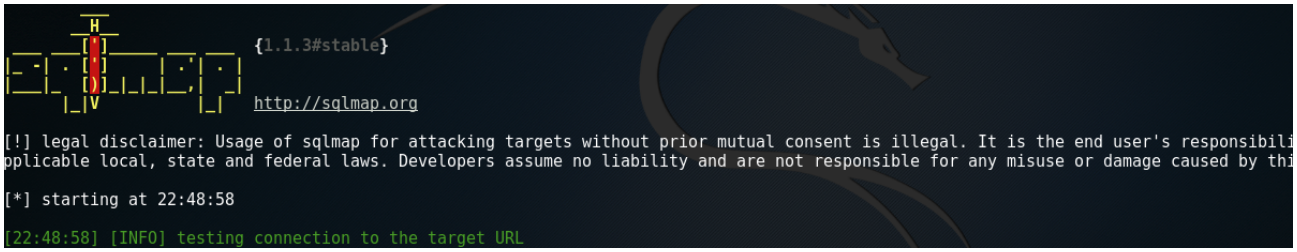
SQLi-Exploits.md

SQLMap

Processos de analise de vulnerabildiades baseados em SQL tendem a ser longos e massivos principalmente quando se trata de Blind SQL Injection, nesses cenário é necessário a execução de multiplas querys e analise de retorno uma vez que as preciosas excessões não são devolvidas em tela, para execução deste tipo de teste pode ser auxiliada por ferramentas como a ferramentas deste teste, o SQLMap;



O SQLMap é uma poderosa e versátil ferramenta do tipo Open Source escrita por Bernardo e Miroslav para detectar e explorar dinamicamente vulnerabilidades baseadas em SQLi. Uma das caracteristicas relativas a essa versatilidade é o suporte a uma boa quantidade de softwares DBMS: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB e HSQLDB.

Informações sobre o projeto podem ser consultadas na página oficial no endereço sqlmap.org;

SQLMap contém uma ampla gama de recursos alguns dos mais relevantes estão listados abaixo:

- Suporte para diferentes tipos de técnicas de injeção SQL como:
    - Error-based Injection;
    - Blind Injection;
    - Time-based Injection;
    - Stacked queries;
- Atuar como um cliente de banco de dados se credenciais forem fornecidas;
- Download e upload de arquivos para o servidor de banco de dados;
- Capacidade de explorar bancos de dados, tabelas e colunas individualmente;
- Suporte interno para cracking de hashes mais antigos como por exemplo MD5;
- Suporte para integração com o framework Metasploit;
- Execução de código explorando recursos do DBMS como por exemplo o xp_cmdshell;

Instalação do SQLMap no ambiente Kali Linux

O SQLMap é um recurso nativo do Kali e portanto já vem pré-instalado, entretanto a versão instalada por default possui certos bugs não sendo a mais recomendada para o uso do mundo real, dessa forma faremos a instalação da última versão estável do SQLMap a partir da sua página no Github.

Faça o donwload do código fonte da ultima release do SQLMap:

```
cd /opt
wget https://github.com/sqlmapproject/sqlmap/archive/1.1.3.tar.gz -q
tar -xvf 1.1.3.tar.gz
cd /opt/sqlmap*
./sqlmap.py -h
```

Você também pode clonar o repositório obtendo a ultima versão do SQLMap:

```
git clone https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

Um bom começo é dar uma olhada no arquivo README.md para obter detalhes sobre o funcionamento da aplicação.

Técnicas de Injection

O SQLmap suporta o uso de técnicas especificas de varredura baseadas no tipo de exploit a ser explorado, para isso utilize o parâmetro ***--technique*** na linha de comando, a tabela abaixo relaciona quais de técnicas disponíveis na ferramenta:

Letter	Technique
B	Boolean-based blind or simply blind injection
E	Error-based injection
U	UNION-query based injection
S	Stacked queries
T	Time-based injection
Q	Inline queries

Um detalhamento de cada uma dessas técnicas pode ser obtido diretamente na Wiki do Projeto SQLMap.

Por padrão, o SQLMap seleciona a técnica mais apropriada de acordo com a URI passada como alvo, na maioria dos casos o efeito natural é que várias técnicas sejam aplicadas a medida que o alvo apresentar-se suscetível a tipos específicos de exploits, por exemplo se uma URI responde diretamente a teste de inserção de código sem tratamento o SQLMap aplicará a técnica ***Boolean-based blind or simply blind injection***

Neste exemplo utilizariamos o SQLmap para escanear epscificamente utilizando a técnica ***Error-based injection*** ou seja, faremos uma análise especifica para uma aplicação que estea retornando mensagens de erro em tela:

```
cd /opt/sqlmap*
./sqlmap.py -u "192.168.X.X/uri..." --technique=E
```

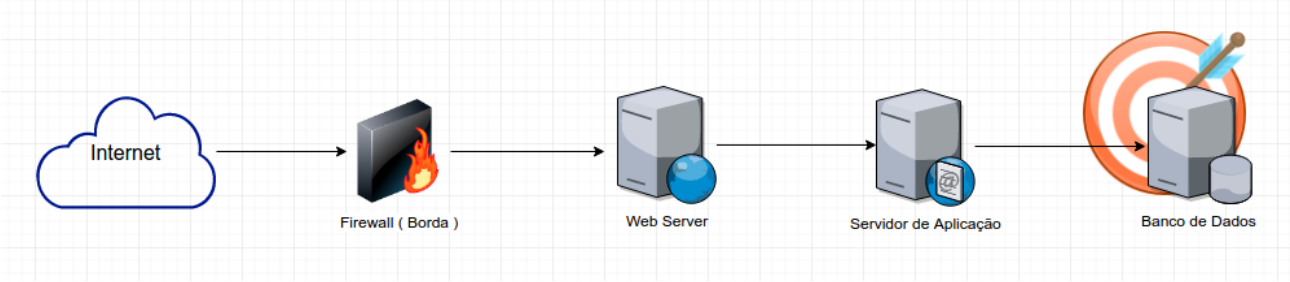
Substitua a URL acima pelo servidor de destino, a URL do multillidae ("OWASP 2013 > A1 - Injection(SQL) > SQLi - Extract Data > User Info (SQL)"), do OWASP Broken Web Applications Project pode ser utilizada como destino valido para seus testes, faça uma tentativa de autenticação simples e copie a URI para o SQLmap, o formato seria algo similar ao endereço abaixo.

> Em casos onde deseja-se ignorar testes genéricos e aplicar um tipo especifico de análise pode ser uma boa idéia forçar manualmente o SQLMap a usar uma técnica das citadas acima usando o parâmetro ***--technique***.

Identificação do DB e do tipo de DBMS

O SQLMap é um suite extramentente complexo e por isso tende a executar consultas demoradas uma vez que a continuação do processo de consulta vai sendo modificada de acordo com os resultados encontrados, por isso a especificação de técnicas de consulta auxiliam na performance de sua busca, outro recurso útil para obter performance e precisão nos resultados é identificar e especificar o DBMS usado no sistema de seu alvo.

Considere novamente o modelo de arquitetura da aula anterior:

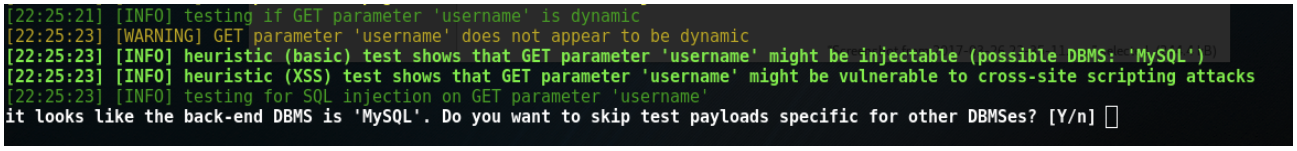


Considerando a arquitetura descrita acima, um banco de dados geralmente não é acessível externamente, o que dificiulta o uso de ferramentas como o nmap para identificação do DBMS, entretanto existe a possibilidade de levantar essas informações através dos campos de formulários que se comunicam com o banco.

Usando o SQLMap faça uma varredura para determinar qual o database da URL de sua aplicação.

```
cd /opt/sqlmap*
./sqlmap.py -u "192.168.X.X/uri..." --dbs
```

Neste exemplo o resultado demonstra que o banco de dados utilizado é o MYSQL:



Outra informação que poderá ser identificada usando SQLMap é o database exato onde a Query está sendo executada, essa informação pode ser obtida a partir do parâmetro --current-db:

```
cd /opt/sqlmap*
./sqlmap.py -u "192.168.X.X/uri..." --current-db
```

Caso bem sucedida a Query trará como retorno o nome exato do database dentro do DBMS MySQL:

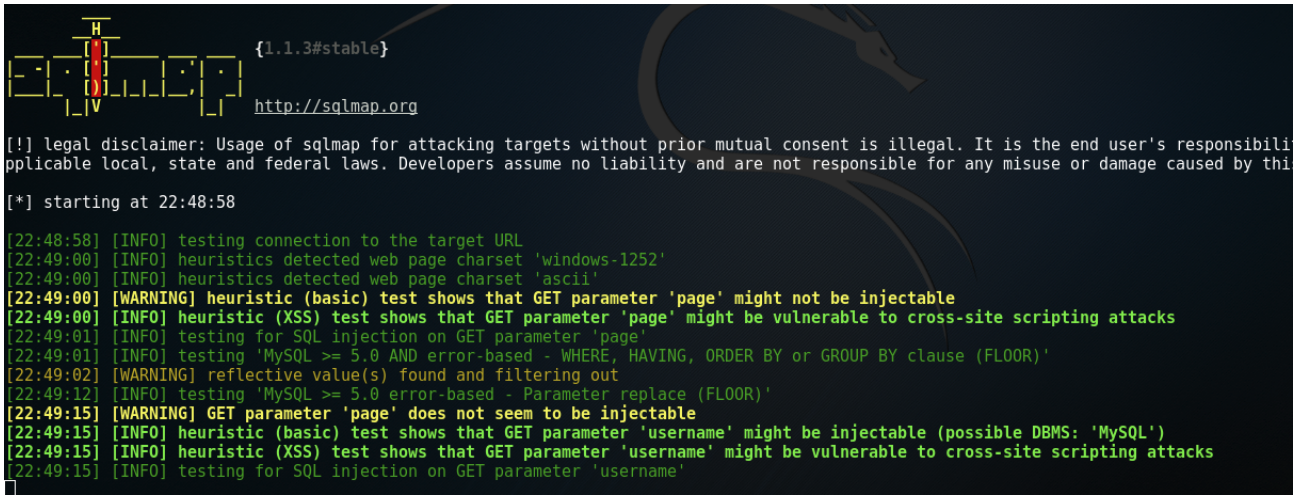
[[images/sqlmap-screen-03.png]]

Neste caso o banco nowasp, essa informação é identificável pela linha: ***current database: 'nowasp'***

Sabendo o banco de dados da aplicação é possível utilizar essa especificação no SQLMap, combinando essa informação a técnica desejada (Neste exemplo "Error-based injection") teremos algo mais ou menos assim:

```
./sqlmap.py -u "192.168.X.X/uri..." --dbms=MySQL --technique=E
```

Considerando uma URI de alvo e o exemplo acima temos um retorno sobre o campo os campos que existiam na consulta (Neste caso o campo "username"



A informação referente ao exploit no exemplo acima aparece na linha: ***[INFO] heuristic (basic) test shows that GET parameter 'username' might be injectable (possible DBMS: 'MySQL')***

Executando Dumping de Dados

Uma vez que tenhamos uma lista específica de databases disponíveis dentro do DBMS e qual o database em uso por uma determinada Query, podemos direcionar nossas consultas para geraçãode dump de tabelas desse banco. Para demonstração, utilizaremos o banco "nowasp".

O SQLMap fornece o parâmetro ***--tables*** para listar as tabelas de um banco, para que funcione esse parâmetro deve ser usado em paralelo com a opção -D, que informa qual banco de dados escolher conforme exemplo abaixo:

```
./sqlmap.py -u "192.168.X.X/uri..." --dbms=MySQL --technique=E -D nowasp --tables
```

Considerando novamente como alvo a Broken Web Application da OWASP e a URL citada no começo do material:

Material de Referência:

Existem boas referências sobre o SQLMap, a principal delas é o próprio site do Projeto:

- SQLMap Project

Boa parte dessa aula baseia-se em uma publicação de Prakhar Prasad pela editora Packt:

- Livro, Mastering Modern Web Penetration Testing

Free Software, Hell Yeah!