

# OWASP ZAP Pentesting

Henrique José Carvalho Faria, nºA82200

Departamento de Informática, Universidade do Minho

**Resumo** Neste artigo científico procedeu-se ao estudo dos sistemas de recomendação iniciando-se com uma introdução ao tema abordado e uma contextualização histórica do mesmo. Seguidamente abordaram-se os cinco paradigmas diferentes: Sistemas baseados em filtros colaborativos (community based recommender systems), Sistemas de Recomendação Baseados em Conteúdo (Content-based recommender systems), Sistemas de Recomendação Baseados em Conhecimento (Knowledge-based recommender systems), Sistemas de Recomendação Demográficos (Demographic filtering), Sistemas de Recomendação híbridos/Agregados (Hybridand Ensemble-based recommender systems) [?]. Por fim, referem-se dois exemplos práticos da utilização de sistemas de recomendação, na área da filmes (Netflix) e na de música (Spotify).

**Palavras-chave:** Sistemas de Recomendação · Netflix · Spotify.

## Introdução

2 Henrique José Carvalho Faria, nºA82200

## 1 OWASP ZAP

## 1.1 OWASP-ZAP Layout

Nesta secção pretendemos explicar todos os recursos importantes disponibilizados pelo ZAP. E a configuração da interface do mesmo.

### 1.1.1 Desktop UI Overview

#### 1.1.1.1 Top Level Menu

Em primeiro lugar vamos referir as opções oferecidas pelo menu superior da aplicação. Este menu oferece acesso a praticamente todos os recursos do ZAP e não só isso como também permite configurar os mesmos.

- File menu

Permite que possamos criar uma nova sessão, importar uma sessão guardada, apagar a sessão atual, guarda-la ou sair da aplicação.

- Edit menu

Permite mudar o modo em que o ZAP opera:

- safe: não são realizadas operações potencialmente perigosas.
- protected: só podemos realizar operações potencialmente perigosas nos URLs especificados
- Standard: podemos realizar qualquer operação que o ZAP suporte
- ATTACK: novos nodos que entram na "mapa" resultados do spider do site sãoativamente scaneados mal seja descobertos.

Permite manter o utilizador autenticado no site alvo com a opção *forced user mode*.

- View menu

Permite ver a *history tab*, que por sua vez mostra todos os pedidos feitos ordenados no tempo possuindo as informações de index, método HTML usado (GET/POST), URL, código de resposta HTTP, explicação do que significa o código anterior, o tempo que demorou o pedido, qualquer alerta de vulnerabilidade no pedido, notas adicionadas ao pedido e tags associadas ao mesmo.

- Analyse menu

Este menu permite modificar uma política de análise<sup>1</sup> ou adicionar uma nova.

- Report menu

permite gerar um relatório com todos os alertas encontrados em formato HTML, XML ou Markdown.

---

<sup>1</sup> Uma política de análise define um conjunto de regras a serem seguidas durante uma análise ativa. Também define quantos pedidos são feitos e quão provável será potenciais problemas serem assinalados.

- Tools menu

Este menu permite aceder ás opções de configuração das ferramentas e recursos disponíveis no ZAP.

- Nas configurações de scan ativo podemos definir:

- O número de hosts a serem analisados ao mesmo tempo;
- Quantas threads existirão por host;
- O número de resultados que aparecerão na tab do scan ativo;
- O tempo máximo que uma regra da política de análise poderá demorar;
- O tempo máximo que um scan pode demorar;
- O delay em (milisegundos) entre pedidos feitos;
- Se queremos que seja injetado em cada pedido o identificador do mesmo no ZAP na forma X-ZAP-Scan-ID;
- Se queremos que durante o scan o zap tente obter tokens anti CSRF<sup>2</sup>;
- Usar a definição padrão de análise ativa;
- Usar a definição padrão de análise de ataque;

- Nos vetores de input do scan ativo podemos definir como inputs:

- Strings de URL específicas após o ?;
- Pares chave-valor nos dados Post pedidos;
- Elementos de caminhos URL;
- Cabeçalhos HTTP;
- Cookies;

- Podemos adicionalmente escolher o formato dos inputs do scan ativo sendo estes:

- \* Multipart Form Data
- \* XML tag/attribute
- \* JSON

---

<sup>2</sup> OS tokens Anti CSRF são parâmetros pseudo aleatórios usados para proteção contra ataques de Cross Site Request Forgery (CSRF).

- \* Google Web Toolkit
- \* OData id/filter
- Podemos definir como queremos que os alertas nos sejam apresentados:
  - Podemos reduzir o tamanho dos relatórios significativamente caso façamos merge de alertas iguais encontrados em pontos diferentes do scan.
  - Podemos também escolher o número máximo de alertas que pretendemos que constem no relatório.
  - Por último é-nos dada a opção de modificar a mensagem associada com um alerta específico para melhor compreensão.
- Outro ponto configurável são os tokens anti CSRF, podemos escolher quais queremos analisar.
- Podemos definir o endereço e a porta em que o ZAP vai ficar à escuta de conexões, caso não sejam definidos o ZAP usa o localhost como endereço e liga-se a uma porta aleatória disponível na máquina.
- O ZAP permite gerar um certificado SSL dinâmico para os casos em que as aplicações usem SSL mútuo.

Para permitir conexões SSL transparentes é necessário encriptar cada pedido feito ao servidor e desencriptar cada resposta do mesmo, como o browser usado já faz isso, a única maneira de o ZAP fazer isto é através de um ataque "man-in-the-middle". Para isto basta que adicionemos aos certificados de confiança do browser escolhido o ZAP Root CA certificate. Desta forma como o ZAP cria um certificado assinado para cada servidor sobre o qual realiza o ataque "man-in-the-middle", usar o ZAP Root CA certificate é uma forma de assegurar ao browser que pode confiar nesse certificado.

- O ZAP também permite definir e configurar o tipo de conexão a ser usada:
  - Podemos definir um timeout que o que facilita o teste de aplicações lentas.
  - Podemos definir o utilizador que o ZAP deve ser ao criar as mensagens HTTP.
  - Podemos definir se o ZAP usa um "Single cookie request header" ou "multiple cookie request header"<sup>3</sup>.
  - Podemos ativar o (Global) HTTP State, isto é, podemos rastrear as cookies guardadas na sessão.

---

<sup>3</sup> O Cookie HTTP request header contém cookies HTTP enviadas previamente em respostas do servidor, o Zap possibilita enviar as cookies todas juntas ou uma de cada vez por HTTP request header.

- Podemos definir por quanto tempo as queries DNS bem sucedidas devem ser guardadas em cache (números negativos representam sem limite de tempo, zero não permite guardar e números positivos representam o tempo em segundos a guardar).

- Podemos ainda definir as versões dos protocolos a usar nas conexões efetuadas. Devemos no entanto ter a opção "SSLv2Hello" selecionada com pelo menos uma versão de SSL/TLS.

- Podemos também definir vários regex para URLs que o ZAP irá ignorar. Estes regex podem ser ainda modificados ou removidos.

#### – Help menu

- Neste menu podemos procurar ajuda sobre como utilizar o ZAP, procurar ajuda para resolver problemas de versões do ZAP e add-ons e para verificar a existência de updates.

### 1.1.1.2 Toolbar Menu

Neste menu podemos encontrar formas de dispor o layout do ZAP para que se torne mais agradável à vista ou para podermos visualizar vários painéis úteis simultaneamente. Neste menu destacam-se algumas features que achamos interessantes nomeadamente:



**Figura 1.** Toolbar Menu

Neste menu podemos definir se queremos colocar os pedidos feitos ao servidor e as respetivas respostas ao servidor lado a lado. Para isso temos 2 opções disponíveis.

1. Colocar os pedidos e respostas em tabs lado a lado.
2. Colocar os pedidos e respostas na mesma tab lado a lado.
3. Colocar os pedidos na parte esquerda de uma tab e as respetivas respostas na direita.

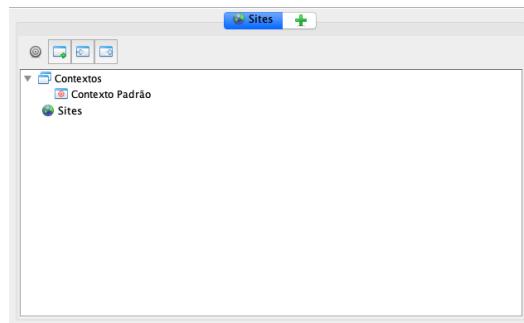
4. Colocar os pedidos na parte superior de uma tab e as respetivas respostas na inferior.

Adicionalmente este menu permite um controlo maior sobre os pedidos e as respostas do servidor, permitindo-nos adicionar um break em cada pedido, resposta ou nos dois e modificar os pedidos, evitar que um pedido seja feito ou remover um par pedido-resposta. O botão usado para implementar os breaks é o circulo assinalado pelo número 5.

Possuimos também auxiliares para trabalhar com os breakpoints assinalados com os números 6,7,e 8 que executam as seguintes ações sobre os pedidos e respostas apanhadas respetivamente saltar para o próximo par pedido resposta, continuar a realizar os pedidos e respostas e parar apenas no caso de termos definido um breakpoint para um URL específico<sup>4</sup> e destruir o par pedido-resposta apanhado pelo breakpoint.

#### 1.1.1.3 Windows

1. Sites Window, nesta podemos ver toda a informação referente aos sites que visitamos, isto inclui o URL, pedidos GET associados ao URL, pastas ocultas encontradas neste entre outros.

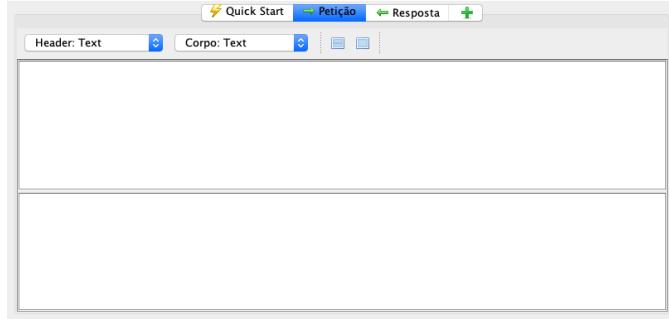


**Figura 2.** Sites Window

2. Workspace Window, nesta podemos visualizar as informações dos pedidos e as respetivas respostas. Também é nesta janela que podemos efetuar alterações aos mesmos quando temos breakpoints definidos.

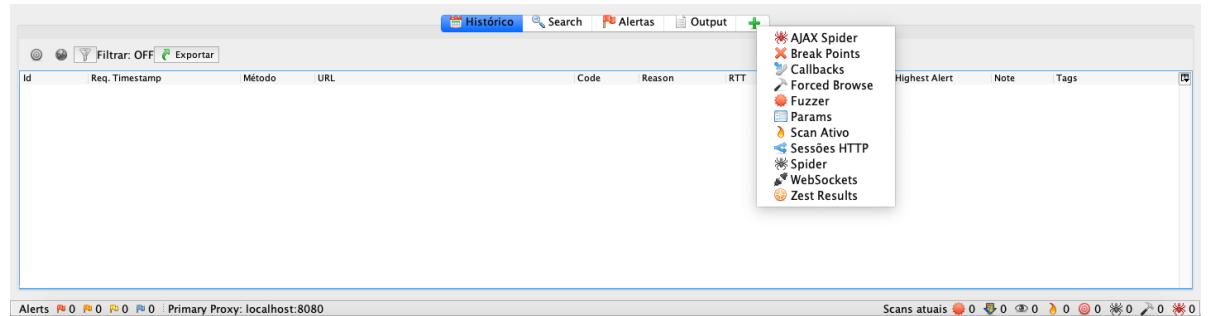
---

<sup>4</sup> Um breakpoint pode ser definido para um URL específico com recurso ao elemento identifocado como 9.



**Figura 3.** Workspace Window

3. Information Window, nesta última janela encontramos todas as informações úteis referentes aos testes de segurança realizados sobre um site. As informações estão organizadas nas seguintes tabs:
  - History tab: Mostra os pedidos pela ordem em que foram feitos, permitindo modificar a ordem pelo atributo que o utilizador quiser (url, método, código, alerta mais importante,...).
  - Search tab: Permite uma pesquisa entre todos os pedidos e respostas de um teste.
  - Breakpoints tab: mostra os breakpoints feitos num teste.
  - Alerts tab: Mostra todos os alertas encontrados na aplicação durante o teste.
  - Active Scan tab: Mostra os scans ativos
  - Spider tab: Mostra os URLs que ainda não foram visitados
  - Params tab: Mostra um sumário dos parâmetros que um site utiliza.
  - Output tab: Mostra várias mensagens informativas sobre o teste realizado.
  - Callbacks tab: Mostra pedidos de callbacks que o ZAP recebeu durante os testes ao site.



**Figura 4.** Information Window

## 1.2 Preparação dos recursos

Agora que já sabemos mexer no ZAP temos de começar por preparar o site sobre o qual vamos efetuar os nossos testes. Para este site usaremos o mutillidae configurando o Metasploitable para o correr e o Firefox para podermos aceder ao mesmo.

### 1.2.1 Metasploitable configurar o ficheiro config.inc:

```
$dbname = 'metasploit' -> $dbname = 'owasp10'
```

```
<?php
/* NOTE: On Samurai, the $dbpass password is "samurai" rather than blank */
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '';
$dbname = 'metasploit';
?>
```

**Figura 5.** Antes da configuração do ficheiro config.inc

```
<?php
/* NOTE: On Samurai, the $dbpass password is "samurai" rather than blank */
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '';
$dbname = 'owasp10';
?>
```

**Figura 6.** ficheiro config.inc corretamente configurado

De seguida temos de ir a etc/php5/cgi e configurar o ficheiro *php.ini*. Neste ficheiro temos primeiro de encontrar os fopen wrappers corretos, para isso podemos usar o comando "ctr + w" para procurar "fopen". Agora basta-nos modificar a opção allow\_url\_include = Off para allow\_url\_include = On, como se vê nas figuras seguintes.

```
; Whether to allow include/require to open URLs (like http:// or ftp://) as files
allow_url_include = Off
```

**Figura 7.** Opção a encontrar para ser modificada

```
; Whether to allow include/require to open URLs (like http:// or ftp://) as fil$  
allow_url_include = On
```

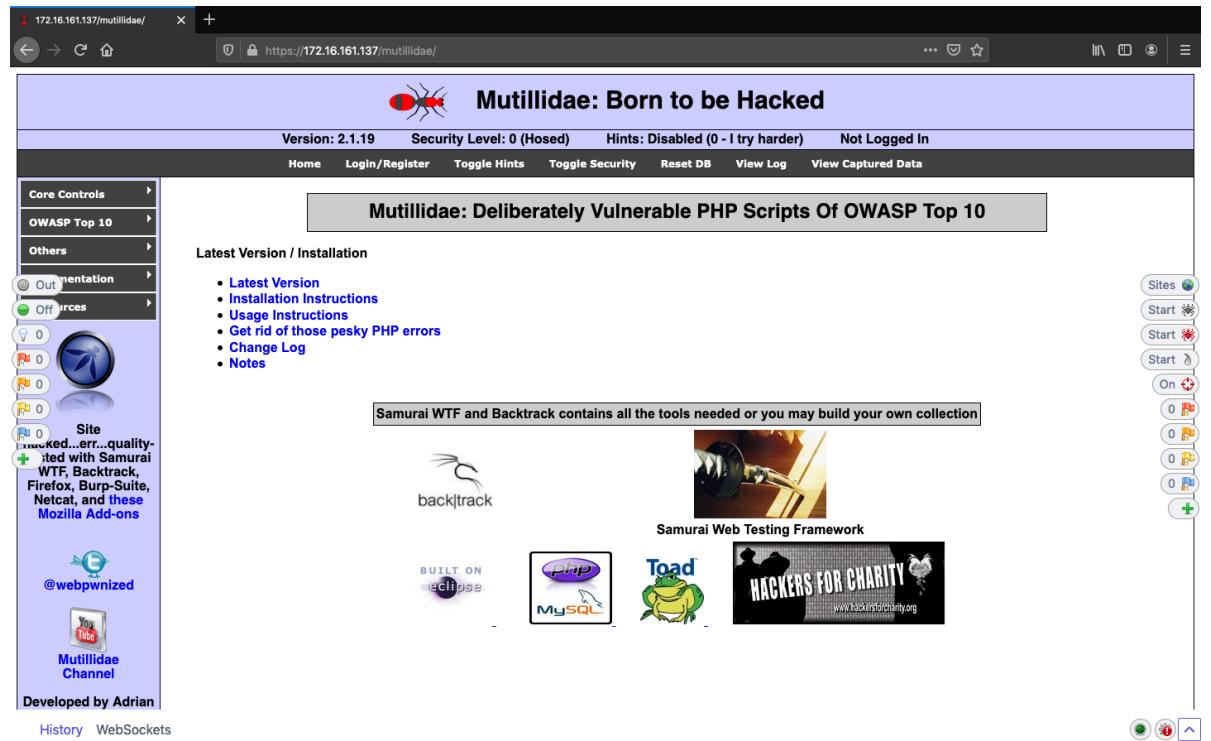
**Figura 8.** Opção modificada

Agora só precisamos de reiniciar o servidor. A figura seguinte mostra o comando usado para reiniciar o servidor apache : "sudo /etc/init.d/apache2 restart" e o comando ifconfig para visualizarmos o endereço IP da Máquina virtual para posterior utilização no Firefox.

```
msfadmin@metasploitable:/etc/php5/cgi$ sudo /etc/init.d/apache2 restart [ OK ]  
* Restarting web server apache2  
msfadmin@metasploitable:/etc/php5/cgi$ ifconfig  
eth0      Link encap:Ethernet HWaddr 00:0c:29:8b:06:6b  
          inet addr:172.16.161.137 Bcast:172.16.161.255 Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe8b:66b/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
            RX packets:45 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:97 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:5296 (5.1 KB) TX bytes:12014 (11.7 KB)  
            Interrupt:17 Base address:0x2000  
  
lo       Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING MTU:16436 Metric:1  
            RX packets:244 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:244 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:78289 (76.4 KB) TX bytes:78289 (76.4 KB)
```

**Figura 9.** Reiniciação do servidor e obtenção do respetivo IP

Por fim podemos aceder ao site usando o URL "<http://172.16.161.137/mutillidae/>" no Firefox.



**Figura 10.** Acesso ao site multillidae

**1.2.2 Firefox** Agora temos de configurar o Firefox para permitir que o ZAP apanhe as trocas de pedidos e respostas entre o Firefox e o Servidor. Primeiro temos de ir ás definições de rede do browser e configurar as definições de ligação.

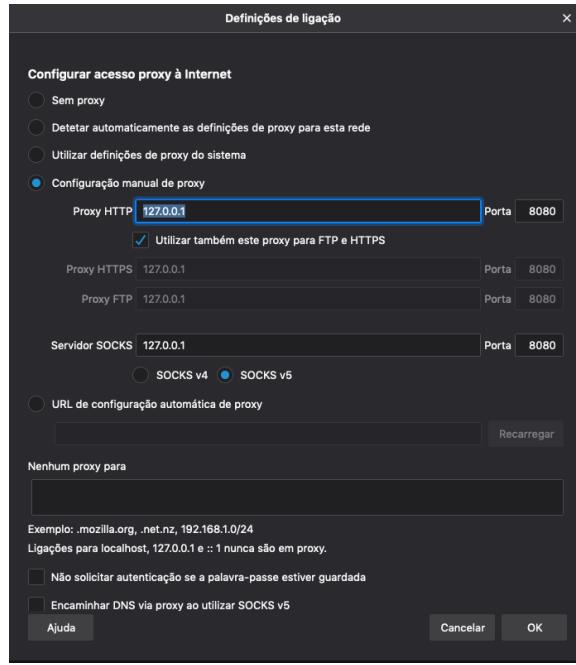
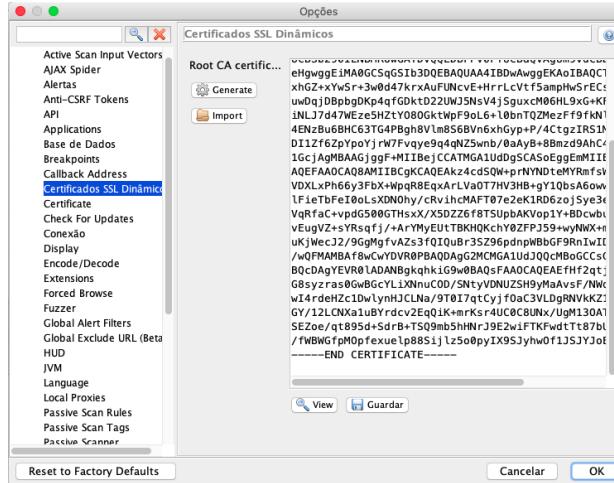


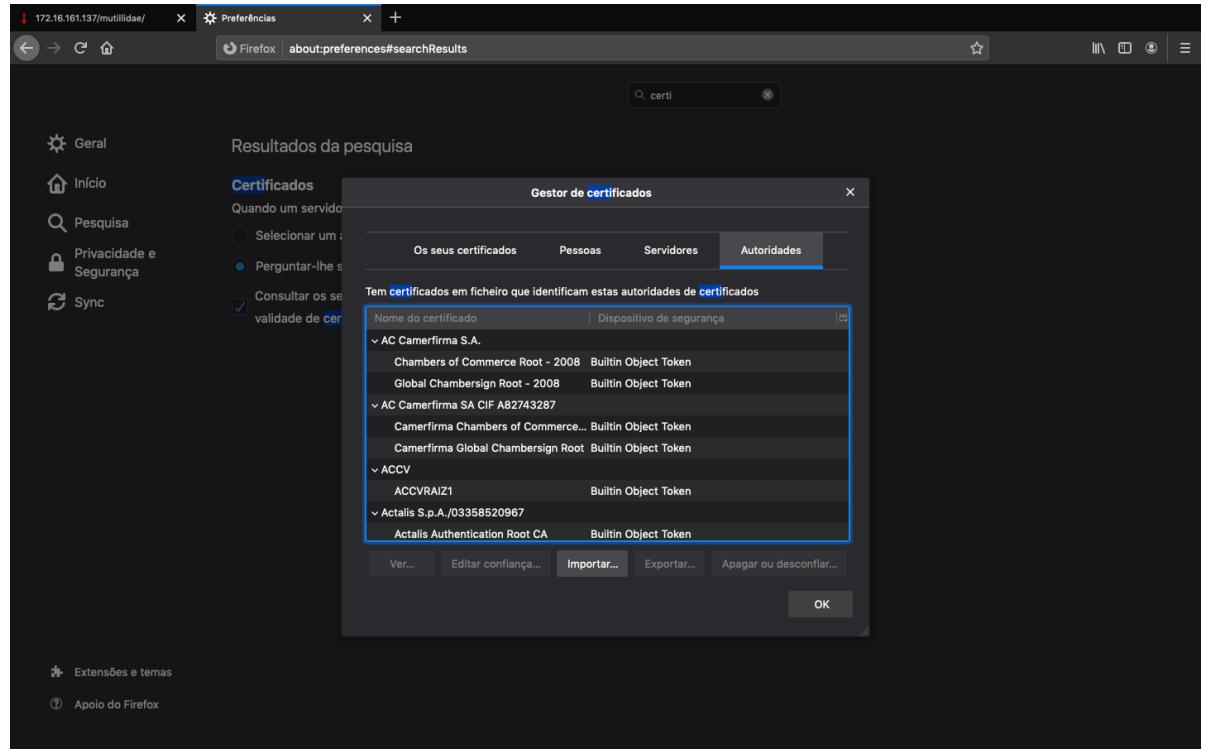
Figura 11. Definições de ligação do Firefox

Como vamos usar o OWASP ZAP temos de importar um certificado do mesmo. Podemos gerar um certificado e guarda-lo onde queremos para adicionar á lista de certificados de confiança do Firefox.



**Figura 12.** Geração do certificado SSL

Como referido anteriormente, agora basta-nos adicionar o certificado á lista de certificados confiaveis do firefox e podemos aceder ao site que preparamos através do mesmo sendo a informação recolhida pelo OWASP ZAP.



**Figura 13.** Adição do certificado SSL aos certificados de confiança do Firefox



**Figura 14.** Informação do acesso ao site através do Firefox coletada pelo ZAP

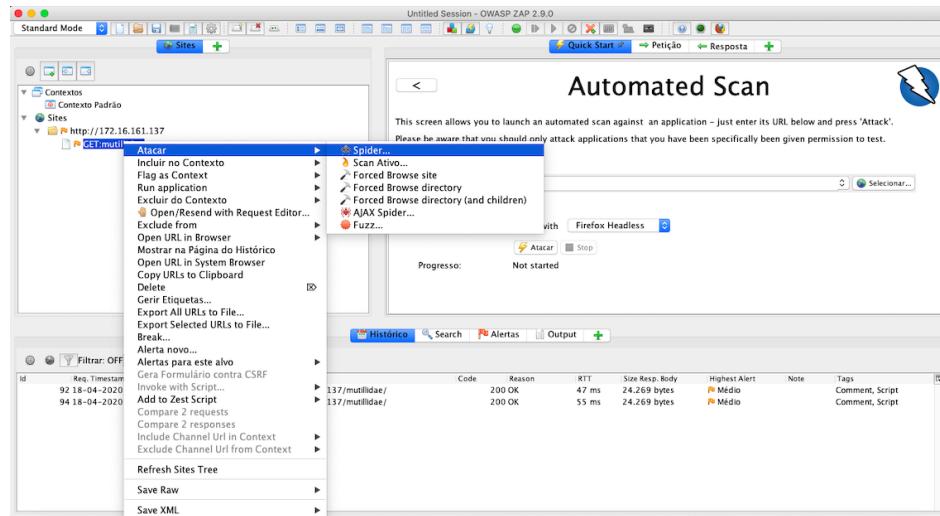
### 1.3 Análises

Nesta secção efetuaremos scans ao servidor apache na máquina virtual metasploitable que corre o mutillidae.

Os exemplos de testes que figuram neste relatório, estão divididos em duas categorias, exemplos básicos e exemplos avançados e são:

- Exemplos Básicos:
  - Spider a website
  - Fuzz a website
  - Scan Ativo
  - Mapear websites autenticados com Ajax
- Exemplos Avançados:
  - SQL Injections
  - Brute Force
  - Cross Site Scripting
  - Obter ficheiros escondidos no Servidor

**1.3.1 Spider a website** Spider é um scan passivo e inofencivo destinado a procurar em pedidos e respostas falhas como security headers ou anti CSRF tokens em falta. Vamos então realizar esta procura passiva fazendo uso do ZAP.



**Figura 15.** Procedimento para realizar um Spider Scan

Após termos realizado este procedimento podemos ver uma alteração na janela que mostrava os sites vistos.

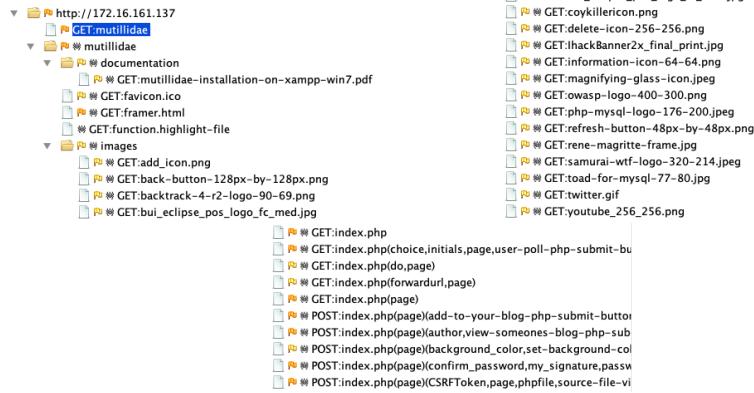


Figura 16. Resultado do Spider Scan

A cor da bandeira ao lado dos pedidos indica falhas na resposta recebida, quanto mais escura a cor mas grave é a falha. Também há uma janela onde são reportados os alertas para melhor visualização dos problemas. Em baixo podemos ver o conteúdo dessa janela.

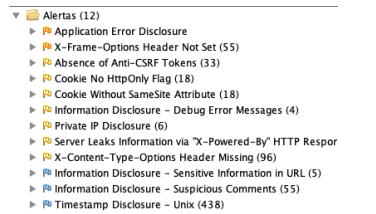


Figura 17. Alertas resultantes do Scan

**1.3.2 Fuzz a website** Para fins demonstrativos vamos usar a página de login para realizar o ataque fuzz. Nas figuras segunites vemos o resultado de tentar fazer login no servidor mutillidae com o username *admin* e a password *wrong-Password* bem como o respetivo pedido apanhado pelo ZAP.

The screenshot shows a login form with a red header bar containing the text "Authentication Error: Bad user name or password". Below this, a green bar says "Please sign-in". The form has two input fields: "Name" and "Password", both currently empty. A blue "Login" button is below the fields. At the bottom, there is a link "Dont have an account? Please register here".

**Figura 18.** Formulario de autenticação

The screenshot displays an HTTP POST request in a browser's developer tools. The request is sent to "http://172.16.161.137/mutillidae/index.php?page=login.php". The headers include standard browser information like User-Agent and Accept. The body of the request contains the parameters "username=admin&password=wrongPassword&login-php-submit-button=Login".

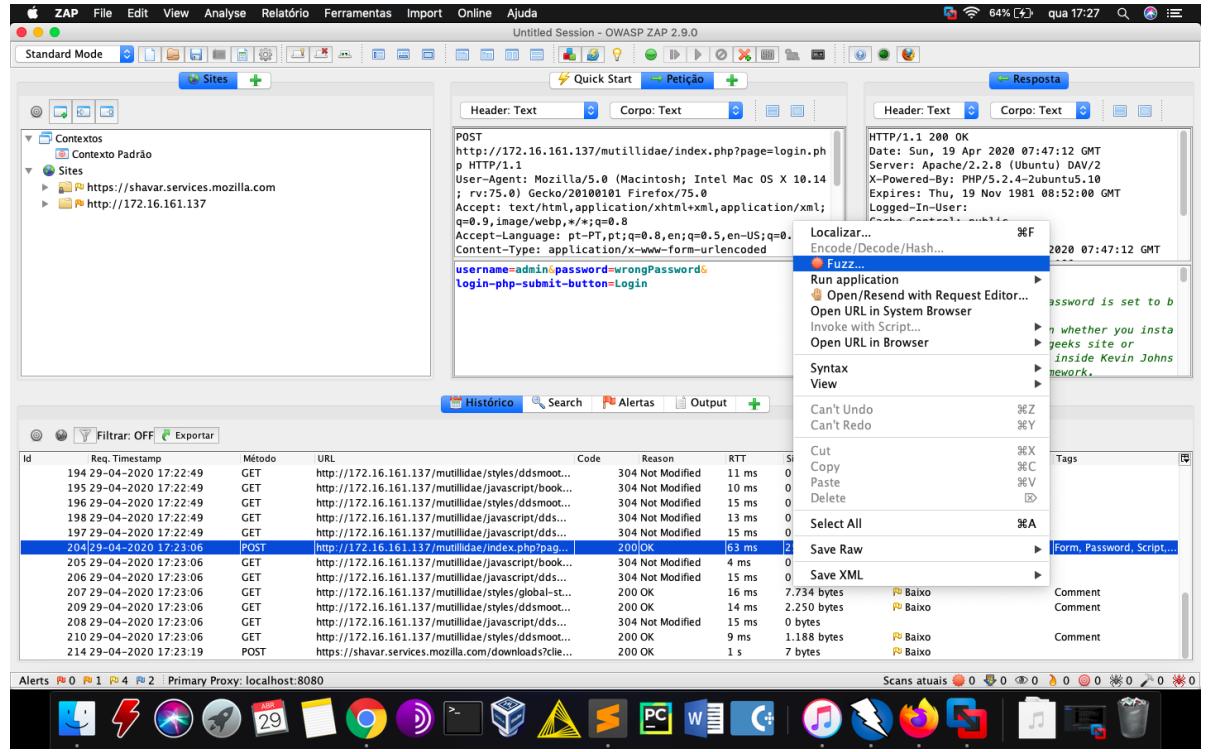
```

POST
http://172.16.161.137/mutillidae/index.php?page=login.php
HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14
; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,image/webp,*/*;q=0.8
Accept-Language: pt-PT,pt;q=0.8,en;q=0.5,en-US;q=0.3
Content-Type: application/x-www-form-urlencoded

username=admin&password=wrongPassword&login-php-submit-button=Login
  
```

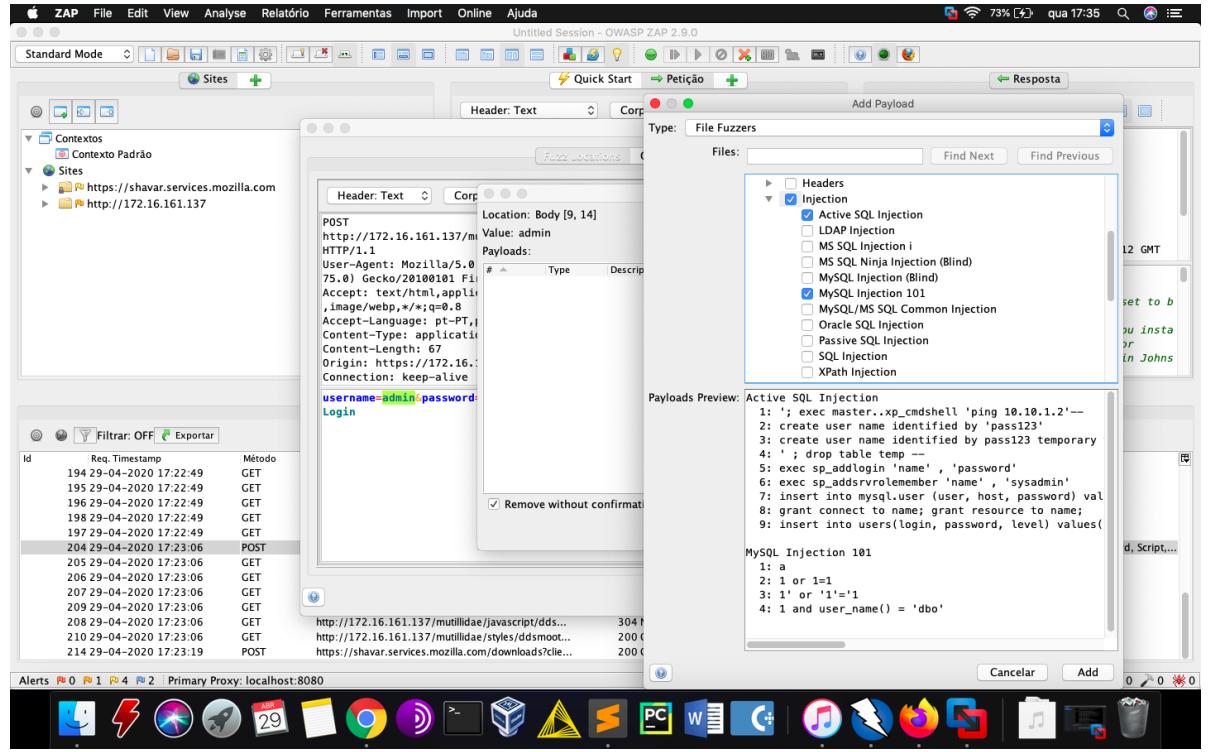
**Figura 19.** Pedido de autenticação submetido

Agora para realizarmos o fuzz temos de selecionar o pedido apanhado pelo ZAP com o username e a password errados e fazendo uso das opções disponibilizadas pelo botão direito do rato escolher Fuzz.



**Figura 20.** Abertura da janela de configuração do Fuzzer

Agora com a janela do fuzz aberta basta-nos selecionar o campo sobre o qual queremos executar o fuzz na mensage interceptada e carregar um payload para substituição. Neste caso como estamos a falar de um autenticação vamos escolher a opção *File Fuzzers* e dentro desta escolher *injeção SQL* e dentro de todas as opções vamos escolher apenas duas para manter o teste simples. Note-se que na segunda figura podemos ver no campo *Payloads Preview* onde é mostrado o que será registado no campo.



**Figura 21.** Configuração escolhida para o Fuzzer

Após escolhermos as opções basta carregar no botão *Start Fuzzer* e esperar. Como o fuzzer executa rápido obtemos respostas que podem ser ordenadas pelo tamanho de corpo de resposta e nas quais podemos ver o payload. Para uma busca mais eficaz podemos usar a barra search sobre o tipo *HTTP Fuzzer Results* para procurar palavras específicas como *error* ou *syntax*. Neste caso usamos a palavra *syntax* e se repararmos temos o campo *username* substituído por uma pilha, o que causa um erro no servidor. Se fizermos a substituição e submetermos o formulário de login na página acabamos por confirmar a existência de um erro como se vê nas figuras abaixo.

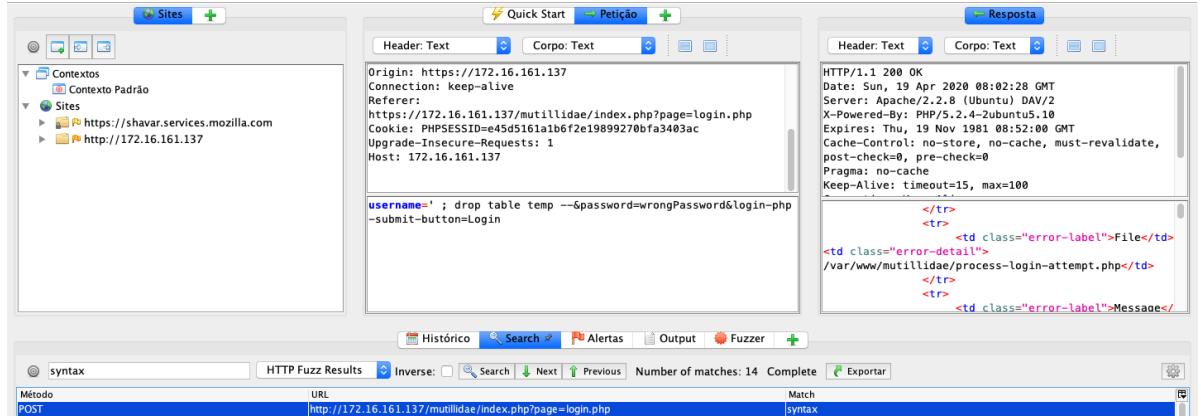


Figura 22. Exemplo de substituição do username

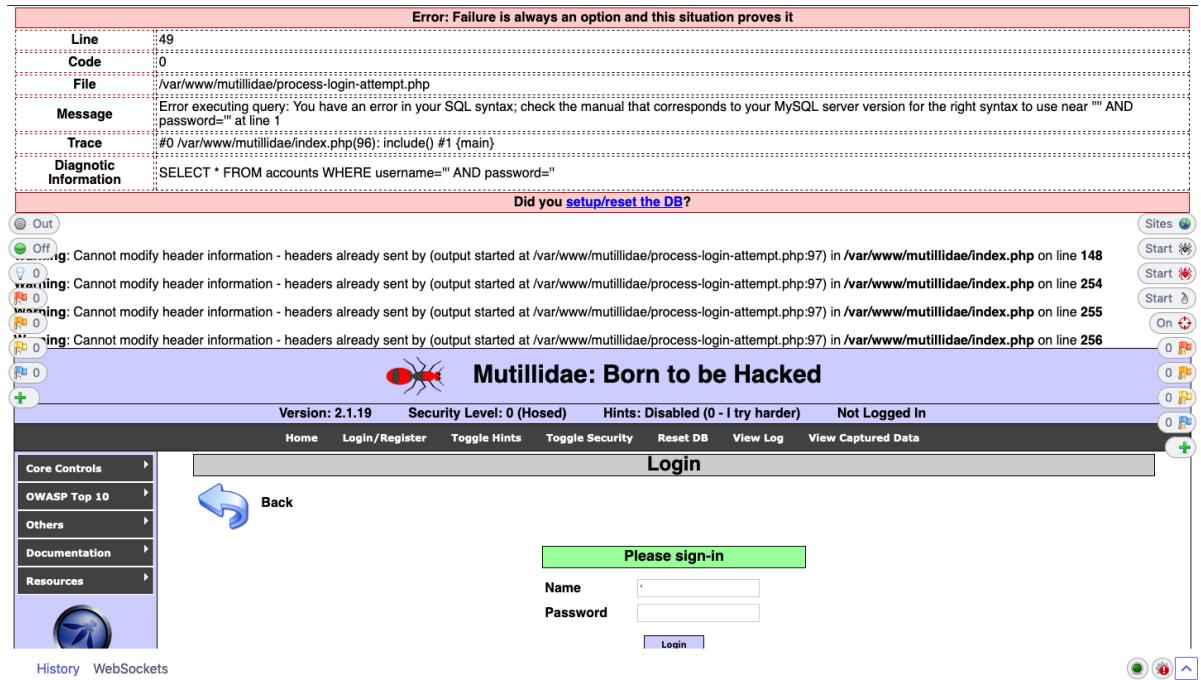
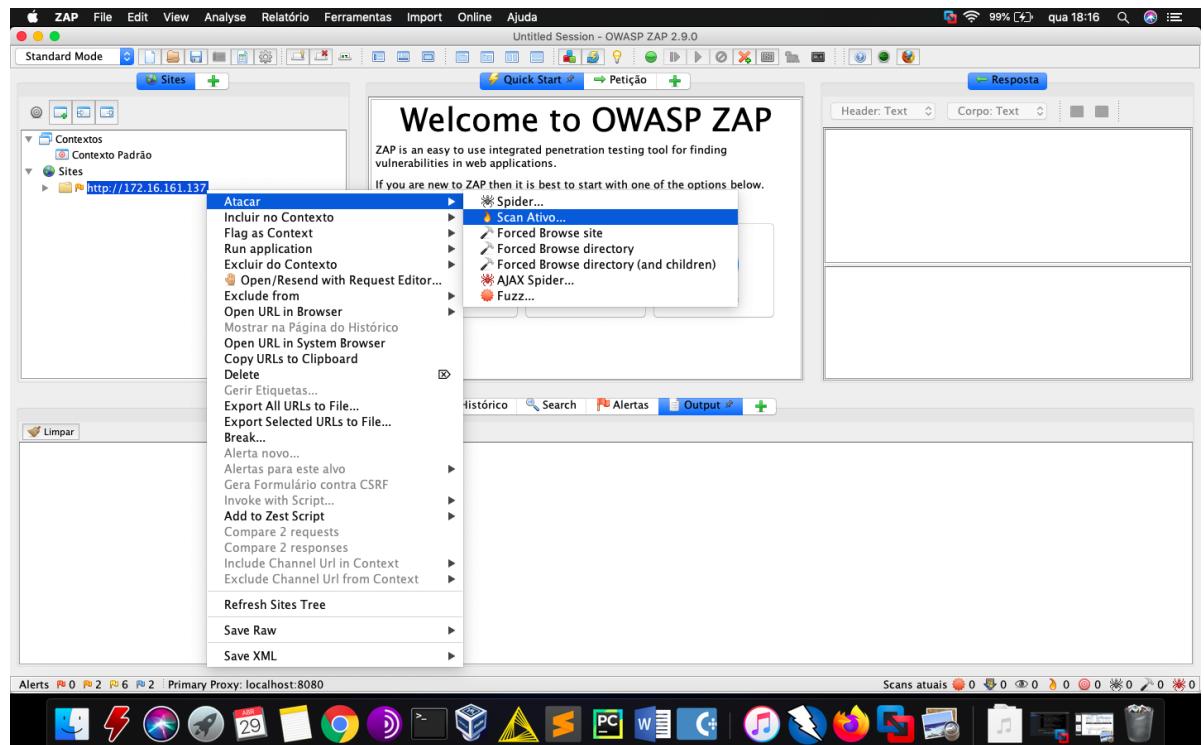


Figura 23. Resultado da substituição do username

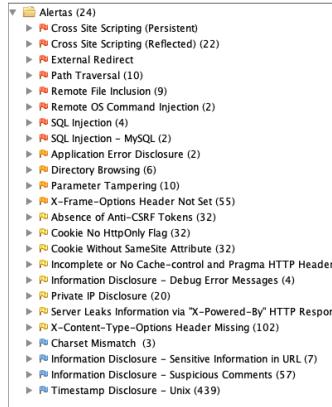
Para além disso note-se também que obtivemos informação útil sobre a base de dados. Agora sabemos que existe uma tabela SQL chamada accounts com os campos username e password.

**1.3.3 Scan ativo** Para fazer o scan ativo basta-nos simplesmente escolher um site e selecionar a opção scan ativo, este correrá um scan padrão pré-definido no ZAP.



**Figura 24.** Procedimento para realizar um Scan Ativo

Podemos ver o resultado do mesmo na figura abaixo, onde na tab Alertas obtemos todos os alertas que o scan padrão detetou etiquetadas pela gravidade da falha identificada pela cor da bandeira.

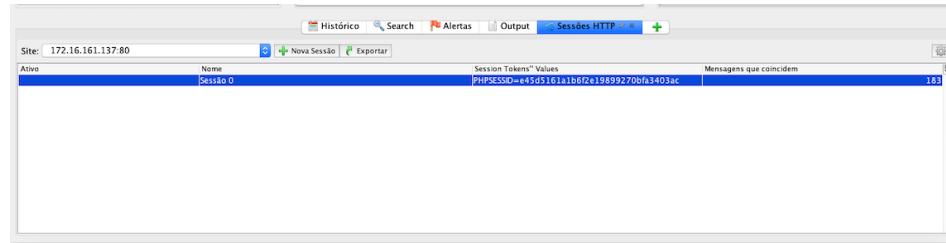


**Figura 25.** Alertas resultantes de realizar um Scan ativo

**1.3.4 Mapeamento de websites com autenticação de utilizador usando Ajax** A vantagem desta técnica de Spider é podermos aceder a URLs aos quais apenas utilizadores autenticados podem aceder, originando assim um mapeamento mais detalhado da organização do site.

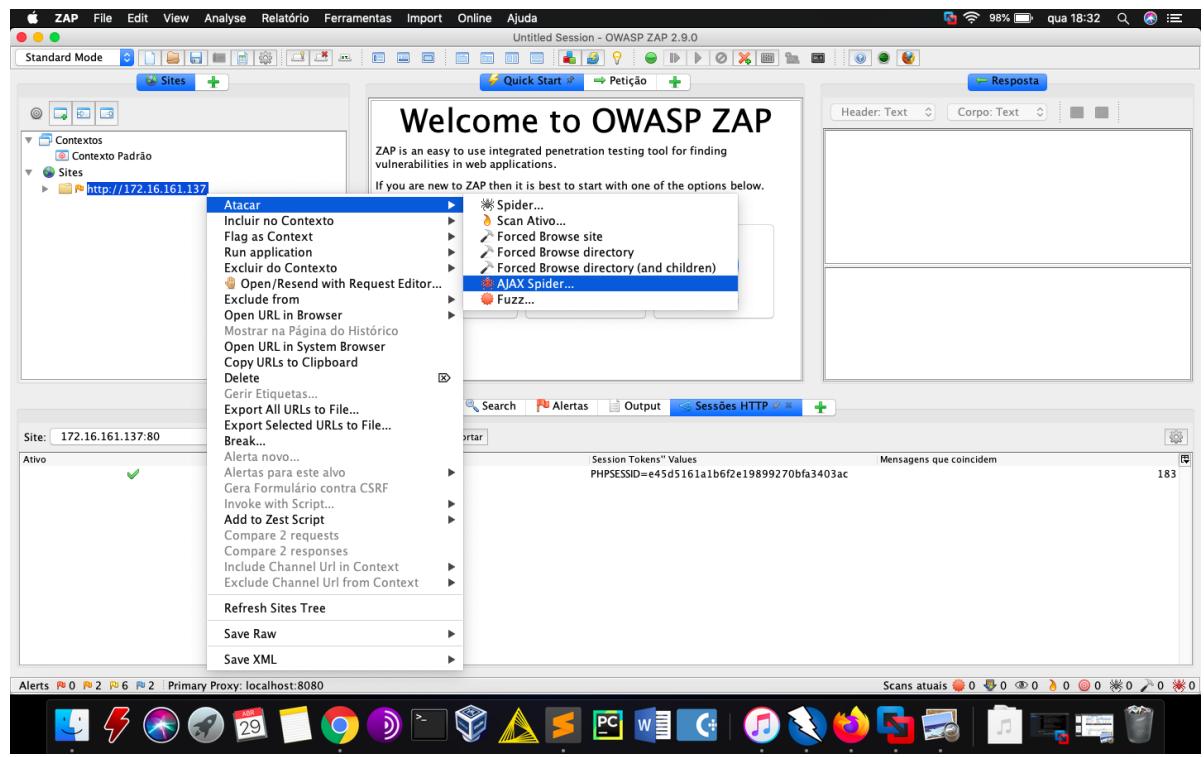
O primeiro passo para realizar este teste é ter descoberto uma combinação username-password válida e realizar a autenticação. Caso ainda não tenha realizado este passo por favor veja a subsecção de Brute Force.

Para garantir que se está autenticado caso observemos a tab HTTP session devemos ver uma imagem parecida com a seguinte.



**Figura 26.** Sessão HTTP por ativar

Devemos então clicar na barra correspondente à secção atual na janela e definir a mesma como ativa. Posteriormente na janela onde visualizamos os sites visitados devemos selecionar aquele com a sessão ativa e escolher a opção *Ajax Spider*. Desta forma o Ajax usará a nossa conta para realizar o Spider do website. Para além disso o Ajax Spider reautentica-se para garantir que caso exista a opção de *log off* o utilizador não seja desconectado.



**Figura 27.** Procedimento para realizar um mapeamento com Ajax após ativação da sessão

Por fim podemos ver os pedidos realizados pelo *Ajax* na janela inferior do ZAP bem como aceder á configuração do site alvo na tab correspondente aos sites.

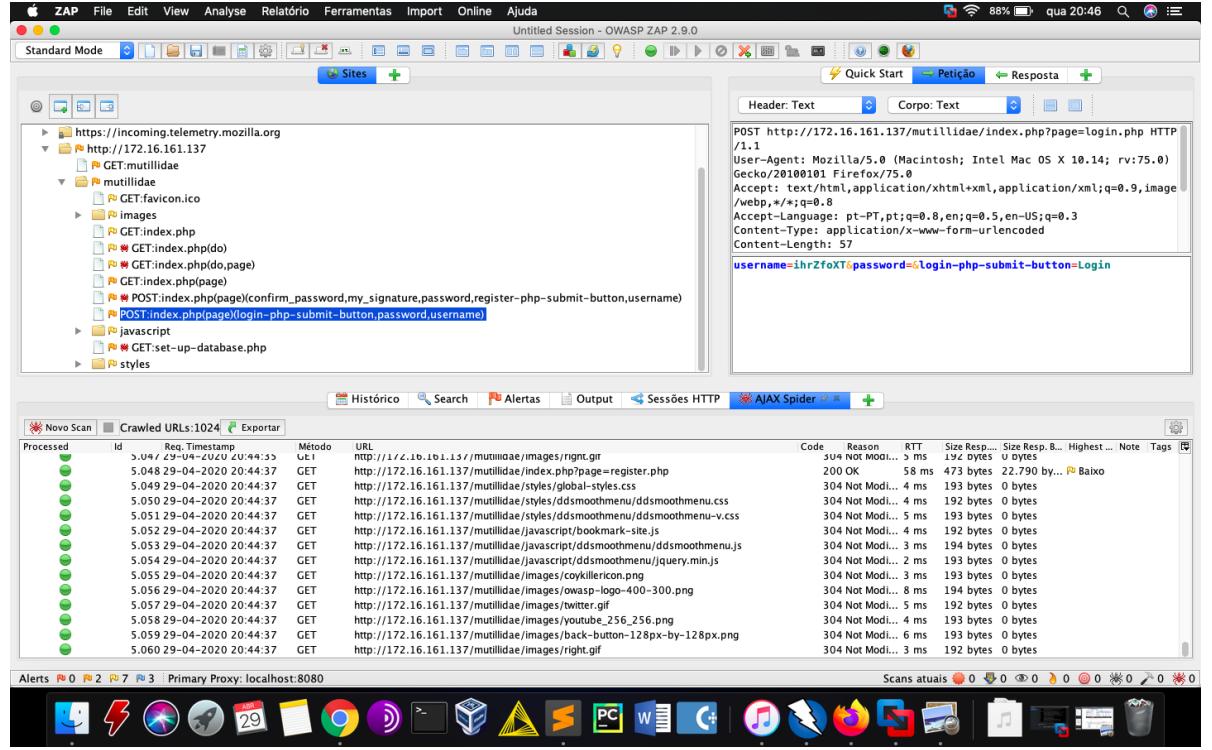


Figura 28. Site mapeado e pedidos feitos pelo Ajax

**1.3.5 SQL Injections** O propósito deste exemplo é demonstrar como injecções de código SQL podem comprometer a segurança. Primeiro vamos autenticar-nos no site e executar um scan ativo para testar o mesmo, como o ZAP esteve a guardar tudo em background ao executar o scan pode revelar mensagens interessantes. Como se pode ver na imagem temos 2 alertas de possíveis injecções de código SQL.

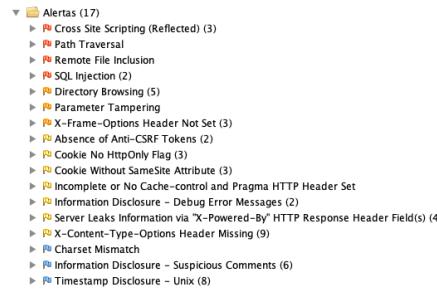


Figura 29. Alertas resultantes do Scan Ativo

Se nos aprofundarmos nos alertas podemos inspecionar o alerta e obter mais informação sobre a falha.

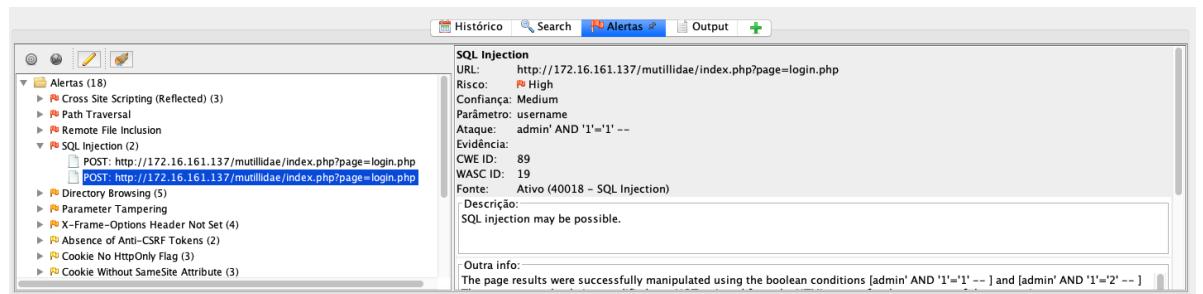
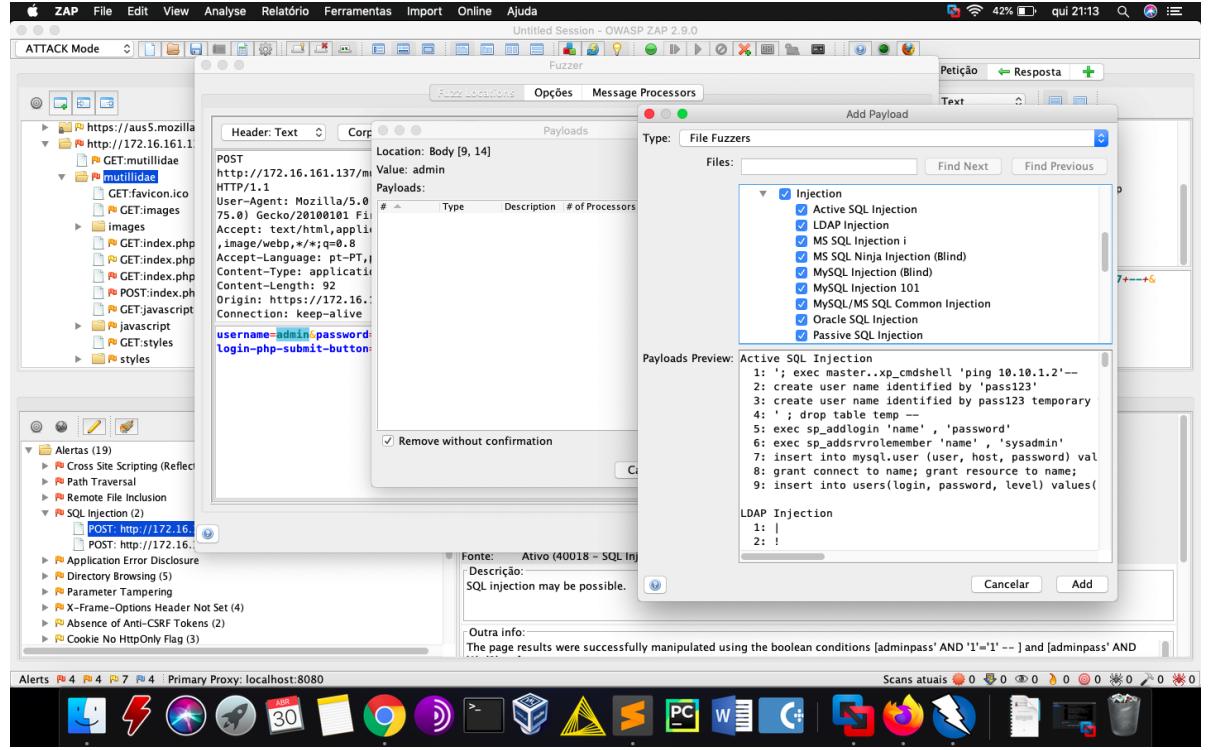


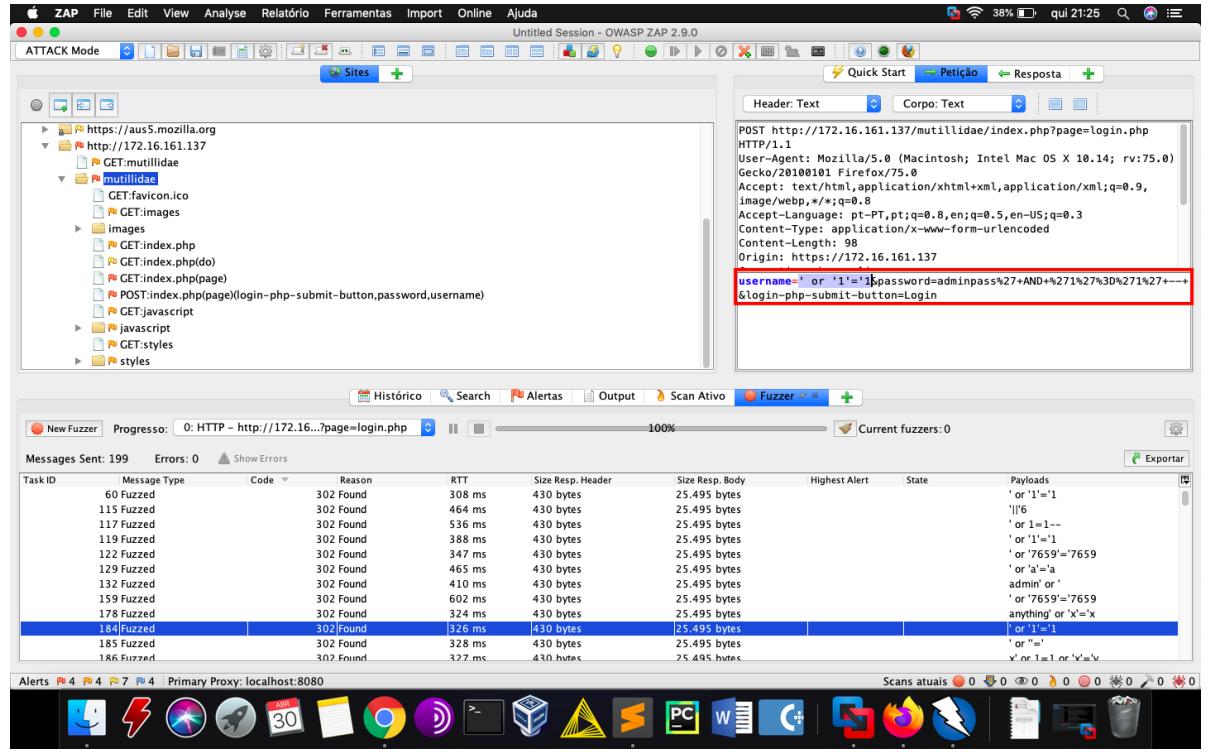
Figura 30. Alertas de SQL injection

Agora, basta-nos selecionar o campo do username e selecionar Fuzzer. No Fuzzer basta-nos adicionar uma nova *Fuzzer Location* carregando no botão add na janela do *Fuzzer*. Em seguida escolhemos todos os tipos de injeções MySQL, começando em seguida o scan.



**Figura 31.** Configuração do Fuzzer para Injeção de código SQL

Se olharmos para uma das mensagens etiquetada com a tag "*Found*", podemos ver o que o ZAP enviou no formulário. No caso da mensagem que podemos visualizar em baixo foi inserido no campo admin o código: '*or '1'='1*'.



**Figura 32.** Exemplo de injeção SQL bem sucedida

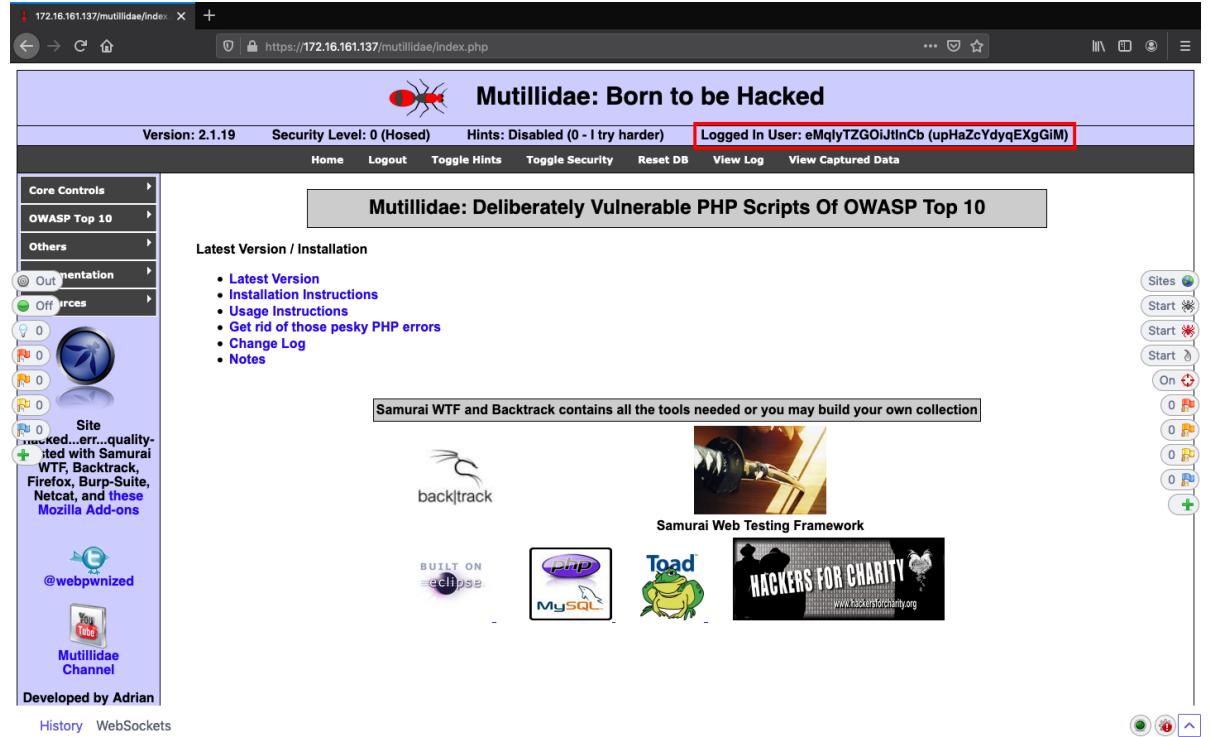
Se testarmos este código no campo do admin conseguimos acesso ao site.

Please sign-in

Name	<input type="text" value=" or '1='1"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

Dont have an account? [Please register here](#)

**Figura 33.** Injeção de código SQL manual

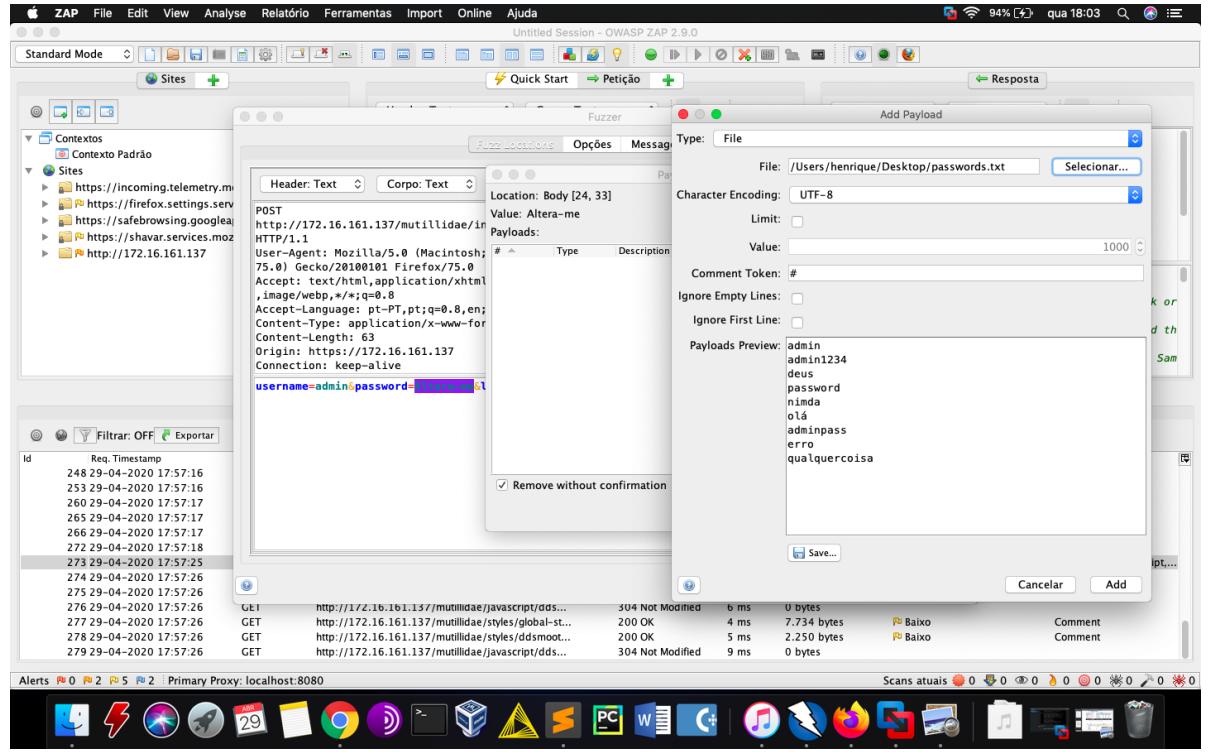


**Figura 34.** Login bem sucedido com injeção SQL

A lição a tirar neste teste é que não é feita nenhuma verificação do que foi inserido pelo utilizador, ou seja quem desenvolveu o acesso á base de dados confia cegamente no input fornecido pelo utilizador o que permite ataques deste tipo.

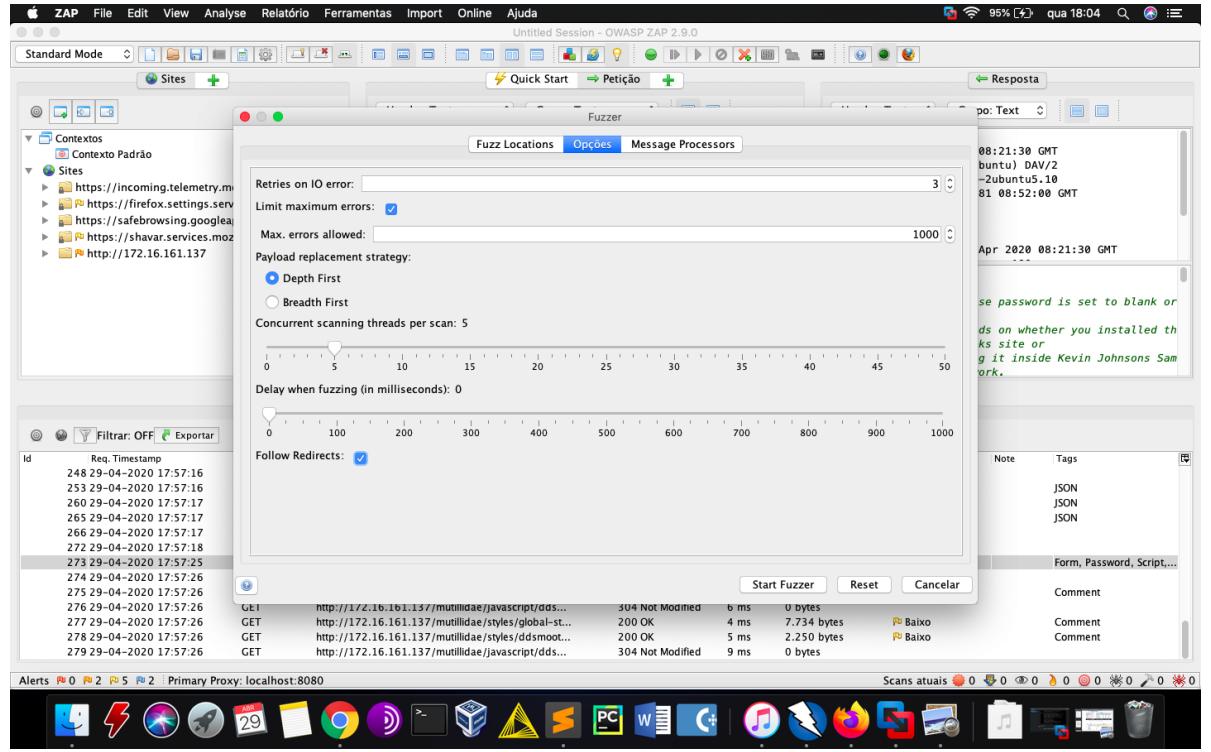
#### 1.4 Ataque Brute Force

Antes de executarmos este ataque é importante lembrar que este é um teste que demora bastante tempo a executar, por isso convém estar atento a mensagens que alguns sites dão como este *utilizador não existe* ou *password inválida*, que nos dão pistas sobre se devemos respetivamente mudar o username ou tentar passwords diferentes visto que o username existe na base de dados. Por isso devemos começar por tentar obter um username. Digamos que o username que identificamos é : *admin*. Para executar este ataque á semelhança do exemplo usado com o Fuzzer devemos colocar no campo de username o username válido e submeter o formulário. Para melhor compreensão o campo password será preenchido com a palavra "*Altera-me*". Neste caso tal como no fuzz selecionamos o pedido interceptado e usamos a opção Fuzzer. Na mensagem selecionamos a palavra *Altera-me* e adicionamos um ficheiro pré-gerado com palavras que podem ser uma password correta: *passwords.txt*.



**Figura 35.** Configuração do Fuzzer para usar ficheiro pré-gerado de possíveis passwords

Para termos uma forma de validação mais simples, caso a password esteja correta, podemos na tab opções escolher "Follow Redirect" para que mal haja uma mudança de página, ou seja, a password inserida foi a correta ou despoletou uma ação que nos levou a uma página diferente, o ZAP pare o scan e possamos observar como o último pacote enviado foi preenchido. Assim podemos tentar substituir a password no site e verificar se é a password correta ou não.



**Figura 36.** Configuração para parar no primeiro redirecionamento

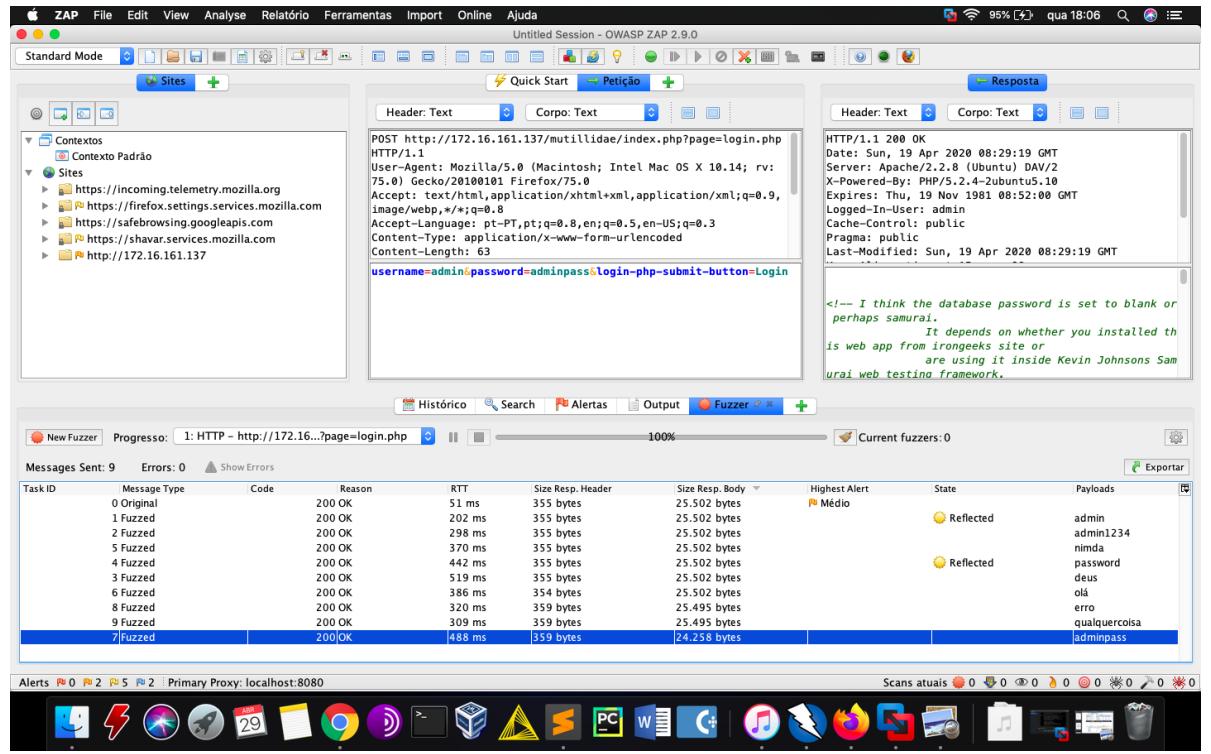
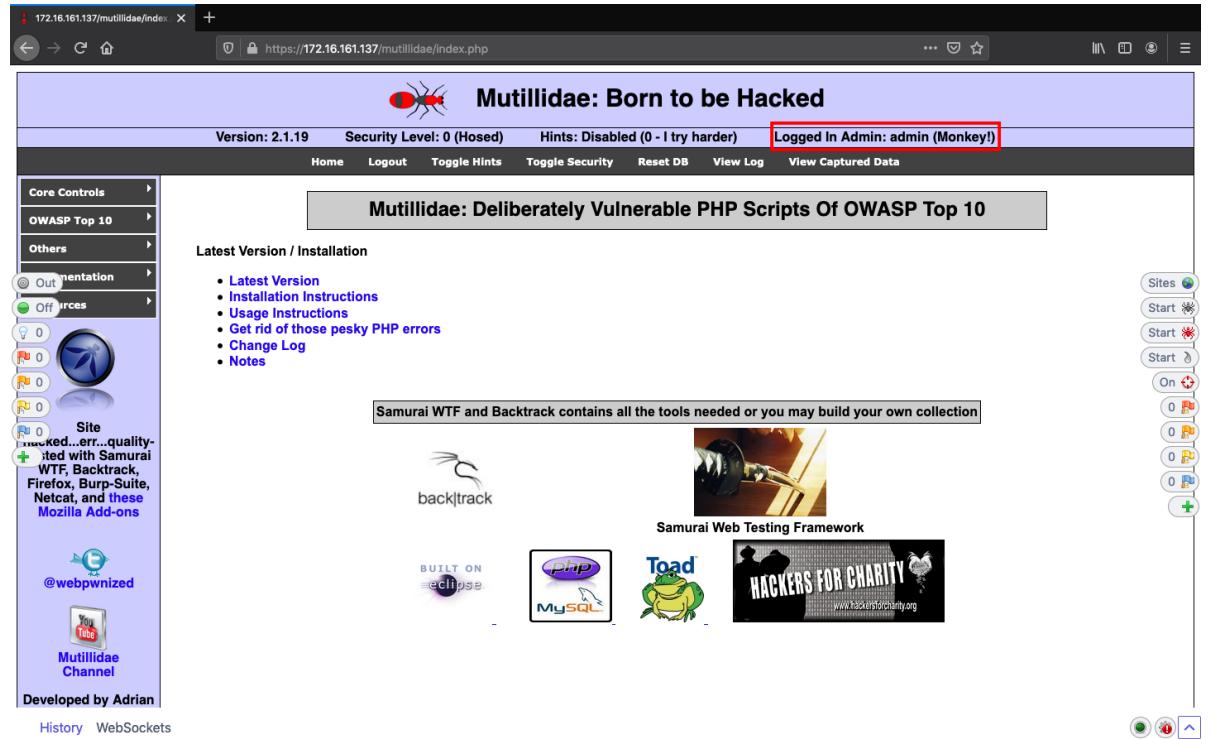


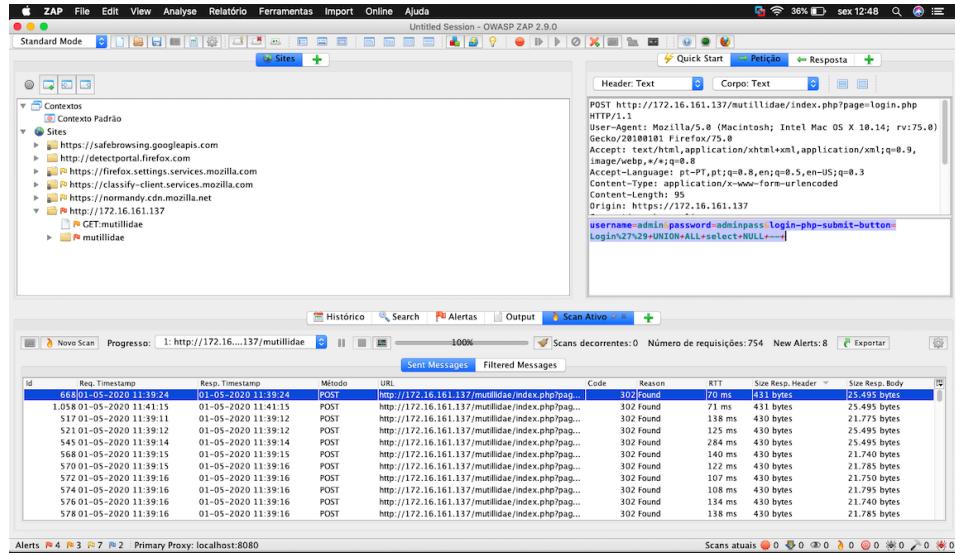
Figura 37. Resultado em que a página foi redirecionada

Na imagem acima podemos ver que o payload do campo password foi preenchido com a palavra *adminpass*, assim ao testarmos na página confirmamos que de facto esta é a password correta.



**Figura 38.** Resultados do teste do resultado anterior no site

**1.4.1 Cross Site Scripting** Neste teste vamos focar-nos em realizar o ataque quando o utilizador efetua o login. Fazendo uso do ZAP é possível efetuar um breakpoint nos pedidos do utilizador e nas respostas recebidas. A imagem seguinte mostra o pedido de autenticação do utilizador.



**Figura 39.** Pedido de autenticação do utilizador

Este ataque consiste portanto, em tentar alterar os campos enviados pelo utilizador para realizar este ataque.

Neste caso, vamos usar o Fuzzer para modificar o campo do username com as opções XSS: URI Cross Site Scripting, XSS Style injection e XSS XML injection<sup>1</sup>.

<sup>1</sup> Note-se que são apenas algumas opções, todas as opções apresentadas pelo ZAP são válidas.

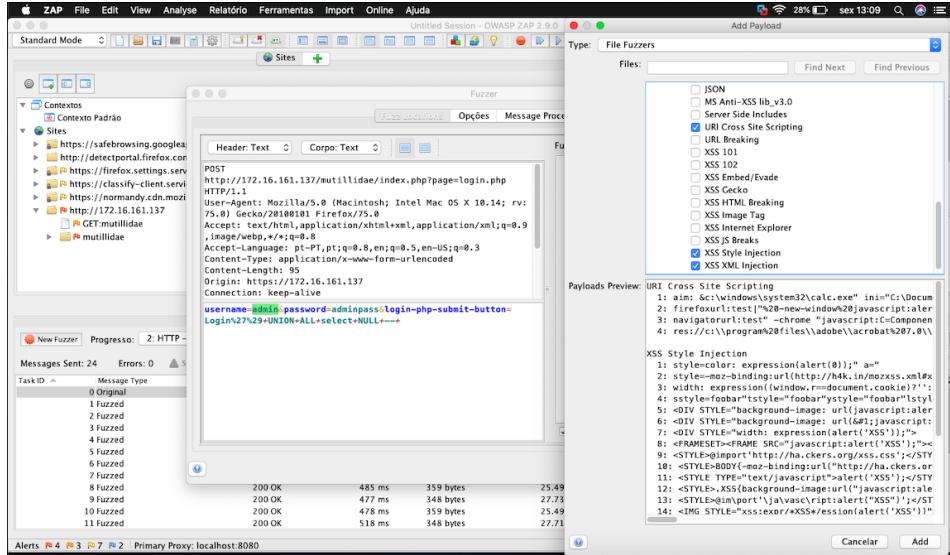


Figura 40. Opções de configuração do Fuzzer

Se observarmos agora os resultados do ZAP vemos que Aparecem alguns simbolos amarelos como a nota reflected ao lado. Estes são os ataques que foram bem sucedidos na injeção de código malicioso no campo do username.

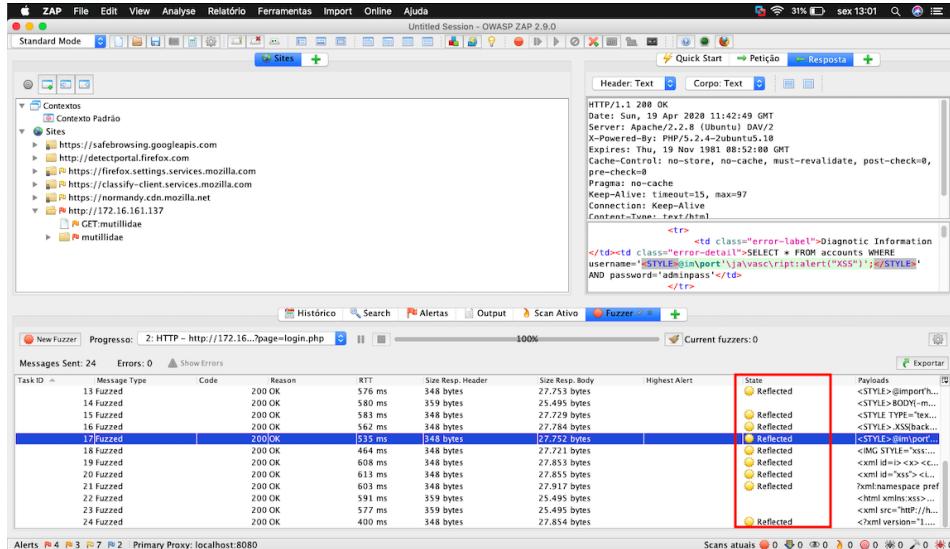
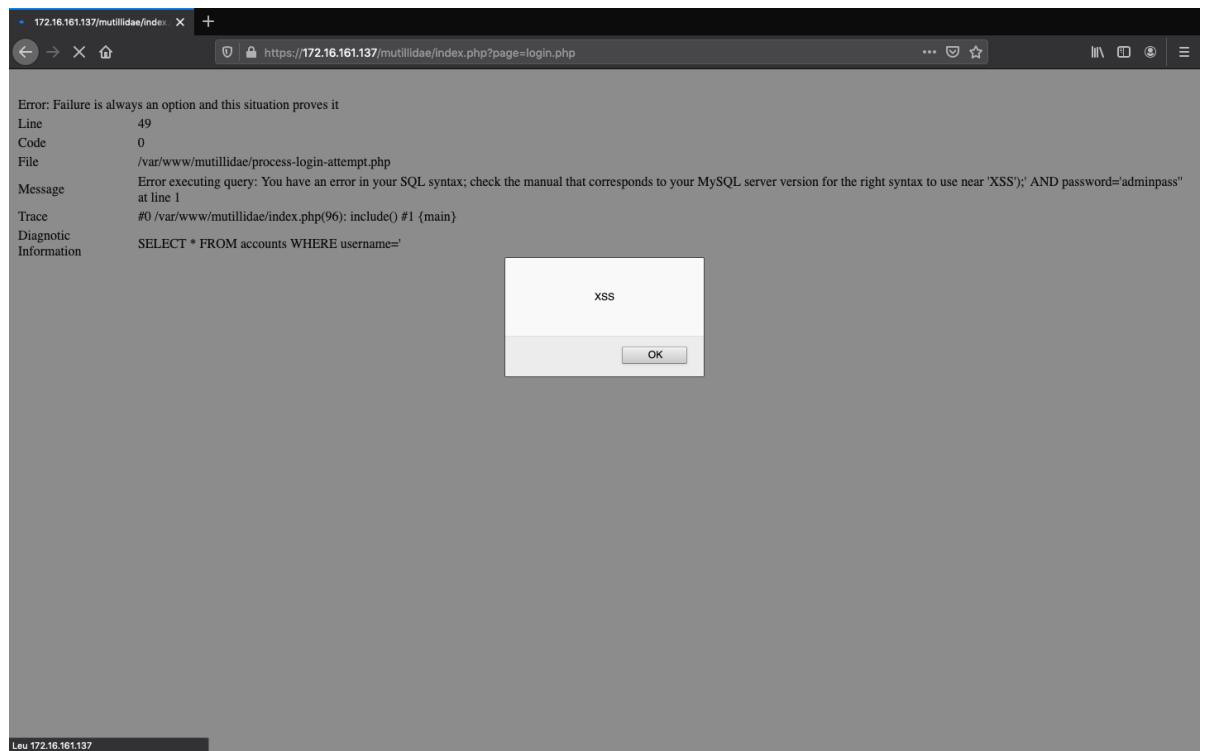


Figura 41. Resultados de injeção de código no campo username

Vamos então verificar o resultado no lado do utilizador caso trocassemos o campo username por um dos payloads possíveis que foram redirecionados. Neste caso para melhor visualização vamos usar o seguinte código: <script>alert('XSS');</script>



**Figura 42.** Teste de utilização de um dos payloads

**1.4.2 Obter ficheiros escondidos no Servidor** Neste ataque nós pretendemos descobrir ficheiros escondidos no servidor do site. Para isso podemos utilizar a opção "force browse directory (and children)". Nós utilizamos esta opção uma vez que a opção "force browse directory" não mostra tudo o que encontra.

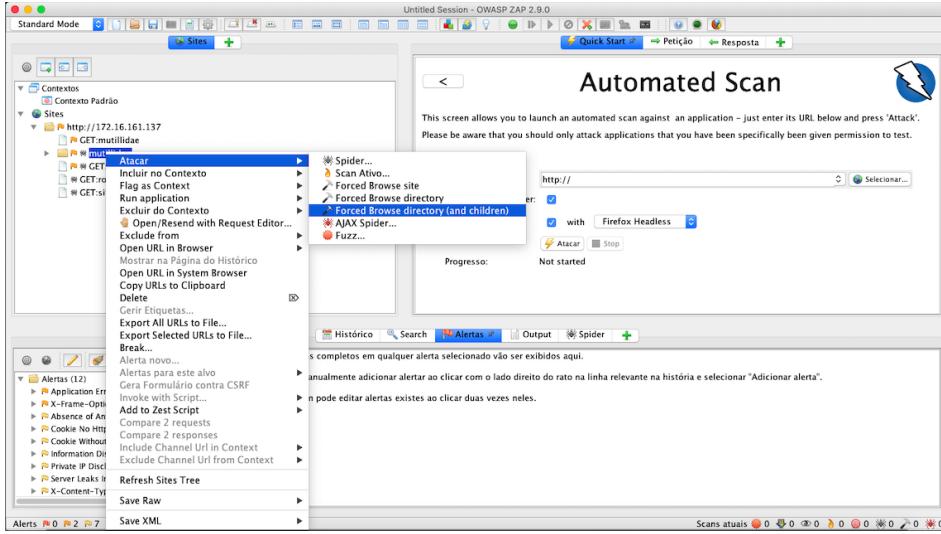


Figura 43. Procedimento de escolha de browse directory

É então aberta uma janela como se vê na figura seguinte. Onde selecionamos a lista que vem por defeito com o ZAP que este usará como base para uma estratégia "brute force" onde tentará encontrar os ficheiros escondidos listando-os. Em baixo podemos ver o resultado da procura.

Req. Timestamp	Resp. Timestamp	Método	URL	Code	Reason	Size Resp. Header	Size Resp. Body
18-04-2020 15:26:38	18-04-2020 15:26:38	GET	http://172.16.161.137:80/cgi-bin/	403	Forbidden	166 bytes	295 bytes
18-04-2020 15:26:38	18-04-2020 15:26:38	GET	http://172.16.161.137:80/	200	OK	176 bytes	891 bytes
18-04-2020 15:26:39	18-04-2020 15:26:39	GET	http://172.16.161.137:80/twiki/	200	OK	240 bytes	782 bytes
18-04-2020 15:26:39	18-04-2020 15:26:39	GET	http://172.16.161.137:80/dav/	200	OK	153 bytes	695 bytes
18-04-2020 15:26:39	18-04-2020 15:26:39	GET	http://172.16.161.137:80/dwqa/	302	Found	399 bytes	0 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/phpMyAdmin/	200	OK	846 bytes	0 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/mutilldiae/	200	OK	326 bytes	0 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/twiki/readme.txt	200	OK	243 bytes	4.334 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/wiki/license.txt	200	OK	244 bytes	19.440 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/TWikiDocumentation....	200	OK	245 bytes	453.477 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/wiki/TWikiHistory.html	200	OK	243 bytes	52.417 bytes
18-04-2020 15:26:40	18-04-2020 15:26:40	GET	http://172.16.161.137:80/wiki/bin/	403	Forbidden	166 bytes	297 bytes

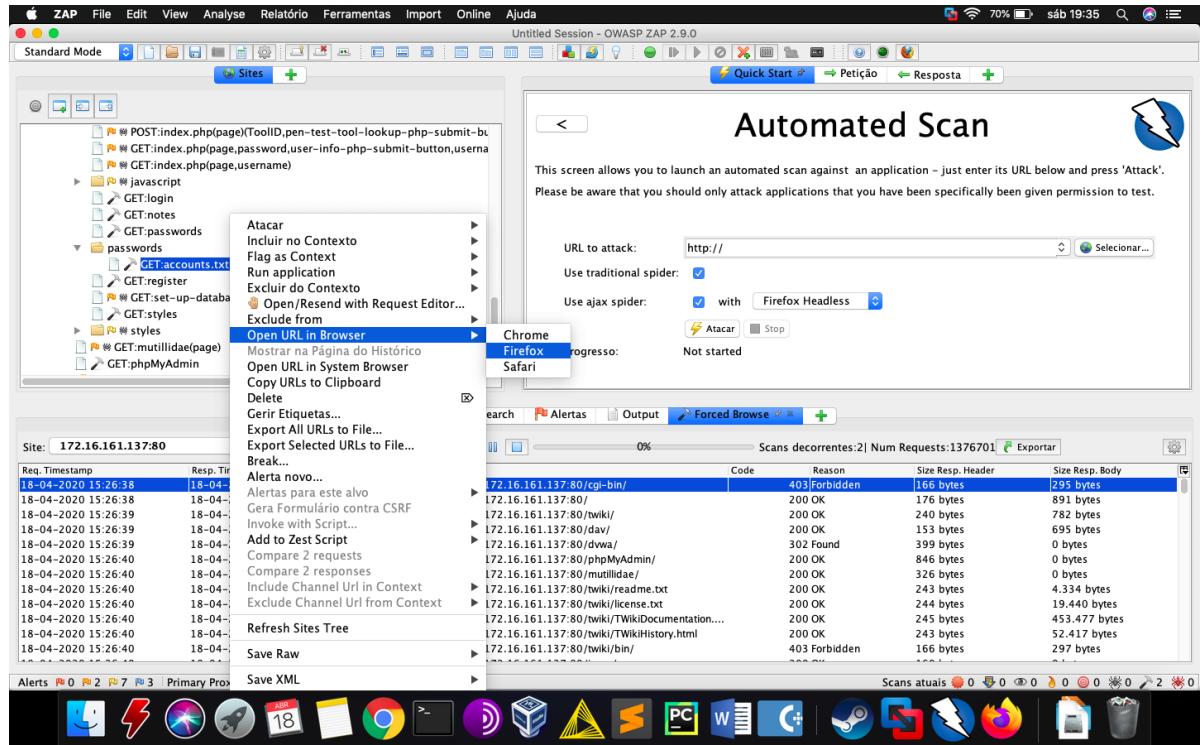
Figura 44. Resultados da procura

Os resultados encontrados são listados como se pode ver na figura assim com vários código associados:

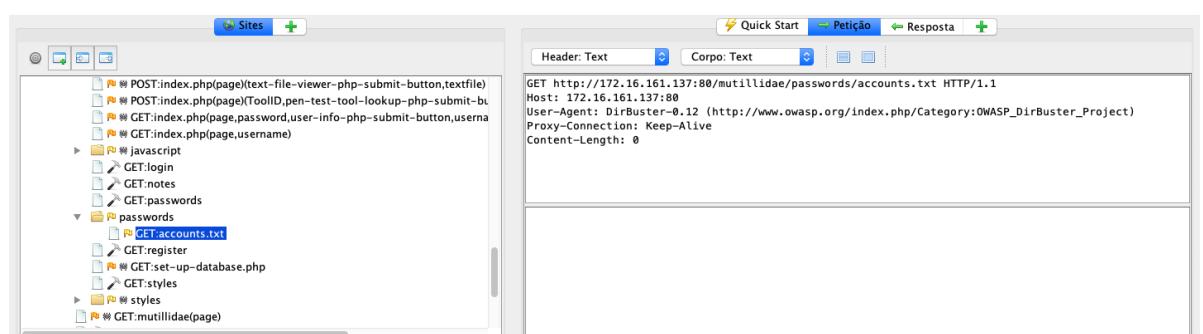
-200 Ok -403 Forbiden -302 Found/Moved -400 Bad Request -500 Internal Server Error

É no entanto mais facil procurar na janela associada aos pedidos "Get" por palavras chave tais como password ou node como se pode verificar na seguinte figura e posteriormente inspecionar o conteúdo encontrado abrindo uma janela

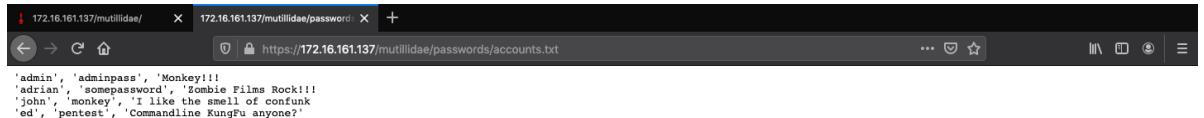
no firefox através do ZAP ou lendo o endereço do pedido get feito e inserindo o url diretamente no firefox obtendo, neste caso, uma lista de usernames e as respetivas passwords.



**Figura 45.** Abertura do URL proibido através do ZAP



**Figura 46.** Pedido com o URL proibido



A screenshot of a web browser window. The address bar shows two tabs: "172.16.161.137/mutillidae/" and "172.16.161.137/mutillidae/password". The active tab displays the URL "https://172.16.161.137/mutillidae/passwords/accounts.txt". The page content is a plain text dump of user credentials:

```
'admin', 'adminpass', 'Monkey!!!'  
'adrian', 'somepassword', 'Zombie Films Rock!!!  
'john', 'monkey', 'I like the smell of confunk  
'ed', 'pentest', 'Commandline KungFu anyone?'
```

**Figura 47.** Resultado da abertura do URL desprotegido

## Referências

1.