Teste realizado dia 24-04-16

Representando a atividade **SQL Injection - Cadastro de lotes.**

Duração dos Testes: 45 min

Duração total da atividade: 1 h e 22 min

**Objetivo**: Tentar realizar consultas não autorizadas no banco por meio

da injeção de SQL nos campos da tela de cadastro de lotes.

**Tentativa 1**

descobrir se o campo Marca/Fabricante é vulnerável a SQL Injection

através das aspas.

O uso de aspas simples aqui é o teste mais básico de vulnerabilidade de um campo para SQL injection. Imaginando que haja por baixo uma instrução do que tenha

*where param1 = <parametro\_na\_interface> and …*

caso não haja o devido tratamento a instrução ficaria

*where param1 = 1 ‘*

*O que ocasionaria um erro de SQL que seria indicado na aplicação.*

**Resultado**: Colocando 1 ‘ no campo Marca/Fabricante o cadastro

aconteceu normalmente. Indicando que talvez haja um tratamento

e o campo não seja vulnerável.

**Tentativa 2**

Usar uma ferramenta automatizada para realizar a tentativa

de injetar SQL na página.

Salvar a requisição POST capturada pela ferramenta burp-suite

num arquivo e indica-lo ao SQL map para que possa utiliza-lo na requisição.

*comando: sqlmap -r params -p --current-db*

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. --current-db pede ao SqlMap que tente descobrir qual o banco da aplicação alvo.

**Resultado**: erro na ferramenta.

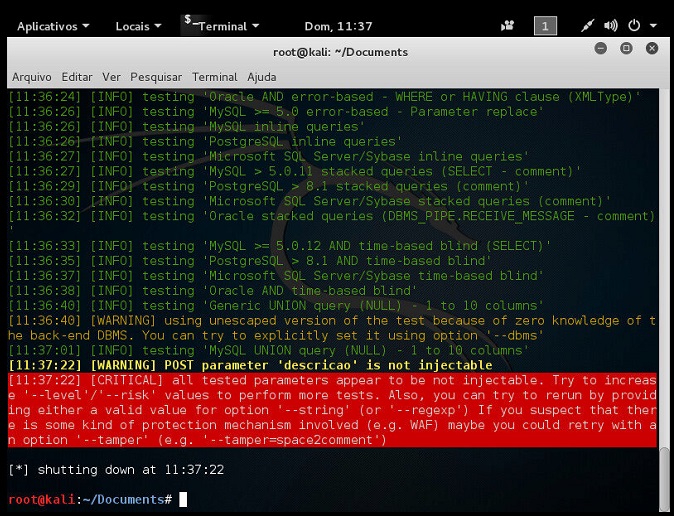
**Tentativa 3**

Alterar o comando para incluir o parâmetro desejado.

O comando do SqlMap pede que seja informado um parâmetro para possível injeção.

*comando: sqlmap -r params -p descricao --current-db*

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. descrição é o parâmetro que tentará ser injetado.
4. --current-db pede ao SqlMap que tente descobrir qual o banco da aplicação alvo.

**Resultado**: 

**Tentativa 4**

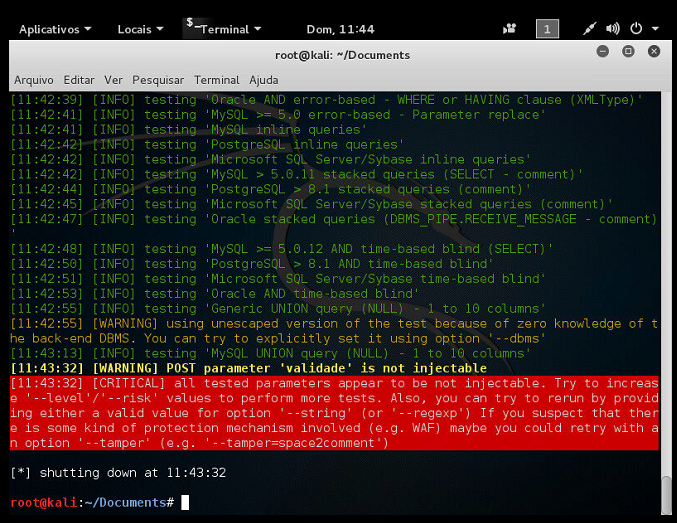
Alterar o comando para incluir o parâmetro validade.

Novamente repetindo o comando, agora tentando a injeção do parâmetro “validade”

*comando: sqlmap -r params -p validade*

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. validade é o parâmetro que tentará ser injetado.

**Resultado**



**Tentativa 5**

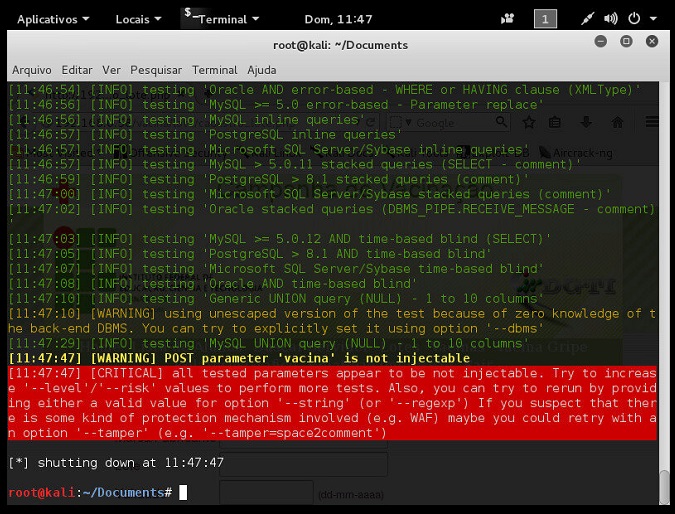
Alterar o comando para incluir o parâmetro vacina.

Agora tentando injetar o parâmetro vacina.

*comando*: sqlmap -r params -p vacina

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. vacina é o atributo que tentará ser injetado.

**Resultado**



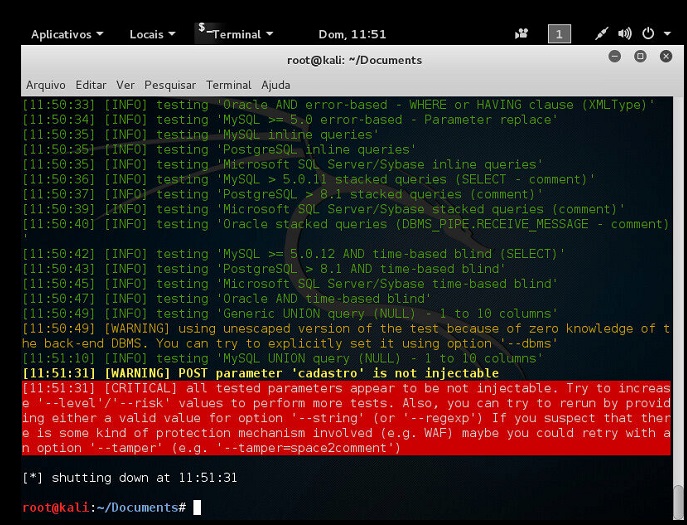
**Tentativa 6**

Alterar o comando para incluir o parâmetro cadastro.

*comando*: sqlmap -r params -p cadastro

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. cadastro é o atributo que tentará ser injetado.

**Resultado**



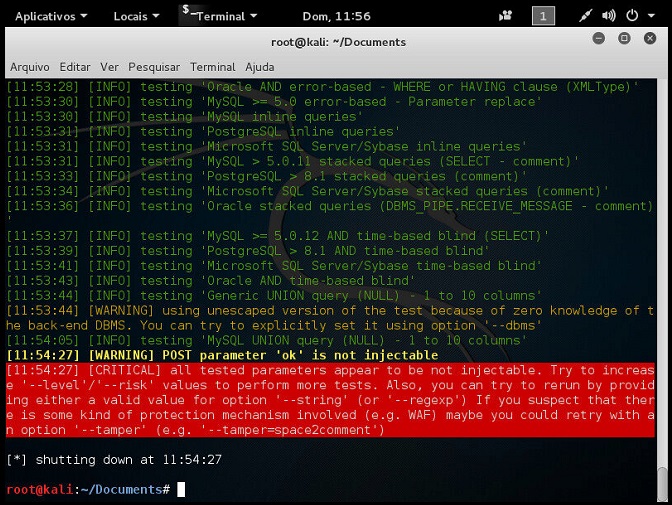
**Tentativa 7**

Alterar o comando para incluir o parametro ok.

*comando: sqlmap -r params -p ok*

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. - p Indica o parâmetro a ser utilizado na consulta
3. ok é o atributo que tentará ser injetado.

**Resultado**



Referencia

[SQLMAP usando POST](https://hackertarget.com/sqlmap-post-request-injection/)