado, como tal é possível ocorrer um cenário em que através dum ataque de Cloud Metadata extrair informações sensíveis sobre o sistema.

Para o prevenir nunca se deve confiar informação de utilizadores aos ficheiros de configuração do NGINX e configurar estes seguindo os padrões de segurança.



2 Configuration and Deployment Management Testing

WSTG-v4.2-CONF-03 Test File Extensions Handling for Sensitive Information
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/03-Test_File_Extensions_Handling_for_Sensitive_Information

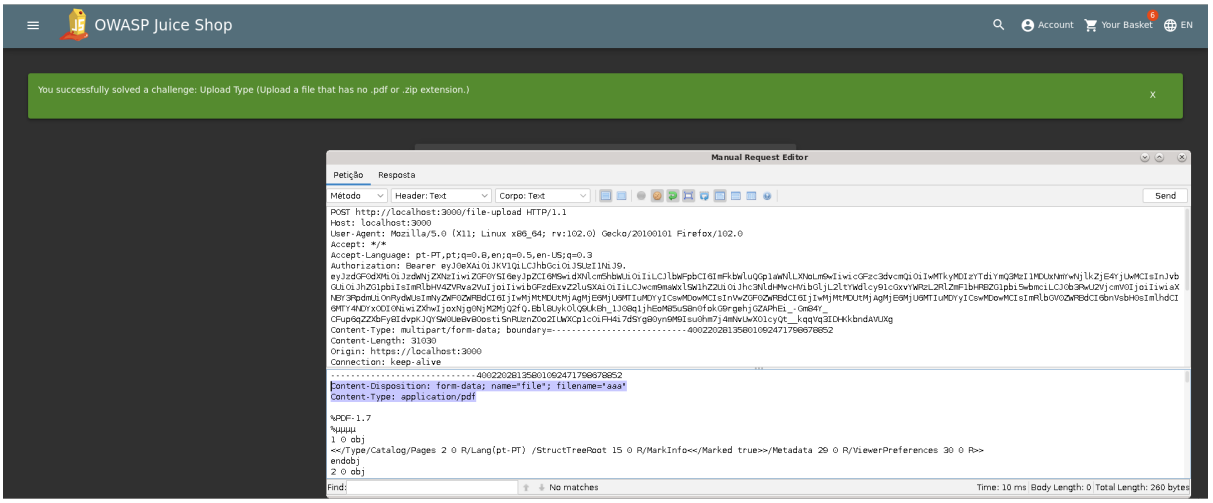
Detectado por um teste manual.

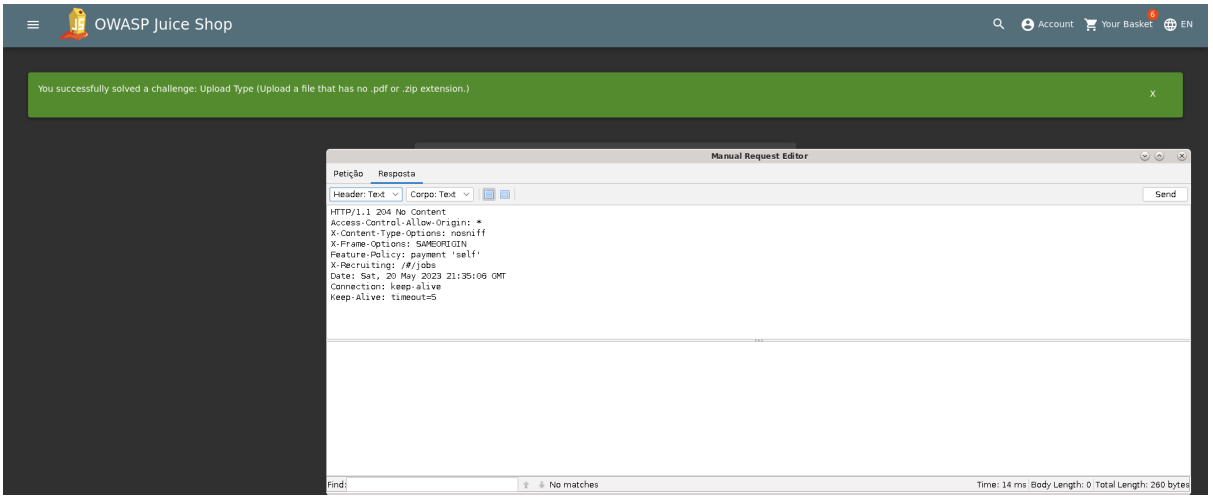
Tentando fazer o upload de um ficheiro na seção de reclamações o file type deste é limitado a .pdf ou .zip, ao ponto que qualquer outro file type é rejeitado.

É possível interceptar o pedido enviado ao servidor para a realização do upload e alterar o valor do campo com o filename para algo que não tenha como extensão .pdf ou .zip, no exemplo das imagens nem extensão alguma foi colocada no nome. Como resultado desta alteração foi possível realizar o upload de um ficheiro com uma extensão diferente do suportado, podendo levar a problemas de integridade do sistema.

Com um WAF é possível mitigar este problema ao confirmar que a extensão presente no nome do file que chega ao WAF corresponde a (.zip ou .pdf) e confirmar que o Content-Type header do file corresponde ao datatype da extensão associada ao nome do ficheiro.

Nota: As categorias 4.2.9, 4.2.10 e 4.2.11 não são aplicáveis a este relatório.


OWASP Juice Shop
Account
Your Basket
EN
You successfully solved a challenge: Upload Type (Upload a file that has no .pdf or .zip extension.)
Manual Request Editor
Petição
Resposta
Método
Header: Text
Corpo: Text
Send
POST http://localhost:3000/file-upload HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: pt-PT,pt;q=0.8,en;q=0.5,en-US;q=0.3
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
Content-Type: multipart/form-data; boundary=--------------------------400220281358010924717906708852
Content-Length: 31030
Origin: http://localhost:3000
Connection: keep-alive
--------------------------400220281358010924717906708852
Content-Disposition: form-data; name="file"; filename="aaa"
Content-Type: application/pdf
%PDF-1.7
%µµµµ
1 0 obj
<</Type/Catalog/Pages 2 0 R/Lang(pt-PT) /StructTreeRoot 15 0 R/MarkInfo<</Marked true>>/Metadata 29 0 R/ViewerPreferences 30 0 R>>
endobj
2 0 obj
Find:
No matches
Time: 10 ms
Body Length: 0
Total Length: 260 bytes

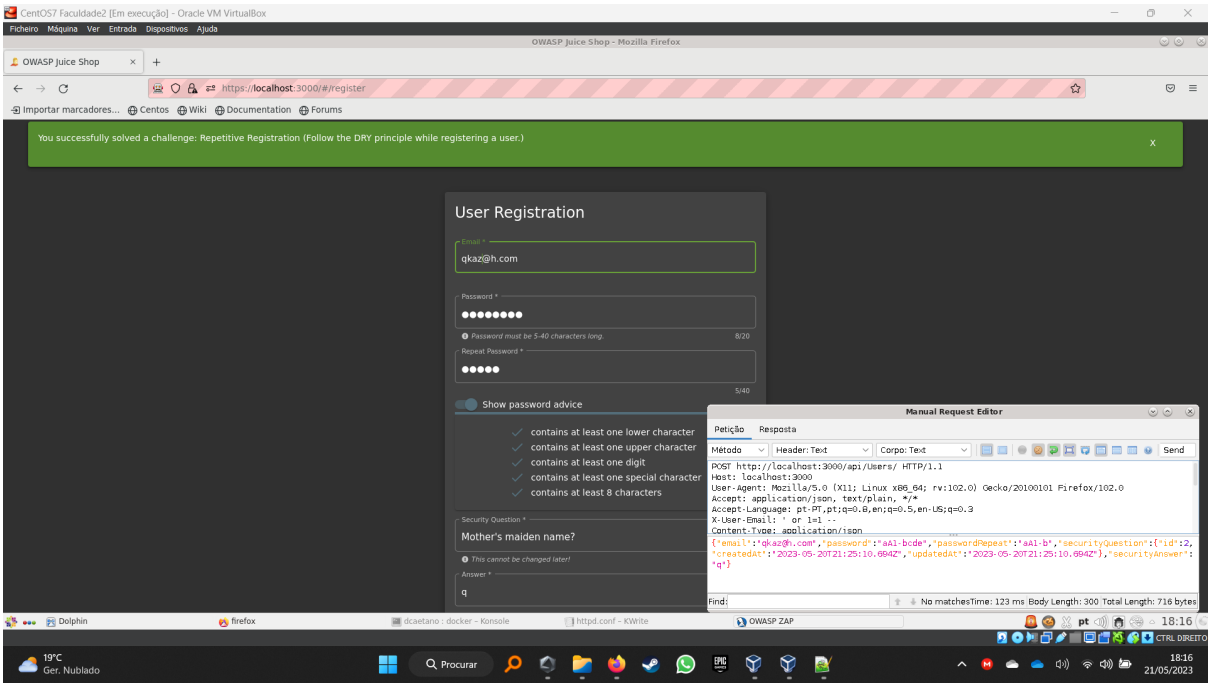

OWASP Juice Shop
Account
Your Basket
EN
You successfully solved a challenge: Upload Type (Upload a file that has no .pdf or .zip extension.)
Manual Request Editor
Petição
Resposta
Header: Text
Corpo: Text
Send
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Date: Sat, 20 May 2023 21:35:06 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Find:
No matches
Time: 14 ms
Body Length: 0
Total Length: 260 bytes

3 Identity Management Testing

WSTG-v4.2-IDNT-03 Test Account Provisioning Process
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/03-Identity_Management_Testing/03-Test_Account_Provisioning_Process

Detectado por um teste manual.

Durante o registo dum novo utilizador é pedido para inserir uma senha e a repetir como confirmação, se 1º preencher o campo mais acima e depois o 2º com a mesma senha estas são aceites, mas se modificar o 1º campo sem tocar no 2º de forma a que fiquem com senhas diferentes, como visível na imagem, a Web-Page contínua a as aceitar, mesmo com uma clara disparidade, contornando assim o mecanismo de validação de input e fazendo o utilizador a ser registado ter a senha presente no 1º campo, como tal o sistema também tem um problema de redundância, pois envia duas strings para o servidor mas na prática só usa uma.

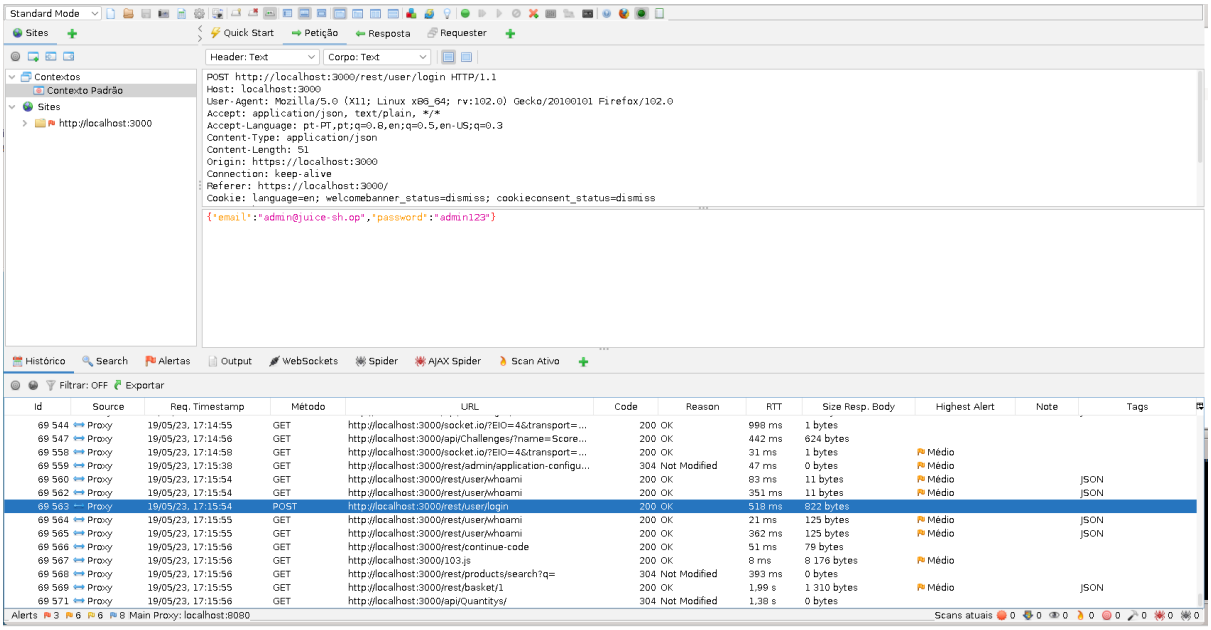


4 Authentication Testing

WSTG-v4.2-ATHN-01 Testing for Credentials Transported over an Encrypted Channel
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/01-Testing_for_Credentials_Transported_over_an_Encrypted_Channel

Detectado por um teste manual.

Através do uso do OWASP ZAP como um proxy a capturar todas as mensagens que partem e chegam ao browser que está interagir com a juice-shop, foi possível capturar o envio das credenciais de login em plain text, sem qualquer forma de encriptação (HTTP), provando assim que o canal de comunicação entre o servidor e cliente não é seguro para a transmissão de dados sensíveis.



5 Authorization Testing

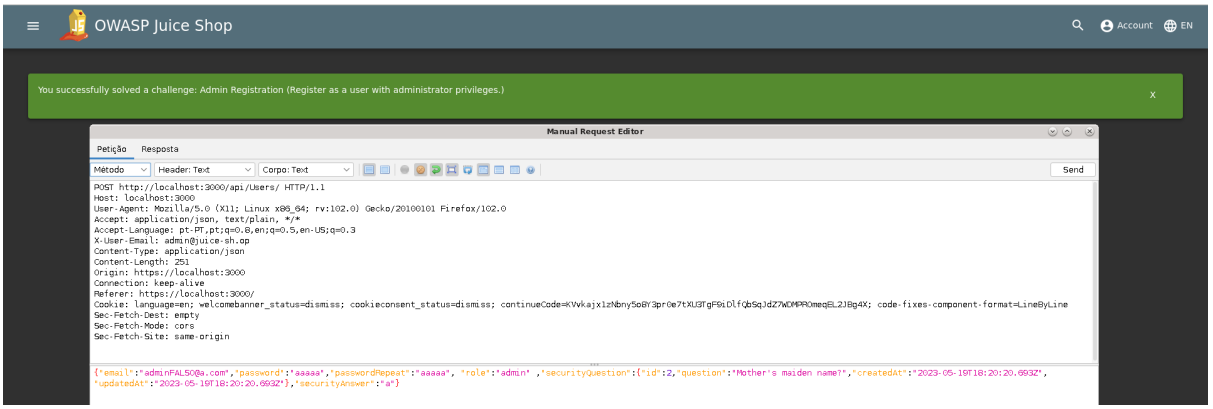
WSTG-v4.2-ATHZ-02 Testing for Bypassing Authorization Schema
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/02-Testing_for_Bypassing_Authorization_Schema

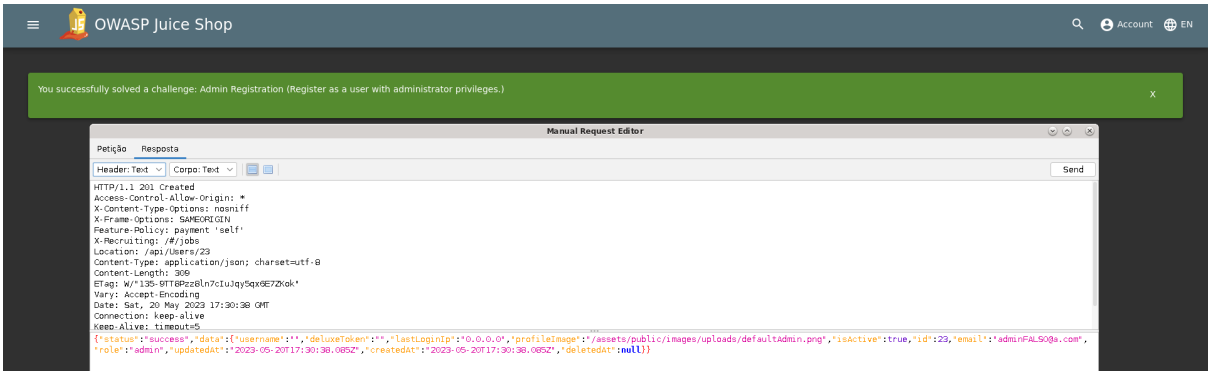
Detectado por um teste manual.

Tendo interpretado o pedido de registo dum user novo, não existia um campo especificando o “role” do user a ser registrado, mesmo este existindo como um campo na Base de Dados, como teste adicionei (“role”: “admin”) num pedido de criação de utilizador e reenviei, o resultado é que o novo user foi dado o cargo de administrador, mesmo não tendo recebido permissões para tal por parte de alguém qualificado, contornando assim o esquema de autenticação.

O sistema usou os campos presentes no body sem antes os validar ou, neste caso do role, sem sequer só selecionar os conteúdos relevantes/corretos para o processo, ao selecionar tudo presente no body, é possível manipular as características do novo utilizador para além do planeado.

Este problema pode ser resolvido com a aplicação dum WAF que haja como um filtro, removendo do pedido do cliente campos indesejados ou que possam dar ao utilizador privilégios indesejados, neste caso apagar o campo “role”, resolveria o problema.





6 Session Management Testing

WSTG-v4.2-SESS-01 Testing for Session Management Schema
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/01-Testing_for_Session_Management_Schema

Detectado com o teste automatizado do OWASP ZAP para XSS e CSRF e o resto por um teste manual.

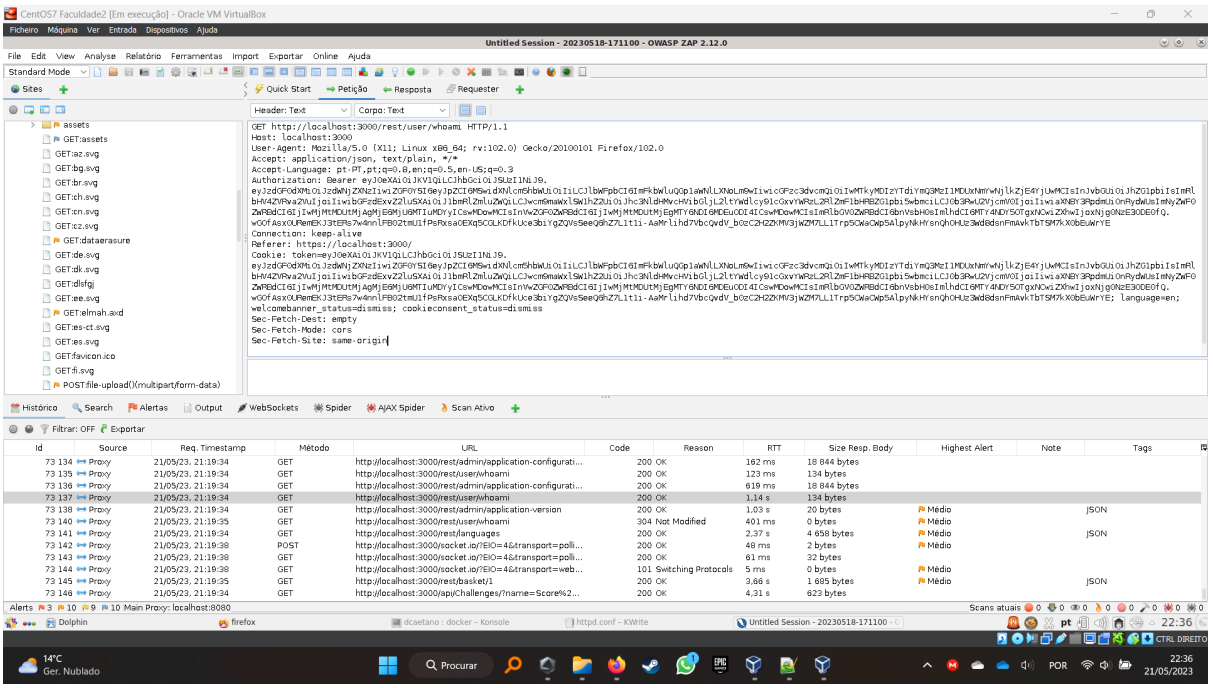
O esquema de sessão implementado é baseado em cookies e há vários problemas com a sua implementação aqui:

1. Os pacotes são transmitidos por HTTP, sem encriptação, permitindo que todos os dados transmitidos caso intentados sejam compreensíveis e usáveis por terceiros.

2. O elemento token, não tem o atributo “Secure” definido este faz com que o token só seja transmissível por HTTPS.

3. O elemento token, não tem o atributo “HttpOnly”, que bloqueia o acesso de javascript do lado do cliente conseguir aceder ao ás cookies, ajudando a prevenir o risco de cross-site scripting (XSS).

4. O elemento token, não tem o atributo “SameSite”, que bloqueia o envio do cookie a outros sites, ajudando a prevenir risco de ataques de cross-site request forgery (CSRF).



7 Input Validation Testing

WSTG-v4.2-INPV-05 Testing for SQL Injection
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

Detectado por 1 Scan Ativo feito por um fuzzer ao formulário de login com o uso do fuzzer FuzzDB Offensive, onde username e password tiveram as payloads loads:

1. fuzzdb -> attack -> authentication
2. jbrofuzz -> ASCII 95 Alphabet
3. SQLinjection
4. Zero Fuzzers ->10 plain requests

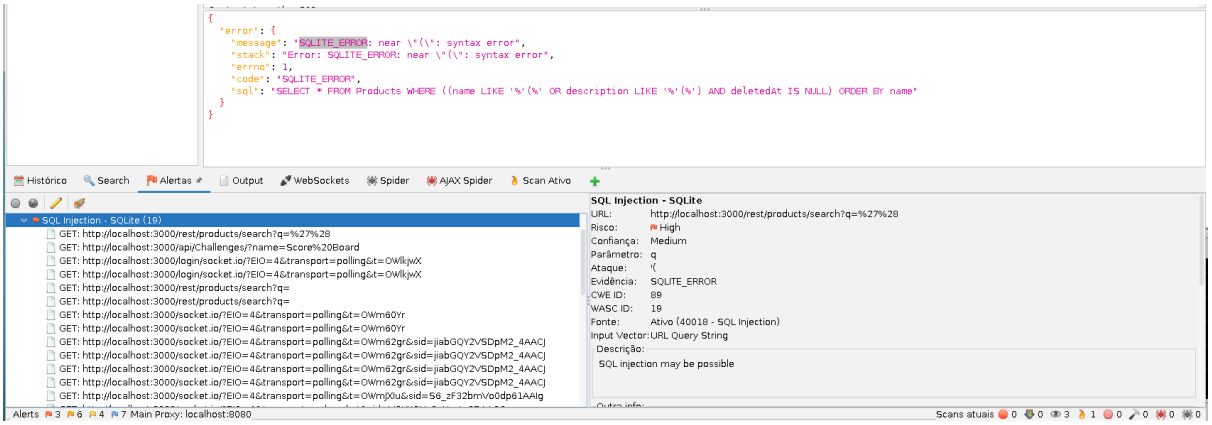
Foi detectado um erro de SQLite, isso significa que o servidor não validou o input do usuário recebido do client-side, e este foi usado numa query enviada para a DataBase e esta gerou um erro detectável do lado do cliente.

Este problema é grave, pois permite que agentes externos, por tentativa e erro, consigam fazer engenharia-reversa do sistema de forma a entender o seu funcionamento para conseguir extrair informações e manipular a Base de Dados sem terem permissões para tal, comprometendo a sua integridade e a privacidade dos clientes nela registados.

No ponto de vista da lógica do sistema, a solução passa por validar os inputs recebidos no server-side, mesmo quando estes já foram validados no client-side, antes de os incorporar em queries parametrizadas a serem enviadas à DataBase.

Foram detectados:

- 2 tipos de SQL Injection: Oracle - Time Based e SQLite
- 5 falhas para Cross Site Scripting:
    - CSP: Wildcard Directive,
    - CSP: script-src unsafe-eval,
    - CSP: style-src unsafe-inline (2),
    - Content Security Policy (CSP) Header Not Set
    - Absence of Anti-CSRF Tokens
- 1 Data Injection attack: Content Security Policy (CSP) Header Not Set



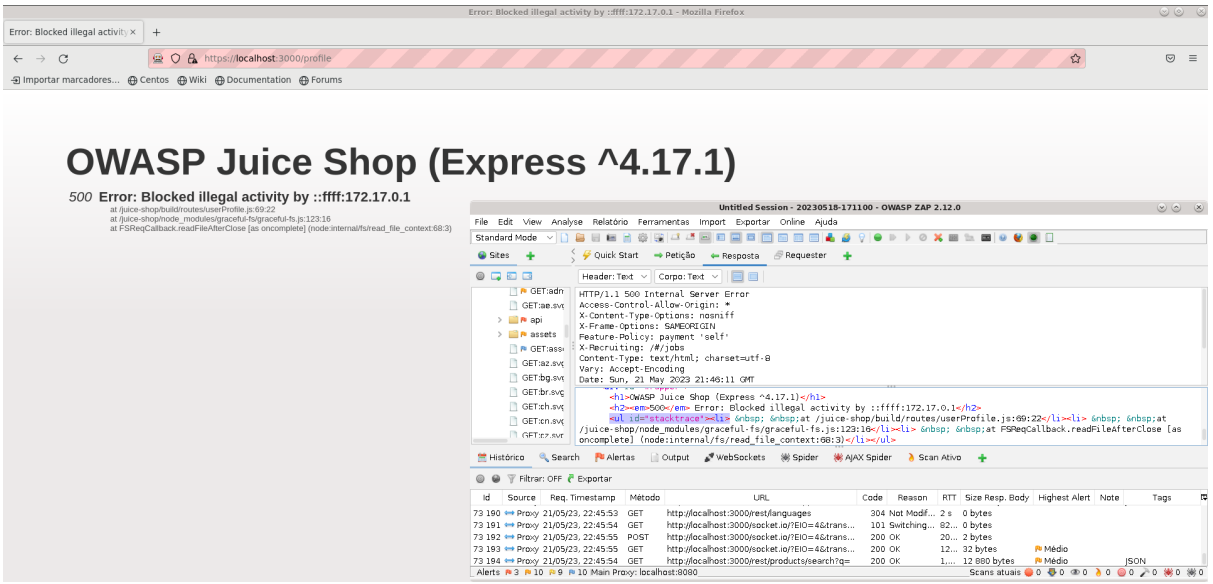
8 Testing for Error Handling

WSTG-v4.2-ERRH-01 Testing for Improper Error Handling
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/08-Testing_for_Error_Handling/01-Testing_For_Improper_Error_Handling

Detectado por um teste manual.

Se usar o URL para aceder ao perfil do utilizador atualmente logado sem ter qualquer utilizador logado, o acesso a esta é bloqueado, como esperado, no entanto ao invés duma simples mensagem de erro, é apresentada uma página com a stacktrace gerada com o levantamento do erro interno do servidor.

Como tal alguém que pretenda entender o funcionamento interno do servidor para o atacar, pode o conseguir através da análise desta stacktrace, devendo por isso esta nunca ser exposta publicamente.

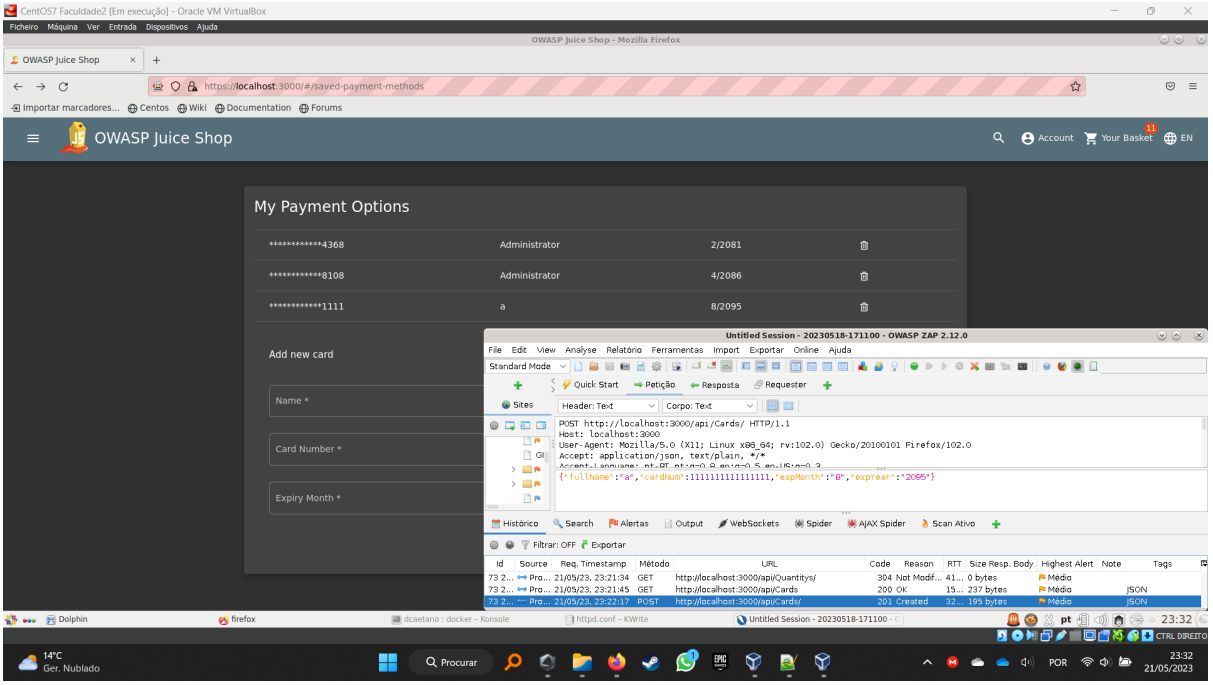


9 Testing for Weak Cryptography

WSTG-v4.2-CRYP-03 Testing for Sensitive Information Sent via Unencrypted Channels
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/03-Testing_for_Sensitive_Information_Sent_via_Unencrypted_Channels

Detectado por um teste manual.

Os dados necessários para registar um novo cartão de crédito para a realização de pagamentos, são enviados para o servidor em plain text com o protocolo HTTP, permitindo que agentes 3º os consigam interceptar e extrair, representando uma grande falha de privacidade.

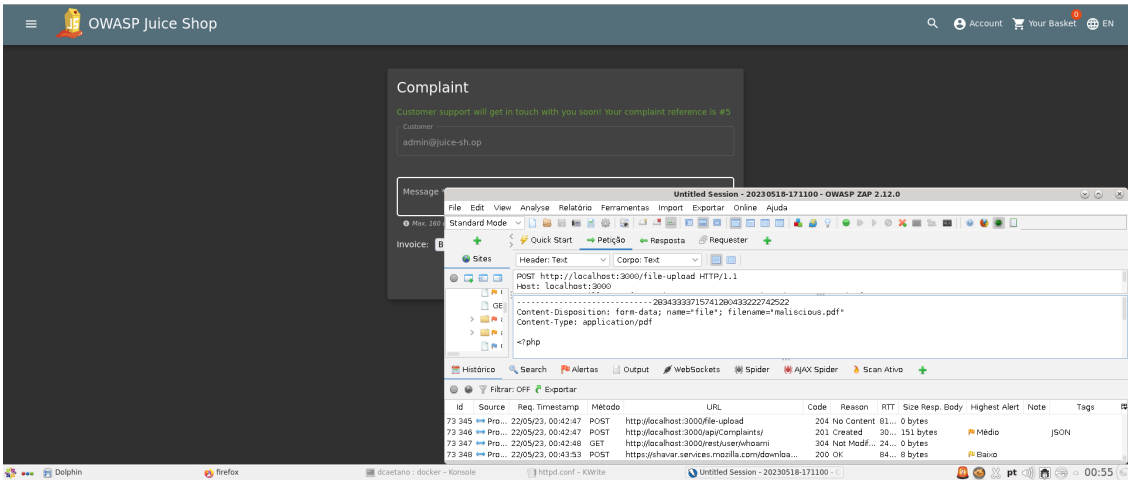


10 Business Logic Testing

WSTG-v4.2-BUSL-09 Test Upload of Malicious Files
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/10-Business_Logic_Testing/09-Test_Upload_of_Malicious_Files

Detectado por um teste manual.

Foi possível fazer upload de um script de PHP para o servidor, bastou mudar a extensão para .pdf e este ficheiro passou a ser aceito pela Web-Page para ser enviado ao servidor. Esta falha permite o envio de payloads e scripts maliciosos para o servidor onde estes poderão ser guardados e/ou executados comprometendo a integridade do servidor.

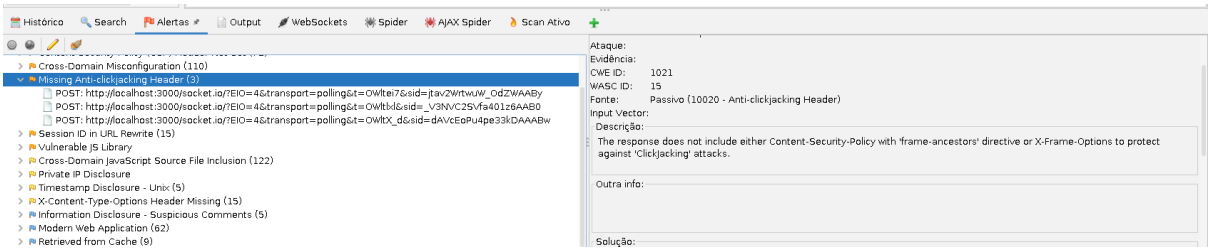


11 Client Side Testing

WSTG-v4.2-CLNT-09 Testing for Clickjacking
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking

Detectado pelo teste automatizado do OWASP ZAP.

O site está vulnerável a ser renderizado dentro dum iframe dum site 3º, capaz de se fazer passar pelo site original, uma das camadas de prevenção usadas para prevenir tal problema é o uso do header de HTTP X-Frame-Options, que permite definir quem é capaz de renderizar as páginas deste, sendo a opção mais segura o uso do valor DENY, quem impede que a página seja alguma vez colocada dentro dum frame de outra página.



WSTG-CLNT-07 Content Security Policy (CSP) Header Not Set

Detectado pelo teste automatizado do OWASP ZAP.

Acabou por não ser avaliado mais a fundo, deixamos como um extra.

4) Firewall de Segurança de Aplicações Web

1 Information Gathering

WSTG-v4.2-INFO-03 Review Webserver Metafiles for Information Leakage

A solução com WAF passa por apagar de pacotes quaisquer menções de metadata nos seus Headers e Bodies.

SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess On

SecRule REQUEST_HEADERS "(?i)\bmetadata\b" "id:1,phase:2,t:none,t:lowercase,t:replaceNulls,t:compressWhitespace,log,deny,msg:'Metadados removidos'"

SecRule RESPONSE_HEADERS "(?i)\bmetadata\b" "id:2,phase:4,t:none,t:lowercase,t:replaceNulls,t:compressWhitespace,log,deny,msg:'Metadados removidos'"

SecRule REQUEST_BODY "(?i)\bmetadata\b" "id:3,phase:2,t:none,t:lowercase,t:replaceNulls,t:compressWhitespace,log,deny,msg:'Metadados removidos'"

SecRule RESPONSE_BODY "(?i)\bmetadata\b" "id:4,phase:4,t:none,t:lowercase,t:replaceNulls,t:compressWhitespace,log,deny,msg:'Metadados removidos'"

2 Configuration and Deployment Management Testing

WSTG-v4.2-CONF-03 Test File Extensions Handling for Sensitive Information

Com um WAF é possível mitigar este problema ao confirmar que a extensão presente no nome do file que chega ao WAF corresponde a (.zip ou .pdf) e confirmar que o Content-Type header do file corresponde ao datatype da extensão associada ao nome do ficheiro e caso não seja o ficheiro é rejeitado.

SecRule FILES_NAMES "@endsWith .zip" "id:5,phase:1,t:none,nolog,pass"
SecRule FILES_NAMES "@endsWith .pdf" "id:6,phase:1,t:none,nolog,pass"

SecRule REQUEST_HEADERS:Content-Type "!@rx ^application/(zip|pdf)$" "id:7,phase:2,t:none,log,msg:'Ficheiro com extensão ou e/ou Content-Type header errada: Content-Type does not match file extension.',severity:2,deny,status:403"

3 Identity Management Testing

WSTG-v4.2-IDNT-03 Test Account Provisioning Process

Com um WAF, é possível validar que as 2 senhas presentes nos campos (npassword e passwordRepeat) do pedido são as mesmas e rejeitar os pedidos nos quais isso não se verifica.

SecRule REQUEST_FIELDS:npassword "@streq %{REQUEST_FIELDS:passwordRepeat}" "id:8,phase:2,t:none,log,msg:'As senhas não coincidem.',severity:2,deny,status:400"

4 Authentication Testing

WSTG-v4.2-ATHN-01 Testing for Credentials Transported over an Encrypted Channel

Com o uso do WAF, é possível obrigar o estabelecimento de uma conexão encriptada (HTTPs) entre este e o cliente, criando um canal de comunicação seguro para a transmissão de dados sensíveis, impedindo assim que agentes terceiros, como o OWASP ZAP, consigam interpretar as informações que venham a interceptar no canal entre cliente e WAF.

Nota: esta solução não resolve possíveis ataques entre WAF e Servidor, seria necessário também criar uma conexão encriptada entre estes 2.

SecRule REQUEST_HEADERS:Upgrade-Insecure-Requests "1" "id:9,phase:1, t:none,log,pass,msg:'Conexão upgraded para HTTPS', redirect:https://%{HTTP_HOST}%{REQUEST_URI}'"

5 Authorization Testing

WSTG-v4.2-ATHZ-02 Testing for Bypassing Authorization Schema

Este problema pode ser resolvido com a aplicação dum WAF que haja como um filtro, removendo do pedido do cliente campos indesejados ou que possam dar ao utilizador privilégios indesejados, neste caso apagar o campo “role”, resolveria o problema.

SecRule REQUEST_FIELDS_NAMES "@rx role" \ "phase:2,log,auditlog,msg:'Campo role removido da request',id:10,t:none,t:lowercase,deny,status:400"

6 Session Management Testing

WSTG-v4.2-SESS-01 Testing for Session Management Schema

Para resolver com um WAF basta:

Para o Problema 1: Forçar Ligações HTTPS.
Para o Problema 2,3,4: No elemento (token) do header adicionar os elementos “Secure”, “HttpOnly” e “SameSite” respetivamente.

Regra com o id=5. e

```
SecRule REQUEST_HEADERS:Token ".*" \
   "id:11, \
   phase:2, \
   pass, \
   t:lowercase, \
   log, \
   msg:'Segurança do token fortalecida', \
   setenv:TOKEN_SECURE, \
   setenv:TOKEN_HTTPONLY, \
   setenv:TOKEN_SAMESITE"

Header always edit Set-Cookie (.*) "$1; Secure" env=TOKEN_SECURE
Header always edit Set-Cookie (.*) "$1; HttpOnly" env=TOKEN_HTTPONLY
Header always edit Set-Cookie (.*) "$1; SameSite=Strict" env=TOKEN_SAMESITE
```

7 Input Validation Testing

WSTG-v4.2-INPV-05 Testing for SQL Injection

Por parte do WAF é possível, procurar por padrões comuns de SQL injection como (or 1=1) mas tal é limitado às formas de validação mais simples, pois o WAF não tem acesso à lógica interna da DB.

Mas para prevenir que o atacante aprenda sobre DB através dos erros que recebe desta, deve-se filtrar a transmissão de respostas para o cliente que tenham no body a presença de mensagens de erro relacionadas à Base de Dados, quando estas estiverem a passar no WAF, impedindo este de obter informações sobre como a DB faz o handling dos seus erros.

```
SecRule RESPONSE_BODY "@contains error" \
   "id:12,\
   phase:4,\
   pass,\
   t:none,\
   nolog,\
   setvar:'tx.error_removed=1',\
   setvar:'tx.error_removed_msg=mensagem de erro removida',\
```

```
    chain"

SecRule TX:error_removed "@eq 1" \
    "chain"

SecRule RESPONSE_BODY "@beginsWith %{tx.error_removed_msg}" \
    "t:none,\
    setvar:'tx.error_removed=0',\
    setvar:'tx.error_removed_msg='"

SecRule TX:error_removed "!@eq 0" \
    "capture,\
    setvar:'tx.msg=%{tx.error_removed_msg}'"
```

8 Testing for Error Handling

WSTG-v4.2-ERRH-01 Testing for Improper Error Handling

Para resolver isto podemos com um WAF apagar do body de qualquer reply o conteúdo associado a um (id="stacktrace"), tal como visto na imagem, desta forma a página de resposta a dizer Erro Interno será apresentada, mas a stacktrace será censurada. Ou simplesmente detectar que se trata de um pacote de erro interno (CODE: 500) e substituir a resposta em HTML com uma mensagem de erro genérica escrita em HTML ou até mesmo em plain text.

```
SecRuleEngine On

SecRule RESPONSE_STATUS "@streq 500" "id:13,log, phase:4,t:none, msg:'stacktrace apagada da resposta', ctl:auditLogParts=+E"

SecRule RESPONSE_BODY "(?i)<\s*div\s*id\s*=\s*\"stacktrace\"[^>]*>.*?</\s*div\s*>" "id:14,t:none,t:htmlEntityDecode,t:removeTagById=stacktrace"
```

9 Testing for Weak Cryptography

WSTG-v4.2-CRYP-03 Testing for Sensitive Information Sent via Unencrypted Channels

Para resolver, com um WAF pode-se atualizar o protocolo para HTTPs, desta forma toda informação transmitida passa a ser encriptada, preservando assim a privacidade dos utilizadores e dos seus cartões de crédito.

Regra com o id=5

10 Business Logic Testing

WSTG-v4.2-BUSL-09 Test Upload of Malicious Files

Para mitigar esta questão podemos usar um WAF que tenha ativado a opção de File Type Validation, esta opção irá, durante o processo de upload, verificar se o conteúdo presente nos ficheiros submetidos realmente corresponde ao filetype declarado dos mesmos (.pdf ou .zip), prevenindo assim que conteúdo malicioso se consiga disfarçar por conteúdo legítimo.

Nota: não resolve o possível cenário do .zip ter algum ficheiro malicioso após descompactado.

```
SecRule REQUEST_FILENAME "@rx .*\.(pdf|zip)$" \
        "id:15,\
        phase:2,\
        t:lowercase,\
        deny,\
        log,\
        status:403,\
        msg:'Ficheiro (potencialmente) malicioso rejeitado'"

SecRule REQUEST_HEADERS:Content-Type "!@pm application/pdf application/zip"
\
        "id:16,\
        phase:2,\
        deny,\
        log,\
        status:403,\
        msg:'Ficheiro (potencialmente) malicioso rejeitado'"
```

11 Client Side Testing

WSTG-v4.2-CLNT-09 Testing for Clickjacking

Para resolver, com o WAF é vamos injetar um header de HTTP X-Frame-Options nos pacotes de HTTP com o valor DENY, que impede qualquer página ser capaz de renderizar o nosso site sem ser o nosso próprio site.

```
SecRule RESPONSE_HEADERS:X-Frame-Options "@eq 0" \
        "id:17,\
        phase:4,\
        pass,\
        t:none,\
        setenv:RESPONSE_HEADERS:X-Frame-Options=DENY,\
        log,\
        msg:'injetado um header de HTTP X-Frame-Options DENY'"
```

5) Conclusão

Neste trabalho prático identificamos diversas vulnerabilidades, como leaking de informações do servidor, manipulação de extensões de arquivos, falhas na validação de senhas, transmissão de credenciais sem criptografia, entre outros.

Juicy-Shop é uma web app desenhada com o propósito de ser propositalmente falha e cheia de más práticas, como tal a quantidade de vulnerabilidades nesta é altíssima, o nosso trabalho revelou várias falhas de segurança, essas vulnerabilidades podem levar a riscos significativos de vazamento de informações, manipulação indevida de dados e comprometimento da integridade do sistema.

Para mitigar esses riscos, é essencial implementar práticas de segurança adequadas, como validação de entrada, criptografia de dados, tratamento adequado de erros e uso de canais seguros de comunicação. Essas medidas são essenciais para proteger a aplicação e garantir a confidencialidade, integridade e disponibilidade dos dados e dos utilizadores.

EXTRA-Todos os alertas levantados