11

Teste com SQLMAP

Transcrição

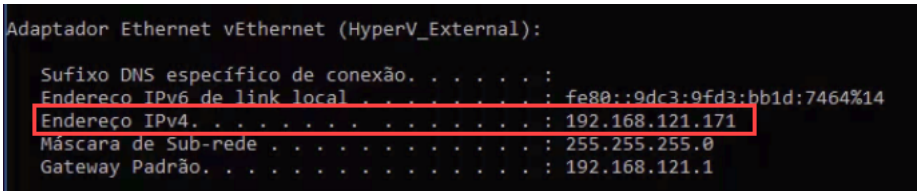
O usuário Alex conseguiu efetuar o login na aplicação da Alura Shows como se fosse o Fernando. Imagine o problema que isso pode causar, o Alex poderia comprar diversos ingressos, pegar informações pessoais, de pagamentos e assim por diante.

Sabemos que esse problema é necessário correção, mas quais outras informações conseguimos pegar com as injeções de SQL?

O Alex, com o intuito de descobrir mais sobre a aplicação, resolveu automatizar o processo de injetar código SQL. Para isso, ele resolveu usar uma distribuição do Linux que já possui diversas ferramentas que testam vulnerabilidades em aplicações. Essa distribuição é o Kali Linux.

Estamos utilizando Windows e para conseguirmos trabalhar com o Kali Linux, precisaremos de um ambiente virtualizado para comportar outro sistema operacional. Instalaremos o Virtal Box e dentro dele teremos a máquina virtualizada com o Linux. Não se preocupe que nos exercícios colocaremos todos os passos para a instalação e configuração.

Começaremos acessando a aplicação da **Alura Shows** no Linux. Vemos que a aplicação está rodando no ***Localhost:8080***, esse domínio é pertencente ao Windows. Para acessarmos do Linux, precisaremos do endereço **IP** do Windows, que é onde o Tomcat está executando a aplicação. No **Prompt de Comandos**, digitaremos `ipconfig` e pressionaremos a tecla "Enter". Em ***Adaptador Ethernet vEthernet (HyperV_External)***, pegaremos o valor **Endereço IPv4**. Lembrando que **em seu computador, o valor de IP será diferente**, no computador da Alura o IP é `192.168.121.171`.



Na máquina do Linux, abriremos o navegador **Firefox** e acessaremos o endereço:

192.168.121.171/alura-shows/

Dessa forma, conseguimos acessar a aplicação da Alura Shows de outro computador. Faremos o *login* convencional utilizando a conta do **Alex**, no campo **E-mail** colocaremos `alex@gmail.com` e no campo **Senha** colocaremos `123`.

Repare que a URL está passando os valores de e-mail e senha. Copiaremos essa URL e utilizaremos na ferramenta do Kali que injetará diversos códigos SQL, tentando obter mais informações do banco de dados. Essa ferramenta é o `sqlmap`. **Mas muito cuidado, não é aconselhável utilizar essa ferramenta em sistemas de terceiros sem a autorização das mesmas, isso é um ato ilegal.**

Vamos perguntar para o `sqlmap`, qual o banco de dados usado na aplicação. Na *Exception* vimos que foi solicitado verificar o manual do MySQL, mas com a ferramenta nós confirmaremos.

Passaremos par o `sqlmap` a URL que ele irá verificar. O comando completo será:

```
sqlmap -u "http://192.168.121.171/alura-shows/login?email=alex%40gmail.com&senha=123&login-subm
```

Após executar o comando, nos será perguntado se queremos continuar.

> how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit]

Como queremos continuar colocaremos "c". Em seguida, o `sqlmap` informará que encontrou um redirecionamento, e pergunta se queremos segui-lo.

> sqlmap got a 302 redirect to 'http://192.168.121.171/alura-shows/usuario (http://192.168.121.171/alura-shows/usuario)'. Do you want to follow? [Y/n]

Colocaremos o "n" pois não é necessário. Agora ele informa que o banco de dados parece ser o MySQL, e pergunta se queremos pular os testes para os outros bancos de dados.

> it looks the back-end DBMS is 'MySQL'. Do you want to skip test playloads specific for other DBMSes? [Y/n]

Colocaremos "Y" para pular os outros testes, temos também a confirmação da *Exception* que o banco é mesmo o MySQL. Depois ele pergunta se queremos fazer mais testes para o MySQL.

> for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]

Colocaremos "Y" novamente. Dessa forma, o `sqlmap` começara a inserir diversos comandos SQL nos parâmetros da URL, que são o `email` e `senha`. O teste pode demorar um pouco, basta aguardar.

Ao terminar os testes, o `sqlmap` informa que o parâmetro `email` é vulnerável, e pergunta se queremos continuar os testes em outros parâmetros.

> GET parameter 'email' is vulnerable. Do you want to keep testing the others (if any)? [y/N]

Como já descobrimos a vulnerabilidade do `email`, colocaremos "n". No fim, ele confirma novamente que o banco é o MySQL.

Agora tentaremos descobrir qual banco a aplicação da Alura Shows está usando **dentro do MySQL** para cadastrar os usuários. Voltaremos o mesmo comando inicial - basta pressionar a seta para cima no teclado - onde passamos a URL, mas dessa vez, adicionaremos `--current-db`, que solicita o banco atual. O comando completo será:

```
sqlmap -u "http://192.168.121.171/alura-shows/login?email=alex%40gmail.com&senha=123&login-subm
```

Ao executar, o `sqlmap` responderá que a aplicação está usando é o `owasp`.

> current database: 'owasp'

Agora tentaremos descobrir as tabelas contidas do banco `owasp`. Para isso, colocaremos o comando o parâmetro `--tables -D owasp`. O comando completo será:

```
sqlmap -u "http://192.168.121.171/alura-shows/login?email=alex%40gmail.com&senha=123&login-subm
```

O `sqlmap` informa que o banco possui quatro tabelas, `depoimento`, `role`, `usuario` e `usuario_role`. Se acessarmos o banco de dados que instalamos para o curso, veremos que contém essas tabelas.

Pediremos para o `sqlmap` descarregar as informações que estão na tabela `usuario`. Para descarregar usamos o `--dump` passando a tabela `-T usuario` e o banco `-D owasp`. O comando completo será:

```
sqlmap -u "http://192.168.121.171/alura-shows/login?email=alex%40gmail.com&senha=123&login-subm
```

O `sqlmap` trouxe todos os resultados da tabela `usuario`, mostrando as informações dos usuários **Alex** e **Fernando**. Essa informações precisam estar protegidas.

Na próxima aula, veremos como corrigir essa vulnerabilidade.