# CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA

# PAULA SOUZA

**FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA**

**CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADOR**

**JEAN KARL DA SILVEIRA**

**MARIO FERNANDES TESTONI JUNIOR**


# UM ESTUDO SOBRE TÉCNICAS E SOFTWARES PARA TESTES DE PENETRAÇÃO EM SERVIDORES DE REDE.


**LINS/SP**

**2° SEMESTRE /2012**

# CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA

# PAULA SOUZA

**FACULDADE DE TECNOLOGIA DE LINS PROF. ANTONIO SEABRA**

**CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADOR**

**JEAN KARL DA SILVEIRA**

**MARIO FERNANDES TESTONI JUNIOR**


# UM ESTUDO SOBRE TÉCNICAS E SOFTWARES PARA TESTES DE PENETRAÇÃO EM SERVIDORES DE REDE.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins para obtenção do Titulo de Tecnólogo em Redes de Computadores.

Orientador: Prof. Mestre. Julio Fernando Lieira


**LINS/SP**
**2° SEMESTRE /2012**

**JEAN KARL DA SILVEIRA**

**MARIO FERNANDES TESTONI JUNIOR**

# UM ESTUDO SOBRE TÉCNICAS E SOFTWARES PARA TESTES DE PENETRAÇÃO EM SERVIDORES DE REDE.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Redes de Computadores sob a orientação do Prof. Me.Julio Fernando Lieira.

Data de aprovação: 11 de dezembro de 2012

_____

Orientador: Prof. Me. Julio Fernando Lieira

_____

Prof. Me. Adriano Bezerra

_____

Prof. Me. Adriano De Souza Marques

Dedico este trabalho à minha família e à minha namorada Louise que tem me apoiado durante toda esta jornada.

**Jean Karl da Silveira**

À minha esposa Valéria e filhos Felipe e Mariana que me dedicam tanto amor e carinho incentivando–me nessa jornada, e aos meus Pais Mario e Teresa a quem tudo devo.

**Mario Fernandes Testoni Junior**

# AGRADECIMENTOS

Inicialmente agradeço a Deus que me conduziu durante todo este árduo e longo trajeto, dando-me condições e capacitando-me para a conquista e realização deste sonho.

Gostaria de agradecer também ao meu orientador Me. Julio Fernando Lieira pela paciência para sanar minhas dúvidas e pelo comprometimento para com o meu trabalho, o que sem dúvidas foi peça fundamental para a sua conclusão.

Agradeço não somente a ele, mas a todos os professores e colegas que desde o primeiro semestre me ajudaram e acrescentaram conhecimento a minha vida.

Sou muito grato ao meu parceiro Mario Fernandes Testoni Junior por complementar minhas idéias, se comprometer com o projeto, pela força de vontade para concluir este trabalho e pelo incentivo que me passava nas horas em que pensei em desanimar.

Agradeço à minha família e à minha namorada que muitas vezes me incentivou e me apoiou durante todo este período.

E por fim, agradeço a todos que direta ou indiretamente participaram ou contribuíram para a conclusão deste trabalho.

**Jean Karl da Silveira**

# AGRADECIMENTOS

Ao final da realização de mais um sonho e início de uma nova jornada vislumbrada graças aos ensinamentos recebidos e amizades conquistadas, expresso meus sinceros agradecimentos ao Professor Me Julio Fernando Lieira, meu orientador, que com sua sapiência soube guiar-me na realização deste trabalho.

Ao companheiro de autoria Jean Karl da Silveira, meu agradecimento e respeito pela parceria neste trabalho.

Aos demais Professores e amigos de turma deixo também meus agradecimentos pela amizade, companheirismo e conhecimentos transmitidos.

**Mario Fernandes Testoni Junior**

# RESUMO

Com esse estudo objetivou-se conhecer os tipos de ataques mais comuns a servidores, bem como as técnicas e ferramentas utilizadas pelos invasores para obter informações sobre o alvo e tentar a possível invasão. Além disso, foi feito um breve relato sobre as metodologias mais utilizadas para teste de penetração. Procurou-se também conhecer o *Backtrack* 5 r3, que é uma distribuição *Linux* focada em teste de segurança e de penetração, para concluir se um administrador de redes com estes conhecimentos pode testar e fortalecer a segurança de seus próprios servidores. Para tanto, foi criado um ambiente de testes, com dois servidores virtualizados, sendo um com sistema operacional *Windows* 2000 *server* e outro com sistema operacional linux *Ubuntu* 12.10 e neles aplicados os conhecimentos adquiridos para comprovação de sua eficiência. Finalizando concluiu-se que a Backtrack se mostrou um bom aliado ao administrador de redes em sua busca por melhoria na segurança, entretanto ficou claro que esta não deve ser a única fonte de recurso para tal finalidade.

Palavras-chave: *Pentest*, *backtrack, s*egurança, servidor.

# ABSTRACT

The Objective of this study was to know the most common types of attacks to servers, the techniques and tools used by invaders for obtain information about the target and the possible invasion, attempt also was made a brief report on the methodologies used to penetration testing. We also sought to know the Backtrack 5 r3, which is a Linux distribution focused on security testing and penetration, to conclude if a network administrator with this knowledge can test and strengthen the security of their own servers. To that end, we created a test environment with two virtualized servers, one running Windows 2000 server operating system and another with linux Ubuntu 12.10 and then applied the knowledge gained to prove its efficiency through Backtrack. Finally it was concluded that the tool has proved a good ally to the network administrator in your quest for improved security, however it became clear that this should not be the only source of funds for this purpose.



Keywords: Pentest, backtrack, security, server.

# LISTA DE ILUSTRAÇÕES

# LISTA DE ABREVIATURAS E SIGLAS

ACK - *Acknowledge*

CERT - Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil.

ICMP - *Internet Control Message Protocol*

IP - *Internet Protocol.*

ISECOM - *Institute for Security and Open Methodologies*

ISSAF - *Information Systems Security Assessment Framework*

PenTest - Teste de Penetração (invasão) em Sistemas Informatizados.

VM - *Virtual Machine*

DDoS - *Distributed Denial of Service*

DLL - *Dynamic-link library*

DNS - Domain Name System

DoS - *Denial of Service*

NIST - *National Institute of Standards and Technology*

OISSG - *Open Information Systems Security Group*.

OSSTMM - *Open Source Secutory Testing Methodology Manual*

OWASP - *Open Web Application Security Project*

RST - *reset*

RPC - *Remote Procedure Call*

RVA - *Risk Value Assessment*

SQL - *Structured Query Language*, ou Linguagem de Consulta Estruturada.

SYN - *synchronize*

SSH - *Secure Shell*

TCP - *Transmission Control Protocol*

TTL - *Time To Live*

UDP - *User Datagram Protocol*

**SUMÁRIO**

# INTRODUÇÃO

É certo que a tecnologia da informação está presente em todos os segmentos da sociedade, tais como as redes financeiras, sistemas de comunicação, controle e geração de energia e sistemas de saúde, todos são dependentes de sistemas computacionais para funcionar. Sendo assim, é grande o número de pessoas que utilizam computadores pessoais para educação, entretenimento, compras e transações bancárias.

O advento da *internet* possibilitou inúmeras facilidades no dia-a-dia das pessoas, como o correio eletrônico e a própria “navegação” na *Web*, permitindo o acesso “anônimo” a quase todo tipo de informação ou sistema.

Nesse contexto, não causa espanto o fato de que essa revolução computacional tenha atingido também o mundo do crime. Segundo o Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil (CERT), em 2011 houve mais de 399 mil incidentes reportados espontaneamente, um aumento de 279% em relação a 2010, ou seja, se forem computados os incidentes que não são reportados esse número torna-se ainda mais assustador. (CERT,2012)

Dessa maneira, esse aumento dramático em tentativa de invasões a sistemas faz com que as autarquias, corporações e empresas em geral que têm seus sistemas conectados de alguma forma a *internet* tenha um cuidado redobrado com a segurança de seus dados, pois a cópia não autorizada (roubo), exclusão (perda) ou alteração (modificação) de informações pode causar grandes prejuízos financeiros, comprometendo dessa forma no futuro da empresa.

Entretanto, o lado financeiro não é o maior prejudicado por uma invasão, pois se um invasor alterar os dados do projeto de algum equipamento de segurança ou a composição de um medicamento ou ainda a de um sistema que controla máquinas e válvulas em indústrias, vidas poderão ser perdidas até que a falha seja corrigida.

Como vimos, a segurança em servidores requer atenção e dedicação de profissionais capacitados.

Diante disso, e como as empresas de uma forma ou de outra precisam de conexões com filiais ou com fornecedores e clientes, torna-se quase impossível manter seus servidores completamente isolados. Sabendo, também, que não existe um sistema de segurança ou antivírus que garanta cem por cento de confiabilidade,

as empresas passaram a buscar no mercado profissionais capacitados a detectar possíveis pontos vulneráveis em seus servidores e criar obstáculos, dificultando a invasão de pessoas não autorizadas. Assim nasceu o conceito de *pentest* ou teste de penetração.

O profissional desta área deve ser profundo conhecedor das técnicas e ferramentas utilizadas pelo invasor, ou seja, para saber se seu servidor ou sistema esta vulnerável o profissional tem que utilizar basicamente as mesmas técnicas e ferramentas que serão utilizadas pelo invasor (*hacker* ou *craker*).

Neste contexto, este trabalho realizou um estudo de metodologias e ferramentas de teste de penetração em servidores de rede. Especificamente, nesta pesquisa, é apresentado um estudo sobre a distribuição *Linux* focada em teste de segurança e de penetração, denominado *Backtrack*, que se encontra na versão 5.0 r3, e que foi escolhida por ser uma distribuição nova e que tem um grande número de ferramentas automatizadas que se bem utilizada pode facilitar o trabalho do administrador, visando concluir que, se conhecendo as formas de ataques a servidores e as ferramentas utilizadas pelos invasores, o administrador da rede poderá utilizá-las para fortalecer a segurança de seus próprios servidores.

O primeiro capítulo aborda as técnicas de ataque a servidores de rede, quais os principais tipos de ataque que os servidores podem sofrer, mostrando seu funcionamento e propósitos, bem como as contramedidas para minimizar os riscos de modo a não afetar a continuidade dos negócios da empresa.

Em seguida, no segundo capítulo, são discutidas algumas metodologias para a aplicação do teste de penetração, tais como *Open Source Secutory Testing Methodology Manual (OSSTMM)*, *Open Web Application Security Project (OWASP)*, *Information Systems Security Assessment Framework (ISSAF)* e *National Institute of Standards and Technology* (*NIST*).

No terceiro e último capítulo, tratou-se da aplicação da distribuição linux *BackTrack* no teste de penetração de servidores de rede e, para tanto, um cenário foi criado e os testes guiados por uma metodologia.

# 1 TÉCNICAS DE ATAQUES A SERVIDORES DE REDE

Este capítulo trata das técnicas de ataques a servidores, portanto vale lembrar que um invasor antes de iniciar o ataque e a possível invasão, busca primeiramente munir-se de informações sobre seu alvo, informações estas que são adquiridas das mais diversas formas e utilizando variadas ferramentas, porém a engenharia social foi e continua sendo uma das técnicas mais utilizadas para esta finalidade. Ela consiste em nada menos do que retirar informações de forma “social” das pessoas que por suas funções dentro da instituição alvo são portadoras de conhecimento que interessa ao atacante; o engenheiro social faz isso aproveitando-se das “fraquezas” humanas de forma inescrupulosa, ou seja o atacante utiliza seu poder de convencimento, as vezes se fazendo passar por outra pessoa (assumindo um personagem) para retirar informações de maneira sutil porém eficiente de outras pessoas, mesmo que para isso seja necessário mexer com os sentimentos da pessoa investigada. (MITNICK, 2006)

> O engenheiro social emprega as mesmas técnicas persuasivas que usamos no dia-a-dia. Assumimos papéis tentamos obter credibilidade. Cobramos obrigações recíprocas. Mas o engenheiro social aplica essas técnicas de uma maneira manipuladora, enganosa, altamente antiética, frequentemente com efeito devastador. (SAGARIN apud MITNICK, 2006, p. 189)

Assim o atacante consegue retirar de forma inescrupulosa informações importantes de maneira tão natural que o informante sequer se da conta do que está fazendo.

Outra forma de conseguir informações do alvo é pesquisar por meio de *sites* na *internet,* como o *Google,* por exemplo, ou ainda por meio de pesquisas nos órgãos oficiais de registro de domínios, no caso do Brasil o “registro.br”.(LIEIRA, 2011)

Entretanto, apesar de serem estas as fontes mais comuns de busca por informações sobre possíveis servidores alvos, o que sem dúvida renderia uma boa pesquisa sobre o assunto, este não é o foco deste trabalho, mas sim as técnicas utilizadas diretamente contra os servidores de rede.

Diversos são os tipos de ataque, entretanto, entre os que são dirigidos diretamente aos servidores, analisaremos os mais comuns e cujas ferramentas para teste fazem parte do pacote *Backtrack* 5.

## 1.1 COLETA DE INFORMAÇÕES NOS SERVIDORES E MAPEAMENTO DA REDE

Quando se pensa em um ataque à servidores e provável invasão o primeiro passo é obter o máximo possível de informações sobre o alvo, como por exemplo, os endereços *Internet Protocol* (IP) dos servidores, quais serviços estão sendo executados nos servidores, qual o sistema operacional, quais *softwares*, quais versões e quais vulnerabilidades, tanto dos *softwares* quando dos sistemas operacionais, além é claro, de nomes de usuários e senhas cadastradas no servidor. (ASSUNÇÃO, 2010)

Obviamente, estas não são informações fáceis de conseguir, e exatamente por isso, é que foram criadas as ferramentas que serão citadas a seguir, entretanto, quanto mais informações melhor para o atacante, pois com base nelas é que serão determinadas as etapas seguintes. (ASSUNÇÃO, 2010)

### 1.1.1 Vasculhando Informações nos Servidores DNS

*Domain Name System (*DNS*)* ou "Sistema de Nomes de Domínios", é um protocolo na *internet* e a função do servidor DNS é traduzir um nome, como www.xxx.com.br em endereço de IP do computador que realmente roda o serviço que se está tentando acessar. Como é mais fácil decorar um nome do que um número do endereço IP, o DNS torna-se um serviço essencial da *internet*, assim, quando é digitado um domínio em um navegador, este faz uma consulta ao servidor DNS, visando resolver este nome, ou seja, saber qual o endereço IP do computador que roda o serviço, para na sequência buscar por ele na rede e mostrar a informação solicitada. (FAIRCLOTH, 2011)

Assim sendo, uma das primeiras pesquisas que devem ser realizadas pelo atacante é a consulta ao servidor DNS para saber quais os endereços IP estão de frente para *internet* e configurados para rodar algum tipo de serviço. Para essa pesquisa existem diversas ferramentas, todavia como este trabalho está baseado no *Backtrack* 5, o *Dnsmap* foi escolhido por ser uma das ferramentas de linha de comando que faz parte de seu pacote de ferramentas. A função desta ferramenta é retornar com os endereços IP dos servidores que estão rodando algum tipo de serviço de frente para *internet* para o domínio consultado, a figura 1.1 mostra um exemplo de consulta DNS feita utilizando a ferramenta *Dnsmap*. (FAIRCLOTH, 2011. BACKTRACK, 2012)

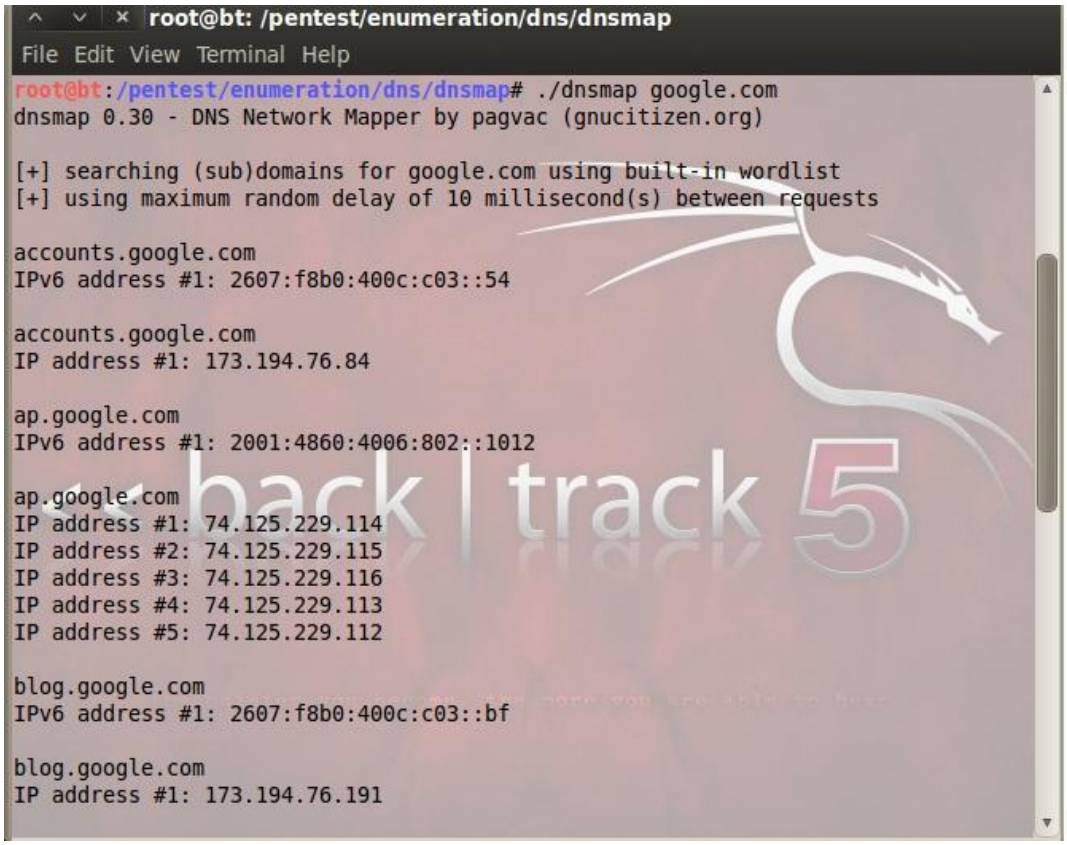


Figura 1.1 – Resultado da ferramenta *Dnsmap* no *Backtrack* 5.
Fonte: Elaborado pelos autores, 2012.

Como pode ser observado na figura 1.1, a consulta foi realizada ao servidor DNS do domínio do Google. A sintaxe utilizada é `./dnsmap Google.com`, e como resposta obteve-se todos os endereços IP de servidores que disponibilizam serviços do domínio pesquisado, inclusive com três deles já utilizando o IPv6, entretanto isso não significa que todos estão ativos neste momento, para isso é utilizado o comando a seguir.

### 1.1.2 Descobrindo Endereços IP

Outra ferramenta indispensável na busca por informações sobre alvo é o FPING, que também faz parte do pacote *Backtrack* 5. O que esta ferramenta faz é enviar uma *solicitação Internet Control Message Protocol* (ICMP) para todos os endereços dentro da faixa de IP solicitada, ou seja, é como enviar um comando *ping* para cada endereço IP que quiser consultar, para saber se este computador está ativo ou não, entretanto este comando faz isso de forma automática e o retorno (resposta) pode ser armazenado em arquivo para posterior consulta, a figura 1.2

mostra a tela de retorno da ferramenta *Fping*. Como as consultas estão sendo feitas pela *internet* esta ferramenta fará uma varredura dos endereços IP válidos do alvo em questão e com o retorno desta consulta juntamente com a consulta do *Dnsmap* podemos descobrir quais são os IP válidos que estão de frente para a internet e, dentre esses, quais são servidores. (BACKTRACK, 2012)

Apesar de ser uma ferramenta interessante, ela pode ser facilmente detectada pelo administrador da rede, uma vez que fica registrado o *log* da consulta, além disso, uma contramedida pode ser facilmente implementada controlando as respostas ICMP e ou limitando-as através de um *firewall.*

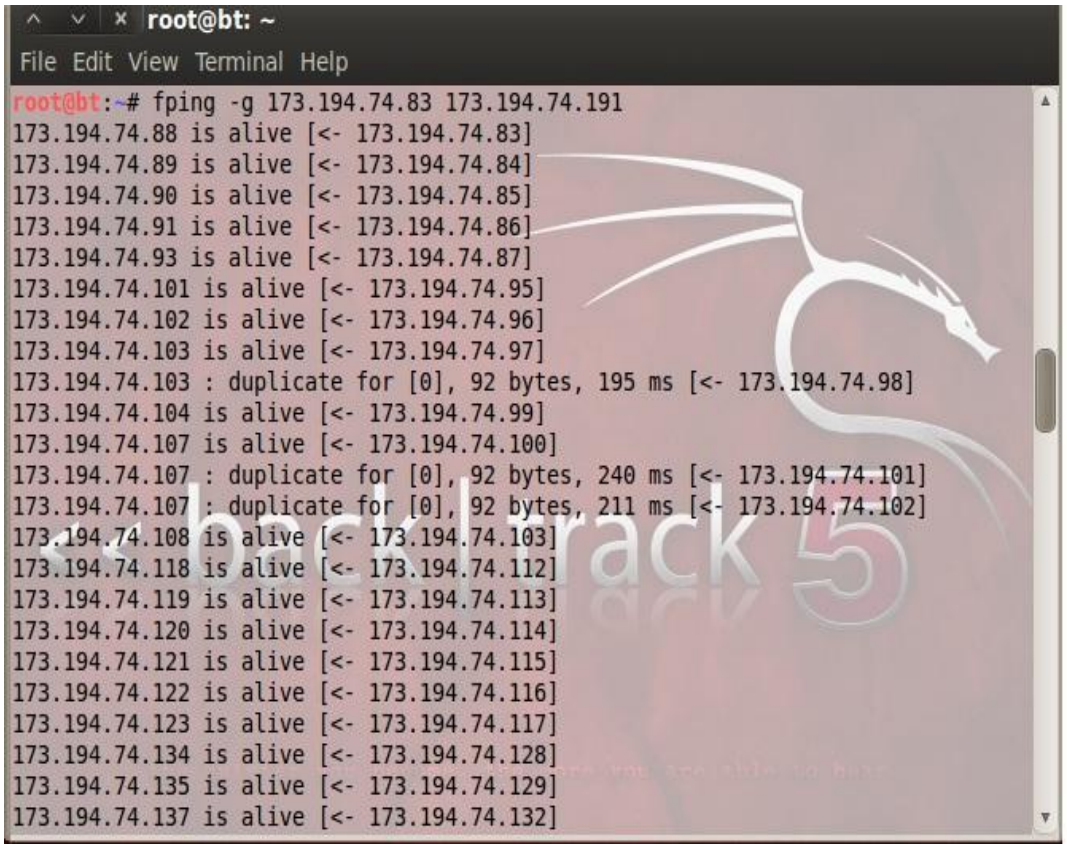


Figura 1.2 – Resultado da ferramenta `Fping` no *Backtrack* 5.
Fonte: Elaborado pelos Autores, 2012.

O resultado (figura 1.2) mostra que foi pesquisada a faixa de IP 173.194.74.83 à 173.194.74.191 e que alguns estão ativos (alive), note que existem intervalos, por exemplo, entre os IP 173.194.74.108 e 173.194.74.118 que não responderam por estarem inativos ou protegidos por regras de firewall para não responder a este tipo de consulta.

### 1.1.3 Traçando a Rota até os Servidores

Ainda na linha de pesquisa sobre o alvo, existem mais duas ferramentas extremante importantes para a obtenção de informações sobre o alvo, são elas o *Traceroute* e o *Nmap*.

O *Traceroute* é utilizado para determinar o caminho até o servidor alvo, é importante saber por onde passam os pacotes até chegarem ao destino e, se neste caminho, existe algum *firewall* ou filtro de pacotes. Na verdade o *Traceroute* foi criado para auxiliar o administrador de rede a diagnosticar falhas na rede, entretanto devido a sua funcionalidade é muito usado por *hackers* para conhecer a topologia da rede e descobrir a existência de filtros ou *firewall* até o destino alvo. O *Traceroute* utiliza o *Time To Live* (*TTL*) de um pacote IP, esta função contém a quantidade de roteadores por onde passa o pacote até chegar ao destino. Quando o pacote passa por um roteador o *TTL* é decrementado em 1, se o *TTL* chegar a 0 o roteador descarta o pacote e envia uma mensagem à máquina que o originou, assim o que o *Traceroute* faz é enviar um pacote com *TTL* 1, depois com *TTL* 2, até chegar a máquina destino, faz isso por padrão em 30 *hops* (roteadores), e com as respostas obtidas traça a rota percorrida. A figura 1.3 ilustra a saída de um *Traceroute* executado no *Backtrack* 5. (LIEIRA, 2011; BACKTRACK, 2012)

Como pode ser observado na figura 1.3, esta consulta foi realizada para saber a rota até o servidor 173.194.74.83 (*Google*) e o retorno da consulta omitiu o IP do primeiro roteador, que pode estar configurado para não responder propositalmente, entretanto mostrou o endereço do segundo, um roteador da Telesp com IP 201.0.86.197 e assim sucessivamente até o décimo segundo que pode ser visto nesta página de consulta (existem outros, porém esta página só mostra os doze primeiros). É possível notar também que além do endereço IP, a ferramenta mostra o tempo de acesso em milissegundos de todos eles e o nome de alguns, com isso podemos observar que a consulta partiu de uma rede dsl (*speedy*) da telesp e que o oitavo roteador já pertence o alvo pesquisado, ou seja, o *Google*. Vale lembrar que o *traceroute*, assim como outras ferramentas aqui demonstradas são largamente utilizadas pelos administradores para monitorar e inventariar sua rede e seus ativos e não apenas por haker ou craker mal intencionado

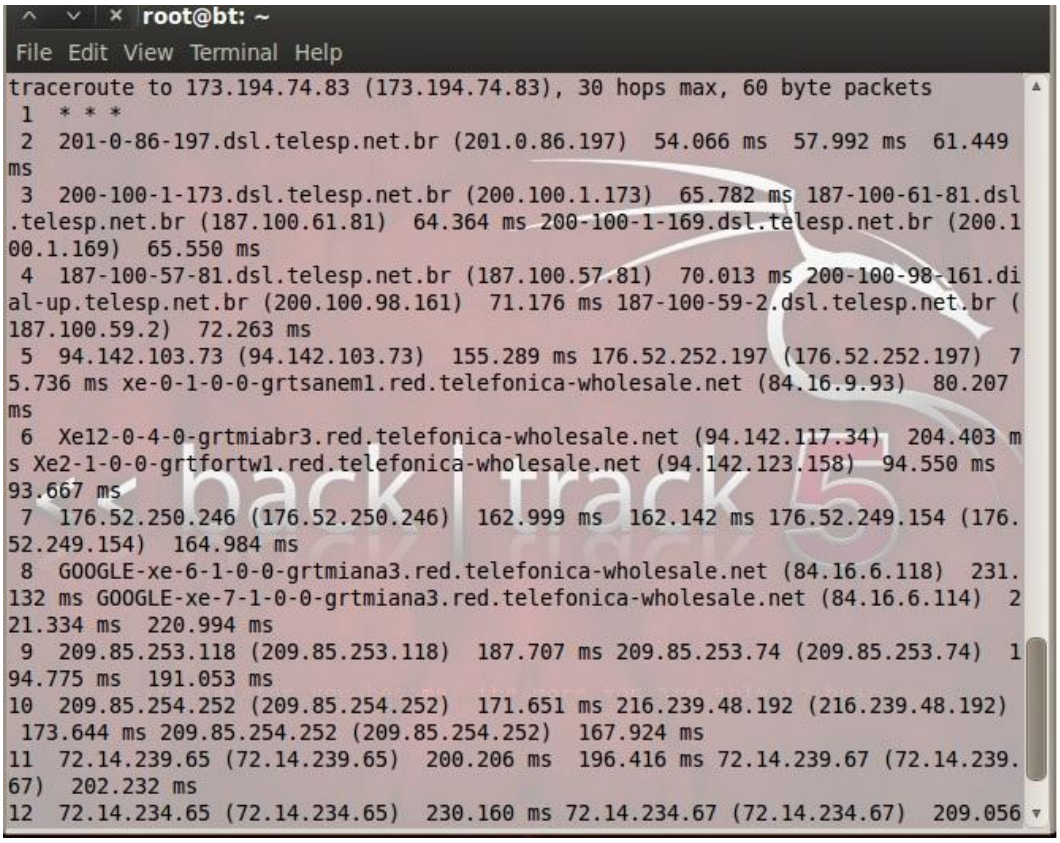


Figura 1.3 – Resultado da ferramenta `Traceroute` no *Backtrack* 5.
Fonte: Elaborado pelos autores, 2012.

Sabendo que neste caminho pode haver um *firewall* que bloqueia a porta *User Datagram Protocol* (*UDP*) que é o padrão do *Traceroute*, podemos usar algumas opções da ferramenta que permitem usar o protocolo ICMP, ou utilizar o *UDP* com uma porta diferente do padrão, a 53, por exemplo. (LIEIRA, 2011; BACKTRACK, 2012)

Como contramedidas para evitar o *Traceroute* existem sistemas que detectam a intrusão e avisam quando o *Traceroute* foi utilizado para traçar rota para as máquinas da rede, existem ainda outros como o caso do *tdetect* o *rotorouter* que detectam e fazem *log* do *Traceroute*. (LIEIRA, 2011. BACKTRACK, 2012)

### 1.1.4 Descobrindo Serviços em Execução

Finalmente, depois de colhidas as informações preliminares, faremos a enumeração de serviços e informações do servidor. Esta técnica consiste em realizar uma varredura nas portas *Transmission Control Protocol* (TCP) e UDP a fim de descobrir serviços ativos, serviços em estado de *listening*, serviços mal

configurados ou com versão de *software* que tenham alguma vulnerabilidade já conhecida e que possam ser exploradas para a invasão. (LIEIRA, 2011)

Para a varredura de portas existem diversas técnicas, como, por exemplo, o TCP *connect scan* que faz um *handshake* completo, ou seja, abre conexão com cada porta do servidor alvo, enviando um *synchronize* (SYN), o servidor caso esteja com o serviço TCP habilitado irá responder com um SYN/ *acknowledge* (ACK), e o invasor por sua vez confirma a conexão enviando um ACK. (LIEIRA, 2011)

Esta técnica funciona bem, mas pode ser facilmente detectada, pois na realidade fez uma conexão normal e gera *log* no servidor. Temos também a técnica TCP SYN SCAN que age basicamente como a técnica anterior, mas não faz o *handshake* completo, o atacante envia um SYN, o servidor responde com SYN/ACK e o atacante responde com *reset* (RST)/ACK, ou seja não gera conexão, sendo portanto mais difícil de ser detectada. Existe, ainda, varredura de portas com UDP SCAN que funciona de forma parecida com os de TCP, entretanto, UDP não é baseado em conexão e por isso funciona da seguinte forma: envia um pacote para as portas UDP e, se elas estiverem fechadas, o servidor retorna com a mensagem ICMP "*port unreachable*" e se não houver esta resposta, pressupõe-se que exista uma aplicação rodando nesta porta. (LIEIRA, 2011)

Para essa varredura a ferramenta mais usada é o *Nmap*, uma ferramenta de *portscan* de uso geral que ajuda a detectar portas abertas tanto na rede local como via *internet,* o que o *Nmap* faz é basicamente o que foi dito nas técnicas acima, porém com algumas particularidades, por exemplo no caso da consulta o *Nmap* envia pacotes SYN para todas as portas do alvo e espera para receber o retorno ACK de confirmação, mas não envia o segundo pacote ACK que abriria a conexão, isso dificulta a detecção de intrusão nos sistemas que só monitoram conexões estabelecidas. (ASSUNÇÃO, 2010)

Vale lembrar que a ferramenta *Nmap* também faz *scan* de UDP, pois apesar de serem raras existem algumas brechas de segurança em serviços *Linux* que podem ser exploradas através de porta UDP, além disso, a maioria dos *firewall* é configurada apenas para bloquear as portas TCP. (ASSUNÇÃO, 2010)

Já para a varredura de vulnerabilidades/falhas de segurança a ferramenta mais utilizada é o *nessus*, a qual faz a varredura do servidor verificando se existe alguma falha no servidor e, principalmente, se o servidor não foi atualizado e a falha for antiga, já documentada. Dessa maneira, a própria ferramenta sugere um *exploit*

(programa que se aproveita das vulnerabilidades de um serviço ou sistema operacional) que possa se aproveitar dessa vulnerabilidade.

Além disso, o *nessus* é uma das ferramentas muito utilizadas pelos invasores, pois em sua varredura envia pacotes para todas as portas TCP e UDP e após verificar se a mesma está aberta, envia um *script* para analisar a vulnerabilidade e então mostra o nível de alerta, uma descrição detalhada da vulnerabilidade e a "solução" para o problema, que na maioria dos casos é a atualização do sistema. (NESSUS, 2012)

## 1.2 TÉCNICAS DE PENETRAÇÃO

Os tipos ou técnicas de penetração de um servidor de rede são as mais diversas possíveis, uma das mais conhecidas é o chamado Cavalo-de-troia ou ainda *Trojan*. Na maioria das vezes esse tipo de ataque usando *Trojan* é direcionado a *desktops*, entretanto, não podemos descartar a possibilidade de serem utilizados contra servidores. Os *Trojan* são basicamente arquivos aparentemente sem perigo algum que contém em seu interior um código malicioso que irá agir após instalado na máquina e, por isso, podem ser adicionados a um pacote compactado de atualização de algum aplicativo, por exemplo. (ASSUNÇÃO, 2010)

Uma vez que o administrador baixa este arquivo para dentro de seu computador e faz a descompactação do mesmo, o arquivo malicioso é instalado de forma oculta e a partir daí faz o que foi estabelecido pelo atacante sem deixar rastros que possam ser facilmente detectados. (ASSUNÇÃO, 2010)

Outro tipo de ataque é o conhecido como força bruta ou *brutal force*. Todo sistema com acesso restrito geralmente utiliza um nome de usuário e senha para acesso, o que a técnica de força bruta faz é tentar de forma aleatória descobrir esta informação tentando logar diversas vezes com nomes e senhas diferentes. (ASSUNÇÃO, 2010)

Daí a necessidade de obter o máximo possível de informações sobre o alvo, e neste caso principalmente com o uso da engenharia social, pois sabendo algumas informações sobre o usuário é possível criar uma lista com os possíveis nomes de usuários e senhas ou utilizar as lista prontas da ferramenta chamadas de *wordlist*, que contém palavras comuns muito usadas como senhas e deixar o ataque rodando de forma automática até que se descubra o nome do usuário e a senha. (ASSUNÇÃO, 2010)

Esse tipo de ataque pode ser utilizado se o atacante descobre que o servidor tem rodando um serviço *Secure Shell* (*SSH*), por exemplo, pois uma vez que tenha acesso vai precisar somente ter permissão de supervisor e consolidar o ataque.

Este ataque é facilmente detectado se o administrador do sistema verificar o arquivo de *log* onde ficará gravado cada tentativa de acesso. (ASSUNÇÃO, 2010)

Para tentar impedir este tipo de ataque existem algumas medidas que podem ser tomadas e que dificultarão a possibilidade de sucesso do atacante. A primeira delas é garantir que todas as contas tenham senha, e que estas sejam trocadas periodicamente e que tenham um número mínimo de caracteres, além, é claro, de verificar sempre se algum aplicativo criou uma conta padrão, e configurar um sistema de alertas quando houver várias tentativas de senhas erradas. (ASSUNÇÃO, 2010)

Ainda sobre tipos de ataques destacamos o chamado *Buffer Overflow***,** também chamado de estouro de pilha ou estouro de *buffer*, que consiste em explorar falhas de programação, ele age da seguinte forma, suponha que um determinado programa em sua rotina criou um *buffer* com 10 caracteres e seja enviado para ele um texto contendo 25 caracteres, como ele tem falhas e não verifica o tamanho do espaço disponível, isso gera um estouro de *buffer*, e aí é que o atacante inclui um código personalizado para ser executado pelo programa, entretanto este tipo de ataque exige conhecimento profundo em programação para criar um *expoit* em linguagem de baixo nível, como o *assembler*, além disso, é preciso informações corretas do sistema operacional e do programa que será usado para causar o *overflow*. Apesar de ser um ataque difícil de se evitar, também o é para ser executado, mas quando se trata de segurança não pode deixar de ser considerado. (ASSUNÇAO, 2010)

Segundo Assunção (2010) outro ataque muito utilizado atualmente é o *Structured Query Language* (SQL) *Injection***,** que na verdade é uma falha de um programa que interage com o banco de dados SQL e não falha do banco como pensam algumas pessoas. O ataque consiste em explorar uma falha do programa que não interpreta corretamente alguns caracteres especiais como barra (/) e aspas simples (‘) permitindo desta forma que sejam incluídos comandos de banco de dados onde deveria entrar dados, simplesmente por que não são filtrados de forma correta, e com isso o atacante consegue muitas vezes acesso completo ao banco de dados.

O que acontece na prática é que se no campo `usuário` o invasor digitar, por exemplo, `pe'dro`, com aspas simples no meio do nome, o sistema entende a aspas como quebra da *string* e aceita o restante `dro` como comando, o que neste caso causaria um erro, porém se o atacante digitar `'; drop table users--` toda a tabela de usuários seria apagada, impedindo o acesso a qualquer usuário. Isso ocorre devido a falha no sistema que interpreta o que o atacante digitou como informações extraídas do banco de dados. (ASSUNÇAO, 2010)

Alguns sistemas limitam o número de caracteres na entrada de dados, visando justamente dificultar os ataques, mas mesmo assim ainda é possível causar danos. Neste caso a única maneira de evitar este tipo de ataque é testar seu programa e caso exista a falha atualizá-lo ou substituí-lo por um mais seguro. (ASSUNÇAO, 2010)

Como pudemos notar as técnicas de penetração são dos mais variados tipos e exploram vulnerabilidades ainda mais diversas, seria quase impossível esgotar o assunto neste trabalho, portanto, é de extrema importância que o administrador da rede mantenha-se sempre bem informado sobre novos tipos de ataque e de prevenções para os mesmos.

## 1.3 ELEVAÇÃO OU ESCALADA DE PRIVILÉGIO

De acordo com Vieira (2011) as técnicas de escalada ou de elevação de privilégios são muito utilizadas quando o invasor já possui acesso ao servidor ou qualquer outro alvo ao qual deseja invadir. Entretanto, estar dentro do sistema não é o suficiente para que se consiga executar todos os ataques desejados, já que algumas ferramentas e ações que o invasor necessita utilizar exigem privilégios de administrador como, por exemplo, a instalação de um *software* qualquer. A elevação ou escalada de privilégio nada mais é do que tornar o usuário comum, no caso o invasor, como um usuário *root* com todos os privilégios dentro do sistema, dando a ele direitos e permissões que antes não possuía. Entretanto esta elevação ou escalada de privilégio não se limita apenas aos sistemas operacionais, pois englobam vários outros sistemas como, por exemplo, conseguindo acesso ilimitado a um servidor de banco de dados dando privilégio de *DataBase Administrator (DBA*) que tornará o invasor capaz de executar várias operações dentro do banco como alterar, excluir e criar tabelas.

Ainda de acordo com Vieira (2011), existem vários servidores que não exigem que o usuário seja *root* para executar operações. Nesses servidores se tornam mais fáceis execuções da invasão, nos casos de servidores que exigem o privilégio *root* como administrador podem ser executados ataque de *Bruteforce,* também conhecido como força bruta, assim como o nome já disse o ataque tenta de forma bruta conseguir acesso testando uma sequência de possíveis usuários e senhas daquele servidor até que uma dê certo.

Segundo Vieira (2011) as aplicações mais exploradas são as que possuem mais falhas no *Kernel* e também aplicações que possuem *suidroot,* ou seja, aplicações que possuem uma permissão especial que é chamada de *suid bit* "s". Quando se consegue utilizar esta aplicação passa se a ter os mesmos privilégios que o *root* possui sobre o sistema. Sendo assim adquire-se controle total sobre o mesmo. Alguns comandos como estes citados abaixo podem ser utilizados para a execução dessas funções**:**

```
# find / -perm -4000 > suidroot.txt
# less suidroot.txt
# find / -perm -04000 -exec ls -l {} \;
```

Seguindo o mesmo raciocínio de Vieira (2011) os comandos citados acima podem ser utilizados para se localizar os arquivos *suidroot* das máquinas acessadas. Já se tratando de explorar elevação de privilégio por falha no *Kernel* se varia muito de acordo com o sistema operacional da máquina atacada.

Dessa maneira, se o servidor desta máquina possuir um *Kernel* cheio de falhas isso pode ser explorado dando ao invasor privilégio de administrador do sistema.

Estas falhas podem ser públicas ou privadas. As falhas públicas ocorrem quando o *software* tem seu código aberto, e suas informações e falhas estão distribuídas na *internet*, em fóruns em que administradores postam problemas detectados durante a sua rotina, ou até mesmo alguns *hackers* que quando conseguem detectar alguma vulnerabilidade a postam para se promover dentro de sua comunidade. Já as falhas privadas diferem-se das falhas públicas pelo fato de que o *software* privado ou comprado de desenvolvedores para alguma aplicação específica, não têm seu código disponibilizado como os *softwares* de código aberto,

para que sejam analisados, esses *softwares* devem ser comprados ou obtidos de forma ilícita, tornando um pouco mais complicado o trabalho dos *hackers*, pois as falhas estão ocultas, e para que suas vulnerabilidades sejam encontradas, o *hacker* deverá estudar todo o código, linha por linha para que possa alcançar sucesso em seu ataque. (Vieira, 2011)

## 1.4 TÉCNICAS PARA MANTER O ACESSO E APAGAR RASTROS

Segundo Ulbrich e Valle (2004) e SmiTh (2005), após o sistema ser invadido o atacante procura uma forma de voltar àquele sistema sem ser notado e sem precisar do mesmo esforço ou trabalho gasto para invadí-lo pela primeira vez. Uma das formas do *hacker* manter seus registros dentro do sistema é instalando novos serviços ou alterando algum serviço já existente para um tipo de serviço que ele desejar. Estas configurações realizadas normalmente são para se manter um acesso remoto ao sistema, e esse tipo de ataque é conhecido como *backdoor* ou porta dos fundos, pois deixa uma porta aberta para os invasores voltarem assim que quiserem. Uma *backdoor* pode ser implantada em vários sistemas operacionais, tais como os que utilizam a plataforma *Windows*, *Unix* e *Mac OS*.

Ulbrich e Valle (2004) ainda afirmam que mesmo após a invasão e instalação de ferramentas ou alteração de códigos que permitam acesso remoto ao sistema e o retorno do invasor, é necessário que estas alterações sejam camufladas para que não sejam detectadas pelo administrador da rede ou por possíveis ferramentas de auditorias no sistema. Uma forma eficaz de se fazer isto é com a utilização dos *rootkits*, que são programas que alteram o funcionamento de ferramentas, aplicativos e utilitários utilizados diariamente pelo sistema por quaisquer outras funções que o invasor solicitar que estes aplicativos, ferramentas ou utilitários façam.

> Os *rootkits* mais comuns fazem, basicamente, quatro coisas: a) abrir um *backdoor* permanente ou ativado por um código, b) mentir sobre o estado do sistema, c) apagar ou destruir alguma coisa, d) descobrir senhas e informações sigilosas. (ULBRICH e VALLE 2004, p. 321)

Seguindo o raciocínio de Ulbrich e Valle (2004) não é nada fácil identificar os *rootkits*, teoricamente o administrador da rede conseguiria identificar alguma atividade suspeita ou fora do normal com o comando *netstat* que é utilizado pra monitorar as conexões de entrada e saída e várias outras informações estatísticas

de utilização de uma rede. Entretanto se o comando estiver *rootkitted* ou melhor, dizendo com seu formato original alterado por algum *rootkit* suas funções normais não serão executadas corretamente e informações como portas abertas estariam ocultadas. Uma forma de se identificar se existem *rootkits* executando em sua rede é observar algumas características que podem mostrar que aquele programa foi alterado como data de criação diferente dos demais programas, o tamanho do arquivo modificado também pode ser diferente do programa original, pois o arquivo *rootkitted* possui algumas linhas mais de comando implementadas dentro dele. Outra forma também adotada pelos administradores de rede é a utilização de controladores de inventários com verificadores de integridade dos arquivos em todas as máquinas, detectando assim qualquer alteração de arquivos feita por algum *rootkit*. Estes controladores enviam mensagens de alerta para o administrador toda vez que algum arquivo existente é modificado.

Ulbrich e Valle (2004) alertam que apenas isso não é o suficiente, pois notando essas atividades dos administradores, os *hackers* desenvolveram *rootkits* com nível mais elevado de camuflagem, não atacando os arquivos e programas e sim trocando o próprio *kernel* do sistema operacional por outro totalmente adulterado, obtendo assim um controle total do sistema a fim de utilizar isso de forma maliciosa para diversos tipos de ataque tais como:

- Camuflar os arquivos do invasor que neste caso é quase impossível de ser detectado, pois os mesmos estão sendo executados pelo *kernel* alterado;
- Camuflar conexões de rede alterando a funcionalidade de comandos utilizados como o exemplo o comando *netstat* que citamos acima modificando os seus resultados e impedindo assim que o administrador note portas abertas e tráfego de dados realizado pelo invasor. Neste caso podem ser detectados com *sniffers* rodando em outras máquinas dentro da rede;
- Camuflar processos se torna mais fácil, pois o *kernel* alterado gerencia todos os processos ativos do computador, facilitando a ocultação dos programas infiltrados;
- Redirecionamento é onde o invasor implanta um programa nocivo semelhante a um programa existente no sistema e toda vez que o sistema operacional solicitar a execução do verdadeiro programa o

> *kernel* alterado vai redirecionar esse pedido ao programa modificado, permitindo assim que toda vez que a integridade do programa original for testada nenhuma alteração será encontrada.

Segundo Ulbrich e Valle (2004) a instalação de *rootkits* no *kernel* pode ser feita de duas maneiras, uma delas é mais utilizada pelos usuários do sistema *Windows* conhecida como *patch* que funcionam de forma semelhante aos *Hotfixes* e *Service Packs* nativos do próprio sistema operacional. Eles alteram completamente ou partes dos arquivos fazendo com que eles respondam de forma diferente do que deveriam responder. Para se instalar um *rootkit* no sistema operacional *Windows* é um pouco mais trabalhoso, mas não muito difícil já que o sistema possui várias bibliotecas de vínculo dinâmico, também conhecidas por *Dynamic-link library* (DLL) que são acessíveis aos usuários, não exigindo prioridade de super usuário para executar estas tarefas como a instalação de *rootkits* no *kernel*.

Os autores ainda afirmam que outra maneira mais preferida pelos usuários da plataforma *Unix* é instalação por módulos carregáveis no *Kernel* - *Loadable Kernel Modules* (*LKM*), que podem ser executados sobre demanda, automaticamente ou remotamente pelo invasor através de *backdoors* escondidos.

## 1.5 ATAQUES DE NEGAÇÃO DE SERVIÇO

Segundo Smith (2005) os *Denial of Service* (*DoS*) também conhecidos como ataques de negação de serviço são ataques que não são considerados como uma invasão de sistema pois apenas visam sobrecarregar os recursos de memória ou processamento de algum servidor escolhido como alvo. Normalmente estes alvos são máquinas de servidores de grandes empresas. Os *DoS* sobrecarregam a máquina enviando vários pacotes de requisições ao mesmo tempo para uma determinada máquina visando fazê-la travar ou até reiniciar por sobrecarga de tarefas dependendo do sistema operacional utilizado por ela, deixando assim os serviços disponibilizados por esta máquina indisponível por um bom tempo. Para exemplificar isso de forma mais clara podemos utilizar o exemplo das propagandas vistas na televisão onde um produto é oferecido a um preço tentador, logicamente que sabendo disso várias pessoas tentariam ligar ao mesmo tempo sobrecarregando a operadora de telefonia durante este tempo, impedindo que ligações àquele destino sejam efetivadas enquanto esta grande quantidade de

pessoas estiver tentando acessar este mesmo número ao mesmo tempo.

Seguindo ainda o raciocínio de Smith (2005) além dos *DoS* existem os *Distributed Denial of Service* (*DDoS*) que são a junção de negação de serviço com introdução distribuída, ou seja, os ataques de *DDos* podem ser considerados como vários ataques *DoS*. Portanto, são ataques de *DoS* em grande escala executados ao mesmo tempo em uma ou mais vítimas destes ataques. Os ataques de *DDoS* partem de inúmeras máquinas espalhadas pelo mundo controladas pelo invasor que por sua vez para conseguir executar esse ataque precisa invadir uma grande quantidade de máquinas interligadas pela *internet*, quanto mais recursos tiver e mais robusto for o servidor atacado maior será o número de máquinas normais atacando ao mesmo tempo para se realizar o *DDoS*. Para se obter acesso as máquinas que realizarão o ataque é necessário:

- Invadir inúmeras máquinas espalhadas pelos 4 cantos do mundo conectadas a *internet* sendo elas *desktops, laptops* entre outras;
- Realizar a elevação de privilégios dando assim ao invasor o privilégio de super usuário dessas máquinas;
- Uma vez com privilégio de super usuário executa-se a instalação do *software DDoS*. Seleciona-se uma pequena quantidade de máquinas que passam mais tempo conectadas por seus administradores a *internet* para serem *masters,* e as demais máquinas, em uma quantidade muito maior, que também estejam conectadas a *internet* com *links* razoavelmente rápidos para se tornarem agentes respondendo a comandos das máquinas *masters*, montando assim uma rede de máquinas dominadas prontas para realizar o ataque
- E por último é só realizar o ataque lançando o *flood* (inundação) de pacotes para uma ou mais máquinas escolhidas como vítimas. Este ataque é efetivado quando o *hacker* manda um comando para as máquinas *masters,* que por sua vez mandam um comando para as máquinas agentes dizendo que podem atacar. Estas por fim enviam um bombardeio de pacotes ao servidor alvo.

Segundo Smith (2005) é muito difícil detectar esse tipo de ataque por serem feitos por meio de criptografia, porém não é impossível. Pode-se identificá-los por meio de auditorias, ferramentas de detecção específicas e sistemas de detecção de introdução.

Ainda seguindo o raciocínio de Smith (2005) para se prevenir de ataques de *DDoS* recomenda-se aumentar a segurança no *host*, sempre manter os *patches* do sistema atualizados, instalar filtros *anti-spoofing* já que os *DDoS* utilizam de *spoofing* para tentar ocultar os endereços *IPs* de onde vem o ataque. Utilizar roteadores que permitam limitar a banda por tipo de tráfego como, por exemplo, os roteadores CISCO que possuem um recurso que permite limitar a banda que poderá ser consumida por esses tipos de pacotes, prevenir que sua rede seja utilizada para envio de pacotes a endereços de *broadcasting* implementando diretivas nos roteadores impedindo os recebimentos desses tipos de pacotes.

Segundo Smith (2005) como não existe sistemas conectados a *internet* que estejam totalmente seguros uma das formas de se manter prevenido é ter um bom plano de contingência, procedimentos de reação bem claros e específicos e também administradores de rede treinados e qualificados para agir o mais rápido possível quando ocorrer esse tipo de imprevisto também é necessário que estes administradores de rede também mantenham um monitoramento constante da rede e processos que estejam executando em sua rede.

# 2 METODOLOGIAS DE TESTE DE PENETRAÇÃO

Para a realização de testes de penetração o conhecimento dos tipos de ataque e das ferramentas utilizadas é de extrema importância, entretanto, é necessário muito mais que isso. Para que um teste seja confiável é preciso que seja feito seguindo uma metodologia que, segundo o Dicionário Silveira Bueno (2000, p. 510), significa: “tratado dos métodos” e método é “ordem que se segue na investigação da verdade”. Portanto entende-se que metodologia no conceito de teste de penetração seja um conjunto de procedimentos a serem seguidos durante os testes de vulnerabilidades e ataque, para que desta forma o mesmo teste pode ser repetido e até comparado. Caso seja realizado utilizando táticas diferentes e sem ser documentado será considerado como mera tentativa de invasão e perde seu valor quanto a confiabilidade dos resultados, descaracterizando o *Pentest* e configurando apenas uma tentativa de ataque.(BORGES, 2011)

Nesta linha de raciocínio e para que haja confiabilidade dos resultados foram criadas por organizações da área de segurança diversas metodologias. Neste capítulo trataremos das mais utilizadas, entretanto, segundo Borges (2011) 94% das empresas utilizam mais de uma metodologia quando necessário e a maioria delas usam também uma metodologia proprietária, o que deixa evidente a carência de metodologias que resolvam cem por cento dos problemas.

## 2.1 *NIST*

A metodologia *NIST* foi disponibilizada em setembro de 2008 pelo Departamento de Comércio dos Estados Unidos e é obrigatório em diversos departamentos americanos, porém pode ser usado livremente por empresas do mundo todo. Para obter o máximo proveito dos testes técnicos de segurança o *NIST* sugere que inicialmente se estabeleça uma política de segurança de avaliações, que se definam os objetivos de cada avaliação de segurança, tentando alcançá-los e que, logo após, sejam analisadas as descobertas para que se possa sanar as vulnerabilidades e minimizar os riscos. (NIST, 2008)

A proposta principal desta metodologia não é a de informar qual a técnica a ser utilizada nos testes, até porque existem diversas e seus objetivos são basicamente os mesmos. Então o que a metodologia propõe é orientar no

planejamento e aplicação dos testes de segurança da informação, análise das descobertas e desenvolvimento de estratégias de mitigação, ou seja, a idéia principal é especificar como utilizar as diversas técnicas para alcançar a efetividade dos testes. (NIST, 2008)

Assim como todas as metodologias estudadas, o documento completo é bem detalhado, no caso do *NIST* 80 páginas, entretanto neste trabalho relatamos apenas os tópicos básicos de cada uma delas e para um conhecimento mais profundo recomendamos a leitura do documento na íntegra. O *NIST* está dividido em basicamente sete seções e vale salientar que dentro do documento são encontradas referências a outras metodologias como o *OWASP* e *OSSTMM*, por exemplo. (NIST, 2008)

Seções:

- Teste de segurança e visão geral dos exames: concentrado em três métodos de avaliação de segurança da informação para comprovar a efetividade das ferramentas, são elas: testes, exames e entrevistas;
- Revisão das técnicas: discute as técnicas para levantar as vulnerabilidades de segurança com a realização de exames passivos nos sistemas, aplicações, procedimentos, redes e políticas;
- Identificação do alvo e técnicas de análise: foca em discutir as técnicas para identificação de dispositivos em atividade, suas portas e serviços associados, visando descobrir vulnerabilidades;
- Técnicas de validação das vulnerabilidades do alvo: com base nas informações obtidas na seção anterior, esta seção explora a vulnerabilidade provando sua existência e demonstrando os riscos existentes;
- Planejamento de avaliações de segurança: esta seção orienta na criação de políticas de testes, selecionando as melhores abordagens, gerenciando as considerações de logística e priorizando as avaliações;
- Execução de avaliações de segurança: esta seção além de chamar a atenção do auditor para pontos chaves durante a execução das avaliações, também discute a análise do processo e fornece recomendações para coletar, armazenar transmitir e apagar dados sobre a avaliação;

- Atividades pós teste: esta seção com base nos resultados dos testes, e analisando as descobertas, visa orientar as organizações a agir, tomando as providências necessárias para obterem um nível maior de segurança de suas informações.

## 2.2 *ISSAF*

O *framework* para Avaliação de Segurança de Sistemas de Informação *ISSAF* é uma metodologia criada em 2006 pelo *Open Information Systems Security Group* (*OISSG*), organização sem fins lucrativos que visa prover conhecimentos de segurança da informação e é formado por profissionais da área de diversas partes do mundo. A versão atual é a 0.2.1, sendo um *framework* de distribuição e utilização gratuita, inclusive para organizações comerciais desde que mantido o *copyright*. (OISSG, 2006)

Esta metodologia é uma das mais extensas, seu conteúdo completo tem mais de 1200 páginas e seus autores acreditam que é melhor fornecer toda a informação que o profissional possa precisar para os testes do que restringi-las a somente os objetivos principais. Cada etapa dos testes tem detalhadas instruções para execução e quais os resultados esperados. (OISSG, 2006)

A estratégia da *ISSAF* é dividida em três fases:

- Planejamento e Preparação: Antes da execução dos testes, na fase de planejamento é preciso garantir proteção legal para os profissionais que realizarão os procedimentos, bem como para a empresa que será “alvo” dos testes, para tanto deve ser firmado um “termo de acordo de avaliação”, nesta etapa são previstas as atividades de identificação dos contatos individuais das partes, metodologias e abordagem, escopo e acordos dos casos específicos de testes e caminhos de escalonamento. (OISSG, 2006)
- Avaliação: nesta etapa é que de fato se realizam os testes de penetração. Esta etapa é composta por 9 áreas, sendo: coleta de informações, mapeamento da rede, identificação de vulnerabilidades, penetração, obtenção de acesso e escalação de privilégios, enumeração, comprometimento de sítios e usuários remotos, manutenção de acesso e encobrimento de rastros. (OISSG, 2006)

- Relatórios e limpeza: nesta fase são disponibilizados os relatórios sobre os testes executados aos clientes, entretanto, se durante os testes alguma vulnerabilidade grave for encontrada, esta deve ser imediatamente relatada ao cliente. Ainda nesta fase devem ser excluídos todos os arquivos gerados no sistema alvo em consequência dos testes e caso não seja possível removê-los remotamente, devem constar nos relatórios para que possam ser removidos posteriormente pelo contratante. (OISSG, 2006)

Objetivos:

Avaliar as políticas e processos de segurança da empresa e verificar se não ferem padrões, regulamentos e leis. Identificar as debilidades da infra-estrutura dos sistemas de informação necessárias à execução com segurança dos processos de negócios da empresa. Realizar avaliações de vulnerabilidades e testes de penetração para identificar as falhas de sistemas, aplicações e redes e priorizar as avaliações segundo critérios de criticidade, custos e benefício potencial. (OISSG, 2006)

Organização:

Segundo (OISSG, 2006) a metodologia *ISSAF* esta dividida em 4 grupos de avaliação técnica de segurança e cada grupo subdividido em diversos tópicos mais específicos de sua área. Cada um destes tópicos é descrito com uma estrutura básica e como já descrito anteriormente cada um tem introdução, objetivos, resultados esperados, listas de testes e contra medidas, entretanto como se trata de uma metodologia muito extensa descrevemos abaixo apenas os 4 grupos gerais. São eles:

- Segurança de Rede: roteador, *firewall*, senha, *switches*, rede privativa virtual, antivírus, sistema de detecção de intrusão, armazenamento de dados, rede sem fio, usuário de *internet*, AS-400 e aplicativos *Lotus Notes*;
- Segurança de *Host*: segurança de sistemas *Unix*/*Linux*, *Windows*, *Novell Netware* e servidores de *internet*.
- Segurança de aplicação: abrange aplicações de *internet* (inclusive injeções de *SQL*), auditoria de código fonte, auditoria em arquivos binários e lista de checagem de avaliação de segurança da aplicação.

- Segurança de Banco de Dados: segurança de banco de dados e engenharia social.

É interessante ressaltar que apesar de extensa, é uma metodologia que sugere formas de evitar as vulnerabilidades que possam ser encontradas em função dos testes.

## 2.3 *OSSTMM*

A metodologia *OSSTMM* foi disponibilizada pelo *Institute for Security and Open Methodologies* (*ISECOM*) originalmente em dezembro de 2000, e com sua última versão a de numero 3.0 publicada em dezembro de 2010, tornou-se a metodologia mais atualizada e seu documento completo têm mais de 200 páginas. Projetada originalmente para ser uma metodologia consistente e que tenha resultados que possam ser reproduzidos, é baseada em conceitos de segurança humana, física, sem fio, de telecomunicações e de rede de dados. (ISECOM, 2010)

A realização de um teste utilizando essa metodologia garante que foram conduzidos completamente, que a postura dos testes aplicados estavam de acordo com a legislação, que incluíram todos os canais necessários, que os testes são consistentes e reproduzíveis, que os resultados dos testes são mensurados de forma quantificável e que estes resultados contém apenas fatos derivados dos próprios testes. (ISECOM, 2010)

Um dos pontos positivos do *OSSTMM* é que se for aplicado corretamente, independente do tamanho da organização, os resultados serão relativamente compatíveis, isso se deve as chamadas regras de engajamento, que são divididas em 9 seções, são elas: Vendas e *marketing* (5 regras); Avaliação, estimativa de entrega (2 regras); Contratos e negociações (9 regras); Escopo (2 regras); Plano de testes (1 regra); Processo de testes (11 regras); Relatórios (12 regras).

O *OSSTMM* também destaca que um computador pode até ficar mais lento de acordo com o volume de tarefas que tem para executar, entretanto não esquece nenhuma delas e nem deixa de realizá-las por seu bel prazer, diferentemente do ser humano que além de esquecê-las pode simplesmente deixar de realizar alguma delas simplesmente por não julgá-la necessária. Portanto para a realização de um teste é necessário antes de tudo um planejamento bem feito e, além disso, que seja realizado seguindo criteriosamente o que foi definido, uma vez que os testes são

complexos e, portanto devem ser tratados por partes, para que nada fique sem ser executado. Para que isso seja feito de forma correta o *OSSTMM* define os tipos de testes como: (ISECOM, 2010)

- *Blind*: é o tipo de teste onde o analista não tem conhecimento sobre os canais, defesa e ativos do alvo, entretanto o alvo tem conhecimento detalhado da auditoria que será realizada. Esta categoria além de tudo testa o conhecimento do analista de segurança;
- *Double Blind*: neste tipo de teste assim como no *Blind,* o analista não dispõe das informações sobre os ativos, canais e defesa do alvo, entretanto o alvo também não tem informações sobre o escopo do teste, portanto visa testar a capacidade do analista e também a efetividade dos controles do alvo;
- *Gray Box*: aqui o analista possui conhecimento limitado sobre os ativos e defesas e conhecimento total sobre os canais do alvo, por outro lado o alvo tem conhecimento detalhado da auditoria. Os testes dependerão das informações passadas ao analista e do conhecimento dele;
- *Double Gray Box*: neste caso o analista tem informações limitadas sobre defesas e ativos e conhecimento total sobre os canais do alvo. O alvo tem conhecimento da auditoria, seu escopo e tempo, entretanto não sabe quais canais serão testados. Os testes dependerão das informações disponibilizadas ao analista e ao alvo bem como do conhecimento do analista;
- *Tandem*: neste tipo de teste tanto o analista quanto o alvo estão preparados e sabem detalhes da auditoria. Este tipo de teste visa conhecer os controles do alvo e capacidade do analista, entretanto é ineficaz para o controle do alvo sobre variáveis desconhecidas;
- *Reversal*: neste caso o analista trabalha no alvo e tem conhecimento total de todos os seus processos e segurança operacional. Entretanto o alvo não tem conhecimento sobre quando e o que será testado.

Outro diferencial da metodologia *OSSTMM* refere-se ao *Risk Value Assessment* (*RVA*) ou Avaliação do Valor de Risco, um índice utilizado para mensurar a segurança. Cada seção recebe um valor, valor este que será aplicado a uma fórmula pré-definida para obter o *RVA*. Este *RVA* é utilizado para mostrar

estatisticamente os riscos ao qual o alvo está exposto e precisa de cuidados, também com base no *RVA* é que são definidas as estratégias a serem seguidas no pós teste, como por exemplo, quanto será necessário investir em segurança, quais as prioridades, quais soluções de segurança serão necessárias e o quanto de melhoria obteremos com estas ações.

## 2.4 *OWASP*

O *OWASP* ou Projeto Aberto para a Segurança de Aplicações *Web foi* fundado em abril de 2004 e segundo a definição do próprio documento original *OWASP* (2008, p.12) “é uma comunidade aberta dedicada a habilitar as organizações a desenvolver, adquirir e manter aplicações que sejam confiáveis”.

Na verdade trata-se de uma comunidade, onde todo o conteúdo está disponível gratuitamente a quem desejar utilizá-lo. Como não tem nenhum vínculo com empresas comerciais do ramo, as publicações têm um bom nível de imparcialidade, entretanto estimula o uso informado de tecnologias proprietárias. (SANTOS, 2010)

A comunidade *OWASP* é organizada em forma de capítulos locais, cada capítulo local é formado por pessoas da região geográfica que tem interesse em contribuir ativamente ao tema. O Brasil tem dois capítulos, um de Brasília e outro de São Paulo. Devido ao fato de ser uma comunidade o *OWASP* tem uma das maiores bases de informações sobre o tema segurança em aplicações, uma vez que possui 21.000 membros, 159 capítulos, 117 projetos e 6.381 artigos em seu sítio eletrônico. (OWASP, 2008)

A metodologia *OWASP* trata da segurança dos sistemas até mesmo antes do desenvolvimento propriamente dito e possui fase de teste para todo o processo desde projeto, criação, implantação e manutenção do sistema, porém neste trabalho o que interessa é a condução do teste de penetração em sistemas já em operação e portanto não é aprofundado nas demais áreas.

Na área de teste de penetração, a metodologia baseia-se na chamada “caixa preta”, onde o analista conhece muito pouco ou nada sobre o funcionamento interno do sistema a ser testado. Visando facilitar os testes, estes foram divididos em dez categorias, sendo que cada categoria possui um número de testes associados, são eles: (OWASP, 2008)

- Coleta de informações: primeira etapa a ser executada e visa obter o máximo de informações a respeito da aplicação a ser testada. Possui seis testes;
- Gerenciamento de configuração: foca em questões arquiteturais do *software* como transmissão e forma de armazenamento de dados, configuração e gerência de infra-estrutura. Possui oito testes;
- Lógica de negócio: Verifica se a aplicação se comporta corretamente quando o usuário executa operações fora do convencional. Possui um teste;
- Autenticação: Testa o correto funcionamento dos processos de configuração das identidades dos usuários. Possui dez testes;
- Autorização: verifica se o sistema dá privilégios a seus usuários corretamente e a possibilidade de aumentar de forma indevida tais permissões. Possui 3 testes;
- Gerenciamento de sessão: verifica se a aplicação *Web* controla e mantém o estado das interações corretamente com o usuário. Possui cinco testes;
- Validação de dados: verifica se os dados recebidos são validados de forma correta. Por ser essa uma das principais falhas de segurança, esta categoria possui dezesseis testes;
- Negação de serviço: testa a resistência da aplicação a ataques que objetivam deixá-la indisponível por algum tempo. Possui oito testes;
- *Web services*: verifica se existem vulnerabilidades nos serviços que compõem a aplicação. Possui sete testes;
- *Ajax*: verifica se a integridade proporcionada pela adição de *Ajax* na aplicação *Web* não esta acarretando novas vulnerabilidades. Possui dois testes.

Por ser uma comunidade de filosofia aberta no que se refere a colaboração, esta metodologia alcançou um grau de credibilidade muito alto e é atualmente uma das metodologias mais utilizadas, tanto na fase de desenvolvimento quanto nos testes de sistemas em uso.

# 3 APLICAÇÃO DA BACKTRACK EM TESTES DE PENETRAÇÃO DE SERVIDORES DE REDE

Este capítulo mostra a aplicação prática do teste de penetração, utilizando a B*acktrack 5 R3* e suas aplicações, em servidores especialmente criados para este trabalho, entretanto, antes dos testes deve-se deixar claro alguns pontos considerados extremamente importantes para realização de teste de penetração em servidores.

É importante esclarecer que este é um trabalho acadêmico para conclusão de curso e, que seus autores condenam todo e qualquer tipo de teste em servidores sem autorização por escrito do responsável pelo sistema alvo; segundo, os testes aqui realizados foram feitos em servidores especialmente montados e configurados para esta função (teste de penetração).

Não é recomendado que os testes e aplicação de *exploits* seja feito diretamente em um servidor em uso (ativo), uma vez que a aplicação dos testes vai consumir banda da conexão e, além disso, quando da varredura de portas o *software* utilizado fará com que o servidor responda a diversas requisições feitas pelo sistema que realiza a referida varredura, o que com certeza causará queda significativa no desempenho do servidor, podendo também, dependendo do *exploit*, causar a paralisação (*crash*) de serviços ou até mesmo do próprio servidor.

Assim sendo recomenda-se que os testes sejam aplicados em uma cópia do servidor e de preferência utilizando máquinas virtuais, para que, assim, evite tráfego desnecessário na rede do servidor, a menos que se conheçam todos os riscos e sejam assumidas todas as responsabilidades pelos danos que possam ser causados em virtude dos testes.

## 3.1 ETAPAS DO TESTE

Como já tratado anteriormente, todo teste deve seguir uma metodologia para garantir a confiabilidade dos resultados. Entretanto, também é sabido que a grande maioria das empresas utiliza um método próprio (capítulo 2), portanto neste teste não será diferente. Sendo que o grande objetivo é exatamente testar uma distribuição nova, o *Backtrack* 5, e descobrir se o administrador de rede conhecendo as ferramentas do sistema pode melhorar a segurança de seus servidores e, com

isso evitar uma possível intrusão, foram analisadas as tradicionais metodologias já citadas no capítulo 2 e ousamos não criar uma nova metodologia, mas sim simplificá-la para a aplicação dos testes realizados neste trabalho.

### 3.1.1 Coleta de Informações

A primeira etapa de qualquer teste de penetração é a coleta de informações sobre o alvo. Nesta fase deve ser coletado o máximo de informações sobre a empresa, ramo de atividade, sócios, servidores, serviços disponíveis, faixa de *IP* utilizada, etc. Para esta finalidade existem diversas ferramentas já citadas no capítulo1, e tudo depende do tipo de testes que será realizado, que pode ser com o conhecimento e colaboração do administrador da rede (o que deixa tudo mais fácil) ou sem o seu conhecimento, o que dificulta a coleta de informações, entretanto, especificamente neste trabalho alguns passos desta fase foram suprimidos, visto que estamos considerando que quem está aplicando os testes é o próprio administrador do servidor e que já se conhece suas características, *IP*, serviços etc.

Assim, iniciamos esta etapa no teste de varredura do servidor, pois mesmo o administrador conhecendo os serviços nos seus servidores, este teste se faz necessário para certificar que nenhum outro serviço estranho esteja executando nos servidores, o que poderia ser um indício de que o servidor já estivesse sido comprometido. Neste teste a ferramenta mais utilizada é o já conhecido *Nmap*, uma ferramenta que faz a varredura do servidor e retorna com o “mapeamento” completo do mesmo, mostrando qual sistema operacional está em uso, sua versão e quais os serviços estão disponíveis na máquina alvo, bem como quais portas estão sendo usadas por estes serviços e os protocolos por eles utilizados.

No *Backtrack* 5 o *Nmap* pode ser utilizado de diversas formas, modo gráfico simples, no modo gráfico com *Zenmap*, que já vem com parâmetros pré definidos (para usuários menos experientes), como linha de comando simples ou ainda com auxílio de *scripts* como é o caso do exemplo a seguir que utiliza o *script* `smb-check-vulns` para verificar as vulnerabilidades já conhecidas. Sua sintaxe é: `# nmap -v --script=smb-check-vulns 10.0.2.16`. A figura 3.1 mostra o resultado da pesquisa.


Figura 3.1 – Resultado da ferramenta *Nmap* no *Backtrack* 5.
Fonte: Elaborado pelos autores, 2012.

### 3.1.2 Analisando as Possíveis Vulnerabilidades

O segundo passo é analisar as informações que foram dadas pelo *Nmap*. Como podemos notar no teste realizado foi detectada uma vulnerabilidade, a já conhecida MS08-067 que aparece no resultado (figura 3.1) como *VULNERABLE*. O próximo passo será explorar esta falha, entretanto outras vulnerabilidades podem aparecer, dependendo do tipo de serviço oferecido pelo servidor, por exemplo, se existir um serviço *Web* e nele houver um formulário, pode-se acessar a página e tentar um *SQLinjection*, como visto no capítulo 1.

O importante nesta fase é anotar as portas e serviços disponíveis, bem como suas versões, pois as maiores vulnerabilidades estão exatamente em versões desatualizadas dos serviços, além disso, as vulnerabilidades irão variar de acordo com a quantidade e tipo de serviços que estiverem ativos no servidor pesquisado.

Para descobrir quais as vulnerabilidades destes serviços e sistemas uma boa pesquisa em *sites* especializados e em fórum sobre o assunto pode ser uma boa alternativa, entretanto cuidado para não disponibilizar dados demais sobre o seu servidor em fórum, pois é exatamente aí que invasores buscam por informações sobre seus próximos alvos.

Para pesquisas sobre as possíveis vulnerabilidades de um sistema ou

serviço, existem diversas fontes de consulta na *internet*, porém deve se ter a maior cautela nessa busca, pois como já dito anteriormente os invasores estão em toda parte e na ajuda “gratuita” pode estar o perigo.

Existem alguns sites que são atualmente referências seguras sobre vulnerabilidades conhecidas tais como o `http//cve.mitre` que é um *site* internacional livre e de uso público que as nomeia e classifica vulnerabilidades, sendo um verdadeiro dicionário sobre as vulnerabilidades; O `http://www.first.org/cvss`, que oferece o Sistema Comum de Pontuação de Vulnerabilidade ou *Common Vulnerability Scoring System* (*CVSS-SIG*), um sistema que analisa e pontua utilizando um método universal e padronizado para avaliação de vulnerabilidades; Existe ainda o `http://www.sans.org/que` que além de boletins de segurança, oferece treinamentos *on-line* sobre o assunto; e por fim o `http://www.securityfocus.com` que é uma lista de discussão sobre o tema e pode oferecer informações valiosas sobre o assunto.

### 3.1.3 Confirmando a Vulnerabilidade

As vulnerabilidades apresentadas, nem sempre são de fato verdadeiras, portanto é necessário confirmá-las. A maneira mais correta de fazer isso é tentar explorá-la. Para isso deve se encontrar um *exploit* que explore a falha, isso da mesma forma deve ser pesquisado em *sites* sobre o assunto, uma boa referência pode ser o http://*metasploit.com*.

Encontrado um *exploit* que explore a falha, o próximo passo será a sua aplicação. Este é um processo bastante trabalhoso, pois dependendo do *exploit* pode ser necessário, por exemplo, compilá-lo em um sistema *Unix*, o que requer um conhecimento sobre o assunto. Entretanto, existem ferramentas que fazem este trabalho de forma automática e que serão utilizadas neste trabalho.

### 3.1.4 Explorando a Vulnerabilidade

Nesta etapa, após conhecer as vulnerabilidades existentes e qual *exploit* pode ser aplicado para tentar explorá-la e obter acesso ao servidor é hora de aplicá-la.

Para a aplicação do *exploit* siga os seguintes passos:

Abra um terminal no *backtrack* e digite o seguinte comando:

`msfconsole`, na tela que se abre digite:

```
> use exploit/windows/smb/ms08_067_netapi
```

Para usar o *exploit* ms08_067_*netapi*;

```
> set RHOST 10.0.2.16
```

*IP* do servidor *Windows*;

```
> set PAYLOAD windows/meterpreter/reverse_tcp
```

Especificando que será utilizado o *payload reverse_tcp*, ele é o responsável por criar a comunicação entre *LHOST* e *RHOST*);

```
> set LHOST 10.0.2.15
```

Informando o host local, que no caso é o endereço *IP* da *VM* com *Backtrack*;

```
> exploit
```

Executando o *exploit*.

Como pode ser verificado pela figura 3.2 foi feita a invasão e existe uma conexão entre o atacante (*Backtrack*) e o servidor (*Windows*).


Figura 3.2 – Resultado do *Msfconsole* no *Backtrack* 5.
Fonte: Elaborado pelos autores, 2012.

### 3.1.5 Consolidando a Invasão

Este é o passo final de uma invasão, onde o invasor instala um *rootkit* (item 1.4) que além de ocultar sua invasão e domínio da máquina também abre uma

*backdoor* para que o atacante possa voltar a invadir quando quiser, porém, como o foco deste trabalho não é ocultar a invasão e sim testar a segurança dos servidores para melhorar a segurança e dificultar a intrusão, esta etapa foi suprimida. Entretanto, vale salientar e alertar que, caso um *rootkit* tenha instalado uma *backdoor* , esta ficará invisível para as ferramentas administrativas do sistema local, podendo ser possivelmente detectada por meio de uma varredura de portas feita de uma outra máquina, como mostrado na seção 3.1.1.

### 3.1.6 Finalizando o teste

Como este é um teste para melhorar o nível de segurança em servidores e não um ataque, o passo final é documentar todos os testes realizados e seus resultados, mesmo que seja um teste feito pelo administrador em seus próprios servidores.

Caso alguma vulnerabilidade tenha sido detectada, a solução para corrigi-la deve constar na documentação e para isso o administrador deve verificar com os fornecedores do *software* a existência de correções oficiais, ou seja, pacote de atualização que corrija esta falha. Caso a mesma já exista, basta atualizar o *software* e tudo estará resolvido, porém se ainda não houver correção e for impossível impedir o ataque de outra forma, há que se considerar suspensão do serviço até que se resolva o problema.

## 3.2 ENTENDENDO O *ARMITAGE*

O *armitage* é uma *interface* gráfica disponível no *Backtrack* que facilita o uso da *framework metasploit*, que por sua vez é uma ferramenta automatizada que varre o servidor alvo em busca de vulnerabilidades. Esta *framework* possui um banco de dados com as vulnerabilidades já conhecidas e outro banco com os *exploits* criados para aproveitar tais vulnerabilidades e instalar o *payload*. Este *payload* é um algoritmo “malicioso” desenvolvido por *experts* com profundo conhecimento de programação. Estes *experts* estudam as vulnerabilidades existentes em cada serviço e ou sistemas operacionais e desenvolvem o *exploit* que de fato explora a falha e estabelece a conexão entre atacante e alvo, daí a importância de manter o *Backtrack* e, consequentemente, o *metasploit* constantemente atualizados, baixando

para seu banco de dados todas as novas descobertas sobre vulnerabilidades, bem como os *exploits* utilizados para explorá-las. (BACKTRACK, 2012)

## 3.3 CENÁRIO DE TESTES

Para a aplicação dos testes foi criado um cenário com três máquinas virtuais, utilizando o *Virtual Box* da *Oracle,* como se estivessem em uma rede exclusiva como mostra figura 3.3.

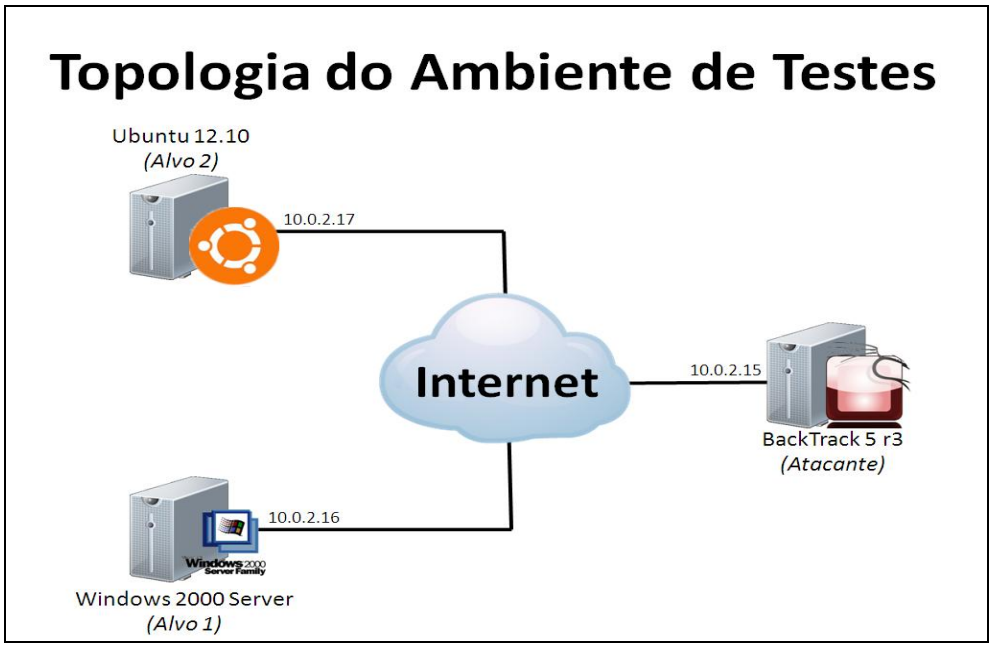


Figura 3.3 – Topologia da rede virtual de testes de penetração.
Fonte: Elaborado pelos autores, 2012.

Computadores utilizados:

**Máquina atacante**: Distribuição linux *Backtrack* 5 r3, criada em uma máquina virtual utilizando o *Oracle Virtual Box*, com 1024MB de memória, disco rígido de 20GB dinamicamente alocado, previamente atualizada, *interface* de rede configurada como rede interna e *IP* estático 10.0.2.15.

**Máquina alvo 1**: Sistema operacional *Microsoft Windows* 2000 *server*, criada em uma máquina virtual utilizando o *Oracle Virtual Box*, com 1024MB de memória, disco rígido de 20GB dinamicamente alocado, sem instalar pacotes de atualização, *interface* de rede configurada como rede interna e *IP* estático 10.0.2.16.

**Máquina alvo 2**: Sistema *Linux Ubuntu* 12.10, criada em uma máquina virtual utilizando o *Oracle Virtual Box*, com 1024MB de memória, disco rígido de 20GB

dinamicamente alocado, previamente atualizada e configurada como servidor *Web*, *interface* de rede configurada como rede interna e *IP* estático 10.0.2.17.

## 3.4 PRIMEIRO TESTE UTILIZANDO O *ARMITAGE*

Após a demonstração e entendimento dos passos a serem seguidos, passamos a aplicação prática do teste, que visando minimizar o tempo gasto pelo administrador foi feita utilizando uma ferramenta nativa do *Bactrack* que como já explicado, faz todo o processo de forma automática, ou seja, realiza o *Nmap*, analisa as vulnerabilidades, verifica no banco de dados quais *exploits* estão disponíveis para estas vulnerabilidades e faz sua aplicação, esta ferramenta é denominada *Armitage*.

Vale lembrar que um teste realizado de forma manual, passo-a-passo, pode obter um resultado mais preciso, entretanto é mais demorada e exige um profundo conhecimento das ferramentas utilizadas e sua sintaxe, podendo ser aplicado de forma diferente em cada caso desde que siga uma metodologia e seja documentado.

Os testes a seguir foram feitos individualmente em cada um dos servidores, inicialmente no servidor *Windows* e em seguida no servidor *Linux.* Cada um deles foi repetido duas vezes, visando retificar ou ratificar os resultados obtidos, como segue:

1. Inicie normalmente o *BackTrack*.


Figura 3.4 - Tela inicial do *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

2. Abra um novo *Shell*, e digite '*armitage*'.

Este comando vai iniciar o *Armitage*, abrindo a sua janela de conexão.

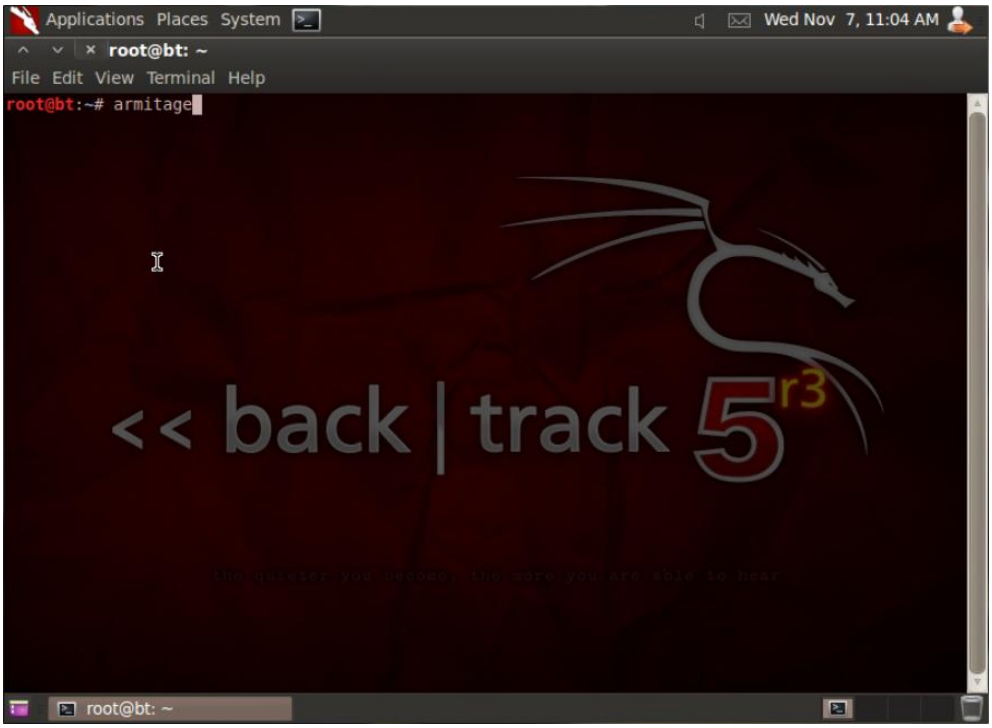


Figura 3.5 - Iniciando *Armitage* do *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

3. Nesta janela, preserve os dados iniciais, e clique em '*Connect*'.

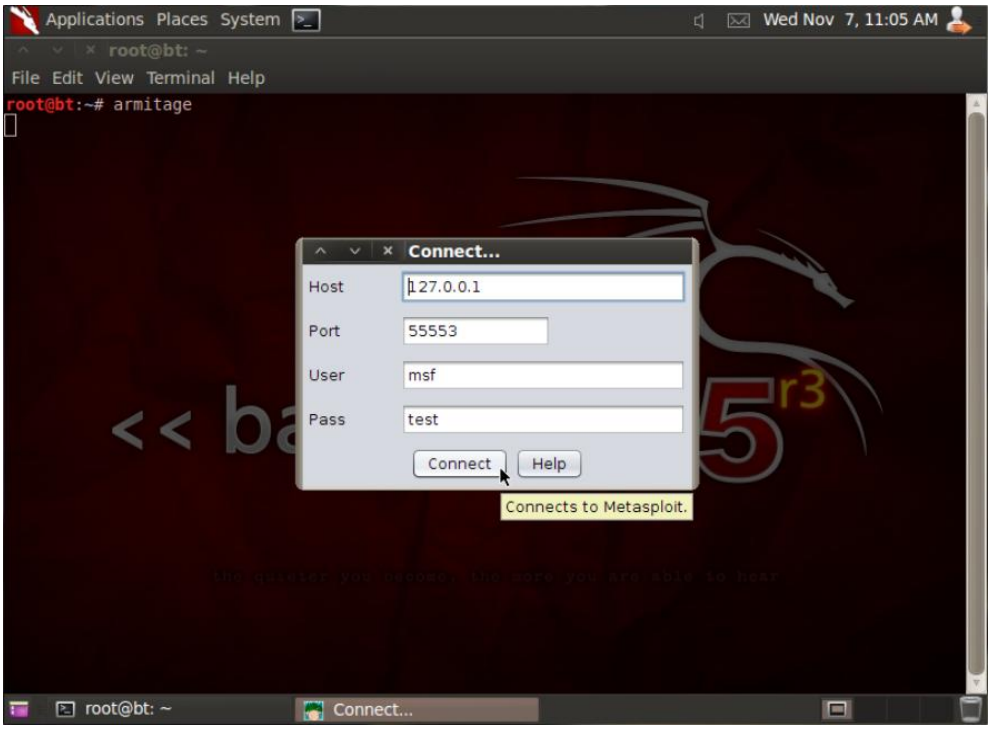


Figura 3.6 - Tela inicial do *Armitage* no *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

4. O *Armitage* pergunta se você deseja iniciar o servidor *Metasploit*. Basta confirmar a ação, clicando em '*Yes'*.


Figura 3.7 - iniciando o *Metasploit* no *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

5. Aguarde o *Armitage* carregar.

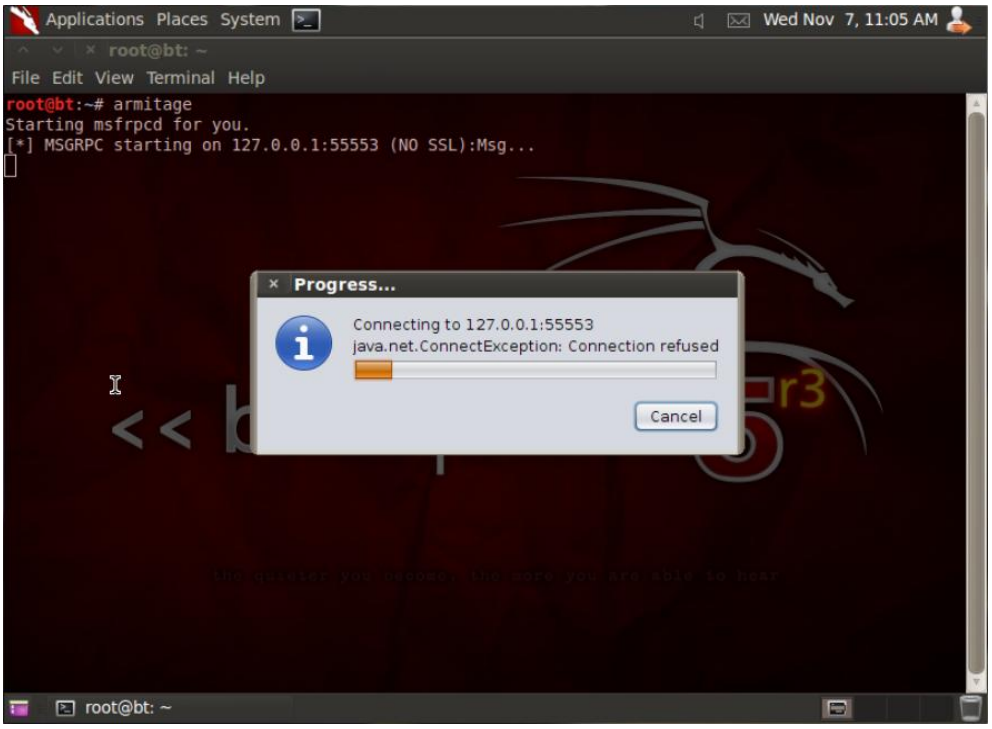


Figura 3.8 - Conectando ao banco do *metasploit* no *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

6. Após o carregamento, inicia-se a tela principal do *Armitage*.

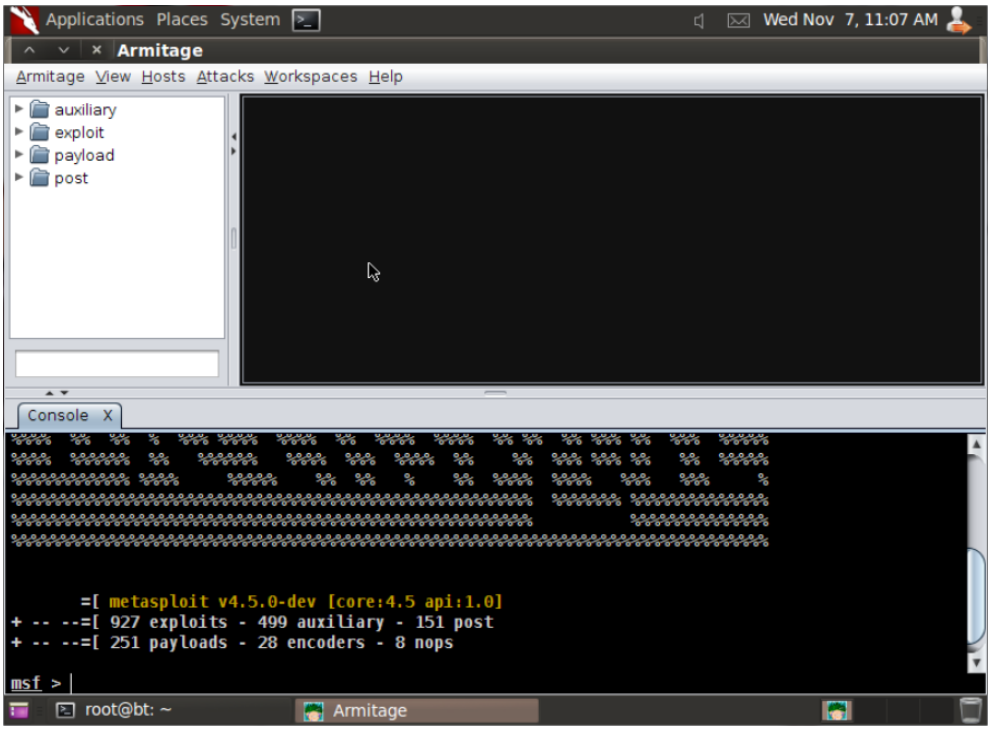


Figura 3.9 - Tela inicial do *Armitage* no *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

7. Aplicando o *'Intense scan'* no alvo.

Acesse o menu '*Hosts > Nmap Scan > Intense Scan'*.

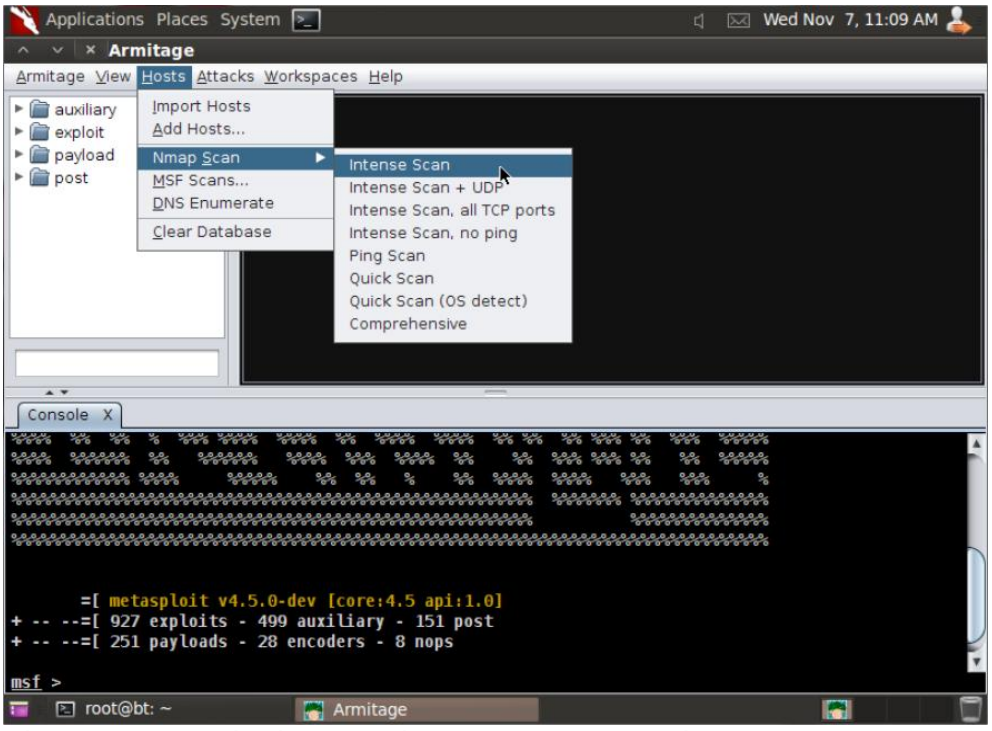


Figura 3.10 - Tela do *Intensive Scan* no Bactrack 5 r3.
Fonte: Elaborado pelos autores, 2012.

8. Nesta janela, digite o *IP* do *Windows* (*10.0.2.16*) e clique em '*OK*'.

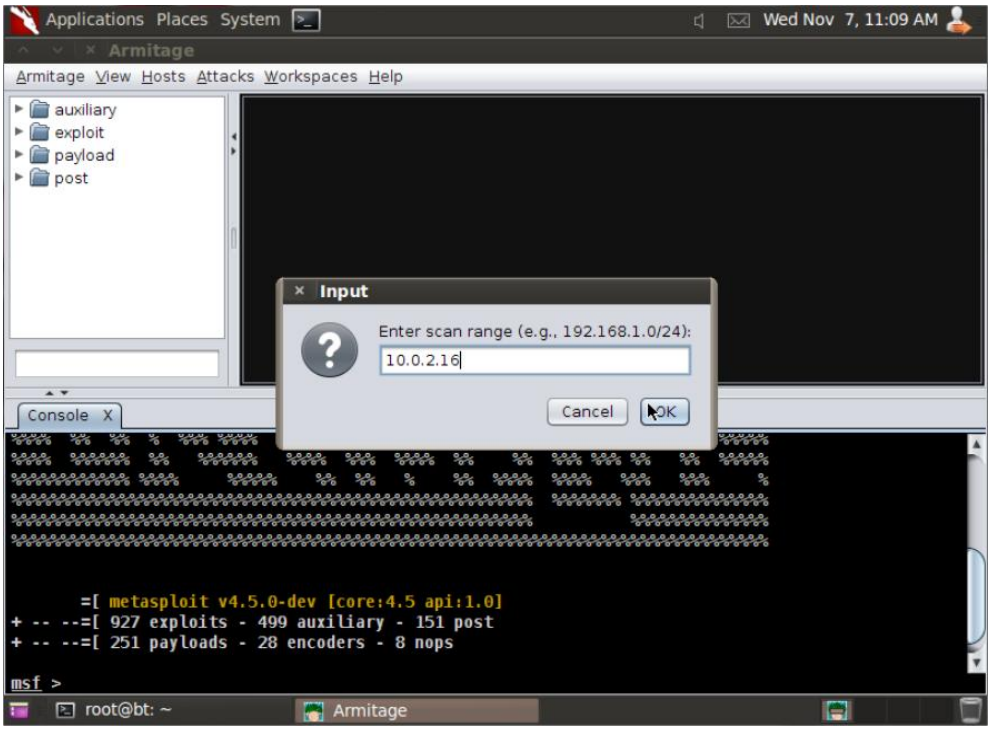


Figura 3.11 - Campo de *IP* do '*Intensive Scan'* no *Bactrack* 5 r3.
Fonte: Elaborado pelos autores, 2012.

9. Aguardar até o '*Scan Complete!*', e clique em '*OK*'.

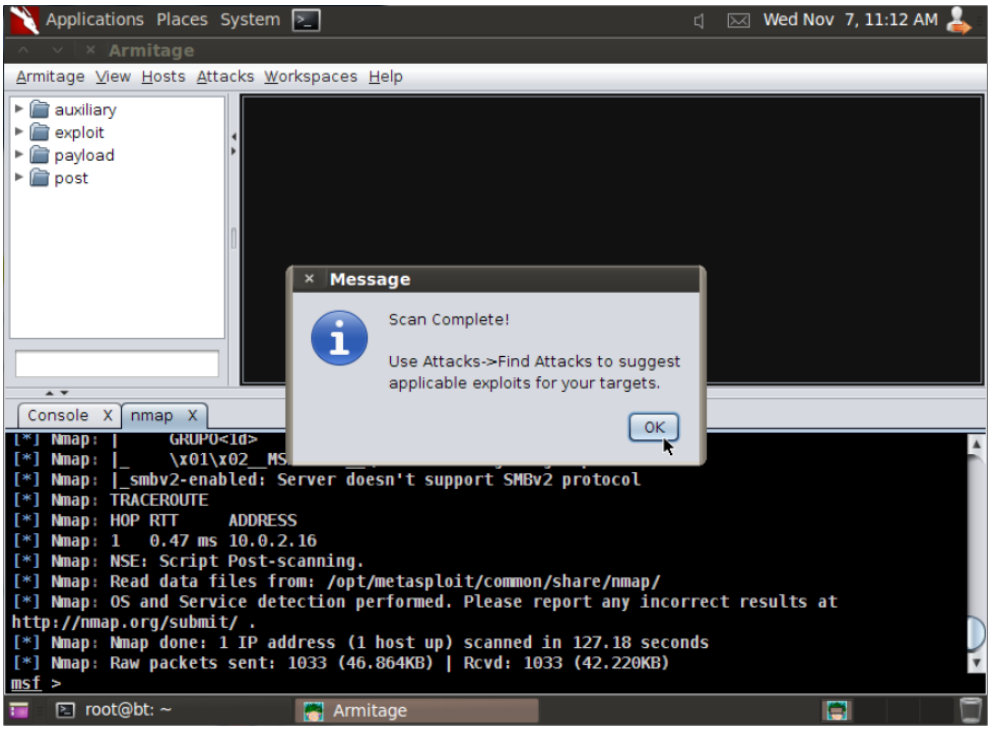


Figura 3.12 - Tela de final de *scan*.
Fonte: Elaborado pelos autores, 2012.

10. Observe que apareceu o *PC* do *Windows* como um *Host*.

Para explorar suas vulnerabilidades, acesse o menu '*Attacks* > *Hail Mary*'.

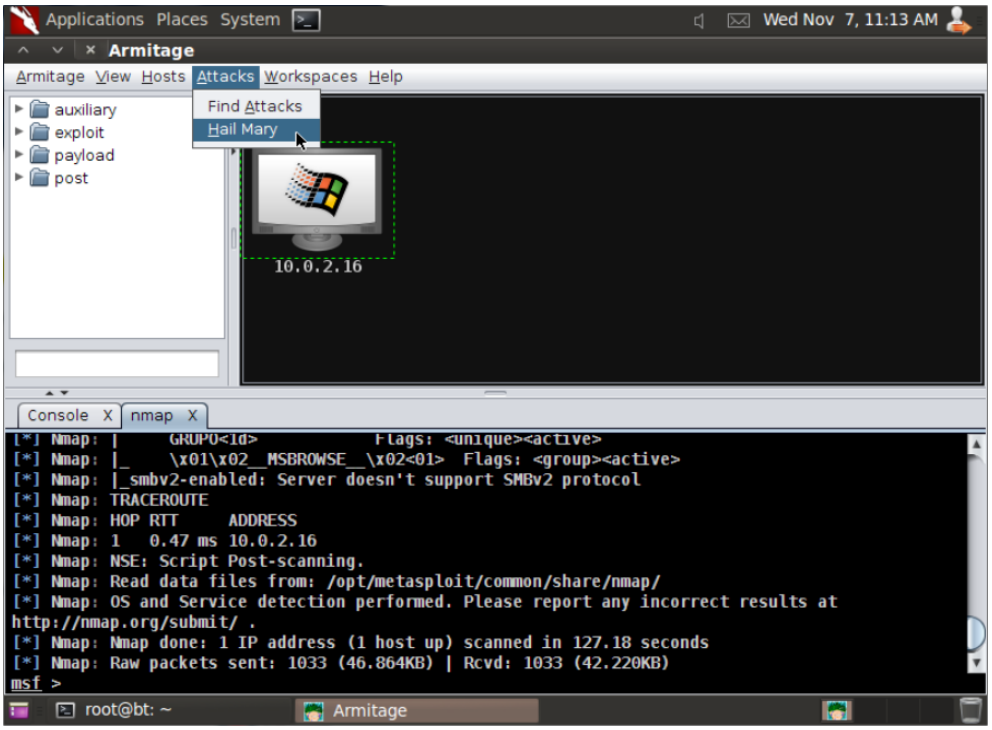


Figura 3.13 - Iniciando aplicação de *exploits*.
Fonte: Elaborado pelos autores, 2012.

11. Aguarde enquanto o *Armitage* encontra um *Exploit* para aplicar no *PC Windows.*

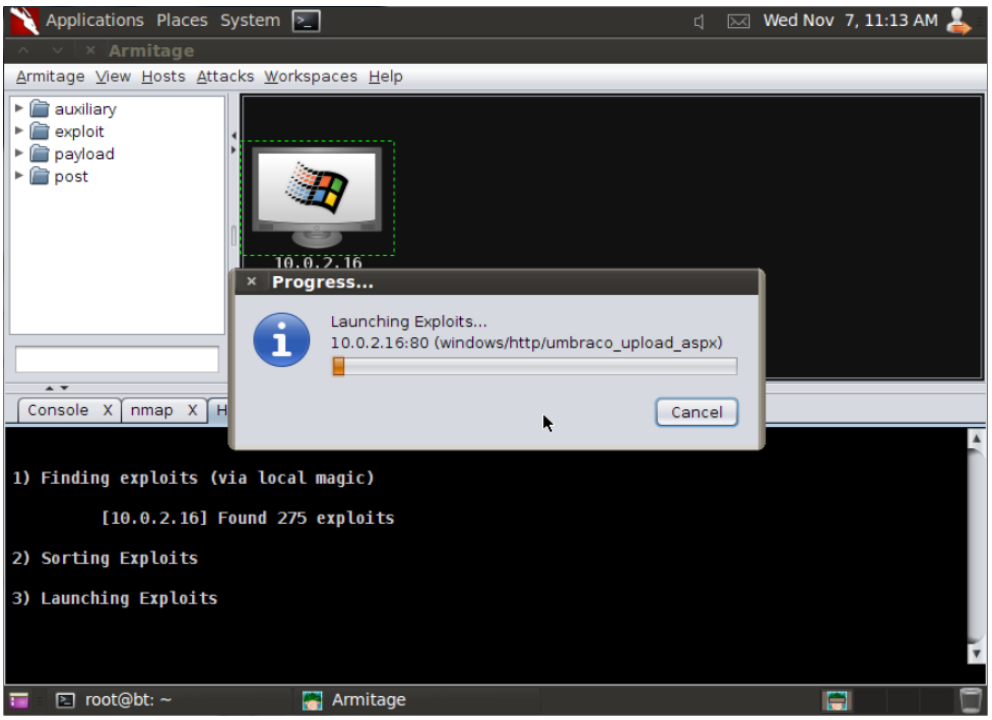


Figura 3.14 - Aplicando *Exploits*.
Fonte: Elaborado pelos autores, 2012.

12. Ao terminar, observe, o ícone ficou vermelho com alguns "raios" sobre ele. Isso significa que o *Armitage* já aplicou um *exploit* e invadiu o servidor.

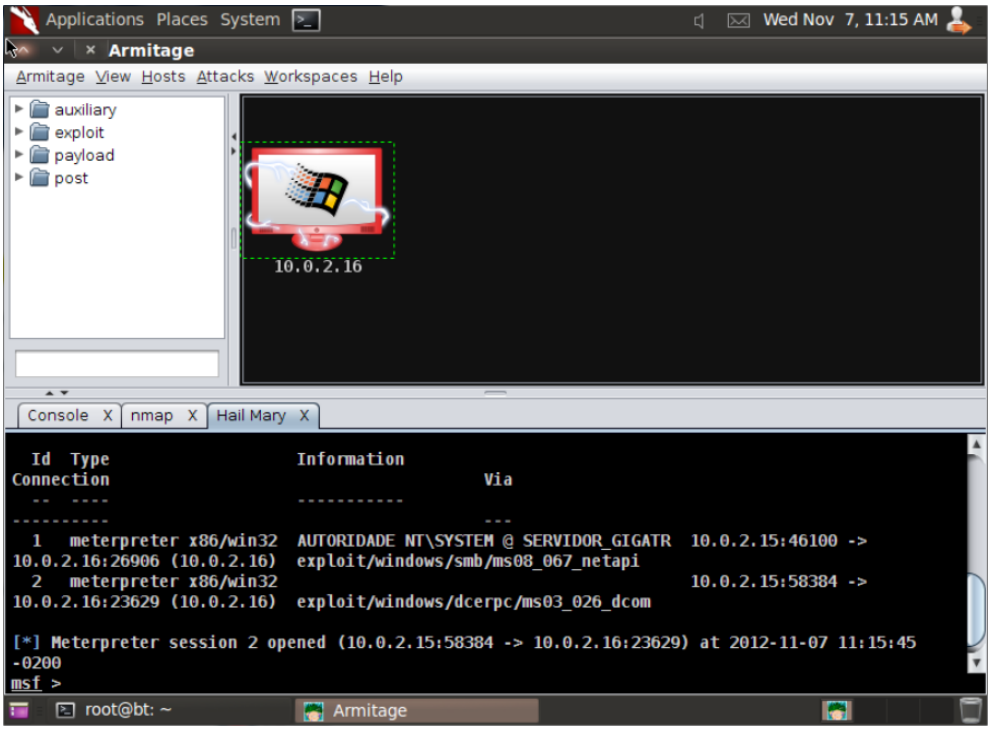


Figura 3.15 - Tela final de aplicação de *exploits*.
Fonte: Elaborado pelos autores, 2012.

13. Para comprovar a invasão, clique com o botão direito sobre o ícone do *Windows*, e vá até '*Meterprete 1 > Explore > Browse Files*'.

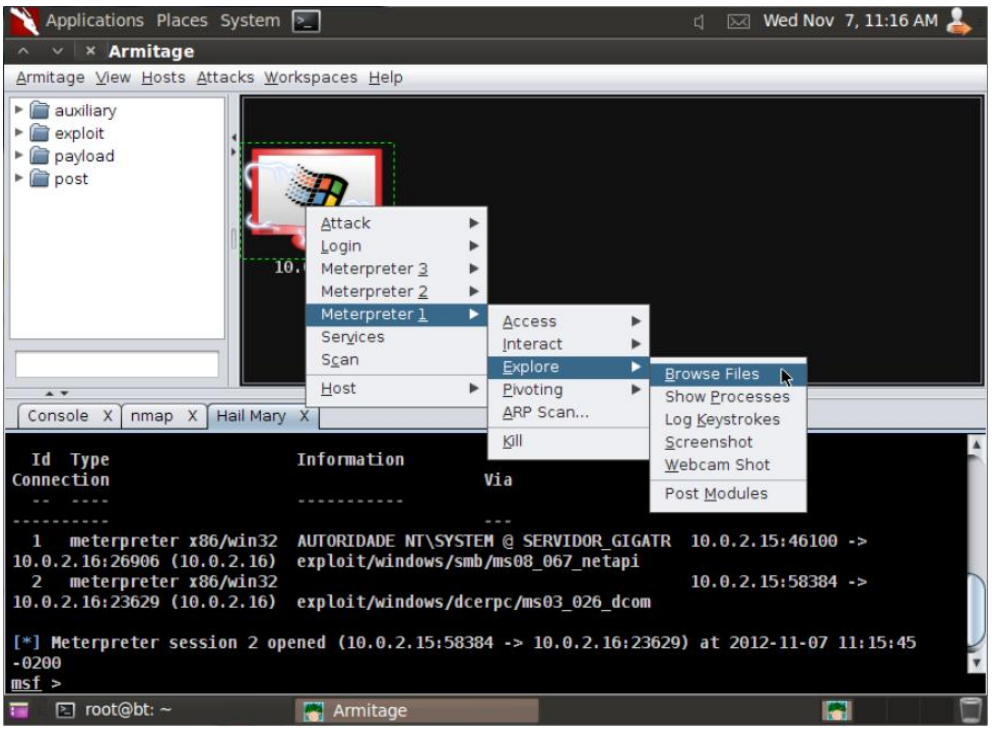


Figura 3.16 - Acessando o servidor invadido.
Fonte: Elaborado pelos autores, 2012.

14. Observe à árvore de arquivos do *PC Windows*.

Utilizando desta vulnerabilidade, pode-se “dominar” o servidor *Windows*.

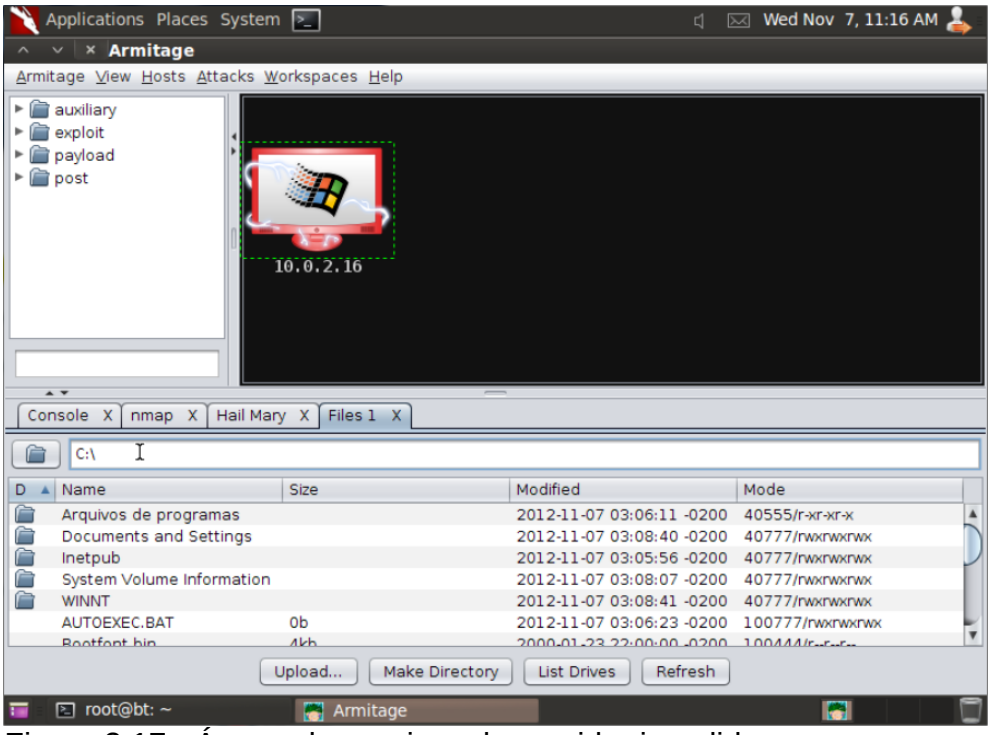


Figura 3.17 - Árvore de arquivos do servidor invadido.
Fonte: Elaborado pelos autores, 2012.

15. Enviando um arquivo ao servidor, observe a pasta '*C:/*' do alvo, antes do envio.

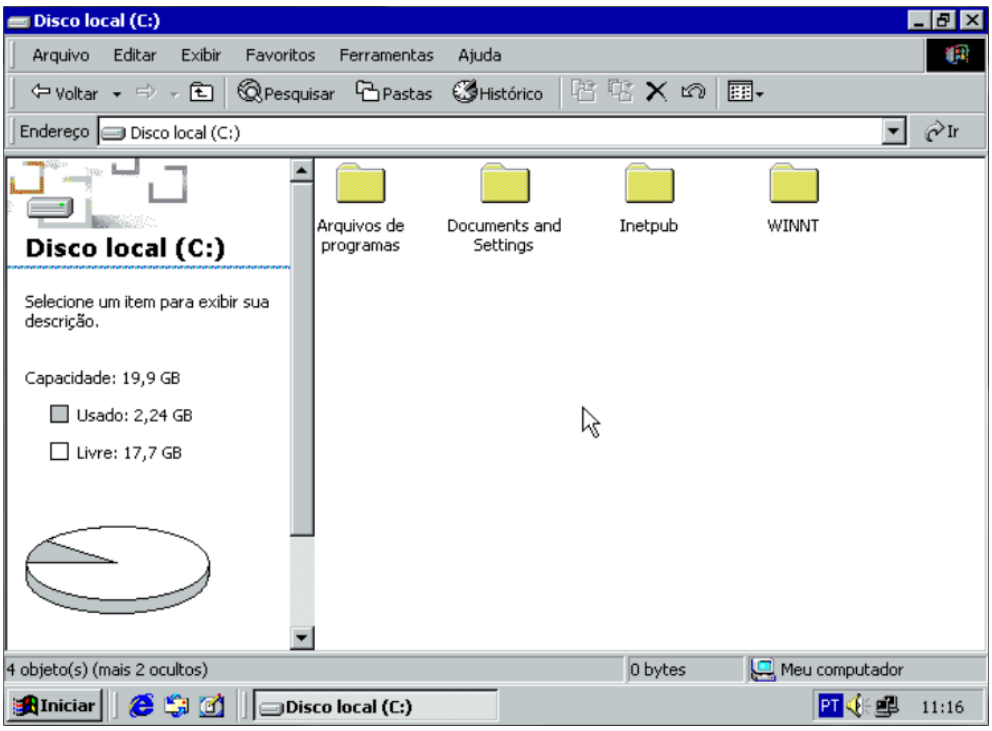


Figura 3.18 - Pasta C:/ do servidor *Windows*.
Fonte: Elaborado pelos autores, 2012.

16. Crie um arquivo txt para enviar ao servidor invadido, através do *Armitage*.


Figura 3.19 - Criando arquivo *txt* no *Backtrack*.
Fonte: Elaborado pelos autores, 2012.

17. De volta ao *Armitage*, clicando em '*Upload*', abre-se uma janela, nela selecionar o arquivo criado para o envio. Depois, clicar em '*Open*'.

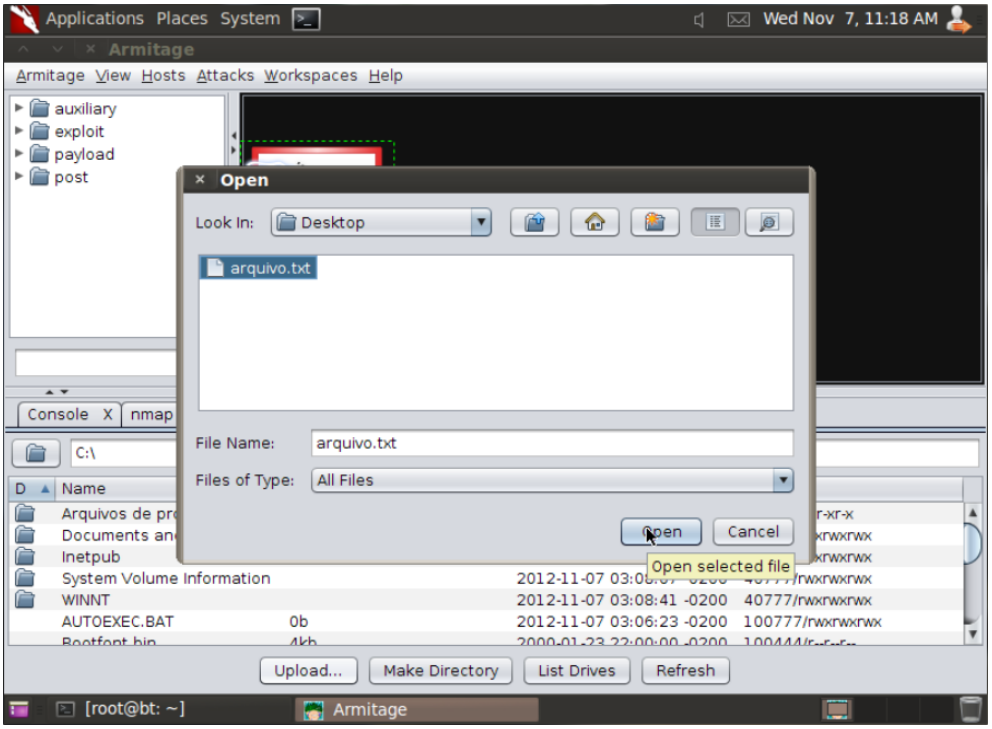


Figura 3.20 - Enviando arquivo ao servidor invadido.
Fonte: Elaborado pelos autores, 2012.

18. Comprove o envio do arquivo no servidor *Windows*, no diretório '*C:/*'.

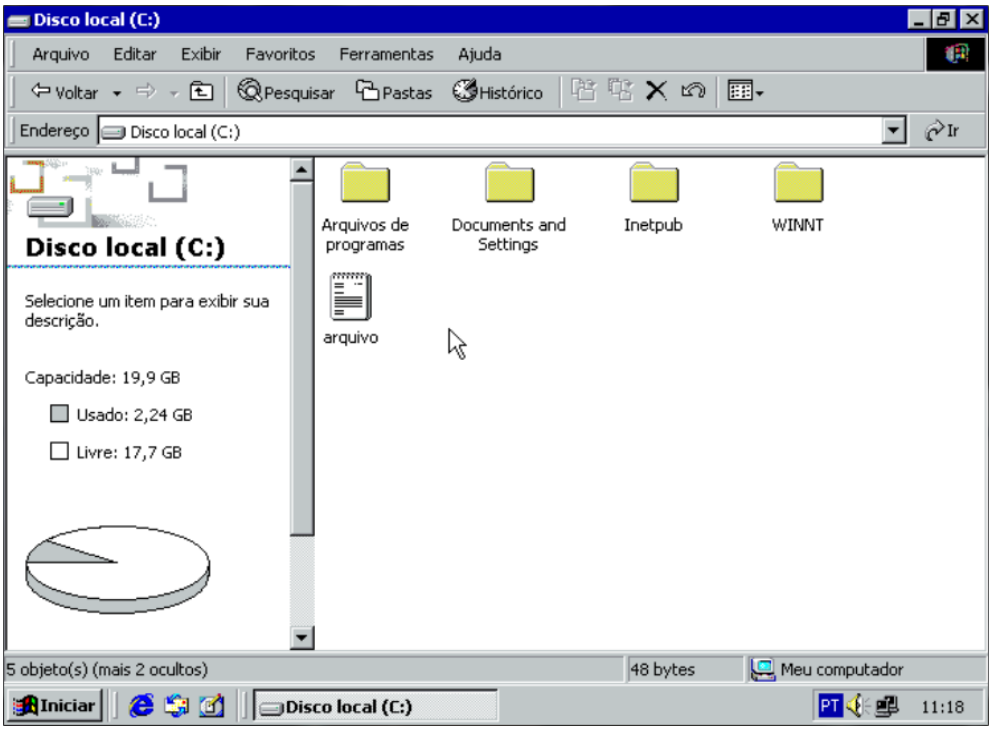


Figura 3.21 - Tela com arquivo enviado ao servidor.
Fonte: Elaborado pelos autores, 2012.

Neste caso, como pode ser notado na figura 3.16, a aplicação encontrou e explorou duas vulnerabilidades no servidor testado, são elas a `ms03_026_dcom` e a `ms08_067_netapi`, a primeira uma vulnerabilidade que foi explorada pelo buffer *overflow* na *interface Remote Procedure Call* (*RPC*) e a segunda, uma vulnerabilidade causada pelo serviço do servidor, que não controla corretamente as solicitações de *RPC* especialmente criadas. Esse serviço roda por padrão diretamente na porta 445 sobre *TCP*, através da *API NetBios*. Ambas já descobertas pelo fornecedor do *software* e constam em seu boletim de segurança, no endereço *http://technet.microsoft.com/en-us/security/bulletin/ms03-026* e *http://technet.microsoft.com/en-us/security/bulletin/ms08-067*, onde se recomenda que o administrador deva instalar o aplicativo de correção imediatamente.

## 3.5 CORRIGINDO AS VULNERABILIDADES E REPETINDO OS TESTES.

Visando comprovar a eficiência da solução proposta pelo fornecedor do *software* usado no servidor alvo 1, uma atualização do mesmo foi realizada com o pacote *service pack* 4 e na sequência os testes foram repetidos, como pode ser notado pelas figuras 3.22, 3.23 e 3.24, as vulnerabilidades foram corrigidas e não foi

possível concretizar a invasão desta vez.

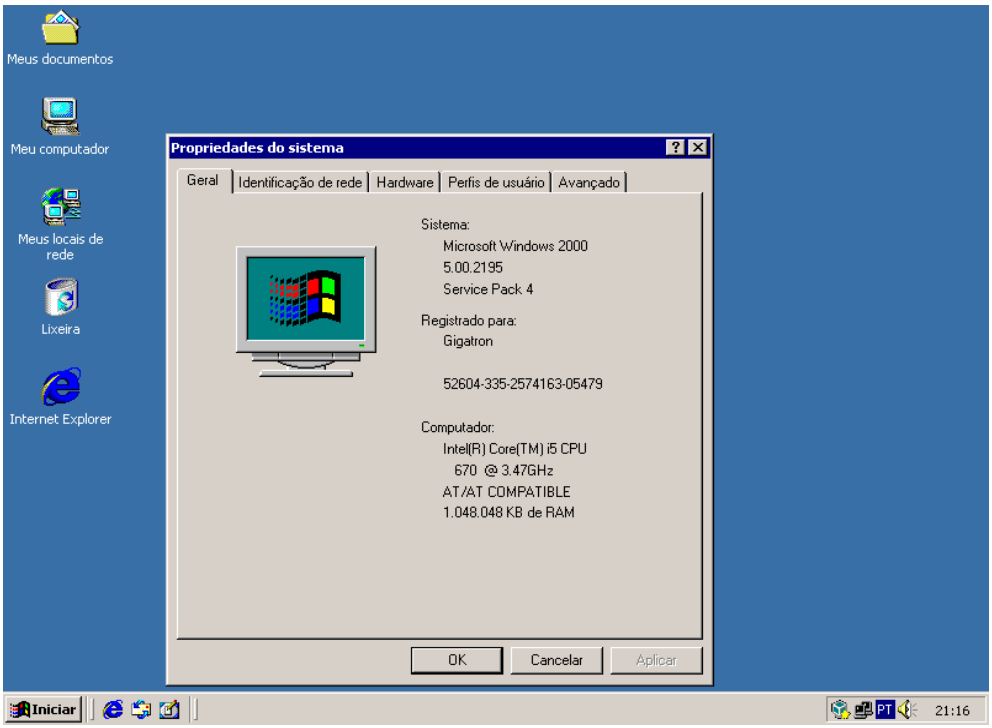


Figura 3.22 - Tela do sistema atualizado (*service pack* 4).
Fonte: Elaborado pelos autores, 2012.

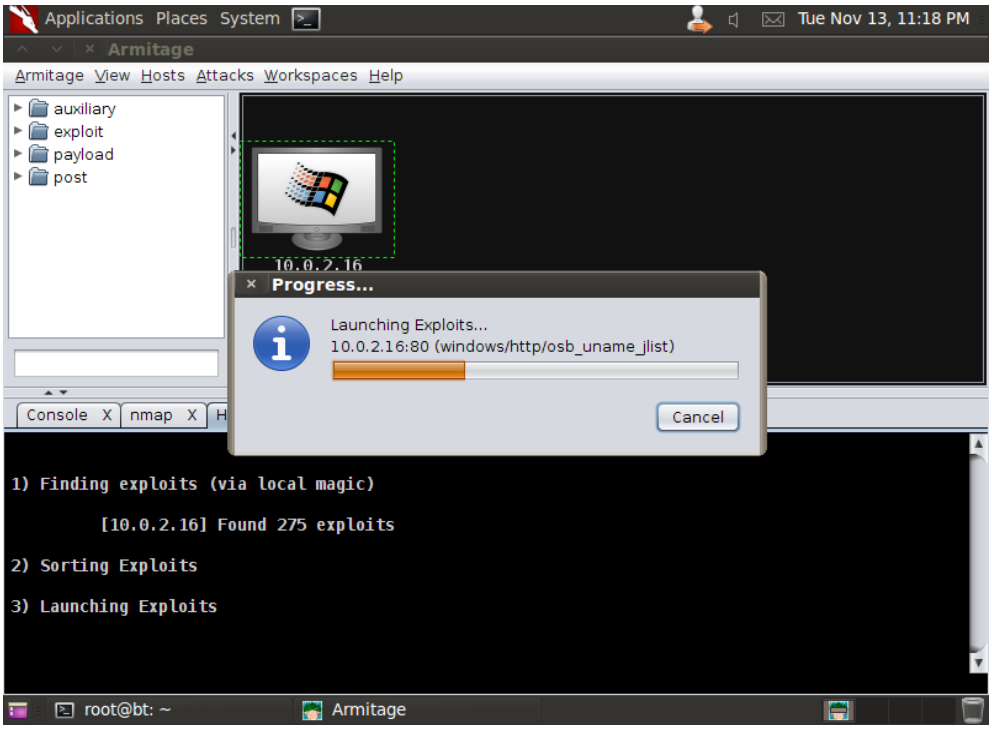


Figura 3.23 – Aplicando *exploits* no sistema atualizado.
Fonte: Elaborado pelos autores, 2012.

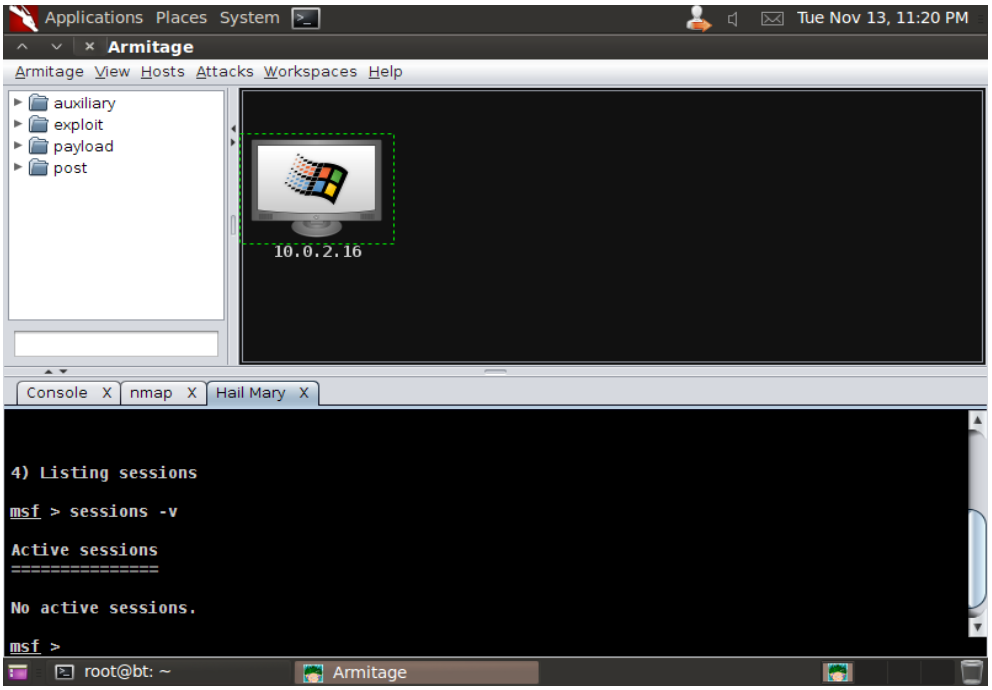


Figura 3.24 – Nenhuma vulnerabilidade encontrada.
Fonte: Elaborado pelos autores, 2012.

## 3.6 SEGUNDO TESTE UTILIZANDO O *ARMITAGE*

No segundo teste foram seguidos os mesmos passos do primeiro, só que desta vez na máquina alvo 2 (sistema *Linux* 12.10), e como pode ser notado os resultados foram diferentes e como mostram as figuras 3.25 e 3.26, nenhuma vulnerabilidade foi encontrada.

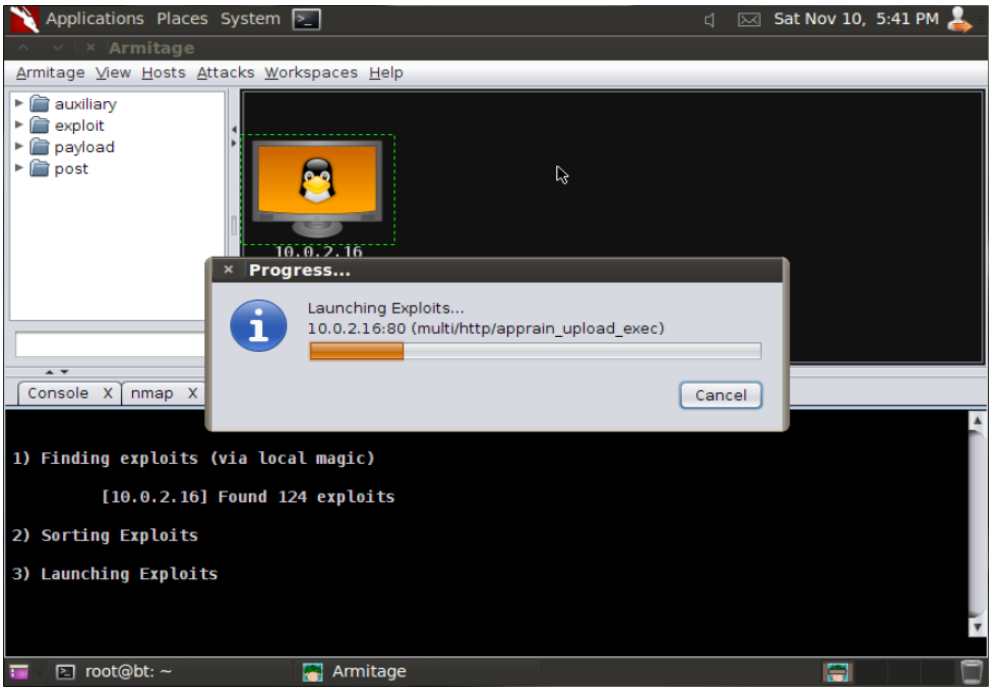


Figura 3.25 - *Armitage* tentando aplicar *exploits no servidor Ubuntu*.
Fonte: Elaborado pelos autores, 2012.

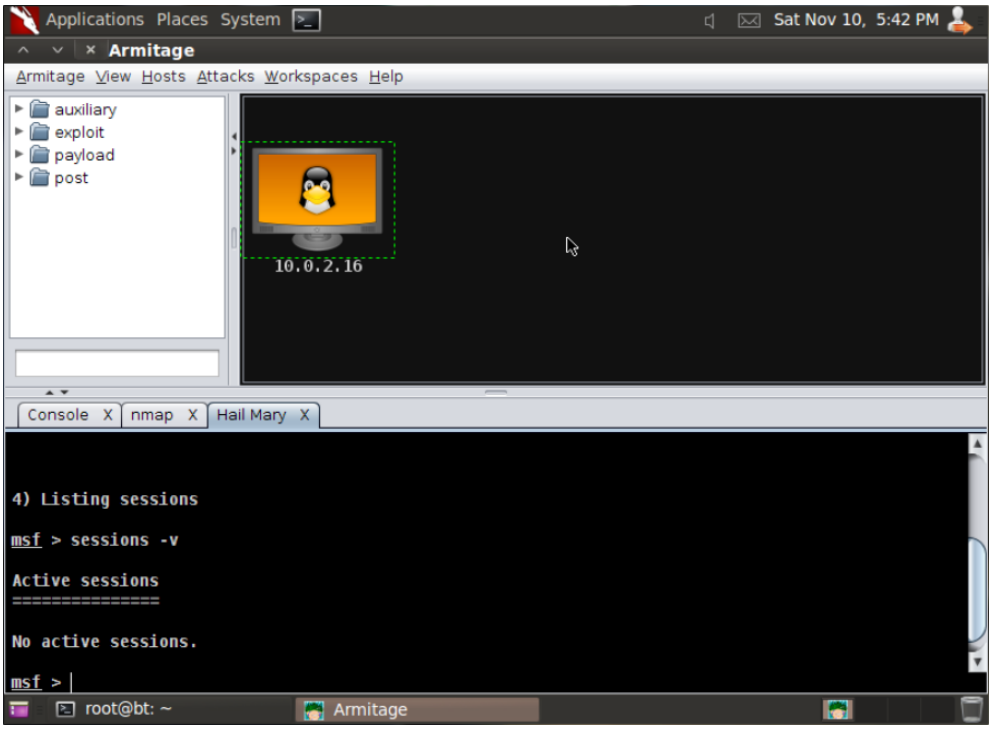


Figura 3.26 - Resultado da tentativa de invasão com *exploits*.
Fonte: Elaborado pelos autores, 2012

Isso não significa necessariamente que um sistema é melhor que o outro e nem é esse o objetivo deste trabalho, mas deve-se observar que a máquina alvo 1 (*Windows*) no primeiro teste não tinha sido atualizada com o pacote de atualizações recomendado pelo desenvolvedor, já no segundo teste com o sistema devidamente atualizado a falha foi corrigida e nenhuma vulnerabilidade conhecida foi detectada.

O sistema da máquina alvo 2 (*Ubuntu*) foi devidamente atualizado antes dos testes e portanto resistiu ao ataque não mostrando vulnerabilidades conhecidas no teste com esta ferramenta. Essa falta de atualização do servidor alvo 1 foi propositalmente deixada de lado para ilustrar que a grande fonte de vulnerabilidades em servidores atualmente é a falta de atualização dos sistemas e serviços e a má configuração dos mesmos, o que às vezes, pelas atribulações do dia a dia, passa despercebida pelo administrador que só irá notar a falha após uma invasão, quando na verdade já é tarde demais.

## 3.7 TESTANDO A PROTEÇÃO DE UM *FIREWALL*

Como visto no item 3.4, quando o servidor *web* estava sem a atualização sugerida pelo fornecedor e de frente para a *internet* os testes realizados mostraram as vulnerabilidades e foi possível a invasão. Como a intenção deste estudo é mitigar

os riscos e ajudar o administrador a proteger sua rede, uma nova bateria de testes foi realizada, porém, desta vez testando a eficiência de um *firewall*. Para tanto um novo cenário foi criado, incluindo uma outra máquina que funcionará como *firewall* da rede, filtrando pacotes indesejados e liberando apenas o que realmente interessar a rede. A figura 3.27 ilustra a topologia da rede no novo teste.

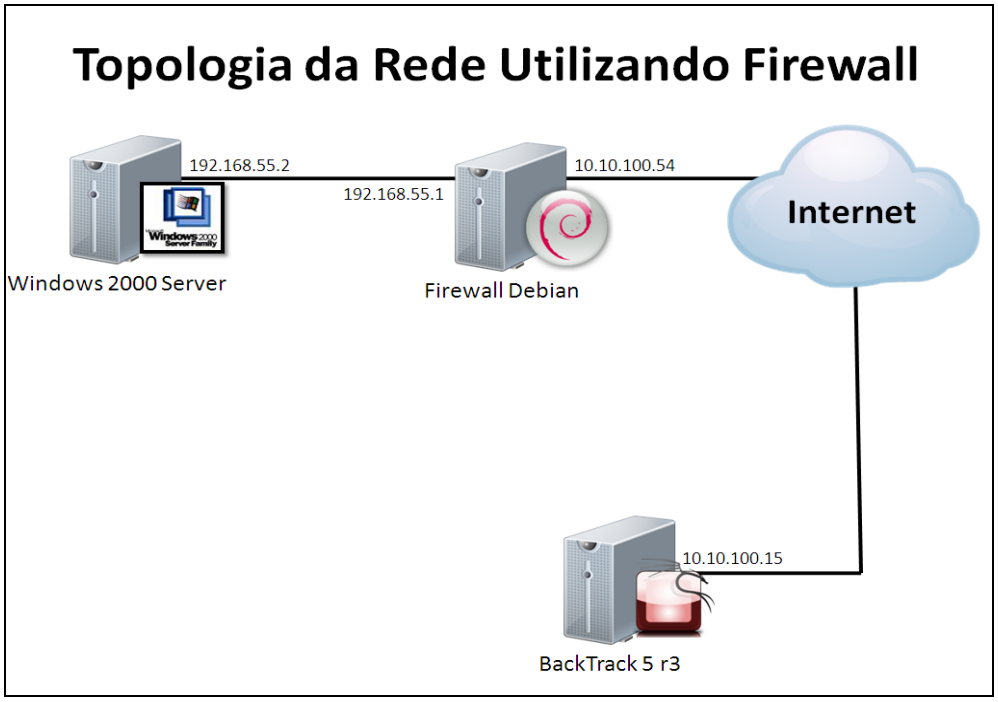


Figura 3.27 – Topologia do novo teste utilizando *Firewall*.
Fonte: Elaborado pelos autores 2012.

A figura mostra a topologia com o *firewall* conectado à núvem (*internet*), como de fato estaria em um ambiente real em que um invasor também conectado a nuvem faria a tentativa de intrusão, porém, no ambiente de teste, esta máquina foi conectada diretamente ao computador atacante (*Backtrack* 5).

### 3.7.1 Configuração da Rede de Testes

Na sequência foi configurada a rede para os testes da seguinte forma: Na máquina atacante (*Backtrack*) a *interface* de rede foi configurada como *host only* e atribuído o IP 10.10.100.15. Na máquina *Firewall* (*Debian*) foram instaladas duas *interfaces* de rede, a primeira (*eth0*) foi configurada como *host only*, atribuído *IP* 10.10.100.54 e colocada de frente para a *internet*, ou seja, como rede externa.

A segunda *interface* (eth1) foi configurada como *host only* #2 e atribuído o *IP*

192.168.55.1 e na máquina alvo a *interface* foi configurada como *host only* #2, atribuído o *IP* 192.168.55.2 e conectada a eth1 da máquina *Firewall* (*Debian*).

Após todas as configurações foram realizados alguns testes, para confirmar o funcionamento da rede e do servidor *Web*. O primeiro foi realizado na máquina atacante (*Backtrack*) visando acessar o site hospedado na máquina alvo, note que o endereço digitado é o do *Firewall* e não do servidor *Web*, o resultado pode ser observado na figura 3.28.

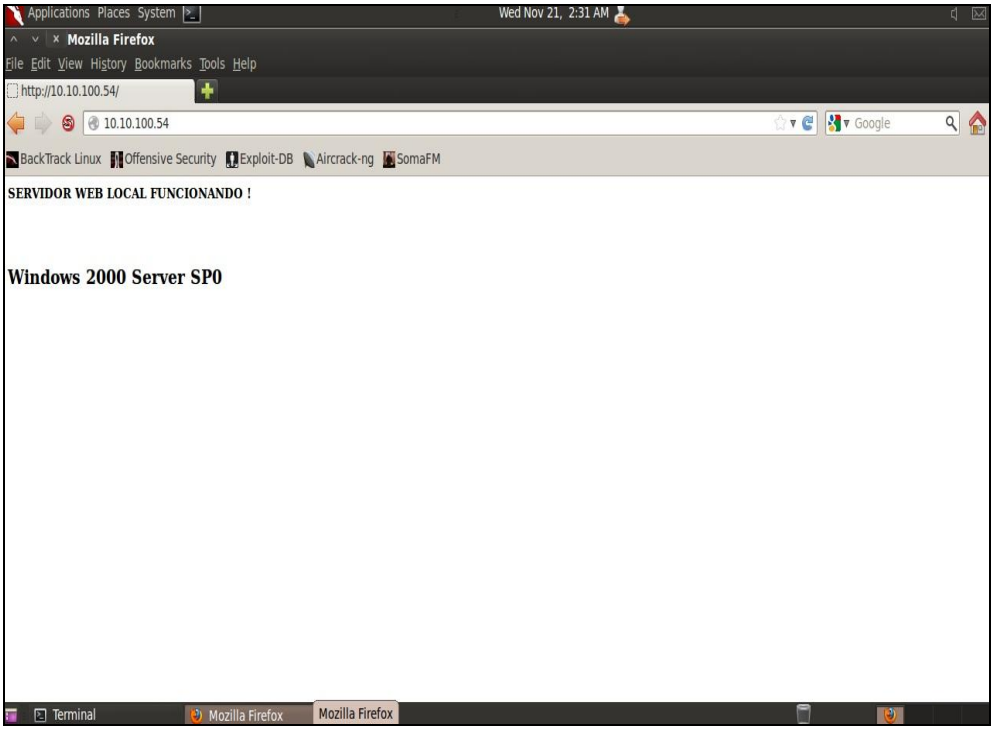


Figura 3.28 - Acessando o servidor com firewall ativo.
Fonte: elaborado pelos autores 2012.

### 3.7.2 Configuração do *Firewall Debian* 6.05

Para o *firewall* foi utilizado uma máquina virtual criada no *Virtual Machine* da *Oracle*, com *HD* dinamicamente alocado de 20GB, memória de 512MB e instalado o sistema *Debian* versão 6.05. Para que funcione como *firewall* foi criado um *script* somente com as configurações básicas de *firewall* e adicionado as seguintes regras:

```
iptables -t nat -A PREROUTING -i $IFACE_EXT -p tcp -s 0/0 --
sport 1024:65535 -d $IPFW_EXT --dport 80 -j DNAT --to-dest
192.168.55.2:80

iptables -A FORWARD -i $IFACE_EXT -p tcp -s 0/0 --sport
1024:65535 -d 192.168.55.2 --dport 80 -j ACCEPT
```

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

A primeira redireciona as requisições que chegarem pela *interface* externa do *firewall*, utilizando o protocolo *TCP*, vindos de qualquer lugar com destino ao *IP* externo do *firewall* e porta 80, para o *IP* do servidor *Web* hospedado na máquina com *Windows Server* 2000 vulnerável, porta 80, ou seja, faz o redirecionamento de porta, trocando o endereço da *interface* externa do *firewall* pelo endereço da *interface* do servidor *Web* hospedado no *Windows Server* 2000.

A segunda libera os pacotes de pedidos de dados vindo da *internet* (*interface* externa do *firewall*) para o servidor *Web* do *Windows Server* 2000. Desta forma o *firewall* libera apenas o que foi definido nas regras e bloqueia tudo o que não estiver especificado.

Por fim, a terceira regra é uma regra de estado responsável por liberar a passagem da resposta do servidor *Web* do *Windows Server* 2000 para os clientes que solicitaram conexão.

## 3.8 APLICAÇÃO DO TESTE NA REDE PROTEGIDA COM *FIREWALL*

Testado o funcionamento da rede e do servidor *web*, foram iniciados os testes de penetração da mesma forma que os anteriores. Desta vez foi possível notar que o *Backtrack* identificou o alvo como sendo um sistema *Windows* (figura 3.29), isso porque apesar da consulta ter sido feita à máquina *Debian*, esta redirecionou a requisição, através das regras do *firewall* ao servidor *web*, que retornou com a resposta solicitada pelo atacante. Foi possível notar também que apesar do servidor ter diversas portas abertas (como visto no teste anterior), a única detectada desta vez foi a porta 80, pois o teste foi dirigido ao *firewall* e não ao servidor *web* e o *firewall* não possui outras portas abertas nem vulnerabilidades conhecidas.

Por meio deste resultado já foi possível notar que o *Firewall* está protegendo o servidor, pois não permitiu que o teste do atacante mostrasse as vulnerabilidades existentes no mesmo, entretanto, visando confirmar o que já se mostra evidente foi finalizado o teste de invasão, aplicando os *exploits* existentes para este serviço e sistema operacional e como pode ser notado pela figura 3.30, não foi possível a intrusão no servidor *Web* (*No active sessions*).

Portanto, fica claro a importância de um *firewall* na proteção de servidores,

porém vale lembrar que deve ser um *firewall* bem configurado e testado, caso contrario pode se ter a falsa noção de segurança, o que de certa forma facilita o ataque.

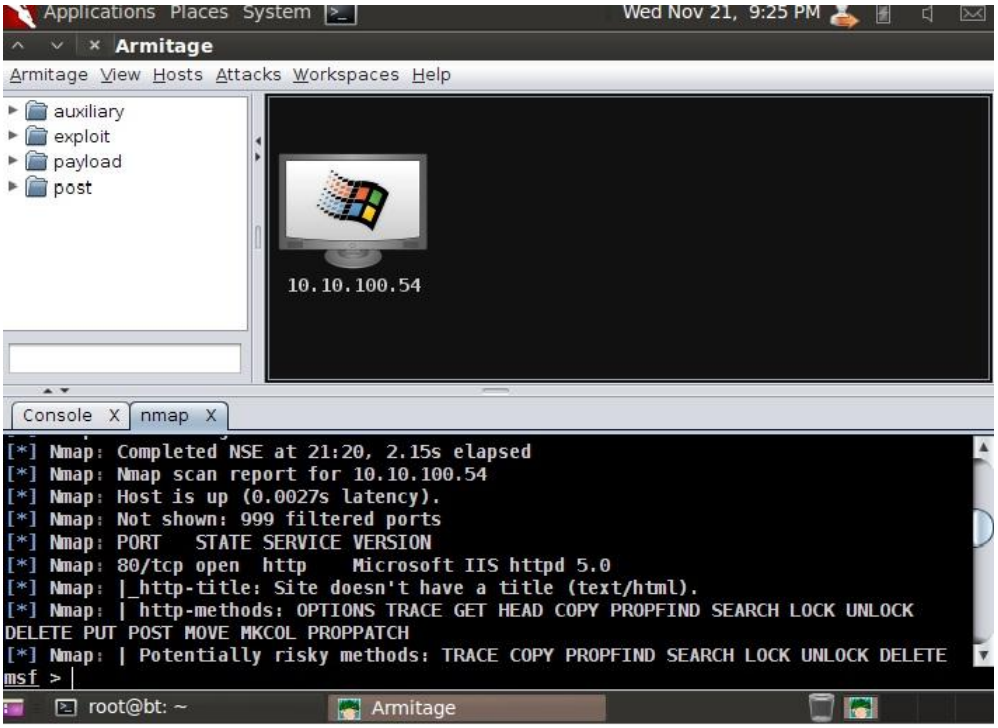


Figura 3.29 – Resultado parcial do *Armitage* no teste com *firewall*.
Fonte: elaborado pelos autores 2012.

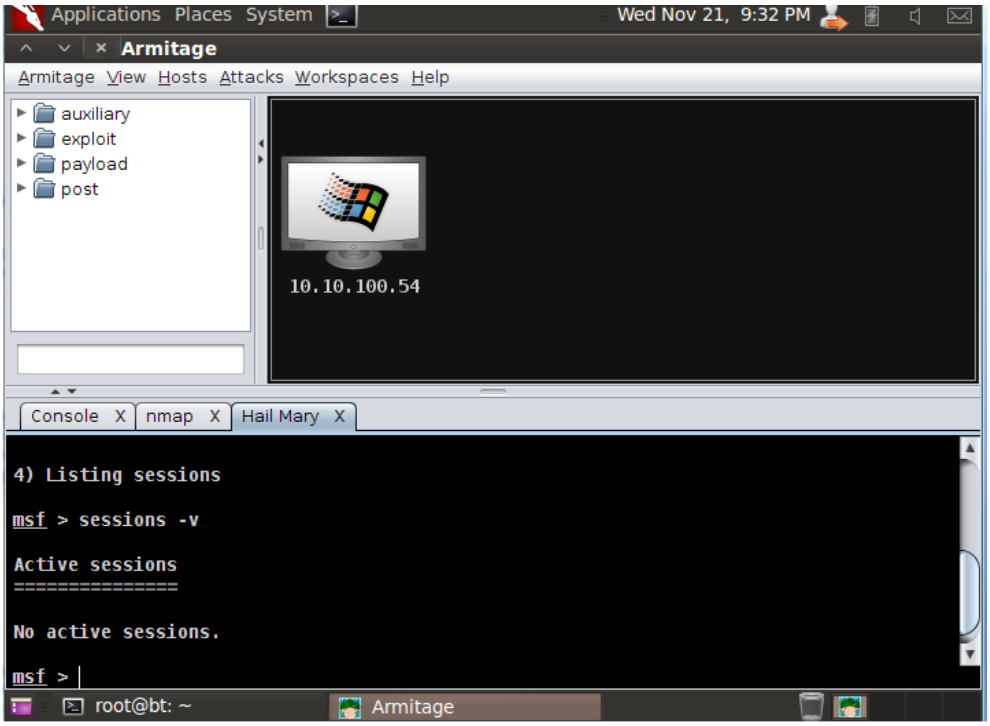


Figura 3.30 – Resultado Final do *Armitage* no teste com *firewall*.
Fonte: elaborado pelos autores 2012

# CONCLUSÃO

Os tipos ou técnicas de penetração de um servidor de rede são as mais diversas possíveis e podem ser feitas de diversas formas. Além disso, novas ameaças aparecem com novos serviços e sistemas. Há de se considerar ainda que invasores são pessoas experientes, hábeis e que não se cansam de estudar e pesquisar (geralmente autodidatas) novas ferramentas e técnicas.

Portanto o profissional responsável pela segurança de redes deve estar atento e buscar constantemente melhorias de segurança para sua rede, lembrando que qualquer serviço que seja instalado em um servidor será um ponto de vulnerabilidade a mais, uma vez que abre uma porta de comunicação, então uma configuração correta e com apenas os serviços estritamente necessários é essencial para elevar o nível de segurança em um servidor.

Mesmo tendo toda segurança possível nos sistemas e serviços é preciso lembrar que quem cria e mantém estes sistemas e serviços são seres humanos e que como tal possuem fraquezas e são sem sombra de dúvida vulneráveis, o que torna o mais perfeito sistema de segurança vulnerável se não houver uma forte confidencialidade por quem detém as “chaves” do sistema.

Confiar plenamente na segurança que se tem pode ser o maior dos erros quando se trata de segurança. Vale lembrar ainda que apesar de não ter sido o foco deste trabalho, existe o constante perigo de um ataque interno, para o qual o *firewall* colocado nos testes não teria efeito algum. Este tema, ataques internos, poderia ser o foco de um novo estudo.

Como visto, não existe uma receita única para manter a segurança e sim diversas atitudes que devem ser tomadas com frequência por quem controla a rede, entre elas, estar atento a qualquer mudança de comportamento em sua rede e serviços, verificar sempre arquivos de *logs*, configurar de forma correta os serviços ativos, manter o sistema atualizado, ter um bom *firewall* instalado e devidamente configurado, ler constantemente os boletins de segurança do fornecedor se seus *softwares* são exemplos de atitudes que, sem dúvida, elevariam o nível de segurança da rede.

Diante do exposto e após as análises realizadas, conclui-se que conhecer o *modus-operandi*, o raciocínio e as ferramentas utilizadas pelos atacantes é fundamental para um administrador de rede na proteção de seus ativos. Além disso,

a *Backtrack* mostrou ser um recurso importante no auxílio ao administrador da rede em testes de penetração de seus servidores, desde que realizados sob critérios pré-estabelecidos e devidamente monitorados.

# REFERÊNCIAS BIBLIOGRÁFICAS


ASSUNÇÃO, M. F. A. **Segredos do Hacker Ético**. 3. ed. Florianópolis: Visual, 2010.

BACKTRACK. **Howtos: BackTrack Linux** – Penetration Testing Distribution. 2012. Disponível em: <http://www.backtrack-linux.org/tutorials/> Acesso em: 05 abr. 2012.

BORGES, C. G. **Estudo comparativo de metodologias de pentests.** 2011. TCC (Pos graduação em segurança de Sistemas) – Universidade Luterana do Brasil Campus Canoas, Canoas - RS.

BUENO, F. S. **Silveira Bueno Dicionário da língua Portuguesa.** São Paulo: FTD, 2000.

CERT**. Centro de Estatísticas, Respostas e Tratamentos de Incidentes.** 2012. Disponível em: <http://www.cert.br/stats/incidentes/> Acesso em: 14 mar. 2012.

FAIRCLOTH, J. **Penetration Tester's Open Source Toolkit.** 3.ed. Waltham: Syngress, 2011.

ISECOM **Institute For Security And Open Methodologies**. **Open Source Security Testing Methodology Manual – OSSTMM v3.0.** 2010. Disponível em: <http://www.isecom.org/mirror/OSSTMM.3.pdf>. Acesso em: 20 jul. 2012.

LIEIRA, J. F. **Segurança de Redes**. 2011, 87p. Apostila da disciplina de Segurança de Redes. Faculdade de Tecnologia de Lins.

MITNICK, K. D.; SIMON, W. L. **A arte de invadir**. São Paulo: Pearson. 2006.

NESSUS, **Guia do Usuário Nessus 5.0**. 2012. Disponivel em: http://static.tenable.com/documentation/nessus_5.0_user_guide_PTB.pdf> Acesso em: 14 mar 2012.

NIST. **Technical Guide to Information Security Testing and Assessment,** 2008. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf> Acesso em: 17 jul. 2012.

OISSG. **Open Information Systems Security Group**. **Information System Security. Assessment Framework Draft – ISSAF 0.2.1.B**. 2006. Disponível em: <http://www.oissg.org/downloads/issaf-0.2/index.php>. Acesso em: 13 jul. 2012.

OWASP. **Open Web Application Security Project. OWASP Testing Guide v3.0**. 2008. Disponível em: <https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf>. Acesso em: 23 jul. 2012.

SANTOS, E. **Avaliando a Importância das Metodologias para Aplicação de Testes de Segurança em Sistemas de Informação.** 2010, 108p. TCC - Instituto Federal de Santa Catarina, Santa Catarina. Disponivel em: < http://www.inf.ufsc.br/~esantos/files/TCC_EduardoDosSantos.pdf>. Acesso em: 23 jul. 2012.

SMITH EBook hackers secrets and confessions. **4Shared**, 25 mai.2005. Disponível em: <http://www.4shared.com/office/-KqRApmX/eBook_Hackers_Secrets_And_Conf.html>. Acesso em: 25 mai. 2012

SYMANTEC. **Glossário De Segurança - Security 101**. 2012. Disponível em: <http://www.symantec.com/pt/br/business/theme.jsp?themeid=glossario-de-seguranca> Acesso em: 10 mai. 2012.

ULBRICH, H. C.; VALLE, J. D. **Universidade Hacker**. 4. ed. São Paulo: Digerati, 2004.

VIEIRA, L**. LINUX: Elevação de privilégios locais** [Artigo]. Viva o Linux, 21 março 2011. Disponivel em: <http://www.vivaolinux.com.br/artigo/Elevacao-de-privilegios-locais/>. Acesso em: 22 maio 2012.

# GLOSSÁRIO

Este Glossário foi criado em conformidade com *SYMANTEC* (2011) e CGIBR (2006) contendo os termos técnicos utilizados do trabalho, com o intuito de dirimir possíveis dúvidas que possam surgir durante sua leitura.

ATACANTE – Pessoa responsável pela realização de um ataque.

ATAQUE – É a tentativa, bem ou mal sucedida, de acesso ou uso não autorizado a um programa ou computador. Também são considerados ataques as tentativas de negação de serviço (*DoS*).

ATAQUES DE FORÇA BRUTA (*Brutal Force*) – São estratégias utilizadas para descobrir senhas, consiste em uma busca exaustiva feita por meio de algoritmos tentando encontrar todas as combinações possíveis até se descobrir a senha correta.

*BACKDOOR* – Código malicioso que abre caminho para o acesso remoto a um computador contaminado sem o conhecimento do usuário.

CAVALO DE TRÓIA (*Trojan Horse*) – È um programa que se mascara ou aparenta possuir um propósito benigno, mas que arbitrariamente realiza funções maliciosas

CÓDIGO MALICIOSO – Termo genérico que se refere a todos os tipos de programa que executam ações maliciosas em um computador.

CRIPTOGRAFIA – Método de embaralhar ou codificar dados para impedir que usuários não autorizados os leiam ou adulterem. É usada, dentre outras finalidades, para: autenticar a identidade de usuários; autenticar transações bancárias; proteger a integridade de transferências eletrônicas de fundos, e proteger o sigilo de comunicações pessoais e comerciais.

*DDOS* (*Distributed Denial of Service*) – Ataque de negação de serviço distribuído, ou

seja, um conjunto de computadores é utilizado para tirar de operação um ou mais serviços ou computadores conectados à *Internet*.

*DESKTOPS* – Computadores de uso pessoal ou como terminais de rede.

DNS (*Domain Name System*) – Serviço que traduz nomes de domínios para endereços *IP* e vice-versa.

ENDEREÇO *IP* - Número único para cada computador conectado à *Internet*, composto por uma seqüência de quatro números que variam de 0 até 255, separados por ".". Por exemplo: 192.168.34.25.

ENGENHARIA SOCIAL – Método utilizado pelos invasores para induzir os usuários a realizarem uma ação que normalmente resulta em consequências negativas, como o *download* de *malware* ou a divulgação de informações pessoais. Ataques de *phishing* exploram essa metodologia.

*EXPLOIT* – Programa que se aproveita das vulnerabilidades de um sistema operacional para instalar um *payload*.

*FIREWALL* – Dispositivo constituído pela combinação de *software* e *hardware*, utilizado para dividir e controlar o acesso entre redes de computadores.

*HANDSHAKE* - Ou aperto de mão é o processo pelo qual duas máquinas afirmam uma a outra que a reconheceu e está pronta para iniciar a comunicação.

INVASÃO – Ataque bem sucedido que resulte no acesso, manipulação ou destruição de informações em um computador.

INVASOR – Pessoa responsável pela realização de uma invasão ou pelo comprometimento do computador.

*LISTENING* – Significa que o aplicativo esta aguardando que outro se conecte com ele.

*LOG* – È o registro de atividades gerado por programas de computador. No caso de *logs* relativos a incidentes de segurança, eles normalmente são gerados por *firewalls* ou por *IDSs*.

*MALWARE* - Descrição genérica geral para qualquer programa de computador que produza efeitos indesejados ou mal-intencionados. São considerados *malwares* os vírus, *worms*, cavalos de tróia e *backdoors*.

*MD* (*message digest*) – Resumo de mensagem.

*METASPLOIT* – *Framework* automatizada para varredura de vulnerabilidades e computadores, bem como aplicação de *exploit*.

*MODUS OPERANDI* - Expressão em latim que significa "modo de operação". Utilizada para designar uma maneira de agir, operar ou executar uma atividade.

*PAYLOAD* - Código (algoritmo) malicioso que explora a vulnerabilidade e proporciona a invasão.

*PHARMING* – Método de ataque que redireciona o tráfego de um site normalmente desenvolvido para imitar o *site* legítimo. O objetivo é fazer com que os usuários ignorem o redirecionamento e repassem informações pessoais, como senhas de bancos *on-line*, no *site* fraudulento. O *pharming* pode ser praticado ao se alterar arquivos no computador da vítima ou explorando uma vulnerabilidade no *software* do servidor DNS.

*PHISHING* – Golpe que usa *spam* e mensagens instantâneas para levar pessoas a divulgarem informações confidenciais, como senhas de banco e dados de cartão de crédito. Normalmente, esses ataques aparentam ser de instituições financeiras

*PING* – O *Ping* funciona de forma parecida a um sonar, porém no mundo virtual, este comando envia pacotes (ICMP) de dados de 32 bits ao computador destino e solicita resposta. Se a resposta voltar significa que houve comunicação e que o

computador destino esta ativo, além disso, o tempo de resposta é medido, visando conhecer a velocidade da rede naquele instante.

*ROOTKIT* –Programa ou um conjunto de programas, usado por um atacante para ocultar sua presença em um determinado sistema e permitir seu acesso futuro

SENHA - Conjunto de caracteres, de conhecimento único do usuário, utilizado no processo de verificação de sua identidade.

*SOFTWARE* – Uma sequência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado/informação ou acontecimento, ou seja, um programa de computador.

*SPAM* – Termo usado para se referir aos *emails* não solicitados, que geralmente são enviados para um grande número de pessoas. Quando o conteúdo é exclusivamente comercial, este tipo de mensagem também é referenciado como *UCE* (do Inglês *Unsolicited Commercial E-mail*).

*SQL* - Structured Query Language, ou Linguagem de Consulta Estruturada, é uma linguagem de pesquisa declarativa para banco de dados relacional(base de dados relacional).

*TROJANS* – São programas aparentemente inofensivos, úteis e com alguma funcionalidade, porém esconde programas nocivos que põe em risco a segurança das informações.

VÍRUS – Programa escrito para alterar a maneira como um computador opera, sem a permissão ou conhecimento do usuário. Deve atender a dois critérios: executar a si mesmo e auto-replicar.

VULNERABILIDADE – É uma falha no projeto, implementação ou configuração de um software ou sistema operacional que, quando explorada por um atacante, resulta na violação da segurança de um computador.

*WORM* – Programa capaz de se propagar automaticamente através de redes, enviando cópias de si mesmo de computador para computador. Diferente do vírus, o *worm* não embute cópias de si mesmo em outros programas ou arquivos e não necessita ser explicitamente executado para se propagar. Sua propagação se dá por meio da exploração de vulnerabilidades existentes ou falhas na configuração de *softwares* instalados em computadores.