Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

JEFERSON ANTONIO DE AQUINO NERIS

TESTE DE VULNERABILIDADE EM SERVIDORES LINUX

Assis/SP
2016

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

JEFERSON ANTONIO DE AQUINO NERIS

TESTE DE VULNERABILIDADE EM SERVIDORES LINUX

Projeto de pesquisa apresentado ao curso Ciências da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Jeferson Antônio de Aquino Neris.
Orientador(a): Prof. Douglas Sanches da Cunha.

Assis/SP
2016

FICHA CATALOGRÁFICA

NERIS, Jeferson Antônio de Aquino.
Teste de Vulnerabilidade em Servidores Linux / Jeferson Antônio de Aquino Neris. Fundação Educacional do Município de Assis –FEMA – Assis, 2016.
63 páginas.

1. Vulnerabilidade. 2. Linux.

CDD: 001.6
Biblioteca da FEMA

TESTE DE VULNERABILIDADE EM SERVIDORES LINUX

JEFERSON ANTÔNIO DE AQUINO NERIS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Prof. Douglas Sanches da Cunha

Examinador: Prof. Esp. Diomara Martins Reigato Barros

Assis/SP
2016

DEDICATÓRIA

Dedico este trabalho a Deus, a minha família e a minha namorada, pois estes foram meu porto seguro diante das dificuldades encontradas.

AGRADECIMENTOS

A Deus por ter me dado saúde e força para superar as dificuldades.

Ao meus pais e família, pelo amor, incentivo e apoio incondicional.

Ao meu orientador Douglas, pelo esforço no pouco tempo que lhe coube, pelas suas correções, incentivos e confiança no meu potencial.

E a todos que participaram direta ou indiretamente da minha formação, o meu muito obrigado.

RESUMO

A distribuição Kali Linux, reúne poderosas ferramentas para testes de vulnerabilidade em sistemas computacionais, redes de computadores e softwares. Estas ferramentas podem ser utilizadas para práticas benéficas como correções de falhas e vulnerabilidades por empresas de auditoria de segurança, administradores de redes dentre outros. Pode ser utilizada também para fins maliciosos, por *hackers* e *crackers*, que usufruem dessas ferramentas para invadir sistemas e computadores, afim de obter informações sigilosas. A maneira automatizada por buscas de vulnerabilidades em aplicações *web*, tem o objetivo de localizar o maior número possível de vulnerabilidades do alvo, e também de informações armazenadas no banco de dados, de uma forma simples, eficiente e rápida.

Palavras-chave: Linux; Vulnerabilidades; *Hackers*.

ABSTRACT

Kali Linux distribution, brings together powerful tools for vulnerability tests in computer systems, computer networks and software. These tools can be used for beneficial practices as bug fixes and vulnerabilities for security audit companies, network administrators and others. It can also be used for malicious purposes by hackers and crackers who enjoy these tools to break into systems and computers, in order to obtain sensitive information. The automated way to search for vulnerabilities in web applications, aims to find the largest possible number of the target vulnerabilities, as well as information stored in the database, in a simple, efficient and fast.

Keywords: Linux, Vulnerability, *Hackers*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Atualmente, qualquer pessoa utiliza a *internet* para realizar inúmeras tarefas, por ser uma fonte de informação muito grande e de fácil acesso. Esse avanço tecnológico proporcionou uma comodidade surreal ao realizar compras, efetuar pagamentos, consultar saldos, comunicar com pessoas de qualquer lugar do mundo em tempo real dentre diversas outras atividades. Com toda essa facilidade concebida, os chamados *hackers*, perceberam a possibilidade de interceptarem informações e dados pessoais das pessoas ou empresas, causando danos financeiros e morais. Existem empresas que realizam testes de vulnerabilidades e auditoria em sites ou servidores de empresas, com o objetivo de encontrar vulnerabilidades conhecidas ou caminhos que possam ser explorados para invasão, mostrando os custos resultantes de um ataque bem-sucedido, por exemplo, o prejuízo causado pela indisponibilidade de um site comercialmente ativo, sendo ele de vendas ou anuncio de vendas. (GIAVAROTO e SANTOS, 2013). Os *hackers* fazem uma avaliação em seu sistema de computador ou rede, a fim de encontrar alguma vulnerabilidade em potencial na qual pode ser resultado de uma má configuração, falhas em *hardwares*/*softwares* desconhecidas, deficiência no sistema operacional e softwares desatualizados, que é a causa de 80% das invasões de computadores de empresas, de acordo com uma pesquisa feita pela F-Secure (2013). As invasões de computadores em geral são feitas por vírus, que são programas maliciosos para comprometer o sistema operacional. Existem na *internet* à fora, milhares de *malware* que não só roubam informações do sistema, mas também danificam componentes do computador, um exemplo é o vírus Win32/CIH (conhecido por Chernobyl), criado em 1998 pelo estudante de engenharia Chen IngHau, que ao atingir seu sistema Windows, era capaz de apagar os dados da memória ROM (memória física virtual temporária), comprometendo o funcionamento da placa-mãe. Segundo Schneier (2011), está cada vez mais difícil de se defender de ataques, pois as vítimas não sabem de onde o ataque foi originado e nem o motivo do mesmo. O desafio atual, em termos de segurança, é como se proteger, pois nesta era tecnológica, é muito mais fácil atacar do que defender. O especialista acredita que o caminho seja encontrar formas de descobrir os ataques com mais rapidez, no mesmo dia, minutos depois do ocorrido.

1.1. OBJETIVOS

O objetivo geral deste projeto é verificar as vulnerabilidades dos sistemas, com a utilização dos métodos atuais de ataque e ferramentas que auxiliam no *pentest* e viabilizam a exploração das vulnerabilidades encontradas no sistema operacional. A partir dos artefatos deste projeto, será possível localizar vulnerabilidades no sistema que necessitam de uma segurança mais elaborada, com a finalidade de dificultar a invasão de *hackers* ou *crackers* e consequentemente, o tornar mais seguro.

1.2. JUSTIFICATIVAS

Tendo em vista o grande número de pessoas que utilizam a *internet* para tarefas diárias, como pagar contas, fazer compras, movimentações bancárias dentre outras infinidades de coisas, a vulnerabilidade dos sistemas operacionais se tornou um ponto chave para os *hackers* invadirem computadores e obter informações sigilosas, causando danos incalculáveis. Tendo em vista as notícias sobre invasões, falhas em sistemas computacionais e até espionagem industrial, é necessário que ocorra alguns *pentest* (FERREIRA, F.; PEREIRA, J.), para verificar a segurança do sistema e assim, proteger o mesmo de possíveis as invasões remotas.

1.3. MOTIVAÇÃO

O desenvolvimento deste projeto de pesquisa consiste no fato de que o *pentest* é um tema ainda pouco explorado e pode contribuir com a segurança do sistema testado. O *pentest* ajuda a manter a segurança dos sistemas desenvolvidos, tais testes, podem fornecer um nível de dificuldade mais elevado a invasões, mantendo a integridade dos dados pessoais e sigilosos trafegados pela *internet*, a fim de proporcionar a execução das tarefas mais protegidas. Outra motivação é a chance de, num futuro não muito distante, o mercado de trabalho expandir a área de teste de invasão, necessitando de profissionais com conhecimento na linha do tema desta pesquisa, uma vez que a área de desenvolvimento de sistemas operacionais, cresce exponencialmente a cada dia.

1.4. ESTRUTURA DE TRABALHO

O capítulo 1, a Introdução, contextualiza a área de estudo e apresentará os objetivos, justificativas, motivação, perspectivas de contribuição e metodologia de pesquisa para o desenvolvimento deste trabalho. O capítulo 2 apresentará as ferramentas que serão utilizadas, juntamente com uma pesquisa detalhada sobre a mesma. O capítulo 3, abordará o levantamento de dados do alvo, ou seja, realizar um reconhecimento do alvo, a fim de coletar informações, descobrindo os pontos vulneráveis e executar ataques com fins acadêmicos no mesmo. O capítulo 4, apresentará a conclusão do estudo de vulnerabilidades em servidores Linux.

2. KALI LINUX

Neste capítulo será apresentada as tecnologias e ferramentas que serão utilizadas nos estudos e testes de vulnerabilidades em servidores Linux. A distribuição Kali Linux fora a escolhida para o aplicação e execução dos testes por ser completamente livre, e especializada em testes de intrusão e auditoria de segurança, com mais de 300 ferramentas especificas para a execução. Os testes de intrusão é uma forma de avaliar a segurança de um sistema de computador ou rede, simulando um ataque.

2.1. PRINCIPAIS CARACTERISTICAS DO KALI LINUX

Segundo Muniz e Lakhani (2013), o Kali Linux é uma reconstrução completa do BackTrack Linux, foram revisadas cada ferramenta, eliminando as ferramentas que não funcionavam e substituindo por ferramentas compatíveis, de mesma funcionalidade. Seu *kernel* já inclui os últimos patches de injeção de pacotes, facilitando a avaliação dos executantes dos testes. Por ter seu código fonte disponível na *internet*, o Kali Linux é completamente livre, podendo ser aprimorado por programadores do mundo inteiro, não possuindo nenhum proprietário, pois seu repositório oficial está disponível para todos que todos a aqueles que desejem adaptar e remontar os pacotes, da maneira que preferir.

O Kali aderiu o *Filesystem Hierarchy Standard* (Padrão de Hierarquia do Sistema de Arquivos), trazendo consistência entre sistemas e distribuições, possuindo a fácil localização de bibliotecas, arquivos binários, de apoio, etc. Além disso, suporta vários dispositivos wireless com entradas USB, o que auxilia nos testes de intrusão em redes sem fio. Embora as ferramentas usadas em testes de intrusão tendam a ser escritas em inglês, o sistema de múltiplos idiomas permite que mais usuários o operassem no seu idioma nativo, e encontrasse as ferramentas de que precisa para realizar suas tarefas. (MUNIZ; LAKHANI, 2013).

2.2. PERFIS

Este subcapitulo trata os perfis que são utilizados para realizar *pentest* por empresas de auditoria, ou por usuários. Para a realização dos testes de invasão, será utilizado um perfil invasor. Segue uma explicação dos perfis existentes:

• *Early Adopter*: Este é o usuário/cliente primário, que em troca da possibilidade de avaliar um produto exclusivo em primeira mão, topa ser usado como cobaia, lidando com falhas e para dar um feedback adequado.
• *Hacker*: São indivíduos com vasto conhecimento tecnológico, que se utilizam do mesmo para modificar hardwares e softwares com métodos novos ou adaptando antigos, com o objetivo de localizar brechas de segurança ou falhas, com o intuito de instruir para uma proteção mais robusta e eficiente à ataques criminosos.
• *Cracker*: São indivíduos com vasto conhecimento tecnológico, que se utilizam do mesmo para quebrar códigos de segurança, senhas de acesso a redes e códigos de programas com fins criminosos.
• *Lammer*: São indivíduos com pouco conhecimento tecnológico ou às vezes nenhum, que se auto nomeia um *hacker*, e que muitas vezes utiliza ferramentas ou programas já prontos, normalmente disponibilizados pela *internet*, para tentar realizar invasões ou ataques.
Para a conclusão dos tais testes, será utilizado o perfil *hacker*, que são indivíduos com vasto conhecimento tecnológico, que se utilizam do mesmo para modificar hardwares e softwares com métodos novos ou adaptando antigos, com o objetivo de localizar brechas de segurança ou falhas, com o intuito de instruir para uma proteção mais robusta e eficiente aos ataques criminosos.

2.3. FERRAMENTAS

Este subcapitulo, aborda um resumo de todas as ferramentas utilizadas para a realização dos testes, desde a sua construção, linguagem utilizada para a programação da mesma, com suas características de execução, estrutura da ferramenta até os métodos utilizados para ataque.

O Kali Linux é um sistema operacional que nasceu e foi lançado em 13 de março de 2013, e possui em sua estrutura mais de 300 ferramentas voltadas para testes de penetração (*pentest*), avaliação de segurança e vulnerabilidades presentes. Essa diversificação de

ferramentas tem o objetivo de descobrir falhas em sistemas operacionais, redes, *internet*, softwares, banco de dados etc. Ele é totalmente baseado no Debian, e tem como antecessor o BackTrack Linux que também fora utilizado para *pestest*. As ferramentas são sincronizadas com os repositórios do Debian e se conectam quatro vezes ao dia. Isso significa que os usuários tenham as últimas atualizações de pacotes e correções de segurança. (BROAD; BINDNER, 2014).

2.3.1. Nmap

De acordo com Messer (2007), o NMAP (*Network Mapper*) que surgiu para o mundo em 1997, desenvolvida pelo autoproclamado *hacker* Gordon Lyon, conhecido como “Fyodor”, foi publicada pela revista Phrack com seu código fonte incluso. É uma ferramenta de *software* livre, ou seja, permite-se modificações ou adaptações conforme a necessidade do usuário, sem que haja necessidade de solicitar permissão ao proprietário para modificá-la.

Considerado um *portscan*, é uma ferramenta de segurança, usada para detectar portas abertas em computadores e serviços rapidamente, seja em redes amplas ou em hosts individuais, criando um “mapa” dessa mesma rede. Conhecido pela sua rapidez e eficiência na execução dos testes, o NMAP é amplamente utilizado por empresas de auditoria de segurança e testadores de testes de penetração (*pentest*). (MESSER, 2007).

Segundo Messer (2007), o resultado do NMAP é uma lista de alvos escaneados, conhecida como “tabela de portas interessantes”, com informações detalhadas de cada uma dependendo das opções utilizadas. Nessa tabela lista o número da porta e o protocolo, o nome do serviço e o estado. O estado pode ser aberto (*open*), filtrado (*filtered*), fechado (*closed*), ou não-filtrado (*unfiltered*).

• Aberta (*open*): Atribui-se o estado *open*, para as portas que estão ativamente aceitando conexões TCP ou pacotes UDP nesta porta. Saiba que uma porta em estado *open*, é um convite para um ataque.

• Fechada (*closed*): Uma porta em estado *closed*, está acessível, porém não possui nenhuma aplicação “escutando” nela. Portas *closed* são úteis para mostrar o endereço IP de determinado host. Filtrada (*filtered*): O nmap não consegue determinar se uma porta está aberta pelo fato da mesma estar com uma filtragem de pacotes que impede a sondagem de alcançar essa porta. Essa filtragem pode ser um dispositivo *firewall* dedicado,

regras de roteador, ou um *software* de *firewall* baseado em host. Em alguns casos essas portas respondem com mensagens de erro ICMP do tipo 3 códigos 13 (destino inalcançável: comunicação proibida administrativamente), mas os filtros que simplesmente descartam pacotes sem mesmo responder são bem mais comuns. Sendo assim, o nmap acredita que os pacotes podem ter sido descartados devido ao congestionamento da rede, forçando-o a tentar várias vezes. Isso reduz a velocidade do *scan* drasticamente.

• Não-filtrada (*unfiltered*): *Unfiltered* significa que a porta está acessível, porém o nmap não consegue determinar se a porta está aberta ou fechada. Apenas o *scan* ACK, que é usado para mapear conjuntos de regras de *firewall*, classifica portas nesse estado. Usar outros métodos de escaneamento, como *Scan Window, Scan Syn, Scan FIN*, podem ajudar a determinar se a porta está aberta.

• *Open* | *Filtered*: Esse estado é usado pelo nmap quando o mesmo não consegue determinar se a porta está aberta ou filtrada. Isso acontece para tipos de *scan* onde as portas abertas não dão nenhuma resposta. Essa falta de pacotes também pode significar que um filtro de pacotes descartou a sondagem ou qualquer tipo de resposta que ela provocou. Os *scans* UDP, IP *Protocol*, FIN, *Null* e Xmas classificam portas nesse estado.

• *Closed* | *Filtered*: Esse estado é usado pelo nmap quando o mesmo não consegue determinar se aporta está fechada ou filtrada. É apenas usado para o *scan* IP ID *Idle Scan*.

2.3.1.1. Hosts

De acordo com Brockmeier (2010), as necessidades para o descobrimento de host são diversas, e o nmap oferece um leque de variedade de customizações possíveis para as técnicas utilizadas. A descoberta de host pode ser é chamada de *ping scan*, mas ela vai muito além dos simples pacotes ICMP de *echo request* associados à ferramenta presente conhecida como *ping*. O administrador de redes, geralmente responsável pela infraestrutura, utiliza o *ping* ICMP para localizar hosts disponíveis na rede interna, enquanto um profissional externo de análise de vulnerabilidades (*Penetration Tester*) pode utilizar em uma tentativa de burlar as restrições do *firewall*.

NMAP *Scan*	Sintaxe do Comando	Privilégio de Acesso Requerido	Identificação de Portas TCP	Identificação de Portas UDP
TCP SYN *Scan*	-sS	SIM	SIM	NÃO
TCP *connect*() *Scan*	-sT	NÃO	SIM	NÃO
FIN *Scan*	-sF	SIM	SIM	NÃO
Xmas *Tree Scan*	-sX	SIM	SIM	NÃO
Null Scan	-sN	SIM	SIM	NÃO
UDP *Scan*	-sU	SIM	NÃO	SIM
ACK *Scan*	-sA	SIM	SIM	NÃO
Window Scan	-sW	SIM	SIM	NÃO
Idlescan	-sI	SIM	SIM	NÃO
IP *Protocol Scan*	-sO	SIM	NÃO	NÃO
FTP *Bounce Attack*	-b	NÃO	SIM	NÃO

Tabela 1: Tabela NMAP.

A tabela acima, demostra um resumo de todos os comandos nmap possui várias técnicas de escaneamento de portas, abaixo segue uma lista explicando cada técnica.

-sS (TCP SYN): É uma técnica muito popular e muito eficiente. Essa técnica pode facilmente determinar se uma porta está *open*, *closed* ou *filtered*, e geralmente é chamada de *half-open scanning*, pois não abre uma conexão TCP completa (não completa o *handshake*) e funciona da seguinte maneira.

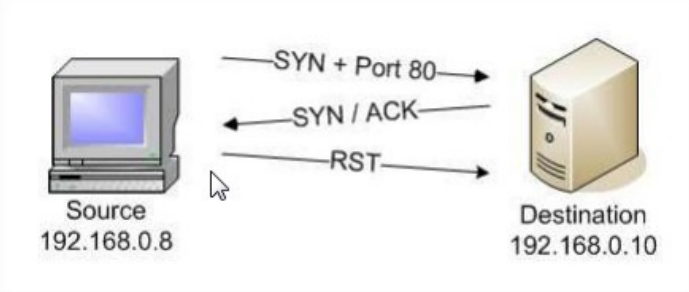

Figura 1: TCP SYN *open* (NETWORKUPTIME, 2016).

Porta fechada (*closed*):

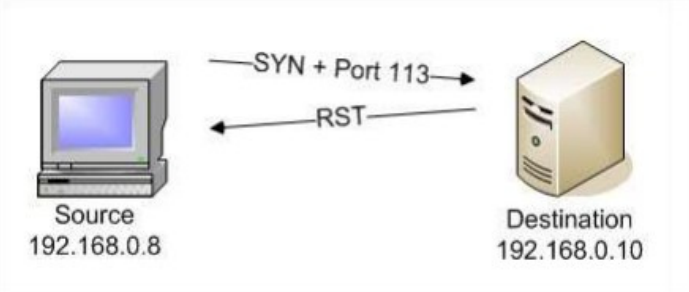

Figura 2: TCP SYN *closed* (NETWORKUPTIME, 2016).

-sT (TCP Connect): Esse *scan* é igual ao *scan* SYN, porém estabelece uma conexão com a máquina e a porta do host (3-*way handshake*) enviando uma chamada de sistema *connect*(). Essa chamada é a mesma utilizada por browsers, clientes P2P, e a maioria das outras aplicações de rede. É a parte da interface de programação conhecida como API de Sockets de Berkeley. O nmap utiliza esta API para obter informações do estado de cada tentativa de conexão. Esse *scan* funciona da seguinte maneira.

Conexão estabelecida (*open port*):

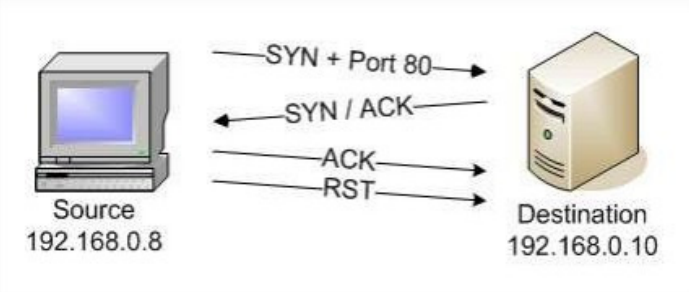

Figura 3: TCP Connect *open port* (NETWORKUPTIME, 2016).

Conexão não estabelecida (*closed port*):

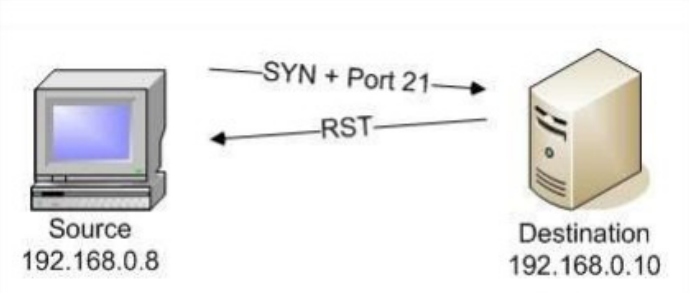

Figura 4: TCP Connect *closed port* (NETWORKUPTIME, 2016).

-sU (UDP *Scan*): É utilizado para escanear portas UDP. É normalmente mais lento e mais difícil que o TCP. Vários serviços utilizam essas portas, tais como DNS (53), SNMP (161/162) e DHCP (67/68), e podem estar vulneráveis. Ele pode ser combinado com algum *scan* de TCP, como por exemplo o SYN *Scan*, averiguando ambos protocolos. Esse *scan* envia um cabeçalho UDP vazio (sem dados) para cada porta. E funciona da seguinte maneira:

Portas abertas (*open*):

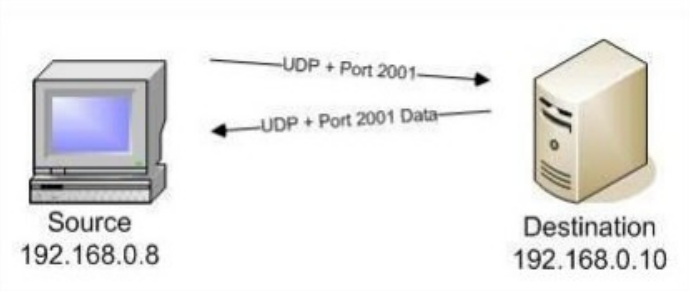

Figura 5: UDP *Scan open* (NETWORKUPTIME, 2016).

Portas fechadas (*closed*):

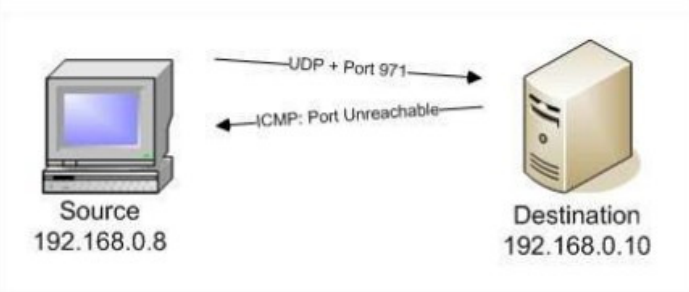

Figura 6: UDP *Scan closed* (NETWORKUPTIME, 2016).

Portas abertas|filtradas (*open*|*filtered*):

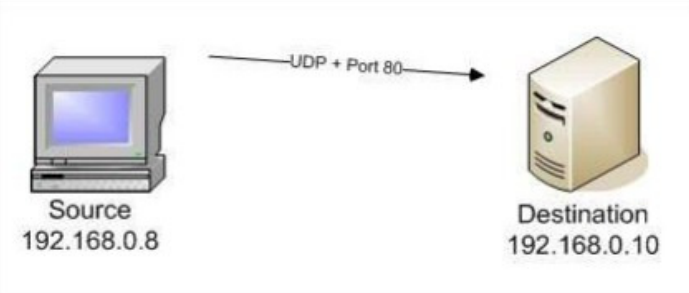

Figura 7: UDP *Scan open*|*filtered* (NETWORKUPTIME, 2016).

Portas filtradas (*filtered*):

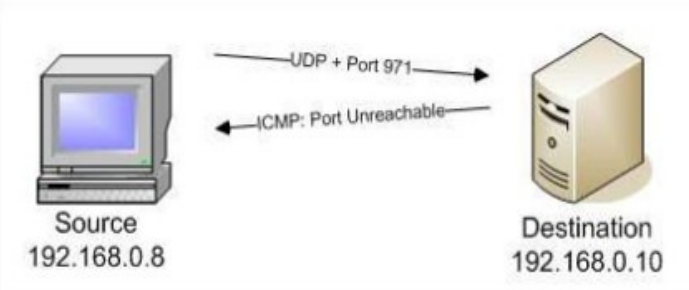

Figura 8: UDP *Scan filtered* (NETWORKUPTIME, 2016).

-sN, -sF e -sX (TCP *Null*, FIN e Xmas): Esses três tipos de *scan* são os mesmos em termos de com*port*amento, exceto pela flag marcadas no pacote de sondagem. A vantagem desses *scans* é que eles podem sondar hosts através de *firewall* não orientado à conexão e roteadores que filtram pacotes. Outra vantagem é que eles são menos perceptíveis que o

Scan SYN. Mas não se engane, IDS's bem configurados podem detectar esses tipos de *scan*. Esse *scan* funciona da seguinte maneira:

Portas abertas|filtradas (*open*|*filtered*):

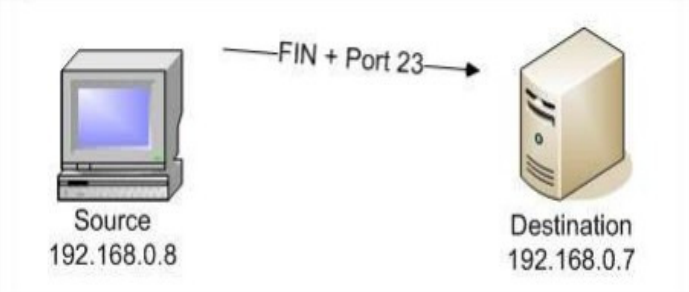

Figura 9: TCP *Null*, FIN e Xmas *open*|*filtered* (NETWORKUPTIME, 2016).

Portas fechadas (*closed*):

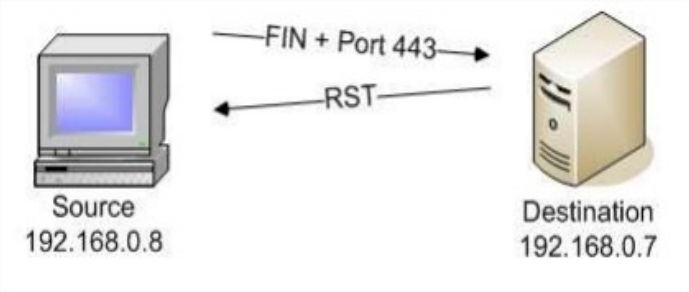

Figura 10: TCP *Null*, FIN e Xmas *closed* (NETWORKUPTIME, 2016).

Portas filtradas (*filtered*):

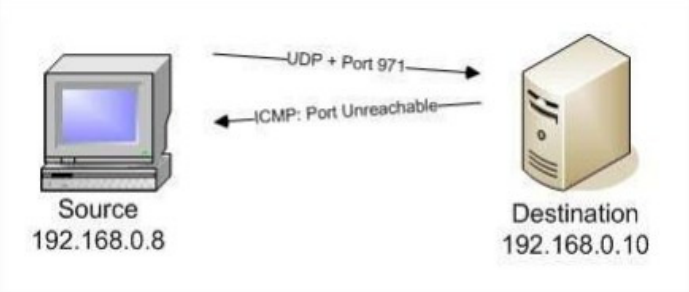

Figura 11: TCP *Null*, FIN e Xmas *filtered* (NETWORKUPTIME, 2016).

-sA (TCP ACK): Esse *scan* nunca definirá uma porta como aberta ou aberta | filtrada. Ele é usado para descobrir regras de *firewall*, determinando se eles são orientados a conexão ou não, e quais portas estão filtradas. Se um sistema é não-filtrado, as portas abertas e fechadas respondem com um pacote RST.

Portas filtradas (*filtered*):

Figura 12: TCP ACK *filtered* (NETWORKUPTIME, 2016).

Portas não-filtradas (*unfiltered*):

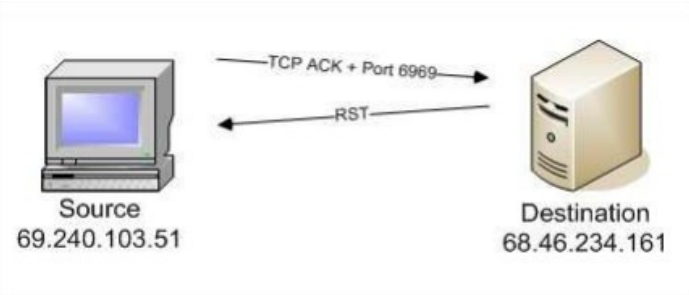

Figura 13: TCP ACK *unfiltered* (NETWORKUPTIME, 2016).

Ou

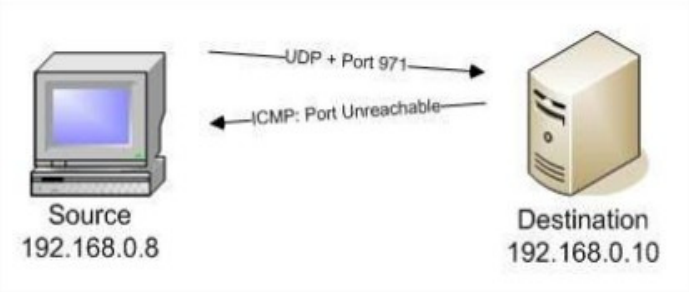

Figura 14: TCP ACK *unfiltered* (NETWORKUPTIME, 2016).

-sW (TCP *Window*): Trabalha igual o *Scan* ACK, mas com um detalhe, ele analisa o campo TCP *Window* to pacote RST de resposta. Em alguns SO's esse valor pode ser positivo, para portas abertas, ou zero, para portas fechadas. Mas como isso não é uma regra, então esse *scan* não é muito confiável. Alguns sistemas que não suportam isso irão normalmente devolver zero, sendo assim, todas portas “estarão” fechadas.

Portas fechadas (*closed*):

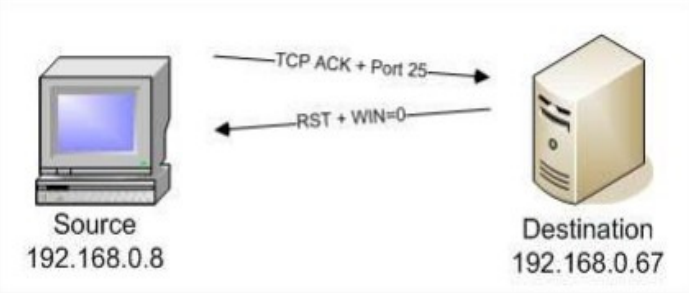

Figura 15: (-sW) TCP Window *closed* (NETWORKUPTIME, 2016).

Portas abertas (*open*):

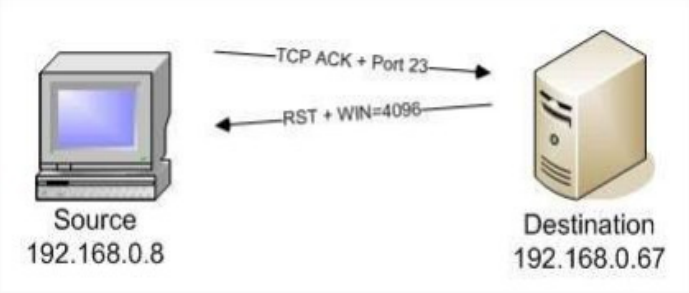

Figura 16: (-sW) TCP Window *open* (NETWORKUPTIME, 2016).

-sI (*Scan Idle*): Esse tipo de *scan*, na verdade, faz um *spoofing* de algum IP que você queira. É bom para descobrir relações de confiança entre hosts, ou seja, se determinado *host* possui relações de confiança com o outro (“spoofado” por você), então mais detalhes serão mostrados. As máquinas que são “spoofadas”, são chamadas de *zombie*. Para iniciar o processo de digitalização ocioso, nmap primeira envia um SYN / ACK para a estação de trabalho *zombie* para induzir um RST em troca. Este quadro RST contém o IPID inicial que nmap vai se lembrar para mais tarde.

Capturando o IPID das máquinas “spoofadas” (zumbis):

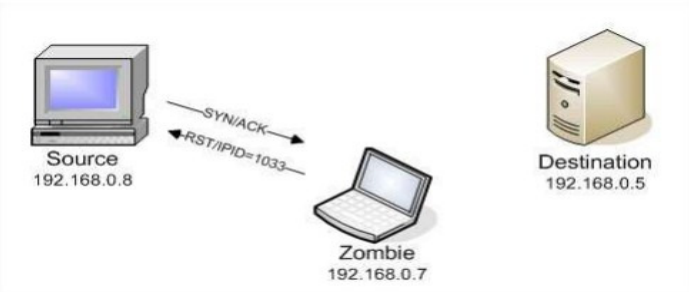

Figura 17: (-sI) *Scan Idle* Capturando IPID (NETWORKUPTIME, 2016).

O nmap agora envia um quadro SYN para o endereço de destino, mas nmap falsifica o endereço IP para fazê-la parecer como se o quadro SYN foi enviado a partir da estação de trabalho zumbi. Se este quadro SYN é enviado a uma das portas abertas do destino, o endereço de destino irá responder com um SYN / ACK para a estação de trabalho *zombie* anteriormente falsificado. A estação de trabalho *zombie* não será esperando o SYN / ACK (afinal, ele nunca realmente enviou o SYN), de modo que o zumbi vai responder à estação de destino com um RST. A resposta RST irá, como esperado, incrementar os zumbis IPID.

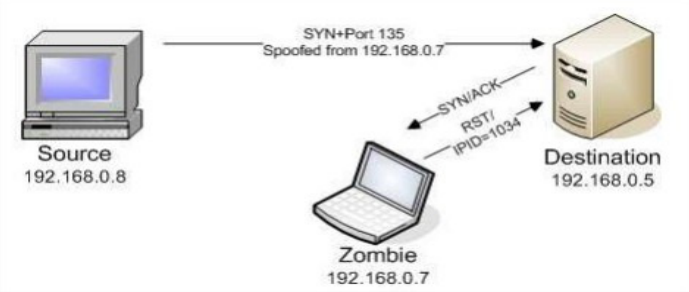

Figura 18: (-sI) *Scan Idle* Aguardando RST (NETWORKUPTIME, 2016).

Por fim, a *Idlescan* faz com que o nmap repita a sondagem SYN/ACK na estação de zumbi. Se o IPID conseguiu incrementar, a porta que foi falsificado no quadro SYN original está aberta no dispositivo de destino. Se o IPID não conseguiu incrementar, a porta está fechada.

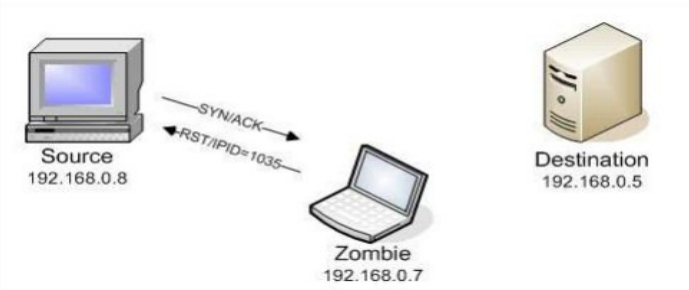

Figura 19: (-sI) *Scan* Idle Sondagem SYN/ACK na estação zumbi (NETWORKUPTIME, 2016).

-sO (*Scan* de protocolo IP): Essa verificação de protocolo IP é um pouco diferente dos outros *scans* do nmap. Esse *scan* é utilizado para descobrir quais protocolos são su*port*ados pelo host alvo (TCP, ICMP, IGMP, etc.).

Protocolos indisponíveis, não respondem ao *scan*:

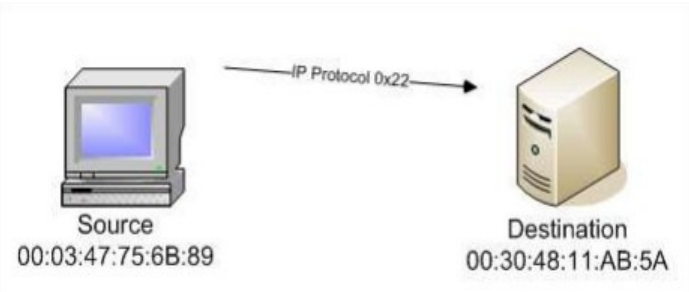

Figura 20: (-sO) *Scan* de protocolo IP: Protocolo indisponível (NETWORKUPTIME, 2016).

Protocolos disponíveis, respondem ao *scan*:

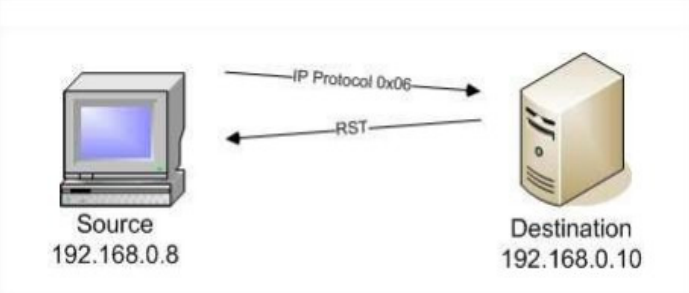

Figura 21: (-sO) *Scan* de protocolo IP: Protocolo disponível (NETWORKUPTIME, 2016).

-b (*Scan* de FTP *bounce*): Esse tipo de *scan* é usado em servidores vulneráveis a Proxy FTP. Isso permite que um usuário se conecte a um servidor FTP, e então solicite que arquivos sejam enviados a um segundo servidor. Tal característica é sujeita a abusos em diversos níveis. Um dos abusos seria fazer um servidor escanear as portas de outro. Para isso, basta solicitar ao servidor para enviar um arquivo para cada porta interessante do alvo. Então a partir da mensagem de erro, é determinado se a porta está aberta ou fechada. Esta é uma boa forma de passar por firewalls, porque os servidores FTP de empresas normalmente são posicionados onde tem mais acesso a outros hosts internos que os velhos servidores da *internet* teriam.

Porta FTP fechada:

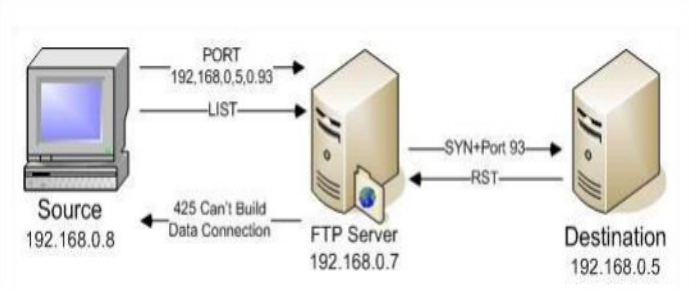

Figura 22: (-b) *Scan* de FTP *bounce*: FTP *closed* (NETWORKUPTIME, 2016).

Porta FTP aberta:

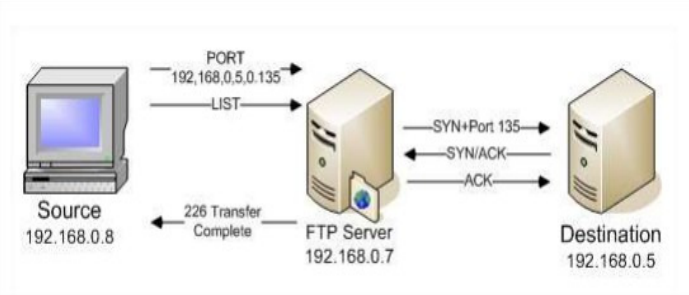

Figura 23: (-b) *Scan* de FTP *bounce*: FTP *open* (NETWORKUPTIME, 2016).

2.3.1.2. Comandos Nmap

Abaixo, estão alguns comandos utilizados no nmap:

```
root@Kali:~# nmap 10.0.0.135

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:44 BRT
Nmap scan report for 10.0.0.135
Host is up (0.0029s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
135/tcp  open  msrpc
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 19.53 seconds
```

Figura 24: Analisando IP.

Verificar se determinado IP é protegido por algum *firewall*:

```
root@Kali:~# nmap -sA  10.0.0.135

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:03 BRT
Nmap scan report for 10.0.0.135
Host is up (0.00015s latency).
All 1000 scanned ports on 10.0.0.135 are unfiltered

Nmap done: 1 IP address (1 host up) scanned in 1.72 seconds
```

Figura 25: Verificando se o IP é protegido por *firewall*.

Mostrar quando determinado IP é protegido por *firewall*:

Figura 26: Verificando quando o IP é protegido pelo *firewall*.

Realizar uma verificação rápida em determinado IP

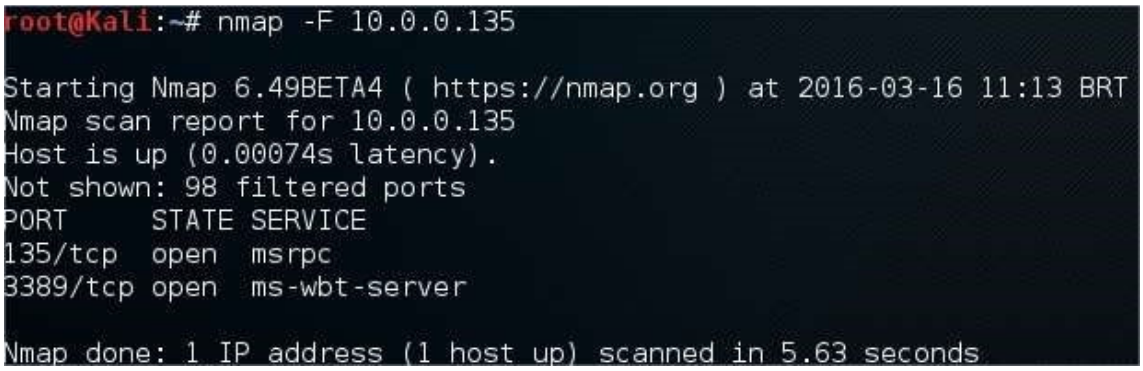

Figura 27: Realizando uma verificação rápida no IP.

Detectar serviços remotos e suas versões de determinado IP

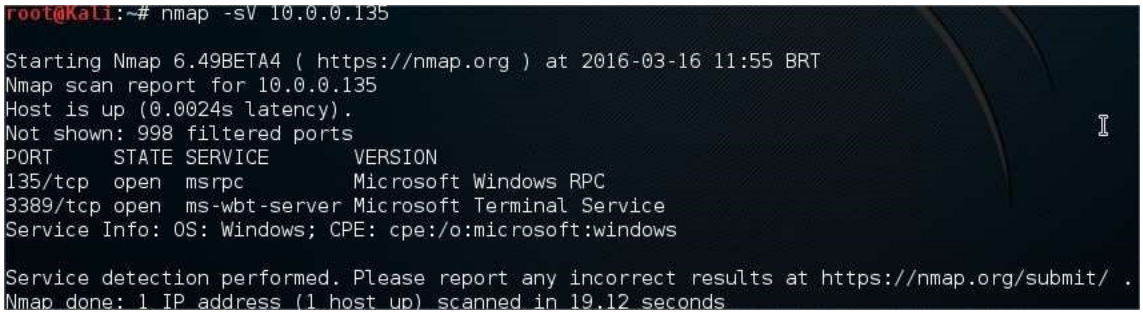

Figura 28: Detectando serviços remotos e versões no IP informado.

Realizar um *scan* de determinado IP.

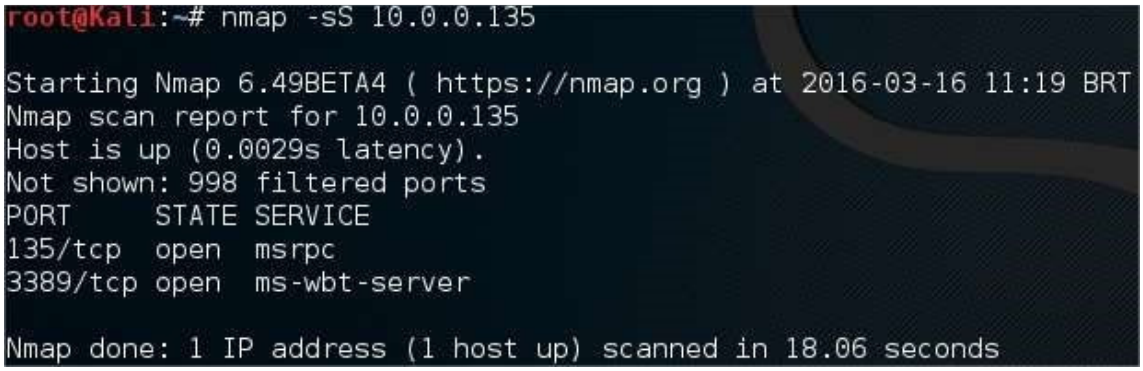

Figura 29: Realizando um *scan* no IP.

Verificar portas mais utilizadas com TCP Connect de determinado IP.

Figura 30: Verificando portas utilizadas com TCP Connect.

Verificar portas mais utilizadas com TCP ACK de determinado IP.

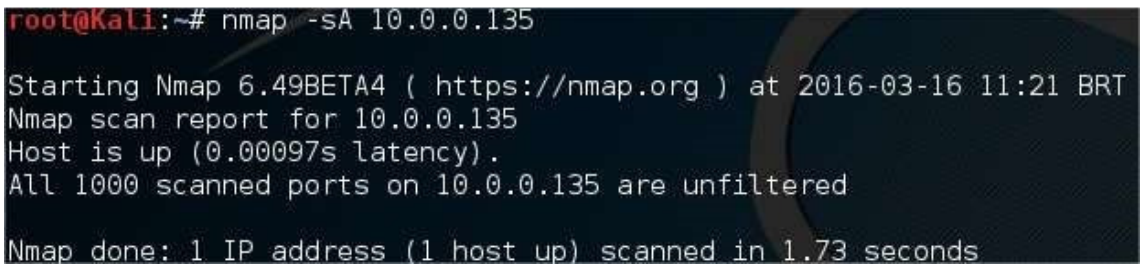

Figura 31: Verificando portas utilizadas com TCP ACK.

Verificar portas mais utilizadas com TCP Window de determinado IP.

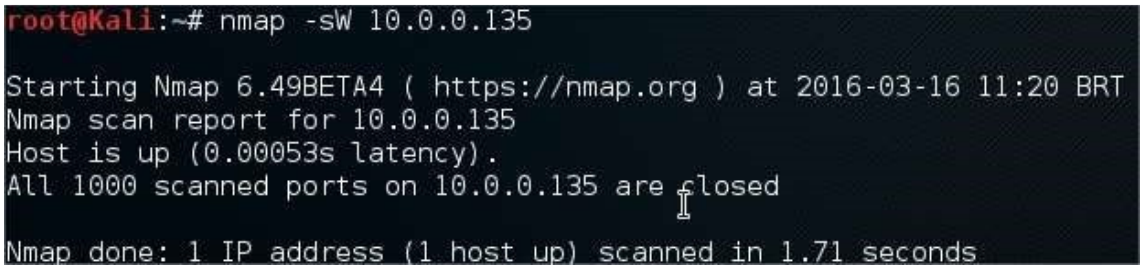

Figura 32: Verificando portas utilizadas com TCP Window.

Verificar portas mais utilizadas com TCP Maimon de determinado IP.

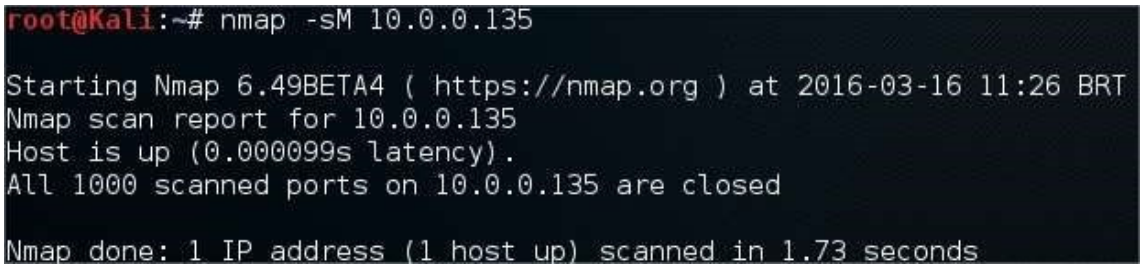

Figura 33: Verificando portas utilizadas com TCP Maimon.

Realizar varredura no *firewall* com TCP Fin de determinado IP.

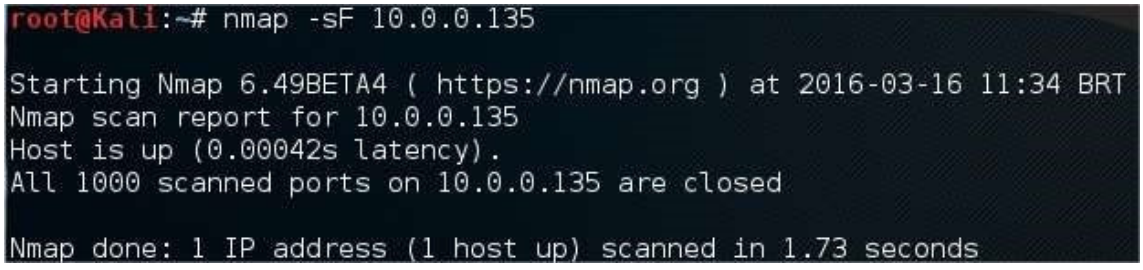

Figura 34: Realizando varredura no *firewall* com TCP Fin.

Realizar varredura no *firewall* com TCP *Null* de determinado IP.

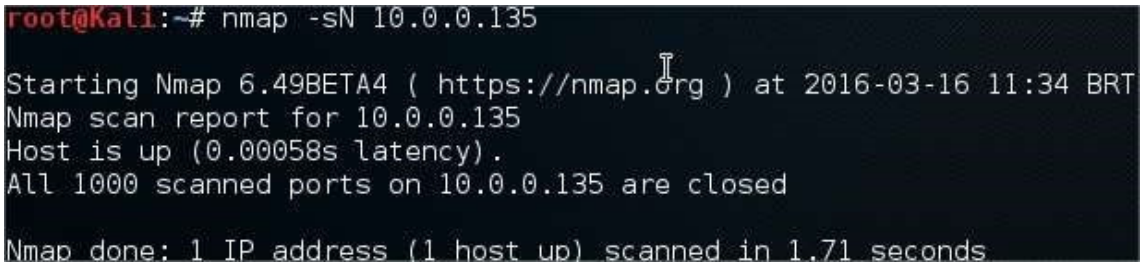

Figura 35: Realizando varredura no *firewall* com TCP *Null*.

Realizar varredura no *firewall* com TCP Xmas de determinado IP.

```
root@Kali:~# nmap -sX 10.0.0.135

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:36 BRT
Nmap scan report for 10.0.0.135
Host is up (0.00025s latency).
All 1000 scanned ports on 10.0.0.135 are closed

Nmap done: 1 IP address (1 host up) scanned in 1.82 seconds
```

Figura 36: Realizando varredura no *firewall* com TCP Xmas.

Realizar *scan* para verificar portas UDP ativas de determinado IP.

```
root@Kali:~# nmap -sU 10.0.0.135

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:25 BRT
Nmap scan report for 10.0.0.135
Host is up (0.00021s latency).
All 1000 scanned ports on 10.0.0.135 are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 5.67 seconds
```

Figura 37: Buscando portas UDP ativas.

Camuflar seu IP para realizar varredura.

```
root@Kali:~# nmap -sS 10.0.0.135 -D 10.0.0.166,200.125.48.78

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:39 BRT
Nmap scan report for 10.0.0.135
Host is up (0.066s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
135/tcp  open  msrpc
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 59.38 seconds
```

Figura 38: Camuflando IP para realizar varredura.

Descobrir qual o sistema operacional que determinado IP utiliza.

```
root@Kali:~# nmap -A 10.0.0.108

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:07 BRT
Nmap scan report for 10.0.0.108
Host is up (0.00045s latency).
Not shown: 993 filtered ports
PORT      STATE SERVICE       VERSION
80/tcp    open  http
```

Figura 39: Descobrindo o sistema operacional utilizado.

```
Host script results:
|_nbstat: NetBIOS name: PRXADM14A, NetBIOS user: <unknown>, NetBIOS MAC: 00:23:ae:b9:00:19 (Dell)
| smb-os-discovery:
|   OS: Windows 7 Professional 7601 Service Pack 1 (Windows 7 Professional 6.1)
|   OS CPE: cpe:/o:microsoft:windows_7::sp1:professional
|   Computer name: PRXADM14a
|   NetBIOS computer name: PRXADM14A
|   Domain name: prx.agr.br
|   Forest name: prx.agr.br
|   FQDN: PRXADM14a.prx.agr.br
|_  System time: 2016-03-16T11:09:49-03:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smbv2-enabled: Server supports SMBv2 protocol
```

Figura 40: Retorno do sistema operacional utilizado.

Realizar *scan* para verificar protocolo de IP.

```
root@Kali:~# nmap -sO 10.0.0.135

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-16 11:23 BRT
Nmap scan report for 10.0.0.135
Host is up (0.0091s latency).
Not shown: 255 open|filtered protocols
PROTOCOL STATE SERVICE
6        open  tcp

Nmap done: 1 IP address (1 host up) scanned in 8.18 seconds
```

Figura 41: Verificando protocolos de IP.

Buscar portas abertas e possivelmente abertas de determinado IP.

Figura 42: Buscando possíveis portas abertas no IP.

2.3.2. Metasploits

O Metasploit é uma ferramenta *open* source, criada por HD Moore, desenvolvida e programada em Ruby, é uma plataforma voltada para testes de invasão (*pentest*). Com ele é possível realizar desde um simples *scan* do alvo, até mesmo a verificação de vulnerabilidade e falhas do sistema operacional, como também de programas instalados, com a ajuda de *exploits* e módulos auxiliares. (KENNEDY et al., 2011).

Essa ferramenta extremamente poderosa, é organizada em vários módulos, contendo programas adequados especificamente para executar *exploits*, carregar módulos auxiliares, efetuar varredura, enumeração e rodar *exploits* em massa contra alvos específicos ou redes inteiras. (KENNEDY et al., 2011).

Os *exploits* são uma sequência de dados, comandos ou parte de códigos de softwares desenvolvidos por *hackers*, com o objetivo de invadir uma máquina aproveitando-se de uma vulnerabilidade, provocando instabilidade no sistema, diminuindo a segurança e finalmente executando ordens no sistema, com o intuito de conseguir informações sigilosas. (KENNEDY et al., 2011).

Os módulos auxiliares são softwares com finalidade de auxiliar na invasão, por exemplo, um *software* que realiza *scan* em portas, captura trafego de rede, etc. Após os *exploits* encontrar alguma vulnerabilidade, os códigos ou comandos conhecidos como *Payload*, são enviados ao alvo com o objetivo de explorar a tal vulnerabilidade encontrada. (KENNEDY et al., 2011).

O Metasploit utiliza do banco de dados PostgreSQL por padrão, sendo assim, as informações obtidas por meio de módulos auxiliares, são armazenadas no próprio banco

de dados do Metasploit, criando um workspace da ferramenta e fornece a infraestrutura para automatizar tarefas. Possui diversos módulos complementares e pode ser utilizado de diversas formas para auxiliar em um teste de invasão. (KENNEDY et al., 2011).

O principal objetivo do metasploits é criar um ambiente de pesquisa, desenvolvimento e exploração de vulnerabilidades de softwares, fornecendo as ferramentas necessárias para a execução de todos os passos da pesquisa.

Descoberta da vulnerabilidade: Onde é descoberto algum erro de programação que pode levar ou não a uma brecha de segurança.

- Análise: Onde é analisada a vulnerabilidade encontrada, e se possível determinar de qual forma essa brecha pode ser explorada.
- Desenvolvimento do *exploit*: Se inicia o desenvolvimento das técnicas de exploração da vulnerabilidade encontrada, buscando a melhor forma para invadir a máquina, utilizando nessa etapa a melhor opção entre engenharia reversa, programação, debugger e etc.
- Teste do *exploit*: É a etapa da execução do *exploit* desenvolvido, para comprovar a falha de segurança na vulnerabilidade encontrada.

2.3.2.1. Tecnologia

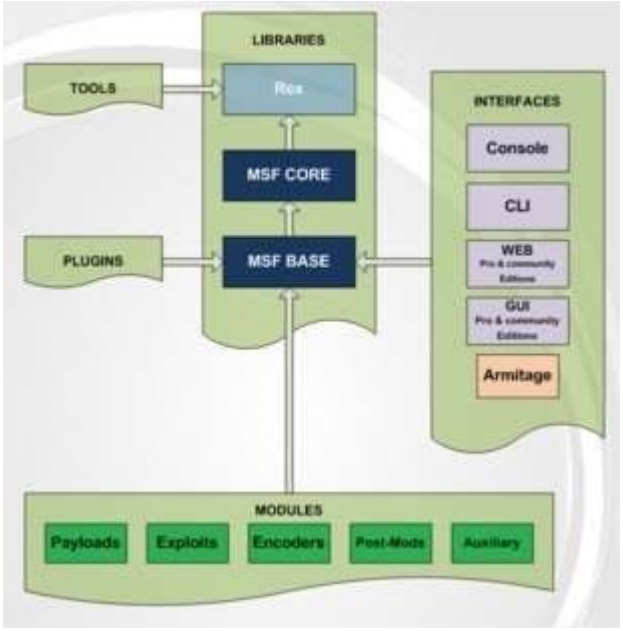

Figura 43: Tecnologia Metasploits (VIVAOLINUX, 2012).

De acordo com Vinicius (2011), o REX (Ruby Extension Library) é o núcleo do Metasploit, ele disponibiliza a API com funcionalidades que ajudam no desenvolvimento de um *exploit*, além de bibliotecas, sockets e protocolos.

- O Core do *Framework* é constituído de subsistemas que controlam sessões, módulos, eventos é a API base.
- O BASE fornece uma API amigável e simplifica a comunicação com outros módulos, interfaces e *plugins*.
- Na camada MODULES é onde reside os *exploits* e *Payloads*, basicamente os *exploits* são programas escritos para explorar alguma falha e o *Payload* é como um complemento para o *exploit*. Basicamente o *Payload* é o código que vai ser injetado no alvo, ao ser injetado alguma ação pré-definida será executada, por exemplo, realizar um download, executar um arquivo, apagar alguma informação ou estabelecer uma conexão com outro sistema.

2.3.2.2. Interfaces

O Metasploits fornece aos usuários três diferentes interfaces no momento da interação:

- *Command Line* Interface: Essa interface de interação automatiza os testes de sequências de *exploits*, sendo executado através do comando “msfcli”.
- Console Interface: Essa interface fornece ao usuário uma interação via linha de comando, que tem como característica a velocidade do seu funcionamento e a sua flexibilidade. Para utilizar essa interface, execute o comando “msfconsole”.
- Interface *web*: Utilizadas em circunstâncias especiais, para utilizar esta versão *web* do Metasploit, execute o comando “msfweb”.

2.3.2.3. MSFCLI

O msfcli fornece uma poderosa interface de linha de comando para o quadro. Isso permite que você facilmente adicionar *exploits* Metasploit em todos os *scripts* que você pode criar.

Benefícios da interface msfcli:

- Dá su*port*e na execução de *exploits* e módulos auxiliares;
- Interface àgil para tarefas específicas;

- Utilizado para testar ou desenvolver novos *exploits*;
- Ferramenta utilizada para a exploração *one-off*;
- Interface utilizada para uso em *scripts* automatizados;
- O único problema real de msfcli é que ele não é su*port*ado tão bem quanto msfconsole e só pode lidar com uma coisa de cada vez, e também não su*port*a recursos de automação avançados de msfconsole.

2.3.2.4. MSFCONSOLE

O msfconsole é provavelmente a interface mais popular para o Metasploit *Framework* (MSF). Ele fornece uma interface que permite acesso eficiente a praticamente todas as opções disponíveis no MSF. O msfconsole pode parecer simples no início, mas uma vez aprendida a sintaxe dos comandos verás o poder desta interface.

Benefícios a usar msfconsole:

- É a única interface que su*port*a acessos à maioria dos recursos dentro do Metasploit;
- Fornece uma interface baseada em console para o usuário;
- Contém a maioria dos recursos e é a interface MSF mais estável;
- Su*port*e, execução e conclusão de comando *readline* completa.

2.3.2.5. MSFCONSOLE *COMMANDS*

Back

Após tiver terminado de trabalhar com um módulo específico, ou se você equivocadamente, selecionou o módulo errado, você pode desfazer a voltar ao estado anterior digitando o comando *back*.

Check

Esse comando *check* é utilizado para verificar se um alvo é vulnerável a um determinado *exploit*, ao invés de realmente explorá-lo.

Connect

É clone do *telnet* embutido no msfconsole que su*port*a SSL, proxy, gerando e enviando arquivos. Ao digitar o comando *connect* com um endereço IP e número de porta, você pode

se conectar a um host remoto de dentro msfconsole, com as mesmas funções do *netcat* ou o *telnet*.

Info

O comando ‘info’ irá fornecer informações detalhadas sobre um módulo em particular incluindo todas as opções, metodos e outras informações.

Irb

Executando o comando ‘irb’ comando irá deixá-lo em modo de interpretador Ruby *Shell*, de onde você pode emitir comandos e criar Metasploit. Esse recurso também é muito útil para entender o funcionamento do *Framework*.

Jobs

Os *jobs* são módulos que estão em execução em segundo plano. Esse comando tem a capacidade de listar os *jobs* e finalizar determinado *jobs*.

Kill

O comando ‘*kill*’ vai finalizar qualquer tarefa em execução quando fornecidos com a ID do mesmo.

Load

O comando ‘*load’* carrega um *plugin* do Metasploit no diretório de *plugins*. Os argumentos são passados como *keys*=val no *shell*.

Route

O comando "*route*" permite seguir rotas através de uma sessão ou 'comm', proporcionando capacidades eficientes de articulação. Para adicionar uma rota, você passa a máscara de sub-rede de destino e de rede seguido por o número da sessão (comm).

Search

O ‘*search’* é utlizado para pesquisar um *Payload*, *exploits*, *scripts* que estão localizados dentro do módulo.

Sessions

O comando ‘*sessions*’ permite listar, interagir e matar sessões abertas. As *sessions* podem ser shells, *sessions* Meterpreter, VNC, etc.

Set

O comando ‘set’ permite que você configure opções de estrutura e parâmetros para o módulo atual que você está manuseando.

Unset

Esse comando anula/remove um parâmetro previamente configurado com set.

Show

O comando ‘show’ no msfconsole exibe todos os módulos dentro do Metasploits.

Auxiliary

Executando o comando ‘show auxiliary’, exibirá uma lista de todos os módulos auxiliares disponíveis no Metasploit.

Exploits

Executando o comando ‘show *exploit*’, exibirá uma lista de todos os *exploits* disponíveis no Metasploit.

Payload

Executando o comando ‘show *Payloads*’, exibirá apenas os pyaloads que são compatíveis com o alvo a ser explorado.

Options

Executando o comando ‘show option’, exibirá quais configurações estão disponíveis ou estão sendo exigidos por esse módulo específico.

Targets

Se você não tem certeza se um sistema operacional é vulnerável a determinado *exploit*, o comando ‘show *Targets*’ exibirá os *Targets* compativeis com a versão do sistema operacional.

Encoders

Executando o comando 'show encoders', exibirá uma lista dos codificadores que estão disponíveis dentro de MSF.

Use

O comando ‘use' é utilizado para selecionar o módulo especifico para a execução do *exploit*.

2.3.2.6. *Payload*

O *Payload* é um módulo utilizado para explorar, e dentro desse módulo existe diferentes tipos de *Payload*:

- *Singles*
- *Stagers*
- *Stages*
- *Inline*

Singles

Singles são *Payloads* que são auto-suficientes e completamente independente, podendo utilizar manipuladores não embutidos no Metasploit, como telnet.

Stagers

Stagers é configuração de uma conexão de rede pequena e confiável entre o *hacker* e a vitima.

Stages

Stages são componentes *Payload* que são baixados por módulos *Stagers* com recursos avançados sem limites de tamanho, como Meterpreter, Injeção de VNC.

Inline (*Non Staged*)

Payload Inline contém no código de exploração, uma grande variedade de *shell* para determinada tarefa, sendo mais estáveis por possuir todos recursos necessários em sua estrutura.

Ao aprender a usar Metasploit, você vai encontrar muitas interfaces diferentes para usar com esta ferramenta de *hacking*, cada uma com as suas próprias forças e fraquezas. Como tal, não há uma interface perfeita para usar o Metasploit, embora o msfconsole é a única maneira su*port*ada para acessar a maioria dos comandos Metasploit. Ainda é benéfico, no entanto, ser familiarizado com todas as interfaces Metasploit.

2.3.3. SQLMap

Segundo Guimarães e Stampar (2006-2016), o SQLmap (*Structured Query Language* ou Linguagem de Consulta estruturada) foi criada em julho de 2006 por Daniele Belluci, e em setembro do mesmo ano, deixou o projeto e Bernardo Damele assumiu. É uma ferramenta *open source*, desenvolvida em python e multi-plataforma, podendo ser instalado em qualquer sistema operacional, é muito utilizada em testes de penetração (*pentest*), por ter seus processos de detecção e exploração de falhas SQL *injection* automatizadas.

Essa ferramenta é utilizada para testar sites com códigos impróprios e URLS ligadas a bases de dados por meio de comandos python em linha. Além de oferecer as funções para detectar e explorar as vulnerabilidades, ele consegue também, realizar a infiltração no sistema banco de dados vulnerável, capturando base de dados do banco, tabelas, credenciais de usuários, usuário administrador e ainda, a manipulação de tabelas. (GUIMARAES; STAMPAR, 2006-2016).

De acordo com Guimarães e Stampar (2006-2016), o SQLMAP possui suporte para diversos banco de dados, entre eles: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB e HSQLDB. Se o banco servidor for MySQL, PostgreSQL ou Microsoft SQL Server, é possível fazer download ou upload de qualquer arquivo de banco de dados. Tem como características, su*port*e para cinco tipos de SQL *injection*:

- *Boolean-based blind*
- *Time-based blind*
- *Error-based*
- UNION *query*
- *Stacked queries*

2.3.3.1. Boolean-*based blind*

Ao utilizar comparações booleanas em conjunto com funções de *string* do banco de dados, força na aplicação um verdadeiro ou falso, montando corretamente ou não a página.

- Técnica de ataque 'cego' baseado em lógica;
- É o tipo mais comum;

- Também conhecido como 'or 1 = 1';

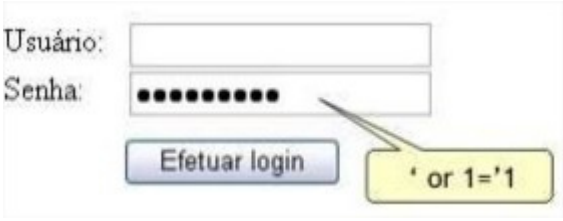

Figura 44: Ataque lógico (DOCPLAYER, 2009).

SQL original:

```
SELECT * FROM usuarios WHERE username = '$u' AND senha = '$p';
```

Figura 45: SQL Boolean-*based* sem *injection*.

SQL com *injection*:

```
SELECT * FROM usuarios WHERE username = '' AND senha = '' or 1 = 1;
```

Figura 46: SQL Boolean-*based* sem *injection*.

2.3.3.2. *Time-based blind*

Neste tipo de ataque a ferramenta cronometra o tempo necessário para executar a requisição à aplicação *web* sem injetar código SQL e, em seguida, injeta códigos SQL que exigem um longo tempo de processamento. Ao comparar o tempo necessário para a aplicação *web* responder entre as diferentes tentativas, é possível identificar se o banco de dados executou o código, o que representaria uma vulnerabilidade. Esse tipo de ataque é bastante utilizado para descobrir qual o banco de dados e versão do alvo.

Mysql	SLEEP(*time*)
SQL Server	WAIT FOR DELAY 'hh:mm:ss'
Oracle	SLEEP(*time*)

2.3.3.3. Error-based

Neste tipo de ataque a ferramenta injeta um comando que causa erros que um banco de dados deve capturar. A ferramenta analisa a resposta da aplicação *web* para verificar se o erro ocorreu, o que significa que o banco de dados executou o código. Esse tipo de ataque exige que o analista de segurança informe o tipo do SGDB previamente.

SQL original:

```
row(1,1) > (select count(*),concat(usuario() ,
0x3a, floor(rand()*2))as x from (select 1 union select 2)
as a group by x limit 0,1)
```

Figura 47: SQL sem *Error-based injection*.

SQL com *injection*:

```
row(1,1) > (select count(*),concat(
(select cast( 0x3c73637269707420737263d27687474703a2f2f3132372e302e312e312f6861636b2e6a73e as char
,0x3a,floor(rand()*2))as x from (select 1 union select 2) as a group by x limit 0,1)
```

Figura 48: SQL com *Error-based injection*.

Após executada a SQL com *injection*, a página *web* retorna um *Error* esperado:

```
ERROR 1062 (23000): Entrada Duplicada '<script
src='http://127.0.1.1/hack.js'></script>:0' para a chave 'group_key'
```

Figura 49: SQL *Error-based injection* retornando erro.

Desta forma, se pode exibir uma mensagem ao usuário que acessar a página:

```
<meta http-equiv="refresh"
content="0;url=javascript:alert('Hacked!!!');"> ou <a href='#'>I'm Ownz You!!
</a> ou </title><script>alert("Hacked!!!!");</script>
```

Figura 50: SQL *Error-based injection* invadida.

2.3.3.4. UNION *query*

Neste tipo de ataque a ferramenta insere código utilizando a cláusula UNION. Esse ataque necessita que a página mostre vários dados de uma tabela, não uma única dupla. Ao

comparar os dados resultantes, a ferramenta identifica se conseguiu adicionar dados na seleção, o que indica que o banco de dados executou o código injetado.

SQL original:

```
SELECT NumeroConta,TipoConta,ValidadeConta,NomeConta FROM CartaoCredito
WHERE (NumeroConta =15874596 AND StatusCartao='Ativo') AND NomeUsuario='Fulano Beltrano'
```

Figura 51: SQL com UNION *query* sem *injection*.

SQL com *injection*:

```
SELECT NumeroConta,TipoConta,ValidadeConta,NomeConta FROM CartaoCredito
WHERE (NumeroConta =15874596) UNION SELECT NumeroConta,TipoConta,ValidadeConta,NomeConta
WHERE 1=1 (AND StatusCartao='Ativo') AND NomeUsuario='Fulano Beltrano'
```

Figura 52: SQL com UNION *query* com *injection*.

2.3.3.5. *Stacked Queries*

Este tipo de ataque ocorre quando um aplicativo está apoiando *stacked queries*. O SQLmap acrescenta um ponto e vírgula (;) para o valor do parâmetro vulnerável e acrescenta instrução SQL para ser executado. Usando esta técnica, é possível levar o sistema de arquivo a ler, escrever e executar comandos de acesso no sistema operacional, e dependendo do SGBD, descobrir acessos privilegiados (SYS, SYSTEM, SYSDBA, etc).

Podemos concluir, que essas ferramentas são essenciais para os testes de invasão. Além de serem extremamente eficientes nas buscas por vulnerabilidades e brechas de segurança, são ferramentas de *software* livre, onde as empresas que utilizam não precisam pagar nenhum centavo pela sua utilização.

3. UTILIZAÇÃO DAS FERRAMENTAS

Neste capitulo, será apresentada as funcionalidades das ferramentas estudas. Estas ferramentas têm por objetivo encontrar de maneira automatizada, o maior número possível de vulnerabilidades do alvo. O mecanismo básico de funcionamento consiste no envio de requisições e análise das respostas obtidas, em busca de evidencias de que alguma vulnerabilidade está presente.

3.1. INVASÃO COM METASPLOIT

Neste subcapitulo será apresentada as etapas de um teste de invasão com Metasploit, mostrando a estrutura dos testes em passos bem definidos, levantando informações, localizando vulnerabilidades existentes e por fim, ganhando acesso total na máquina do alvo.

3.1.1. Levantamento de Informações e Vulnerabilidades

O levantamento de informação é o primeiro passo para a realização bem-sucedida do teste de invasão, conhecer o seu alvo, descobrir algumas informações, facilitaram a invasão.

Para a invasão com Metasploit, a ferramenta utilizada para levantar informações do alvo foi o Nmap, por ser considerado um *portscan*, é uma ferramenta de segurança, usada para detectar portas abertas em computadores e serviços conhecido pela sua rapidez e eficiência na execução dos testes.

Primeiramente, vamos verifica se o alvo é protegido por algum *firewall* e saber quais portas são as mais utilizadas:

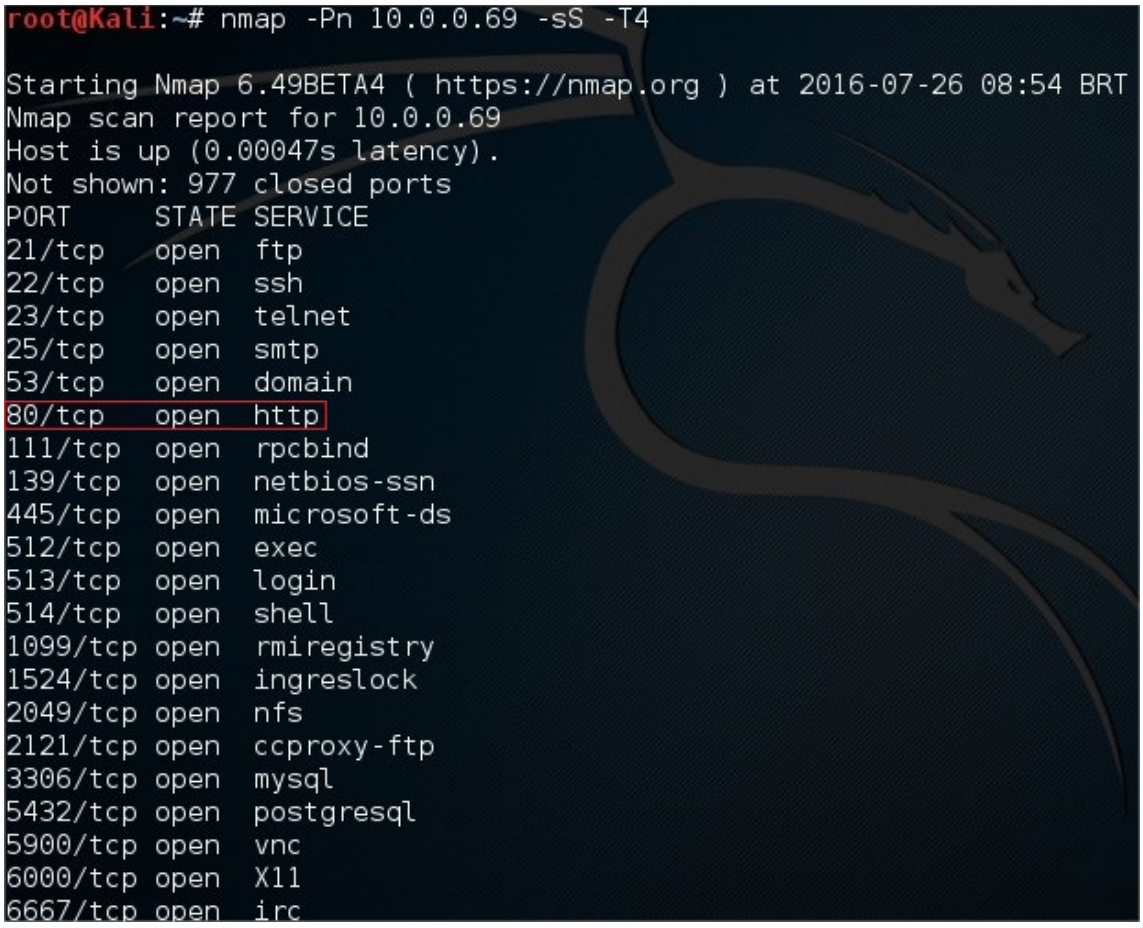

Figura 53: Localizando portas abertas.

Após a verificação realizada, nota-se a porta 80/tcp aberta (*open*) e vulnerável a invasões. A próxima etapa, é verificar o sistema operacional rodando na maquina do alvo pela porta 80/tcp:

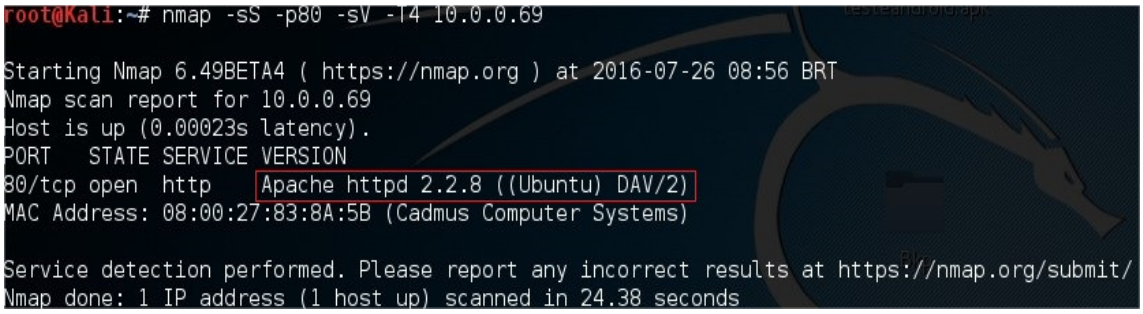

Figura 54: Verificando serviços ativos.

Com as informações obtidas, o próximo passo é explorar as vulnerabilidades encontradas.

3.1.2. Exploração

Para a exploração das vulnerabilidades encontradas, será utilizado o Metasploit, por conter em sua estrutura programas adequados especificamente para executar *exploits*, carregar módulos auxiliares.
Para executar o Metasploit, abra o terminal e digite “msfconsole” e aguarde enquanto a ferramenta é carregada.

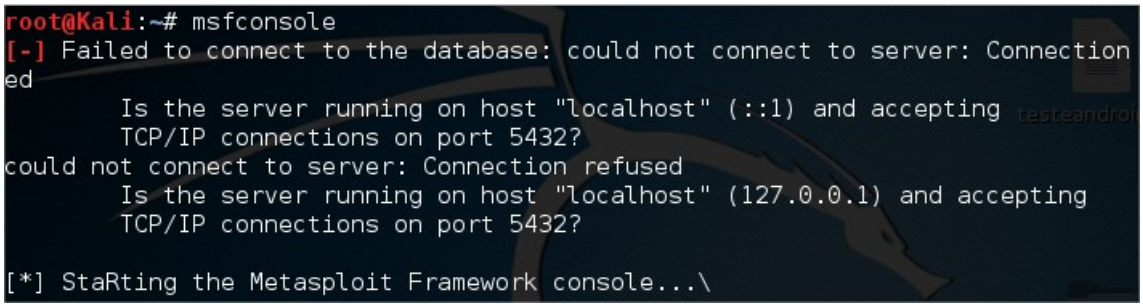

Figura 55: Iniciando o Metasploit.

Como será explorada a vulnerabilidade encontrada na porta 80/tcp, o *exploit* utilizado será o *webapp* por ser especificamente utilizado nesse tipo de ataque.

Figura 56: Carregando *exploit* WEBAPP.

Após carregado o *exploit*, será informado o IP do alvo, e carregado o *Payload* usado para ler e escrever dados em conexões de rede usando o protocolo TCP/IP.

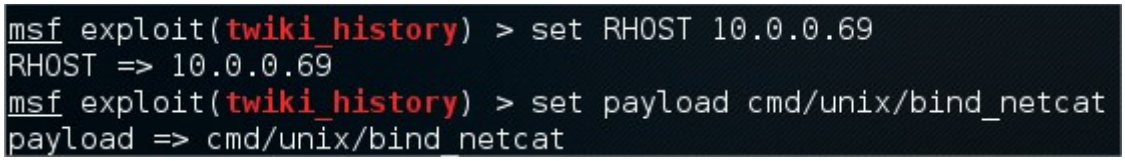

Figura 57: Definido host e carregando *Payload*.

Com o *Payload* carregado, será executado o *exploit BIND_NETCAT,* com o parâmetro –j para que ele trabalhe em *background.*

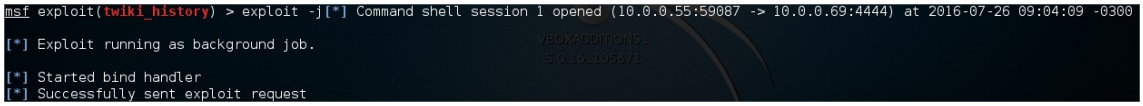

Figura 58: Executando *exploit* em *background*.

Com a execução do *exploit* em andamento, será aberta uma *session* em *background* ligada diretamente a máquina do alvo. Para ver se a *session* foi aberta, digite o comando *sessions* –l.

Figura 59: Verificando *session* aberta.

Após localizada a *session,* selecione a mesma para explorar a conexão criada com a máquina do alvo, com o um *exploit* baseado em interpretador *shell*.

Figura 60: Carregando *shell* e selecionando *session*.

Com o *exploit* carregado e a *session* selecionada, basta executá-lo.

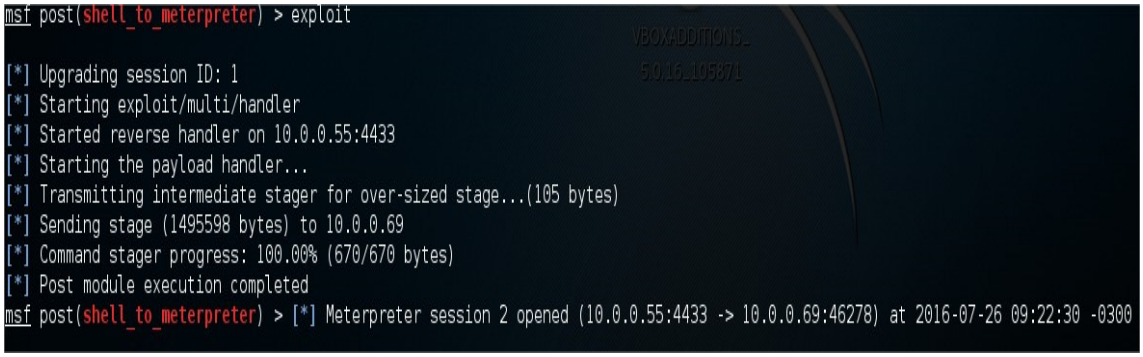

Figura 61: Executando *exploit*.

Após o *exploit* ser executado, será criada outra *session* ligada diretamente com a máquina do alvo.

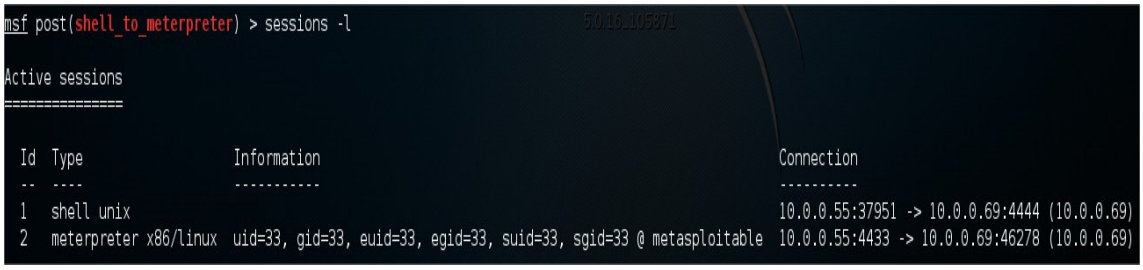

Figura 62: Verificando *sessions* abertas.

Para selecionar a segunda *session* ligada diretamente a máquina do alvo, digite o comando *set session* 2, e para interagir diretamente com a da máquina do usuário, digite *sessions* –i 2. O parâmetro –i serve para especificar o intervalo de tempo no qual as linhas de texto serão enviadas ou recebidas.

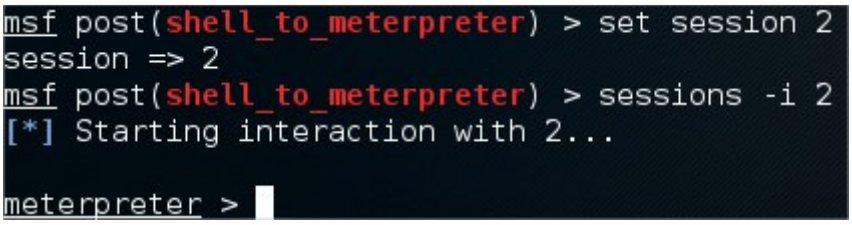

Figura 63: Selecionando *session* e abrindo interação meterpreter.

Com o acesso diretamente a máquina do alvo obtido, basta digitar o comando *shell* para abrir o terminal do alvo.

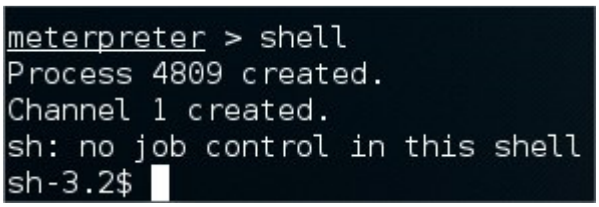

Figura 64: Acesso total a máquina do alvo obtido.

Com o terminal aberto e um *hacker* com conhecimento avançado em comandos Linux, o alvo terá informações sigilosas descobertas sem se quer saber.

Concluímos que os usuários devem se atentar a atualizações de sistema e resguardarem o mesmo com regras de *firewall*, configurações corretas, mesmo não existindo um sistema 100% seguro, uma boa configuração de *firewall* forçará o invasor mais recursos para invasão, levando o mesmo a uma possível desistência.

3.2. INVASÃO COM SQLMAP

Neste subcapitulo, será apresentada a funcionalidade da ferramenta SQLmap, a ferramenta que auxilia os analistas de segurança, *crackers* ou *hackers*, na execução de processos repetitivos e detecção de alguns tipos de vulnerabilidades na base de dados da aplicação *web*. Se for detectada alguma vulnerabilidade na base de dados da aplicação, o usuário pode escolher entre uma variedade de opções que o SQLmap disponibiliza para explorar os dados armazenados dentro do banco de dados deste sistema ou site, tais como, extrair a lista de usuários, senhas, privilégios, tabelas dentre outras informações.

3.2.1. Levantamento de Informações e Vulnerabilidades

Para executar o SQLmap, é necessário executar o parâmetro GET (exemplo: www.site.com/index.php?id=1‘). Neste exemplo será utilizado um site legalmente voltado para testes de SQL *injection*.

Após aberto o SQLmap, execute o comando abaixo:

Figura 65: Definindo site à ser explorado.

--dbs: Lista os bancos de dados do site.

-u: URL, endereço do site.

Durante o processo de execução o SQLmap já detecta que o banco de dados do site é MySQL e o usuário pode ser questionado, se deseja pular a verificação para outros tipos

de banco de dados ou se você que fazer um teste mais profundo sobre o banco de dados detectado, neste caso responda Y (Yes/Sim).

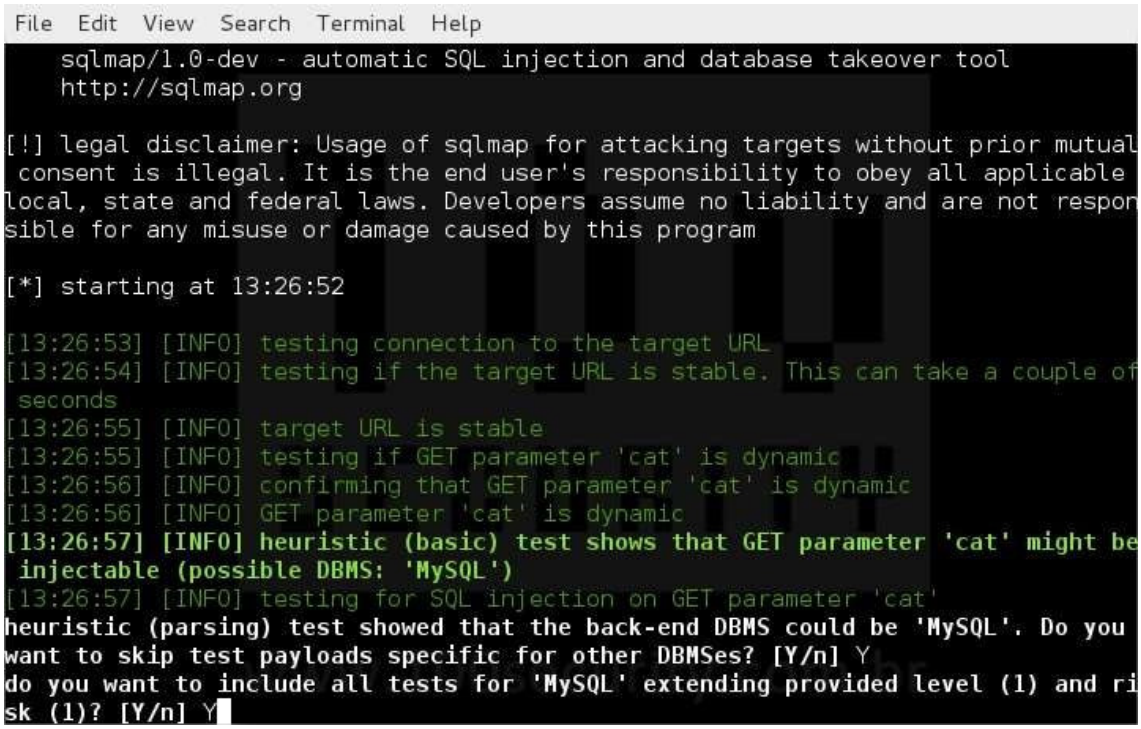

Figura 66: Realizando exploração e encontrando banco de dados.

O SQLmap detectou que o parâmetro cat está vulnerável e o usuário será questionado se deseja realizar outros testes, no momento selecione N (Not/Não).

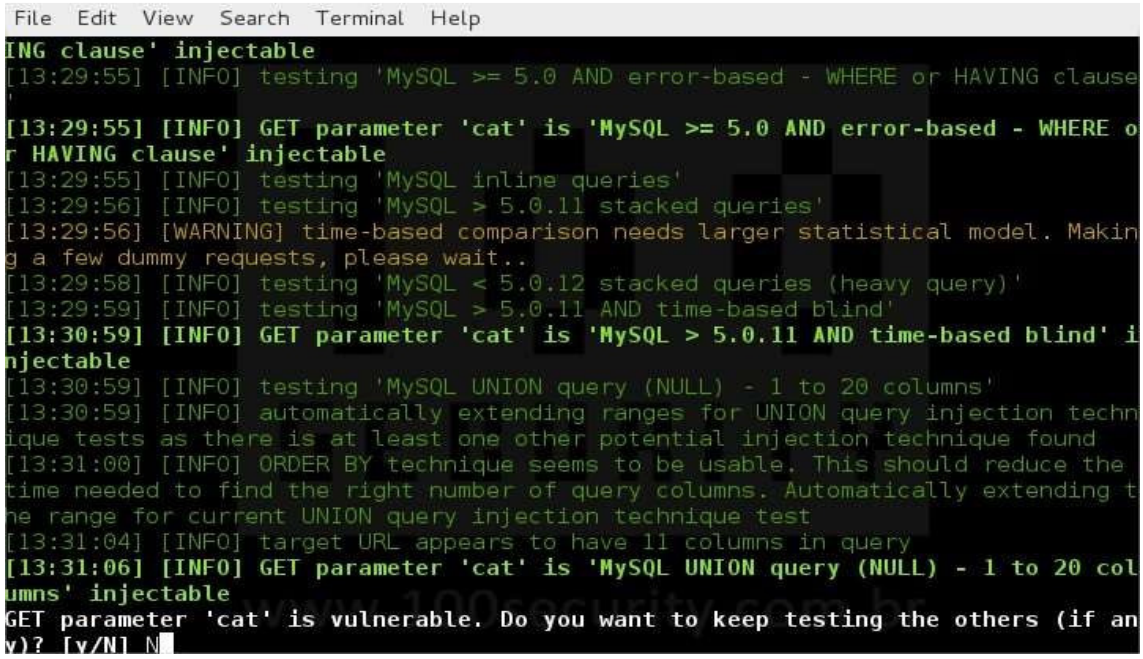

Figura 67: Explorando parâmetro CAT.

Foram encontrados dois bancos de dados.

Figura 68: Banco de dados encontrados.

3.2.2. Exploração

Após serem listados os bancos de dados, será realizada a extração das tabelas do banco de dados acuart.

Figura 69: Extraindo tabelas do banco de dados acuart.

-D: Define o banco de dados que será avaliado.

acuart: Banco de dados do site.

--tables: Lista todas as tabelas do banco de dados informado.

Figura 70: Tabelas do banco de dados acuart listadas.

Com as tabelas listadas, agora será realizada a extração todas as colunas da tabela users.

Figura 71: Extraindo colunas da tabela users.

-D: Define o banco de dados que será avaliado.

acuart: Banco de dados do site.

--tables: Lista todas as tabelas do banco de dados informados.

-T: Define a tabela que será avaliada.

users: Tabela do banco de dados.

--columns: Lista todas as colunas da tabela users.

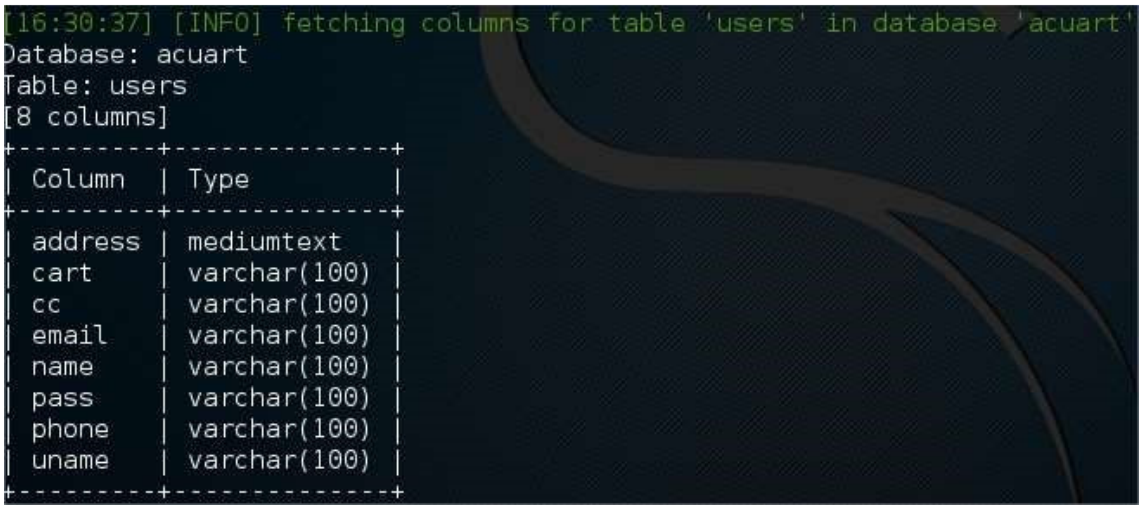

Figura 72: Colunas da tabela users listadas.

Após ser listada as colunas da tabela, será feito um *dump* dos campos name, uname, pass e email.

Figura 73: Extraindo *dump* de campos da tabela users.

-C: Define as colunas que será avaliada.

name, uname, pass, email: Lista das colunas da tabela users.

--*dump*: Extrai todas as informações contidas nas colunas da tabela informada.

Figura 74: Dados dos campos obtidos.

Após o *dump* ser realizado, foi descoberto o usuário e senha do site explorado. Para verificar se os dados extraídos estão corretos, basta ir ao site e logar.

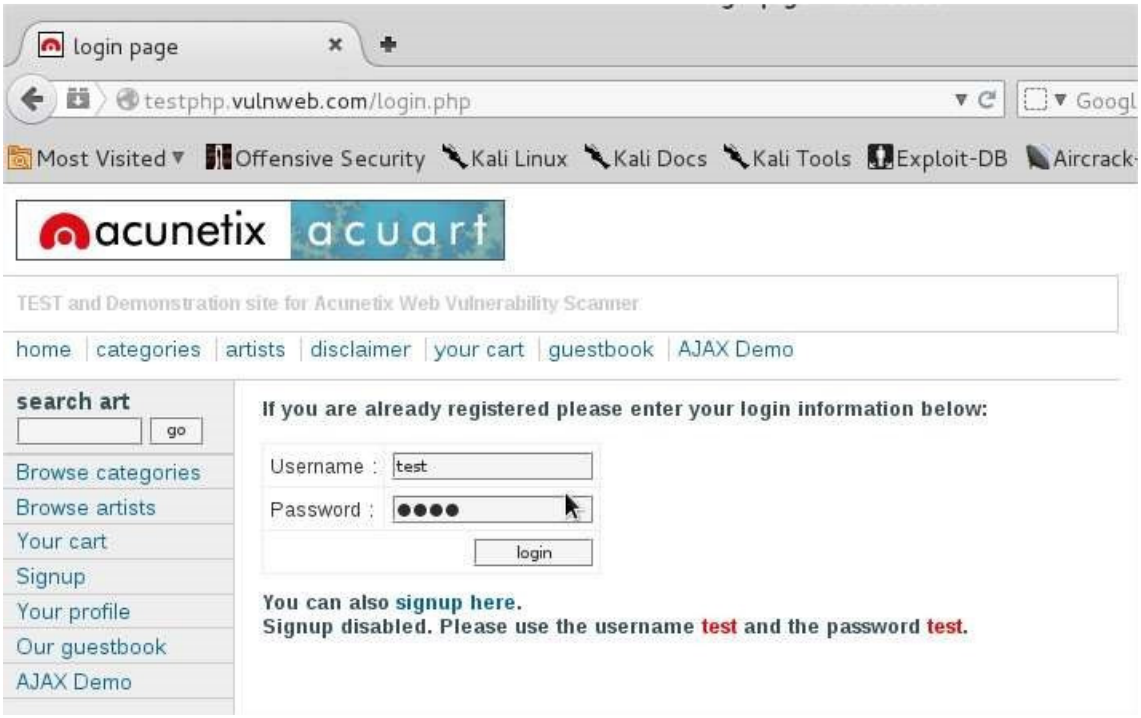

Figura 75: Acessando o site com as informações obtidas.

Login realizado com sucesso.

Figura 76: Login realizado com sucesso.

Podemos concluir, que o SQLmap é uma ferramenta com alto potencial de invasão a aplicações web, além de detectar e explorar vulnerabilidades de SQL *injection* em sites e aplicações *web*, possui recursos que possibilitam extrair lista de usuários, senhas, tabelas, privilégios que estão armazenados no banco de dados, de uma forma simples, eficiente e rápida.

4. CONCLUSÃO

Como podemos observar ao longo deste trabalho, o acesso à informação tem ficado cada vez mais fácil, servidores e máquinas são alvos constantes de ataques, por isso devem ser bem configurados para que não seja fácil o acesso a eles, mesmo com a instalação zerada, ou seja, sem nenhum serviço além do básico, podemos encontrar algumas brechas no Linux.

Estas ferramentas podem ser utilizadas para práticas benéficas como correções de falhas e vulnerabilidades por empresas de auditoria de segurança, administradores de redes dentre outros. Pode ser utilizada também para fins maliciosos, por hackers e crackers, que usufruem dessas ferramentas para invadir sistemas e computadores, afim de obter informações sigilosas, podendo causar danos financeiros e morais a vítima.

No Linux, por ser falado sempre que é o mais seguro, encontramos falhas de configuração. Mas o fato é que o Linux se bem configurado, com um *firewall*, serviços atualizados se torna muito mais seguro. Para concluir, o im*port*ante sempre é estar de olho nos serviços, atualizações e quando possível fazer um teste de penetração, o ideal seria fazer de 6 em 6 meses, pois diariamente são descobertos novos métodos de ataque.

REFERÊNCIAS

Broad, James. Bindner, Andrew. **Hacking com Kali Linux: Técnicas práticas para testes de invasão**. 1. Ed. Novatec Editora, 2014.

BROCKMEIER, Joe 'Zonker'. **Beginner's Guide to Nmap**. Disponível em <http://www.networkuptime.com/nmap/index.shtml>. Acesso em: 04 mar. 2016.

FERREIRA, F.; PEREIRA, J. **Revista Segurança Digital** - 1ª Ed, Junho 2011, 2011. Disponível em: <http://www.segurancadigital.info/sdinfo_downloads/1_edicao_julho_01_07_2011.php>. Acesso em: 12 out. 2015.

From SQL Injection To 0wnage Using SQLMap. Disponível em <http://niiconsulting.com/checkmate/2014/01/from-sql-injection-to-0wnage-usingsqlmap/>. Acesso em: 04 mar. 2016.

GIAVAROTO, Silvio César Roxo. SANTOS, Gerson Raimundo dos. **Backtrack Linux: Auditoria e Teste de Invasão em Redes de Computadores**. Rio de Janeiro, Ciência Moderna Ltda. (2013).

Guia de Referência do Nmap. Disponível em <https://nmap.org/man/pt_BR/>. Acesso em: 04 mar. 2016.

GUIMARES, Bernando Damele Assumpcão. STAMPAR, Miroslav. **SQLmap**. Disponível em: <http://sqlmap.org/>. Acesso em: 03 mar. 2016.

HENRIQUE, Marcos. **SQLmap na Prática.** Disponível em <http://www.100security.com.br/sqlmap/>. Acesso em: 03 mar. 2016.

KENNEDY, O'GORMAN, KEARNS e AHARONI. METASPLOIT. **The Penetration Tester's Guide**. São Francisco: No Starch Press, 2011.

MESSER, James. **Secrets of Network Cartography: A Comprehensive Guide to nmap**. Disponível em <http://www.networkuptime.com/nmap/index.shtml>. Acesso em: 04 mar. 2016.

MUNIZ, Joseph; LAKHANI, Aamir. **Web Penetration Testing with Kali Linux**. 2013. Packt Publishing Ltd. Livery Lugar 35 Livery Rua Birmingham B3 2PB, Reino Unido, 2013.

O que é Kali Linux?. Disponível em <http://br.docs.kali.org/introduction-pt-br/o-quee-o-kali-linux>. Acesso em: 05 mar. 2016.

Offensive Security Training, Certifications and Services. Disponível em: <https://www.offensive-security.com/metasploit-unleashed/multiple-os-post-gather-modules/>. Acesso em: 07 jun. 2016.

PARMAR, Primesh. FALEOL, Alex. Aplicação de *Pentest* em Sistemas Computacionais para Análise de Vulnerabilidades: Um Estudo de Caso. In: **ENCONTRO REGIONAL DE COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO, 2013.** Manaus, Brasil. Anais do Encontro Regional de Computação e Sistemas de Informação. Manaus: FUCAPI, 2013.

VINICIUS, Elger. **Metasploit Framework.** Disponível em <http://securityint.org/artigos/2010/11/15/metasploit-*Framework*> Acesso em: 05 mar. 2016.