MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
SEÇÃO DE ENGENHARIA DE COMPUTAÇÃO

TEN BRUNO MEDEIROS FRAGA
TEN RANMSÉS E. MARTINS BASTOS

IMPLEMENTAÇÃO DE UMA HONEYNET UTILIZANDO MÁQUINAS VIRTUAIS

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA

TEN BRUNO MEDEIROS FRAGA
TEN RANMSÉS E. MARTINS BASTOS

IMPLEMENTAÇÃO DE UMA *HONEYNET* UTILIZANDO MÁQUINAS VIRTUAIS

Monografia de Projeto de Fim de Curso apresentada ao Curso de Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Maj Sérgio dos Santos Cardoso Silva – M. Sc.

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP 22290-270

004.6 Fraga, Bruno Medeiros.
F811i Implementação de uma honeynet utilizando máquinas virtuais / Bruno Medeiros Fraga, Ranmsés E. Martins Bastos – Rio de Janeiro: Instituto Militar de Engenharia, 2010.

73 p. : il.

Projeto Final de Curso – Instituto Militar de Engenharia, 2010.

1. Redes de computadores – segurança 2. Honeynet virtual. 3. Linux. I. Implementação de uma honeynet utilizando máquinas virtuais. II. Instituto Militar de Engenharia.

CDD 004.6

INSTITUTO MILITAR DE ENGENHARIA

TEN BRUNO MEDEIROS FRAGA

TEN RANMSÉS E. MARTINS BASTOS

IMPLEMENTAÇÃO DE UMA *HONEYNET* UTILIZANDO MÁQUINAS VIRTUAIS

Monografia de Projeto de Fim de Curso apresentada ao Curso de Engenharia de Computação do Instituto Militar de Engenharia.
Orientador: Maj Sérgio dos Santos Cardoso Silva – M. Sc.

Aprovada em 13 de agosto de 2010 pela seguinte Banca Examinadora:

_____

Maj Sérgio Dos Santos **Cardoso** Silva – M. Sc. – Presidente

_____

Professora Raquel Coelho Gomes Pinto – D. Sc.

_____

Cap **Anderson** Fernandes P. Dos Santos – D. Sc.

Rio de Janeiro

2010

SUMÁRIO

LISTA DE ACRÔNIMOS

CPU: Central Processing Unit
DNS: Domain Name Server
DoS: Denial of Service
DDoS: Distributed Denial of Service
FTP: File Transfer Protocol
HIDS: Host Intrusion Detection System
HTTP: Hyper Text Transfer Protocol
ICMP: Internet Control Message Protocol
IDS: Intrusion Detection System
IP: Internet Protocol
IPS: Intrusion Protection System
NIDS: Network Intrusion Detection System
SSH: Secure Shell
SO: Sistema Operacional
TCP: Transmission Control Protocol
UDP: User Data Protocol
TI: Tecnologia da Informação
VM: Virtual Machine

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

RESUMO

O presente trabalho trata da implementação de um sistema que se destina a funcionar acoplado a um projeto de segurança em redes de computadores, colaborando para garantir um bom nível de segurança a uma rede que se pretende proteger. Para a concretização do sistema, foi construída uma *honeynet* virtual em uma máquina do Laboratório de Redes da SE-8 (Subseção de Computação) do Instituto Militar de Engenharia. Dentro deste ambiente virtualizado foram instalados vários serviços e realizou-se ataques à *Honeynet* com os respectivos registros pelas ferramentas específicas de monitoramento.

O software de virtualização utilizado foi o VMware Server em máquina Linux cuja distribuição é o Ubuntu Server. Como ferramenta de monitoramento e detecção de ataques utilizou-se o Snort em conjunto com o Guardian, que foi responsável por tomar as medidas de reação.

ABSTRACT

The goal of this work is to implement a system that is attached to a security project in computer networks, helping to ensure a good level of security for a given network to be protected. For the implementation of the system, we constructed a virtual honeynet on a machine on Laboratório de Redes of SE-8 (Subseção de Computação) of Instituto Militar de Engenharia. Some services were installed in this virtualized environment and attacks were carried out on the Honeynet, while specifics monitoring tools were recording them.

The virtualization software used was VMware Server on Linux machine whose distribution is the Ubuntu Server. As a monitoring tool and attack detection we used Snort together with Guardian, which was responsible for taking response actions.

1 INTRODUÇÃO

1.1 FINALIDADE

Os avanços trazidos pelas redes de computadores proporcionaram maior facilidade e agilidade para várias atividades do ambiente de trabalho e do cotidiano das pessoas. Comunicação, acesso à informação, movimentações financeiras são exemplos disso. Entretanto, surgiram problemas de acesso a dados por pessoas não autorizadas. Isto inclui o roubo de informações pessoais ou até mesmo roubo de informações confidenciais de uma corporação ou governo. Com o objetivo de conter estas ações danosas, muitas ferramentas e técnicas de monitoramento e detecção de intrusos vêm sendo desenvolvidas o que, de fato, movimenta o mercado e exige o conhecimento e o aprimoramento contínuo dos profissionais da área.

Neste cenário, tem-se como proposta a implementação de uma *honeynet* composta por máquinas virtuais. Uma *honeynet* é uma rede de *honeypots*, que são máquinas projetadas para atraírem atacantes e serem facilmente atacadas sem que haja danos colaterais a máquinas oficiais da rede. Portanto uma *honeynet* se trata de uma rede de computadores projetada especificamente para ser comprometida e que contém mecanismos de controle para prevenir que seja utilizada como base de ataques contra outras redes. Ela é utilizada para observar o comportamento dos invasores, possibilitando análises das ações realizadas pelos atacantes, estudo dos ataques e suas motivações e também das vulnerabilidades exploradas.

Este trabalho faz uso de máquinas virtuais. A principal razão de se utilizá-las é a possibilidade de se recorrer a uma solução de baixo custo para fornecer confiabilidade, isolamento e escalabilidade ao sistema de defesa que se quer projetar. Além disso, aproveita-se ao máximo a capacidade de processamento da máquina real explorando a ociosidade da máquina física e também a praticidade para testes, já que é possível salvar o estado do sistema e voltar a este ponto de forma rápida. Devido a esta característica, também podemos restabelecer a rede imediatamente caso esta seja comprometida por algum tipo de ataque.

1.2 OBJETIVOS

Este trabalho tem como objetivo implementar uma *honeynet* que utilize máquinas virtuais para executar serviços, ou simulações dos mesmos, como Servidor de Nomes (DNS), servidor *web*, servidor de *Email*, servidor *File Transfer Protocol* (FTP), etc. Será configurada uma máquina virtual para monitoramento, que conterá ferramentas apropriadas para monitorar e detectar ações maliciosas de intrusos. Com este sistema, será possível mapear cada tentativa de ataque à *Honeynet* e tomar as medidas preventivas ou de contra-ataque cabíveis. Para concluir com sucesso o objetivo geral supracitado serão alcançados os seguintes objetivos específicos:

- Estudar o assunto máquinas virtuais e virtualização abordando tipos, formas, aplicações, benefícios e dificuldades ao se adotar máquinas virtuais;
- Estudar *honeynet*;
- Instalar e configurar monitor de máquinas virtuais (VMware Server), máquinas virtuais, serviços nas máquinas virtuais;
- Estudar segurança na rede e principais ataques;
- Definir e configurar a topologia da *Honeynet*;
- Estudar *Intrusion Detection System* (IDS) e *Intrusion Protection System* (IPS);
- Instalar e Configurar IDS / IPS na máquina virtual que gerenciam a rede de máquinas virtuais que contém os serviços instalados;
- Simular ataque à rede de máquinas e mostrar a *Honeynet* atuando;
- Definir políticas de contra-ataque;
- Produzir imagens das máquinas virtuais com os serviços e IDS / IPS instalados.

2 MÁQUINAS VIRTUAIS E VIRTUALIZAÇÃO

Máquina virtual entende-se como uma duplicata eficiente e isolada de uma máquina real. Trata-se de um ambiente criado por um *Hypervisor* ou monitor de máquinas virtuais. Este último trata-se de um software de virtualização que tem como principais atribuições o escalonamento das máquinas virtuais e o controle de recursos compartilhados por estas. Mas antes de aprofundar o assunto, vamos a alguns conceitos fundamentais. (LAUREANO, 2007)

2.1 CONCEITOS

Virtualização permite que em uma mesma máquina sejam executados simultaneamente vários ambientes distintos e isolados com compartilhamento de recursos ou não. Neste contexto é importante definir o que são instruções privilegiadas e não privilegiadas relativamente a uma arquitetura x86. As instruções não-privilegiadas são aquelas que não modificam a alocação ou o estado de recursos compartilhados por vários processos simultâneos, tais como processadores, memória principal e registradores especiais. Em oposição a essas instruções, temos as instruções privilegiadas, que podem alterar o estado e a alocação desses recursos.

Um computador pode operar em dois modos distintos, o modo usuário e o modo supervisor. O modo usuário, também chamado de espaço de aplicação, é o modo no qual as aplicações normalmente são executadas. Neste modo, não é possível executar as instruções privilegiadas, que são restritas ao modo de supervisor. O modo de supervisor tem o controle total sobre o processador, podendo executar todas as instruções do processador em questão, tanto as não-privilegiadas quanto as privilegiadas. O sistema operacional é executado neste último modo. Antes de o sistema operacional passar o controle da CPU para uma aplicação do usuário, o bit de controle de modo é configurado para o modo de usuário.

Em um ambiente virtualizado tem-se os conceitos de sistema operacional hospedeiro ou *Host System* e sistema operacional visitante ou *Guest System*. No

primeiro, o sistema operacional executa diretamente sobre o hardware físico. Trata-se do SO nativo. Este executa um *software* de virtualização que por sua vez gera um *hardware* virtual. O segundo refere-se ao sistema operacional que é executado sobre o hardware virtualizado. Uma máquina real pode contar com apenas um *Host System* sendo executado por vez. No entanto, podem ser executados diversos *Guest Systems* simultaneamente.

O Monitor de Máquinas Virtuais (*Virtual Machine Monitor*) ou *Hypervisor* é um componente de software que hospeda as máquinas virtuais. De maneira específica, é ele o responsável pela virtualização e controle dos recursos compartilhados pelas máquinas virtuais, tais como, processadores, dispositivos de entrada e saída, memória. Também é função dele escalonar qual máquina virtual vai executar a cada momento, semelhante ao escalonador de processos de um sistema operacional.

O *Hypervisor* é executado no modo supervisor, no entanto as máquinas virtuais são executadas em modo usuário. Quando as máquinas virtuais tentam executar uma instrução privilegiada, é gerada uma interrupção e o Monitor de Máquinas Virtuais se encarrega de emular a execução da instrução. (LAUREANO, 2007)

2.2 VANTAGENS

A razão das máquinas virtuais serem amplamente utilizadas hoje está no fato de as vantagens mostrarem-se mais relevantes do que as desvantagens na maior parte dos casos. Abaixo seguem as principais vantagens encontradas.

2.2.1 Isolamento

Usando máquinas virtuais, pode ser definido qual é o melhor ambiente para executar cada serviço, com diferentes requisitos de segurança, ferramentas diferentes e o sistema operacional mais adequado. A vulnerabilidade de um serviço não prejudica os demais já que cada máquina virtual é isolada das outras. (LAUREANO, 2007)

2.2.2 Flexibilidade

A utilização de máquinas virtuais proporciona flexibilidade à medida que facilita a instalação e manutenção de servidores virtuais, os quais têm sido cada vez mais utilizados em ambientes de produção. Dessa forma, o trabalho dos administradores e gestores de TI torna-se muito mais simples, rápido e eficaz. (LAUREANO, 2007)

2.2.3 Custo

A redução de custos é evidente à adesão das máquinas virtuais.

> *"Analistas da indústria revelam que entre 60% e 80% dos departamentos de TI estão engajados em projetos de consolidação de servidores em máquinas virtuais"* (WATERS, 2007)

O motivo é simples: com a redução de servidores reais, automaticamente economiza-se a energia não só gasta por estes servidores, mas também pelo sistema de refrigeração para manter esta grande quantidade de equipamentos funcionando em perfeito estado.

Outro aspecto relevante a ser destacado: o mau uso de grande quantidade de servidores reais para utilização de aplicações independentes tais como servidores Proxy, de banco de dados, de e-mail, etc.

> *"Em média, os servidores reais utilizam só 5% a 10% da sua capacidade, segundo estimativa da empresa de software de Virtualização VMWare"* (STRATTUS SOFTWARE, 2007)

Pode-se também citar mais aspectos, como por exemplo: com a crescente adesão de outros SO no mercado, empresas tendem a desenvolver seus aplicativos para rodar em diversas plataformas, aumentando sua competitividade. Sem a utilização da virtualização estas empresas deveriam ter diversas máquinas (não-virtuais) preparadas para esses testes, o que envolveria um grande custo.

2.2.4 Adaptações às diferentes cargas de trabalho

Variações na carga de trabalho podem ser tratadas utilizando-se ferramentas autônomas que podem realocar recursos de uma máquina virtual para a outra. (LAUREANO, 2007)

2.2.5 Balanceamento de carga

Toda a máquina virtual está encapsulada no *Hypervisor*. Sendo assim é relativamente fácil trocar a máquina virtual de plataforma, a fim de aumentar o seu desempenho. (LAUREANO, 2007)

2.2.6 Suporte a aplicações legadas

É possível manter um sistema operacional antigo sendo executado em uma máquina virtual. Com isso é possível conciliar sistemas legados com os mais atuais. A virtualização pode ser útil para aplicações que são executadas em hardware legado que está sujeito a falhas e tem altos custos de manutenção. Com a virtualização desse hardware, é possível executar essas aplicações em equipamentos mais novos, com custo de manutenção mais baixo e maior confiabilidade.

Dentre todos os aspectos vantajosos citados até o momento, é relevante acrescentar a economia de espaço físico. Além disso, se algum dos sistemas virtuais falharem, por qualquer que seja o motivo, poderá ser reiniciado sem que este afete o funcionamento das outras máquinas virtuais. (LAUREANO, 2007)

2.3 DESVANTAGENS

Algumas características da virtualização impõem vantagens e desvantagens. Por exemplo, a flexibilidade citada como uma vantagem pode também acarretar num ponto negativo. E se o computador que abriga todos os servidores que rodam em máquinas virtuais parar de funcionar? Todos os aplicativos que nele estão sendo executados irão parar também. Isso deve ser levado em conta no processo de decisão na implantação dessa tecnologia.

As principais desvantagens encontram-se explicadas a seguir.

2.3.1 Gerenciamento

Os ambientes virtuais necessitam ser instanciados, monitorados, configurados e salvos. Existem produtos que fornecem essas soluções, mas esse é o campo no qual estão os maiores investimentos na área de virtualização, justamente por se tratar de um dos maiores contratempos na implementação da virtualização. (LAUREANO, 2007)

2.3.2 Desempenho

A introdução de uma camada extra de software entre o sistema operacional e o hardware, o *Hypervisor*, gera-se um custo de processamento superior ao que se teria sem a virtualização.

Além disso, quando se tem um servidor destinado a fazer uma tarefa específica e este tem seus recursos de hardware compartilhados entre algumas máquinas virtuais, eventualmente este servidor deixará de ter o rendimento antes obtido como servidor dedicado.

Outro ponto é que não é simples dizer quantas máquinas virtuais podem ser executadas por processador sem que haja o prejuízo da qualidade de serviço. (LAUREANO, 2007)

2.4 FORMAS DE VIRTUALIZAÇÃO

Para entender como um ambiente virtualizado funciona, é fundamental conhecer as principais categorias de virtualização existentes.

2.4.1 Virtualização Total

A virtualização total tem por objetivo fornecer ao sistema operacional visitante uma réplica do hardware subjacente. Dessa forma, o sistema operacional visitante é executado sem modificações sobre o *Hypervisor*, o que traz alguns inconvenientes. O primeiro é que o número de dispositivos a serem suportados pelo sistema operacional hospedeiro é extremamente elevado. Para resolver esse contratempo, as implementações da virtualização total usam dispositivos genéricos, que funcionam bem para a maioria dos dispositivos disponíveis, mas não garantem o uso da totalidade de sua capacidade.

Outro inconveniente da virtualização total é o fato de o sistema operacional visitante não ter conhecimento de que está sendo executado sobre o *Hypervisor*, então as instruções privilegiadas executadas pelo sistema operacional visitante devem ser testadas pelo *Hypervisor* para que depois sejam executadas diretamente no hardware, ou executadas pelo *Hypervisor* e simulada a execução para o sistema visitante.

Por fim, o último inconveniente da virtualização total é o fato de ter que contornar alguns problemas gerados pela implementação dos sistemas operacionais, já que esses foram implementados para serem executados como instância única nas máquinas física, não disputando recursos com outros sistemas operacionais. Um exemplo desse último inconveniente é o uso de paginação na memória virtual, pois há disputa de recursos entre diversas instâncias de sistemas operacionais, o que acarreta em uma queda de desempenho. (WATERS, 2007)

2.4.2 Para-Virtualização

Nesse modelo, o sistema operacional é modificado para chamar o Hypervisor sempre que executar uma instrução que possa alterar o estado do sistema, uma instrução privilegiada. Isso acaba com a necessidade de o *Hypervisor* testar instrução por instrução, o que representa um ganho significativo de desempenho.

Outro ponto positivo da para-virtualização é que os dispositivos de hardware são acessados por drivers da própria máquina virtual, não necessitando mais do uso de drivers genéricos que inibiam o uso da capacidade total do dispositivo.

A principal desvantagem de se utilizar a para-virtualização é a limitação a poucos sistemas operacionais pois não se pode utilizar qualquer sistema operacional como *Guest System.* Este deve estar ciente de que está executando com a existência de um Hypervisor ao contrário do que ocorre na virtualização total onde o sistema operacional visitante executa sobre um hardware emulado. (MATTOS, 2009)

Outra desvantagem da para-virtualização é a obrigatoriedade de se utilizar um kernel modificado para o sistema operacional hospedeiro.

Tendo em vista as técnicas apresentadas, a decisão de qual a melhor técnica de virtualização para um dado ambiente está intimamente ligada a qual processador da máquina bem como se o processador possui ou não uma extensão no seu conjunto de instruções que suporte a virtualização.

3 ATAQUES COMUNS A REDES DE COMPUTADORES

Para que um sistema de segurança seja bem-sucedido, é necessário incluir a prevenção e a resistência ao maior número possível de ataques. No tocante a Redes de Computadores, o sistema deve estar sempre atualizado, prevenindo os mais novos tipos de ataques desenvolvidos para garantir o máximo de segurança. Entretanto é uma tarefa que exige um intenso trabalho da equipe desenvolvedora do sistema, uma vez que esta deve estar sempre a par das mais recentes ações de *hacking*[1] no mundo. Sendo assim, é conveniente que se tenha conhecimento dos ataques mais comuns na atualidade, já que a probabilidade do sistema ser vítima de um destes é bem maior do que ser vítima de um ataque recém-desenvolvido.

A seguir estão descritas as três etapas iniciais de um ataque genérico e por fim há uma abordagem do principal tipo de ataque da atualidade: Negação de Serviço.

3.1 ETAPAS DE UM ATAQUE

Em sentido geral, um ataque eficiente normalmente segue um padrão de procedimento. Por exemplo, um assalto a banco para ser bem-sucedido deve contar com planejamento, monitoramento das atividades diárias do banco, conhecimento de ações táticas para assalto, etc. A analogia serve para ataques a redes de computadores. A seguir seguem as etapas fundamentais de um ataque a uma máquina ou a um grupo de máquinas.

3.1.1 FOOTPRINTING – AQUISIÇÃO DE ALVOS

Trata-se do primeiro passo a ser tomado quando se deseja efetuar um ataque. Consiste na busca detalhada de informações sobre o alvo com o intuito de traçar um perfil contendo os mais diversos tipos de dados relevantes, como informações sobre domínio, serviços disponíveis, topologia da rede, sub-redes, sistema operacional da

[1] Métodos e técnicas utilizadas por *hackers.*

máquina-alvo, dentre outros. Uma vez feito o levantamento dessas informações, o atacante será capaz de identificar pontos vulneráveis e a partir disso planejar a estratégia do ataque. (McCLURE, 2009)

3.1.2 VARREDURA

Se o *footprinting* é o equivalente a buscar informações de localização de determinado lugar, a varredura é o equivalente a procurar nas paredes todas as portas e janelas deste lugar. Por meio do *footprinting* se obtém os endereços IP utilizados e por meio de varredura se determina quais sistemas estão ativos e alcançáveis a partir da Internet. (McCLURE, 2009)

3.1.3 ENUMERAÇÃO

Trata-se da identificação de contas de usuários válidas ou recursos compartilhados em rede mal protegidos. A enumeração envolve conexões ativas em sistemas e consultas dirigidas, ou seja, o nível de invasão é que diferencia aqui a enumeração da simples coleta de informação. Depois de se ter conhecido a existência de recursos e contas, é questão de tempo para que um invasor descubra senha ou identifique um ponto fraco num protocolo de compartilhamento. (McCLURE, 2009)

3.2 NEGAÇÃO DE SERVIÇO - DENIAL OF SERVICE (DoS)

Os ataques *Denial of Service* (DoS) têm o propósito de tornar os serviços de um sistema indisponíveis. Por exemplo, é possível tornar um servidor web indisponível enviando um grande volume de requisições para acesso às páginas hospedadas. Se o servidor não possuir nenhum tipo de filtro ou regra de *firewall* que limite o volume de páginas servidas a um único endereço, ele passará a simplesmente tentar responder a todas as requisições, o que irá saturar o servidor ou irá consumir todos

os seus recursos fazendo com que ele deixe de responder as requisições de usuários válidos.

Programas para ataques remotos de DoS surgiram em meados dos anos 90, causando problemas. Para usar estes programas, era necessária uma conta em um grande computador ou rede, para obter efeito máximo.

Como exemplo de técnica de ataque DoS pode-se citar o *Smurf* (ataque por reflexão). A técnica se baseia na utilização de servidores de difusão (*broadcast*) que são capazes de replicar uma mensagem para todas as máquinas presentes numa dada rede. O atacante envia um pedido *ping*[2] a um ou mais servidores *broadcast* modificando o endereço IP original (endereço do atacante) para o endereço da máquina vítima. Os servidores *broadcast* repassam o *ping* para uma dada rede de computadores. Os computadores desta rede respondem o *ping* para os servidores de *broadcast* que, por sua vez, repassam para a máquina vítima comprometendo-a por meio da quantidade elevada de pacotes a ela destinados.

Antes do fim da década de 90, os atacantes resolveram explorar a possibilidade de simplesmente enviar centenas de pacotes ao alvo, o que era possível se o atacante utilizasse uma conexão com velocidade muito mais alta que a vítima. Dessa forma ele impediria facilmente a conexão da vítima, que seria prejudicada a ponto de tornar-se inútil e com velocidade de resposta praticamente nula.

A partir de 1998, as velocidades de conexão já se tornavam menos heterogêneas e ataques *Smurf* eram fáceis de serem combatidos. Neste ponto surgiu o DDoS, com atacantes passando a tentar obter controle de máquinas alheias para conseguir inundar a conexão da vítima com tráfego suficiente. Ataques explorando vulnerabilidades de sistemas continuaram, e passou-se a combinar várias falhas DoS em uma única ferramenta, usando *shell scripts* em Unix.

Seguindo esta linha de raciocínio, os ataques *Distributed Denial of Service* (DDoS) consistem na combinação entre negação de serviço e computação distribuída.

Basicamente, os ataques DDoS são executados em 3 passos. O primeiro envolve a escolha da vítima e o mapeamento das suas vulnerabilidades. Caso sejam várias máquinas, é feita então uma lista com os IP das mesmas.

[2] É um comando que usa o protocolo ICMP para testar a conectividade entre máquinas de uma rede. Seu funcionamento consiste no envio de pacotes para o equipamento de destino e na "escuta" das respostas.

O próximo passo consiste na escolha dos agentes (máquinas “zumbi”) e mestres. Os agentes são as máquinas que irão iniciar o ataque contra as vítimas, enquanto os mestres são as máquinas que irão comandar os agentes e enviarão instruções para realização dos ataques como ilustrado na FIG 3.1. Uma vez que sejam escolhidas, as ferramentas DDoS devem ser instaladas para possibilitar a realização do ataque. Os mestres irão executar a aplicação cliente DDoS, que por sua vez irá controlar os agentes por meio do *daemon* DDoS que estará em execução esperando por instruções.

Por fim, no último passo o ataque é iniciado e o atacante determina por quanto tempo o ataque deve ocorrer contra as máquinas contidas na lista que contém os IP das vítimas.

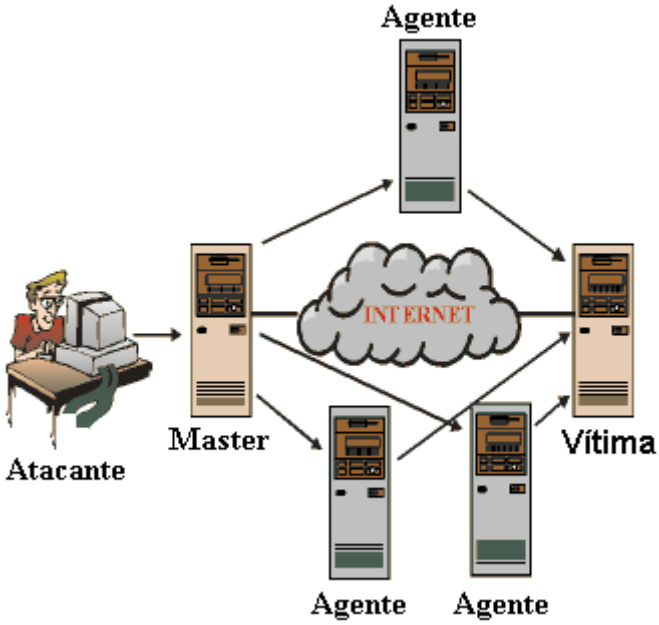


FIG. 3.1 – Ataque DDoS

Neste trabalho serão utilizadas algumas técnicas de ataques DoS para a execução de testes na *Honeynet*, a saber: ICMP *flood* e UDP *flood*.

3.3 FERRAMENTAS DE ATAQUE DDoS

Um bom *hacker*[3] costuma desenvolver seus próprios códigos de ataque. Entretanto muitos códigos prontos estão disponíveis e de fácil acesso no mundo da Internet. Como o objetivo deste trabalho não é desenvolver ataques novos, algumas ferramentas de ataques já consagrados foram utilizadas e estão descritas abaixo.

3.3.1 TRIN00

O Trin00 é uma ferramenta usada para lançar ataques DDoS do tipo UDP *flood.*

Uma rede Trin00 é composta por um número pequeno de *masters* e um grande número de agentes. O controle do *master* Trin00 é feito por meio de uma conexão TCP remota via porta 27665/tcp por uma máquina que se denomina atacante. Após se conectar ao *master*, o atacante deve fornecer uma senha que determina uma ação. Por exemplo, a senha "*betaalmostdone"* determina a interface para comunicação de comandos proveniente do *master* para os agentes realizarem o ataque ao *host* ou grupo de *hosts* vítimas*.*

A comunicação é feita via pacotes UDP na porta 27444/udp ou 31335/udp. Também há comunicação via pacotes TCP na porta 1524/tcp.

O *daemon* consiste no software executado pelo agente que permite que o mesmo seja controlado pelo *master*. Quando um *daemon* é inicializado num agente, ele anuncia a sua disponibilidade enviando uma mensagem ("*HELLO*") ao *master*, que por sua vez mantém uma lista dos IP das máquinas agentes ativas.

Tipicamente, a aplicação cliente que roda no *master* tem sido encontrada sob o nome de **master.c**, enquanto que os *daemons* do Trin00 instalados em máquinas comprometidas têm sido encontrados com uma variedade de nomes, dentre eles: **ns**, **http**, **rpc.Trin00**, **rpc.listen**, **trinix**, etc. Tanto o programa cliente (que roda no *master*) quanto o *daemon* (que roda no agente) podem ser inicializados sem privilégios de usuário *root.* (The DoS Project's "trinoo",2010)

[3] Indivíduos que elaboram e modificam software e hardware de computadores, seja desenvolvendo funcionalidades novas, seja adaptando as antigas.

3.3.2 STACHELDRAHT

Stacheldraht (em alemão arame farpado) é outra ferramenta usada para lançar ataques DDoS coordenados a uma ou mais máquinas vítimas, a partir de várias máquinas comprometidas que denominamos agentes. Como sua predecessora TFN (*Tribe Flood Network*), ela também é capaz de gerar UDP *flood*, TCP *flood*, ICMP *flood* e Smurf/fraggle.

Funcionalmente, o *Stacheldraht* combina basicamente características das ferramentas Trin00 e TFN, mas adiciona algumas características como criptografia para comunicação entre o atacante e o *master*. A idéia surgiu porque uma das deficiências encontradas na ferramenta TFN era que a conexão entre atacante e *master* era completamente desprotegida, sujeita a interceptações de ataques TCP conhecidos (*hijacking*, por exemplo). O *Stacheldraht* lida com este problema incluindo um utilitário "telnet criptografado" na distribuição do código que usa criptografia simétrica para proteger as informações que trafegam. Este utilitário se conecta em uma porta TCP, comumente na porta 16660/tcp.

Outra característica acrescentada é a atualização dos binários dos *daemons* instalados nos agentes. Essa atualização é realizada via serviço rpc (514/tcp).

Uma rede *Stacheldraht* é composta por um pequeno número de *masters* onde rodam os programas *handlers* (comumente encontrados sob o nome de **mserv**), e um grande número de agentes, onde rodam os processos *daemons* (comumente encontrados sob o nome de ***leaf*** ou ***td***). Todos eles devem ser executados com privilégios de *root.*

Diferentemente do que ocorre com o Trin00, que utiliza pacotes UDP na comunicação entre os *masters* e os agentes, e do TFN, que utiliza apenas pacotes ICMP, o *Stacheldraht* utiliza pacotes TCP cuja porta padrão é a 65000/tcp e ICMP do tipo ICMP_ECHOREPLY. (The "*stacheldraht*" distributed denial of service attack tool,2010)

3.4 FERRAMENTA PARA CONTRA-ATAQUE

A função de uma ferramenta de contra-ataque é tomar uma ou mais ações frente a um alerta gerado pelo IDS ou NIDS. Estas ações abrangem bloquear o IP do atacante, tirar a máquina vítima da rede temporariamente ou permanentemente, alterar o IP da vítima, entre outros.

Neste trabalho a ferramenta adotada foi o Guardian. A escolha deste aplicativo se justifica por ser simples, de fácil configuração e ser totalmente integrado ao Snort.

3.4.1 Guardian

Trata-se de um aplicativo de segurança que trabalha em conjunto com o Snort para atualizar o IPTables automaticamente, de acordo com as regras baseadas nos alertas gerados pelo Snort. Estas regras são descritas para bloquear todos os dados vindos do endereço IP da máquina que está realizando o ataque (máquina responsável por gerar o alerta no Snort). Também há possibilidade de configurar o Guardian para nunca bloquear endereços IP relevantes, como por exemplo servidores DNS, *gateways*, e outros, porque estas máquinas podem acabar gerando falso-positivos.

4 INTRUSION DETECTION SYSTEM

Um *Intrusion Detection System* (IDS) ou em português Sistema de Detecção de Intrusão é um software utilizado como retaguarda num sistema de segurança. Após todos os softwares e dispositivos tradicionais de segurança terem falhados, o IDS pode indicar uma possível brecha de alguma ação maliciosa contra o sistema. Por exemplo, imaginando uma rede onde exista um *firewall*. Este segue regras de maneira a permitir que determinado tráfego de entrada ou saída da rede sem se preocupar com o conteúdo dos pacotes o que não isola a possibilidade de ocorrerem ataques. Desta maneira a utilização de um IDS é muito importante.

4.1 TIPOS DE IDS

Existem basicamente duas formas de se implementar um IDS que deve ser escolhida de acordo com o que se deseja proteger: *Host-Based Intrusion Detection System* e *Network-Based Intrusion Detection System*.

4.1.1 Host-Based Intrusion Detection System (HIDS)

O *Host Intrusion Detection System* é um tipo de IDS baseado em *host* e tem como finalidade controlar e prevenir anomalias no computador. Em outras palavras, ele detecta utilização indevida e excessiva de memória, processos estranhos com comportamento suspeito, conexões de rede suspeitas, utilização da CPU, utilização das chamadas de sistema, utilização do disco em termos de leitura, gravação, criação e exclusão de arquivos e pastas. Normalmente um servidor de administração central do HIDS recebe atualizações à medida que cada *host* detecta algum tipo de mudança. A FIG. 4.1 ilustra o posicionamento de um HIDS.

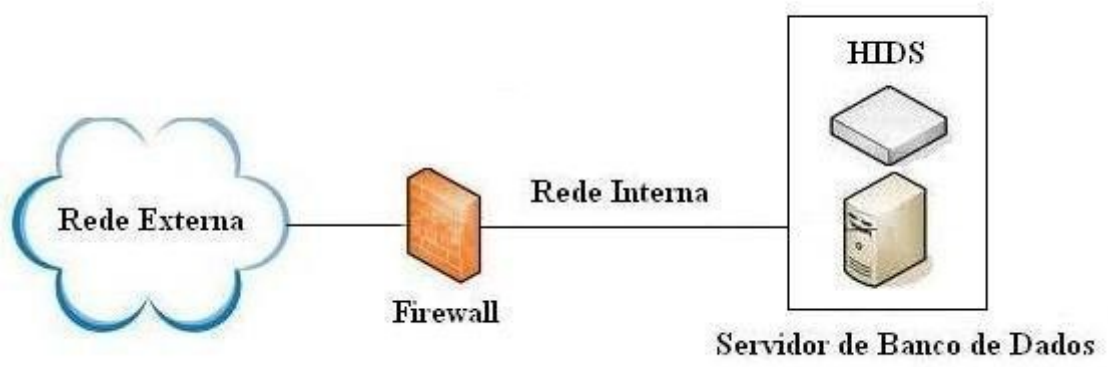


FIG. 4.1 – Posicionamento de um HIDS

4.1.2 Network-Based Intrusion Detection System (NIDS)

Ferramentas baseadas em rede que são capazes de detectar ataques e resolver problemas de uma rede, isto é, aplica o conceito de IDS para redes como um todo. Em vez de se posicionar numa máquina específica para analisar os pacotes que por ela trafegam (função do HIDS) ele monitora o tráfego na rede. O *Network-Based Intrusion Detection System* (NIDS) poderá possuir diversos sistemas de *log*[4] espalhados pela rede monitorando todos os segmentos. Um exemplo de NIDS é o software Snort.

Neste trabalho o interesse maior é a atuação do IDS para redes. Isso justifica o termo NIDS ser empregado de maneira mais recorrente, entretanto não se descarta a utilização do termo IDS que pode ser utilizado num sentido mais abrangente. A FIG. 4.2 ilustra uma configuração onde existe uma *DeMilitarized Zone*[5] (DMZ) ou Zona Desmilitarizada e um NIDS monitorando a rede interna.

[4] Processo de registro de eventos relevantes num sistema computacional.
[5] Pequena rede situada entre uma rede confiável e uma não confiável cuja função é manter todos os serviços que possuem acesso externo separados da rede interna, limitando assim o potencial dano à rede local em caso de comprometimento de algum destes serviços de acesso externo por um invasor.

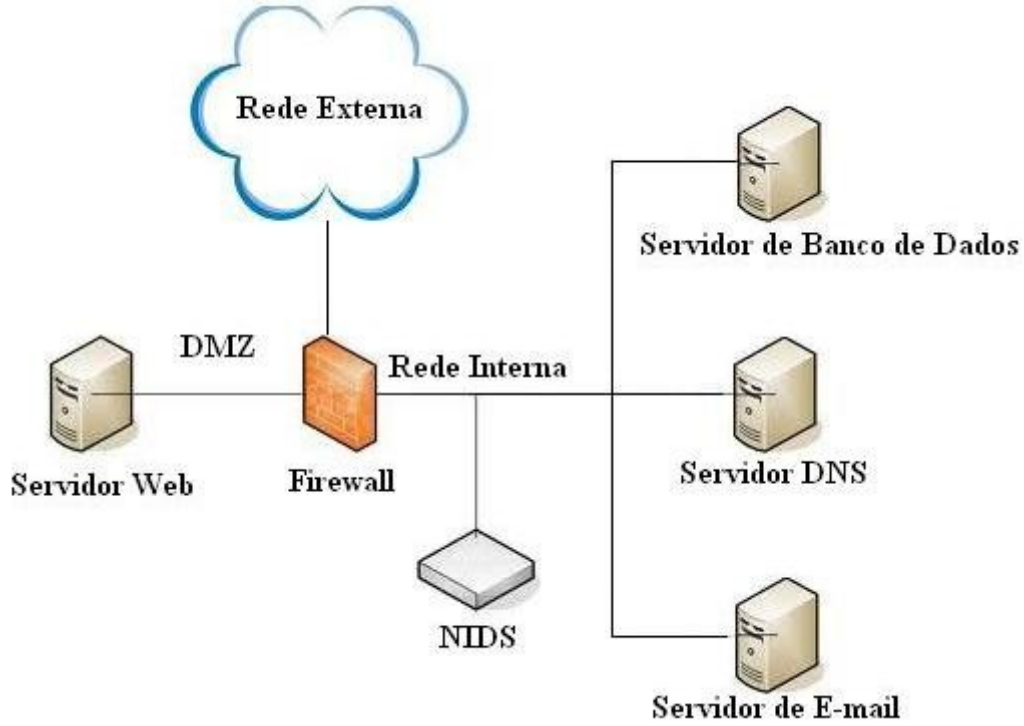


FIG. 4.2 – Posicionamento de um NIDS

A detecção de comportamento malicioso num sistema pode ser realizada de duas maneiras: detecção baseada em assinatura e baseada em anomalia.

4.2 DETECÇÃO BASEADA EM ASSINATURA

A detecção baseada em assinatura tem por finalidade identificar ataques conhecidos analisando trechos de informações. Assinatura é um trecho de informação que um pacote deve conter. Por exemplo, um pacote destinado a um servidor HTTP/WEB contendo um pedido para o diretório ***/*admin**. Isso pode ser uma tentativa de identificar um possível diretório de administração do site. Com um NIDS monitorando o tráfego, provavelmente a invasão seria acusada. Entretanto, deve-se frisar que esse tipo de detecção só funciona com assinaturas conhecidas, ou seja, se uma assinatura foi alterada propositadamente para evitar detecção então o NIDS não acusará o problema. Cabe ao administrador da rede analisar os *logs* registrados e avaliar a situação.

As vantagens deste tipo de detecção resumem-se a velocidade e a identificação da maioria dos ataques, tendo em vista que normalmente são feitos por *script kiddies*[6].

As desvantagens são: necessidade de atualizações freqüentes do banco de dados de assinaturas do NIDS, já que sem as assinaturas atualizadas, a maioria dos ataques não será acusada e a possibilidade de haver falso-positivos[7] aumenta gerando muito lixo nos arquivos de *log*.

Um software popular que age por detecção baseado em assinatura é o Snort. Este foi o software escolhido para monitoramento neste trabalho. Tal escolha ocorreu pela existência de uma sólida comunidade de suporte que mantém e atualiza o software e suas regras periodicamente.

4.3 DETECÇÃO BASEADA EM ANOMALIA

Este tipo de detecção não precisa de assinaturas ou regras específicas. Ele baseia-se em estatística. Funciona da seguinte maneira: estipula-se de antemão o perfil diário do fluxo na rede, ou seja, espera-se que o tráfego seja mais intenso em determinadas horas do dia e mais escassos em outros momentos diários. Estes dados estatísticos definem um modelo padrão de comportamento de rede. Se alguma coisa fugir muito a este padrão, o NIDS irá indicar uma anomalia. Assim como o tipo de detecção baseada em assinatura, este tipo de detecção não é perfeito já que podem ocorrer falso-positivos. Por exemplo, imagine que o comportamento diário do fluxo numa organização indique que durante o horário de almoço o fluxo na rede esteja no patamar mais baixo em comparação aos outros momentos do dia. Imagine que num determinado dia os funcionários combinaram de utilizar o horário do almoço para utilizar a rede para alguma finalidade diversa como baixar músicas ou algo similar. Neste caso o NIDS apontará que houve um ataque naquele intervalo o que não necessariamente ocorreu.

[6] Nível mais básico daqueles que executam *hacking*. Iniciante em *hacking* que utiliza códigos prontos de ataques.
[7] Ocorre quando a ferramenta classifica uma ação como uma possível intrusão, que na verdade trata-se de uma ação legítima.

5 HONEYPOT e HONEYNET

5.1 HONEYPOT

O termo *honeypot* surgiu no início dos anos 90 quando foram registradas as primeiras caças a “cybercriminosos”. *Honeypot* (pote de mel em inglês) é uma palavra usada por aqueles que trabalham em segurança da informação para se referir a um *host* que funciona como uma armadilha.

Para compreender o funcionamento de um *honeypot* e seus tipos é necessário entender os tipos de serviços existentes.

5.1.1 TIPOS DE SERVIÇOS

São classificados de acordo com o nível de interação, ou seja, o nível de atividade que o serviço permite ao atacante. São eles:

5.1.1.1 Serviços de Alta Interação

Ao criar uma armadilha o ideal é adicionar serviços que possam atrair invasores. Estes serviços podem ser aqueles mais comuns encontrados em redes regulares para se aproximar da realidade. É possível até mesmo disponibilizar ao invasor o controle de acesso total do *host* fornecendo o *shell* de comando de um sistema operacional. Isso pode se tornar perigoso já que um invasor teria a sua disposição os recursos de um ou mais sistemas operacionais para atacar outras redes. O administrador da rede deve se preocupar em analisar a finalidade real de se utilizar o que se denominam serviços de alta interação. Em suma, são serviços reais como aqueles disponibilizados em ambientes de produção.

A vantagem de recorrer aos serviços de alta interação é que dificilmente um *honeypot* que utiliza este tipo de serviço será percebido como uma armadilha pelo invasor justamente por se tratar de serviços reais.

O principal exemplo de *honeypot* que utiliza serviço de alta interação é o *Mantrap*.

5.1.1.2 Serviços de Baixa Interação

São mais simples que os de alta interação, uma vez que todos os serviços são simulados. O invasor nunca obterá acesso ao sistema real por meio de serviços de baixa interação, exatamente porque são versões simuladas. O emprego deste tipo de honeypot é vantajoso do ponto de vista da segurança, no entanto qualquer invasor com uma experiência a mais em *hacking* descobrirá rapidamente que tudo se trata na verdade de um ambiente simulado. Por outro lado, só o fato do invasor investigar o sistema já permitirá registrar sua tentativa de ataque o que já possibilita um resultado que pode ser interessante.

Exemplos de *honeypots* que utilizam serviços de baixa interação: *Specter*, *Honeyd* e *KFSensor*.

5.1.2 TIPOS DE *HONEYPOT*

De acordo com a finalidade que se destina, podemos classificar os honeypots em dois tipos:

5.1.2.1 *Honeypots* de Pesquisa

O objetivo deste *honeypot* é ser atacado. Dessa forma, toda a ação do invasor pode seja analisada, isto é, todos os comandos, aplicativos maliciosos porventura utilizados e vulnerabilidades exploradas serão devidamente registrados para posterior análise. Há várias possibilidades que ilustram a utilização desde tipo de

honeypot. É possível por exemplo montar um *honeypot* com a finalidade de definir estatísticas de ataque e a partir daí desenvolver novas soluções para proteção de sistemas reais, o que seria útil para qualquer empresa ou organização que possua parte da sua receita ligada à área de TI.

5.1.2.2 *Honeypots* de Produção

Sua intenção é basicamente detectar intrusos na rede. Após a detecção, cabe ao administrador tomar as providências em resposta às ações suspeitas.

Nesse caso não se deseja fornecer acesso real ao invasor. Deseja-se que ele se mantenha afastado do sistema legítimo. Cabe também ao administrador decidir se usará serviços de baixa ou alta interação em conjunto com este *honeypot*.

5.2 HONEYNET

Trata-se de um conjunto de *honeypots* em rede, ou seja, leva o conceito de *honeypot* a um sentido mais amplo. Tudo que se aplica a *honeypot* também se aplica a *honeynet* só que agora levando em consideração se tratar de redes.

Um dos objetivos de se ter uma *honeynet* seria em segurança de redes. Dentro de um projeto de segurança pode-se tê-la como um meio auxiliar. A idéia é utilizá-la como uma rede vulnerável que fica em evidência de modo a desviar a atenção de possíveis invasores, camuflando a rede real que deve estar devidamente protegida.

Outro objetivo de se ter uma *honeynet* é acadêmico, utilizada para estudos tanto de ataques já conhecidos e consagrados quanto de novos. Percebe-se que estes objetivos são abstraídos de *honeypot* de pesquisa e produção.

5.2.1 TIPOS DE HONEYNET

Existem dois tipos de classificação para uma *honeynet* de acordo com a finalidade do projeto de segurança que se deseja adotar na rede. De acordo com o tipo escolhido alguns quesitos como custo e eficiência são alterados.

5.2.1.1 *Honeynet* Real

Neste tipo de *honeynet* todos os dispositivos que a compõem são físicos. Tais dispositivos são *honeypots*, mecanismos de contenção, mecanismos de alerta e coleta de informações.

Um bom exemplo de configuração para uma *honeynet* Real apresenta os seguintes dispositivos:

- Diversos computadores, cada um contendo um *honeypot* com um sistema operacional, aplicações e serviços reais instalados;
- Um computador com *firewall* instalado;
- Um computador com IDS instalado, atuando como mecanismo de geração de alertas e de coleta de dados;
- Um computador atuando como repositório dos dados coletados;
- Hub, switch ou roteador, se for o caso, para fornecer a infra-estrutura de rede.

As principais vantagens deste tipo se encontra na tolerância à falhas já que se trata de um ambiente distribuído. Além disso, por interagir com ambientes reais, acaba sendo uma melhor armadilha por se aproximar mais de uma rede normal.

As principais desvantagens são a manutenção que é mais difícil e trabalhosa, necessidade de mais espaço físico para os equipamentos e custo total que tende a ser mais elevado que num ambiente virtual. (CERT.BR, 2009)

5.2.1.2 *Honeynet* Virtual

Baseia-se na idéia de ter todos os componentes de uma *honeynet* implementados em um número reduzido de dispositivos físicos. Geralmente utiliza-se um único computador com um software de virtualização que permite executar os diversos dispositivos e aplicativos que estão presentes numa *honeynet*.

As *Honeynets* Virtuais ainda são subdivididas em duas categorias: de Auto-Contenção e Híbridas. Na primeira todos os mecanismos (contenção, captura e coleta de dados, geração de alertas e *honeypots*) estão em um único computador e são implementados por meio de um software de virtualização. Na segunda, os mecanismos de contenção, captura e coleta de dados e geração de alertas são executados em dispositivos distintos e os *honeypots* são executados em um único computador que contém um software de virtualização.

As vantagens são a manutenção mais simples, necessidade de menor espaço físico para os equipamentos e custo total que tende a ser mais baixo se comparado com uma *Honeynet* Real e também a possibilidade de se recuperar rapidamente de uma falha ou ataque já que por se tratar de uma rede virtual existe a possibilidade de restabelecimento a um estado salvo anterior.

As principais desvantagens trata-se da baixa tolerância a falhas (muitos componentes concentrados em um único ponto), o software de virtualização amarrar a escolha do hardware e sistemas operacionais utilizados e, no caso de uma *honeynet* mal-implementada ou mal-estruturada, a possibilidade de o atacante obter acesso a outras partes do sistema pois todos os dispositivos acabam compartilhando os recursos por se encontrarem na mesma máquina (no caso da categoria de Auto-Contenção). Além disso, sempre existe a chance de o atacante descobrir que está interagindo com um ambiente virtual. (CERT.BR, 2009)

Nesta obra, tem-se por finalidade implementar uma *honeynet* Virtual de Auto-Contenção.

6 IMPLEMENTAÇÃO DA HONEYNET

Para a instalação do gerenciador de máquinas virtuais foi escolhido o VMware Server para Linux. Foram instaladas e configuradas máquinas virtuais com o sistema operacional Ubuntu Server, uma distribuição Linux leve e eficiente para a finalidade almejada. Os serviços comumente encontrados em redes corporativas foram instalados e configurados, tais como servidor de SSH, servidor de e-mail *Postfix*, servidor web, servidor DNS e servidor FTP. Os tutoriais de instalação e configuração podem ser encontrados na seção “Anexos” desta monografia.

A ferramenta Snort, um NIDS amplamente conhecido e utilizado, foi instalada na máquina virtual de monitoramento. Como já foi dito, pode-se justificar o emprego deste software para o trabalho apontando algumas de suas características. Inicialmente, se fez necessário uma ferramenta para monitoramento do tráfego nas redes, ou seja, era necessária a utilização de um NIDS. Outro fato crucial em relação a escolha é que se trata de um software livre que dispõe de toda uma comunidade de desenvolvedores apoiando e contribuindo para que esta ferramenta se mantenha atualizada e disponível para utilização. Por último, trata-se de uma ferramenta leve e eficiente para verificar anomalias dentro de toda a rede.

Convém ressaltar que o funcionamento do Snort inspeciona tráfego baseando-se em assinatura, o que se espera ser suficiente para detectar ataques a *Honeynet*. Ele é ideal para monitorar redes TCP/IP pequenas em que se pode detectar uma grande variedade de tráfego suspeito dado que é possível avaliar tanto o cabeçalho quanto o conteúdo dos pacotes trafegados. A análise pode ser realizada por meio dos alertas em tempo real ou registrá-los de acordo com configuração pré-estabelecida para análise posterior. Este registro pode ser configurado para armazenar pacotes decodificáveis a uma estrutura de diretório baseada em IP, ou no formato binário do *tcpdump* em um único arquivo. Desta maneira o administrador da rede dispõe de informações para tomar decisões de segurança.

A título de ilustração pode-se citar outros softwares como o *RealSecure*, *Shadow*, *NetRanger*, *PortSentry*, NFR, etc. que se comportam de maneira semelhante mas sempre apresentam alguma característica indesejável para o presente trabalho. Tomando como exemplo o *NetRanger*, este software tem como

desvantagem ser proprietário da Cisco (há um custo para aquisição) além do projeto ter sido descontinuado o que aumenta o risco do suporte se tornar escasso.

6.1 TOPOLOGIA DA *HONEYNET*

A estrutura foi montada utilizando uma máquina real (CPU Core2Duo E7200 2.53GHz, 1GB RAM, 145GB HD) que abriga as máquinas virtuais. Esta máquina será referenciada no texto pelo codinome “Real”. A topologia da rede definida é apresentada na FIG. 6.1 e está atualmente montada no laboratório de redes da SE/8.

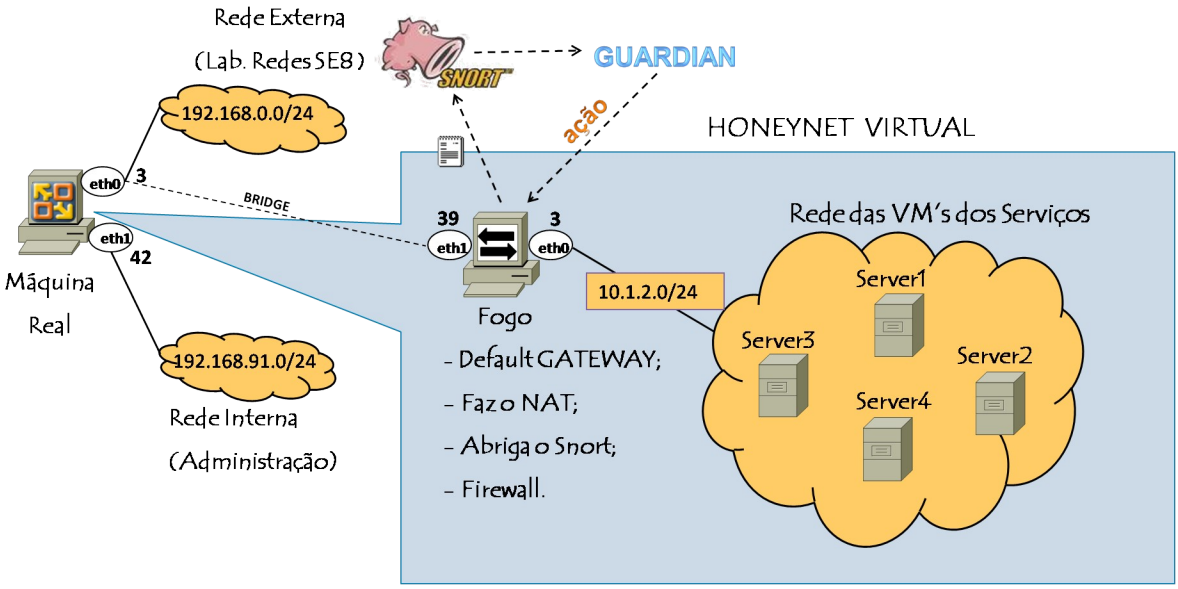


FIG. 6.1 – Topologia da *Honeynet*

Para definir a topologia acima, foi necessário estabelecer uma interface (interna) exclusiva para que o administrador possa operar remotamente a máquina Real que contém o VMware e por conseqüência a *Honeynet*. É interessante que esta interface seja alcançada somente por uma rede interna onde se encontra o administrador. Definiu-se então uma outra interface (externa) na máquina Real que provê acesso público à *Honeynet*. Vale ressaltar que, por meio de regras de IPTables, esta interface impede o acesso externo ao IP alocado à máquina Real. Deste modo evita-se que ações maliciosas direcionadas a administração do VMware ocorram,

restringindo-se a administração apenas a operadores autorizados por meio da interface interna.

O desafio agora passou a ser definir estas interfaces. Primeiramente imaginou-se utilizar apenas uma placa de rede na máquina real para as duas interfaces: eth0 e eth0:1, sendo esta última uma interface virtual. Alguns testes foram realizados para analisar a segurança desta solução. Fez-se passar tráfego numa das interfaces (pacote ICMP *ping*) e na outra interface utilizou-se o capturador de pacotes *tcpdump*. Constatou-se que esta última conseguia “escutar” o tráfego que passava pela outra. Realizou-se o mesmo experimento invertendo as interfaces e obteve-se o mesmo resultado. Assim sendo, esta solução foi considerada ineficiente. A próxima sugestão seria utilizar duas placas de rede: uma para cada interface. Os mesmos testes foram realizados neste novo contexto e o resultado se mostrou satisfatório quanto a segurança, uma vez que uma interface não conseguia “escutar” o tráfego que passava na outra interface.

Desta forma ficou definido para a máquina Real duas interfaces: eth0 e eth1. A interface eth1 está conectada a rede interna e tem finalidade de prover manutenção e administração da rede. A interface eth0 conecta-se diretamente à *Honeynet* por meio da máquina de monitoramento denominada Fogo (default gateway das máquinas da *Honeynet*), na qual foram instalados o Snort para realizar o monitoramento da rede e o Guardian para atuar de acordo com os alertas gerados pelo Snort. Mais detalhes sobre a atuação do Guardian são mostrados na seção 6.3.

Um ponto a ser considerado é que todo o tráfego da *Honeynet* deve obrigatoriamente passar pela máquina Fogo para que o Snort possa atuar.

O primeiro passo para cumprir esta tarefa era obrigar toda requisição à rede externa realizada por qualquer uma das máquinas Servers da *Honeynet* passar pela máquina Fogo através da interface eth0 10.1.2.3 (IP estático). O teste inicial foi realizado com o pacote ICMP *ping* encaminhado para a rede externa partindo de uma máquina Server. Para obter sucesso foi necessário configurar alguns arquivos da máquina Fogo e inserir algumas regras de IPTABLES nesta máquina.

Para confirmar que o tráfego estava fluindo pela interface eth0 10.1.2.3 e também por eth1 192.168.91.39 na máquina Fogo utilizou-se mais uma vez o *tcpdump*.

O segundo passo era fazer com que as máquinas “Server” fossem transparentes à rede externa por meio de endereços IP estáticos na máquina Fogo da seguinte maneira: criaram-se as interfaces eth0:1 , eth0:2, eth0:3, eth0:4 às quais encaminham as requisições respectivamente às máquinas Server1, Server2, Server3 e Server4. A implementação destas operações foram realizadas com comandos de IPTables na máquina Fogo.

Desta maneira todo usuário de fora da rede enxerga as máquinas (caso realize uma varredura usando o comando *nmap*[8]) com os seguintes IP:

Server1 com IP 192.168.0.34

Server2 com IP 192.168.0.35

Server3 com IP 192.168.0.36

Server4 com IP 192.168.0.37

Toda esta configuração que permite o encaminhamento de requisições passando pela máquina Fogo é conhecida como NAT 1:1, onde NAT significa *Network Address Translate* e 1:1 porque o pacote é redirecionado de um endereço IP para outro endereço IP univocamente. Por exemplo, pacotes endereçados para a máquina 192.168.0.34 são encaminhados para o endereço 10.1.2.4 bem como pacotes para 192.168.0.35 são encaminhados para 10.1.2.5 e assim por diante. Na FIG. 6.2 visualiza-se a topologia da *Honeynet* completa com o conceito NAT 1:1.

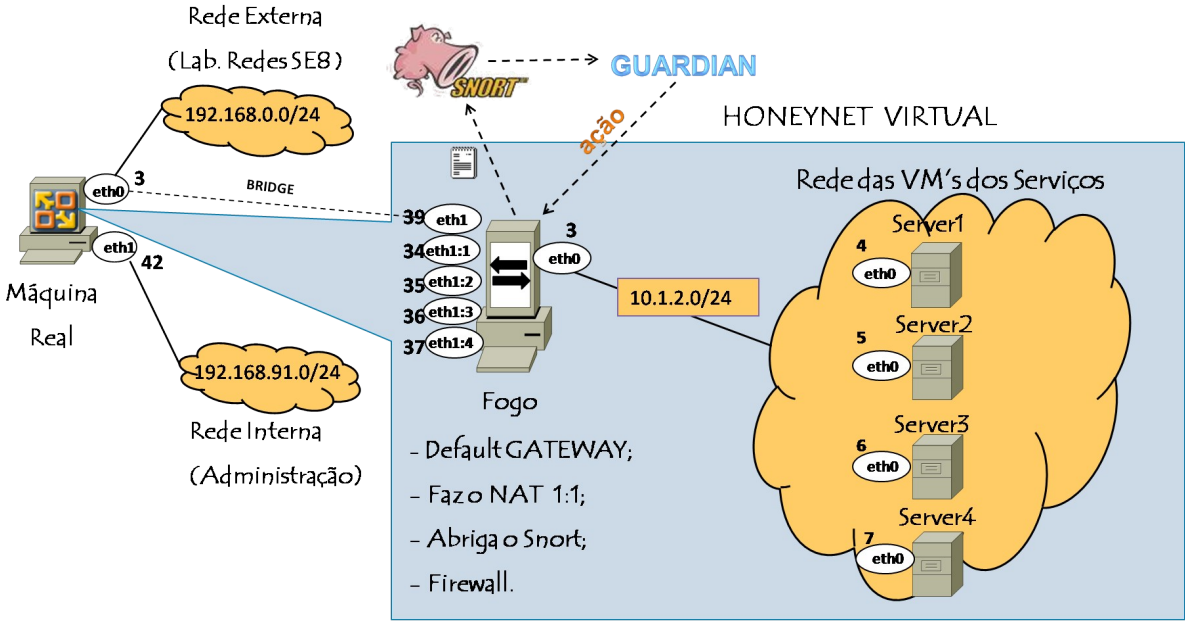


FIG. 6.2 – Topologia da *Honeynet* completa

[8] Software livre utilizado para varredura e mapeamento de portas.

Nas tabelas 6.1, 6.2 e 6.3 abaixo estão expostas as configurações de rede das máquinas que compõem o sistema que inclui a *Honeynet*.

TAB. 6.1 – Configuração da máquina Real

Nome	Aplicativo	Interfaces
Real	VMware Server	eth1 static ip 192.168.0.3/24
		eth0 static ip 192.168.91.42/24

TAB. 6.2 – Configuração da máquina Fogo

Nome	Aplicativo	Interfaces
Fogo (VM)	Snort , Guardian	eth1 static ip 192.168.0.39/24
		eth1:1 static ip 192.168.0.34/24
		eth1:2 static ip 192.168.0.35/24
		eth1:3 static ip 192.168.0.36/24
		eth1:4 static ip 192.168.0.37/24
		eth0 static ip 10.1.2.3/24

TAB. 6.3 – Configuração das máquinas Server

Nome	Aplicativo	Interfaces
Server1 (VM)	DNS	eth0 static ip 10.1.2.4/24
Server2 (VM)	Postfix (SMTP)	eth0 static ip 10.1.2.5/24
Server3 (VM)	FTP	eth0 static ip 10.1.2.6/24
Server4 (VM)	LAMP, Tomcat, Postgres, SSL-TLS	eth0 static ip 10.1.2.7/24

6.2 ATAQUE À REDE DE MÁQUINAS VIRTUAIS

Com o intuito de testar o funcionamento do Snort, submeteu-se a *Honeynet* a ataques DDoS do tipo ICMP Flood e UDP Flood. Realizou-se alguns cenários de testes para ataques. Para todos os casos, o Snort está monitorando a máquina Fogo, que tem sua interface eth1 configurada com endereço IP 192.168.0.39 (Rede Externa) e interface eth0 10.1.2.3 (*Honeynet*). Todas as detecções foram baseadas nas assinaturas dos ataques.

A Tab. 6.4 contém os endereços IP dos elementos componentes do ataque para cada cenário, a saber: atacante, *master*, agente e vítima. Tais cenários foram concebidos de modo a permitir que fossem capturados todos os tipos possíveis de pacotes de comunicação entre os componentes dos ataques. Dessa forma, buscou-se validar o funcionamento da *Honeynet* incluindo tanto a possibilidade de os

ataques estarem sendo originados por agentes localizados fora da rede quanto por agentes dentro da mesma.

TAB. 6.4 – Cenários dos testes

Cenário	Atacante	Master	Agente(s)	Vítima(s)
1	192.168.0.67	192.168.7.4	192.168.0.8 192.168.5.5	192.168.0.37
2	10.1.2.6	10.1.2.4	10.1.2.5	10.1.2.7
3	192.168.0.67	10.1.2.4	10.1.2.5	10.1.2.7
4	192.168.0.67	192.168.0.39	192.168.7.4 192.168.0.8	192.168.0.34 192.168.0.37

As seções 6.2.1 a 6.2.4 contém detalhes a respeito dos resultados dos testes para cada cenário experimentado. Há ilustrações que tratam dos alertas ocasionados pelos testes na rede monitorada, e estas ilustrações em geral seguem o padrão exemplificado pela FIG. 6.2. Como mostra a figura, um alerta contém uma mensagem descritiva do ataque, a prioridade do alerta, a data e a hora em que o mesmo ocorreu, os IP e porta de origem e destino do pacote, o protocolo (TCP ou UDP), a identificação do alerta, assim como outros detalhes do cabeçalho IP.

```
[**] MensagemDescritivaDoAtaque [**]
[Prioridade: n]
Mes/Dia-HH:MM:SS.N  IP_Origem:Porta -> IP_Destino:Porta
PROTOCOLO  TTL:__  TOS: ____  ID: ID_alerta  IpLen:__  DgmLen:__
NSeq: ___   NAck: _____ NWin: ____ TcpLen: __
```

FIG. 6.2 – Modelo de Registro Snort

6.2.1 Primeiro cenário

Neste primeiro teste foi utilizada a ferramenta Trin00 e com a mesma foi executado um ataque UDP *flood* contra a vítima. Neste cenário a vítima estava sendo monitorada, uma vez que era a única pertencente à *Honeynet*.

A FIG. 6.3 representa um alerta gerado pela detecção do ataque do tipo UDP Flood efetuado por um dos agentes.

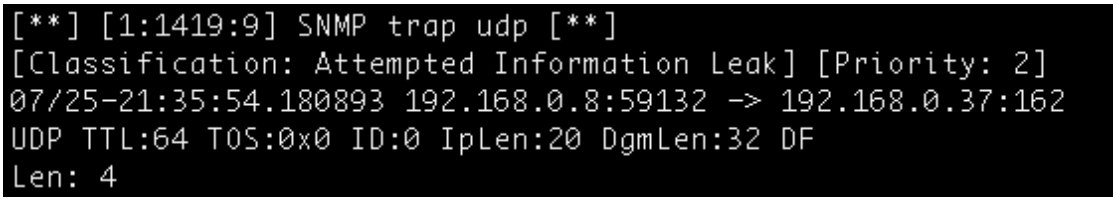


FIG. 6.3 – Registro Snort. Causa: UDP Flood

A FIG. 6.4 representa um ataque mal interpretado ocasionado pela forma de detecção nativa do Snort (detecção por assinatura). O alerta gerado acusa comunicação entre *master* e *daemon*, quando na verdade a figura representa um alerta gerado por um pacote que se originou num *daemon* com direção à vítima. Logo, este faz parte do *flood* de pacotes UDP direcionado à vitima. Tal interpretação ocorreu porque a detecção está ocorrendo a partir da análise das portas utilizadas para comunicação entre os componentes do Trin00. Uma destas portas, a 27444, é utilizada para comunicação entre *master* e *daemon*, e coincidentemente um dos pacotes do *flood* (que acontece em portas randômicas) utlizou esta porta, sendo portanto identificado de forma equivocada pelo Snort.

FIG. 6.4 – Registro Snort – falso-positivo. Causa: ataque com Trin00

Neste cenário foi possível verificar que a estrutura da rede pode influenciar de forma significativa na realização de um ataque. A quantidade de alertas gerados pelos os pacotes provenientes do agente 192.168.0.8 destinados a máquina vítima é muito maior que o registro dos pacotes provenientes da máquina 192.168.5.5. Isto se deve ao fato de haver limite de tráfego em alguns roteadores quando se trata da comunicação entre redes diferentes (192.168.5.0/24 – agente e 192.168.0.0/24 – vítima).

6.2.2 Segundo cenário

Neste cenário, todos os componentes do ataque estão dentro da *Honeynet*. Novamente foi efetuado um ataque de UDP *flood* a partir da ferramenta Trin00.

A FIG. 6.5 representa um alerta gerado pela detecção do ataque do tipo UDP Flood efetuado por um dos agentes.

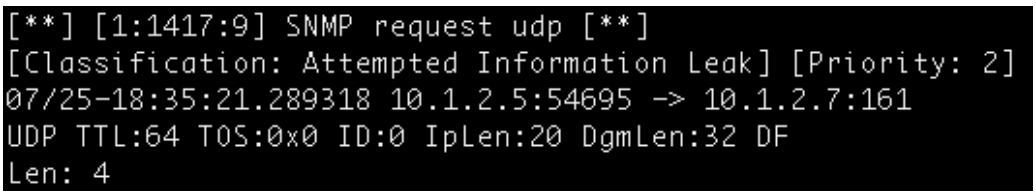


FIG. 6.5 – Registro Snort. Causa: UDP Flood

A FIG. 6.6 representa um alerta gerado pela identificação de um pacote de comunicação entre o atacante e o *master*. Tal detecção ocorreu por meio da análise das portas utilizadas e do conteúdo do pacote, que se mostrou idêntico à senha que o atacante utiliza para se autenticar ao master quando da execução do comando *mdie*, que tem a função de “desligar” todos os agentes.

FIG. 6.6 – Registro Snort. Causa: ataque com Trin00

6.2.3 Terceiro cenário

Neste cenário o atacante encontra-se fora da *Honeynet*, ao contrário de todos os outros componentes do ataque. Novamente utilizou-se o Trin00 para efetuar o ataque UDP *flood* contra a vítima.

A FIG. 6.7 representa a detecção de um pacote no qual um agente (executando o *daemon*) anuncia sua presença ao *master* por meio do envio da mensagem "*HELLO*".

FIG. 6.7 – Registro Snort. Causa: ataque com Trin00

A FIG. 6.8 representa a detecção de um pacote de comunicação entre o atacante e o *master*.

FIG. 6.8 – Registro Snort. Causa: ataque com Trin00

A FIG. 6.9 representa o mesmo tipo de pacote que a FIG. 6.6, mas dessa vez o atacante encontra-se fora da *Honeynet*, como mostra corretamente o alerta gerado. É importante notar que, exatamente por esse motivo é que o endereço IP do *master* exibido é o externo (192.168.0.34), enquanto que no caso anterior o alerta mostrou-o como sendo o interno (10.1.2.4).

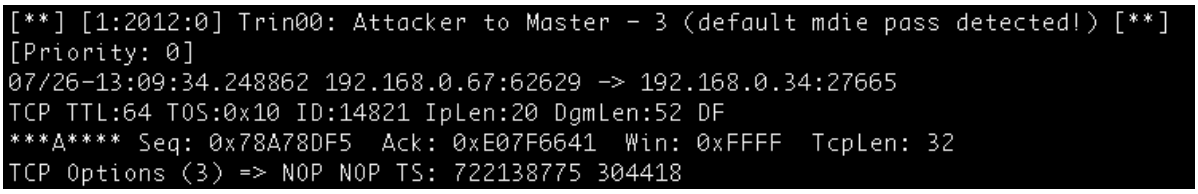


FIG. 6.9 – Registro Snort. Causa: ataque com Trin00

6.2.5 Quarto cenário

Neste cenário foi empregada a ferramenta Stacheldraht para efetuar um ICMP *flood* contra uma máquina da *Honeynet*. O *master* (no Stacheldraht, o *master* também é conhecido como *handler*) também pertencia à *Honeynet*, mas os agentes eram externos.

A FIG. 6.10 ilustra o alerta gerado por um pacote ICMP fora do normal, por ocasião do ICMP *flood*. O mesmo foi originado por um dos agentes e foi destinado à vítima.

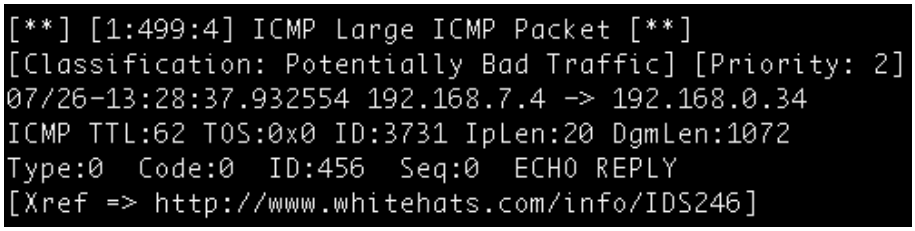


FIG. 6.10 – Registro Snort. Causa: ICMP *flood*

A FIG. 6.11 representa o alerta gerado pela identificação de um pacote de comunicação entre um agente e o *master*.

FIG. 6.11 – Registro Snort. Causa: ataque com Stacheldraht

As FIG. 6.12 e FIG. 6.13 ilustram a captura de pacotes referentes a ataques ICMP *flood* à segunda vítima. É interessante notar que tais figuras ilustram a utilização de mais uma técnica de ataque em conjunto com o ICMP *flood*: o IP *spoofing*. A técnica de IP *spoofing* consiste na modificação de um pacote com o objetivo de inviabilizar a descoberta do IP da máquina que o originou. Dessa forma a reação de bloquear o IP de origem do pacote malicioso não tem efeito, uma vez que

nenhum dos IP exibidos na figura representa corretamente o IP dos agentes, mas sim endereços IP forjados.

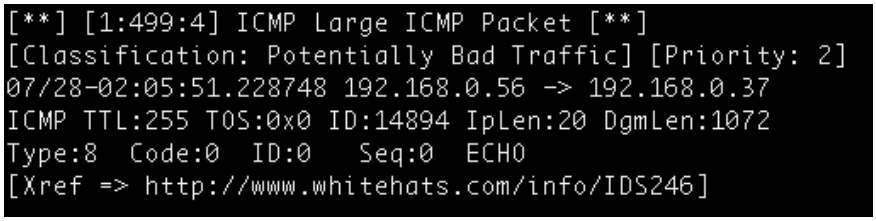


FIG. 6.12 – Registro Snort. Causa: ICMP *flood* com IP *spoofing*

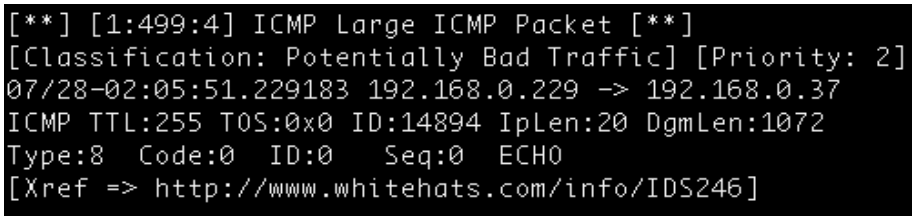


FIG. 6.13 – Registro Snort. Causa: ICMP *flood* com IP *spoofing*

6.3 POLÍTICAS DE CONTRA-ATAQUE

Definir uma política de contra-ataque não necessariamente significa que se deve revidar o ataque sofrido com um outro ataque. Como já mencionado, entende-se aqui por contra-ataque: rastreamento de ações suspeitas (essenciais numa prática de perícia por exemplo), bloqueio da máquina atacante, reversão ao ponto anterior ao ataque ou até mesmo interrupção do acesso à *Honeynet*. Outras medidas de resposta a incidentes de rede podem ser concebidas de acordo com as peculiaridades do sistema a ser protegido e recursos disponíveis.

Neste trabalho a ação adotada foi bloquear o IP da máquina que gera o ataque. Esta tarefa pode ser executada por um aplicativo que trabalha em sintonia com a ferramenta de monitoramento. Desta maneira, evita-se a necessidade de o administrador da rede estar presente para tomar uma atitude no momento em que um ataque ocorrer. Entretanto deve-se tomar cuidado com a configuração adotada já que falso-positivos podem causar bloqueio em IP válidos na rede. O aplicativo Guardian foi utilizado para desempenhar o papel acima.

A FIG. 6.14 representa o resultado da atuação do Guardian por ocasião dos testes realizados no cenário 1. É importante ressaltar que a medida de reação adotada pode ter uma ação não-desejada caso a ferramenta não seja bem configurada. A primeira linha da FIG. 6.14 representa uma tentativa de ataque proveniente do IP 10.1.2.3, enquanto que a segunda linha executa o script “guardian_block.sh”, que bloqueia tal IP. Recordando a FIG. 6.2, o host 10.1.2.3 é aquele que direciona todos os pacotes que chegam na *Honeynet* às respectivas máquinas de destino, sejam estes maliciosos ou não. Dessa forma, posteriormente foi necessário configurar o arquivo “guardian.ignore” para não bloquear a máquina Fogo. Já a terceira e a quarta linhas da FIG. 6.14 representam o bloqueio correto de um agente do cenário 1 (192.167.0.8) quando da detecção de um pacote malicioso destinado à vítima. Nas duas últimas linhas há o registro de que os bloqueios expiraram de acordo com o período de tempo estipulado no arquivo de configuração.

```
Sun Jul 25 21:35:32 2010: 10.1.2.3      [1:2013:0] Trin00: Master to Daemon - 1
Running '/usr/local/bin/guardian_block.sh 10.1.2.3 eth1'
Sun Jul 25 21:35:57 2010: 192.168.0.8   [1:1419:9] SNMP trap udp
Running '/usr/local/bin/guardian_block.sh 192.168.0.8 eth1'
Sun Jul 25 21:36:14 2010: 192.168.0.8   [1:2015:0] Trin00: Daemon to Master - 1
Sun Jul 25 21:36:15 2010: 10.1.2.3      [1:2013:0] Trin00: Master to Daemon - 1
Sun Jul 25 21:36:17 2010: 10.1.2.3      [1:2339:2] TFTP NULL command attempt
Sun Jul 25 21:36:17 2010: 192.168.0.8   [1:1417:9] SNMP request udp
Sun Jul 25 21:36:18 2010: 10.1.2.3      [1:1417:9] SNMP request udp
Sun Jul 25 21:36:18 2010: 192.168.0.8   [1:1417:9] SNMP request udp
Sun Jul 25 21:36:25 2010: 10.1.2.3      [1:2339:2] TFTP NULL command attempt
Sun Jul 25 21:36:28 2010: 10.1.2.3      [1:2015:0] Trin00: Daemon to Master - 1
Sun Jul 25 21:36:48 2010: 192.168.0.8   [1:2339:2] TFTP NULL command attempt
Sun Jul 25 21:36:49 2010: 192.168.0.8   [1:2013:0] Trin00: Master to Daemon - 1
expiring block of 10.1.2.3
expiring block of 192.168.0.8
```

FIG. 6.14 – Arquivo guardian.log referente ao cenário 1

7 CONCLUSÃO

O trabalho realizado tornou prático conhecimentos teóricos nas áreas de redes de computadores, instalação e configuração de aplicativos específicos para redes, servidores, virtualização, ataques a redes e, envolvendo tudo isto, familiarização com Software Livre. Além disso travou-se contato com um assunto muito em evidência e de essencial importância: segurança em redes.

Num primeiro momento houve a necessidade de se buscar os conhecimentos teóricos que serviram de base para o desenvolvimento do produto.

Na etapa de implementação definiu-se primeiramente como seria a topologia da Honeynet. Com o aplicativo VMWare Server foi possível utilizar máquinas virtuais para a rede. Foram instalados e configurados servidores na *Honeynet*. Ainda na etapa de implementação foi necessário configurar regras de IPTables e instalar o Snort, software de monitoramento para a *Honeynet*. Vários conceitos de redes foram abordados tanto a nível físico, ao configurar switches no laboratório de redes da SE/8, quanto a nível lógico, ao configurar um NAT 1:1 na máquina Fogo por exemplo.

Na última etapa foram utilizadas ferramentas para ataques à rede no intuito de testar o funcionamento do Snort. Ataques do tipo DDoS foram postos em prática. Percebe-se que é limitada a técnica de detecção por assinatura em se tratando de ataques DoS em geral. Geralmente por utilizar IP *spoofing* e porta randômica nos ataques UDP *flooding,* torna-se difícil a detecção por assinatura. Sugere-se adicionalmente a utilização de um software que se baseia em anomalia para auxiliar na detecção deste tipo de ataque. Entretanto, a comunicação entre Daemon e Master foi detectada com sucesso pelo Snort. Foi instalado o Guardian, software que atua de acordo com os alertas gerados pela ferramenta de detecção. Quando um ataque era registrado com sucesso, o Guardian bloqueava o IP origem dos dados.

Com todas estas atividades, cumpriu-se os objetivos principais propostos. A implementação de uma *honeynet* em máquinas virtuais é viável porque o ambiente virtual representa fidedignamente o ambiente real além de propiciar os benefícios já citados, tornando bastante conveniente o emprego da utilização de virtualização. Quanto ao software de monitoramento, constatou-se neste trabalho que para

ataques do tipo DoS se faz necessário uma ferramenta de detecção baseada em anomalia.

Uma idéia bem coerente para trabalhos futuros seria conciliar dois tipos de NIDS na *Honeynet:* um baseado em assinatura e outro baseado em anomalia, para maximizar o número de ameaças capturadas. Indica-se também o desenvolvimento de uma interface para configuração e gerência da *Honeynet*.

Muitos aprendizados foram colhidos neste trabalho. Destaca-se o grande contato com Linux que é de fundamental importância para um administrador de redes e desenvolvedor de sistemas. Destaca-se também o tema Segurança da Informação que possui grande valor no mercado e também fundamental interesse para instituições militares.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ASSUNÇÃO, Marcos Flávio Araújo; **Honeypots e Honeynets: aprenda a detectar e enganar os invasores**. Florianópolis.Visual Books, Agosto de 2009.

CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL, CERT.BR. **Honeypots e honeynets: Definições e Aplicações**.Disponível em < http://www.cert.br/docs/whitepapers/honeypots-honeynets/>. Acesso em: 30 set 2009.

CISCO US. Disponível em: < http://www.cisco.com/en/US> Acesso em: abril 2010.

Instalando e configurando o Guardian. Disponível em: <http://www.vivaolinux.com.br/artigo/IDS-com-Snort-+-Guardian-+-Debian-Lenny?pagina=8>. Acesso em: Julho de 2010.

LAUREANO, Marcos; **Máquinas Virtuais e Emuladores. Conceitos, Técnicas e Aplicações**. São Paulo. Novatec Editora, 2007.

Malware FAQ: Analysis on DDOS tool Stacheldraht v1.666. Disponível em: <http://www.sans.org/security-resources/malwarefaq/stacheldraht.php>. Acesso em: Julho de 2010.

MATTOS, Diogo Menezes Ferrazani.**Virtualização: VMWare e Xen.** GTA/POLI/UFRJ. Disponível em < http://www.gta.ufrj.br/>. Acesso em: 30 set 2009.

McCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hacking Exposed: Network Security Secrets and Solutions**. Sixth Edition, McGraw-Hill, 2009.

RealSecure. Disponível em: < http://www.iss.net/> Acesso em: abril 2010.

Shadow. Disponível em: < http://www.nswc.navy.mil/ISSEC/CID/> Acesso em: abril 2010.

Snort Rules for Trin00. Disponível em: <http://www.ussrback.com/docs/distributed/snort-ids.trinoo.txt>. Acesso em: Julho de 2010.

SNORT Brasil. Disponível em: < http://www.Snort.com.br> Acesso em: fevereiro. 2010.

STRATTUS SOFTWARE. Virtualização: rode vários sistemas operacionais na mesma máquina. São Paulo, 2007. Disponível em: <http://www.strattus.com.br/noticias.asp?CodNoticia=14>. Acesso em: maio 2010.

The DoS Project's "trinoo" distributed denial of service attack tool. Disponível em: <http://staff.washington.edu/dittrich/misc/trinoo.analysis>. Acesso em: Julho de 2010.

The "stacheldraht" distributed denial of service attack tool. Disponível em:<http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>. Acesso em: Julho de 2010.

Trinoo (ns.c/master.c). Disponível em: <http://www.merit.edu/mail.archives/nanog/2000-02/msg00366.html>. Acesso em: Julho de 2010.

WATERS, John K. **ABC da Virtualização**. Disponível em: <http://cio.uol.com.br/tecnologia/2007/08/14/idgnoticia.2007-08-4.5515750576/>. Acesso em: agosto de 2009.

9 ANEXOS

9.1 COMANDOS UTILIZADOS NA CONFIGURAÇÃO DA MÁQUINA FOGO

Todo o tráfego da *honeynet* deve obrigatoriamente passar pela máquina Fogo para que o SNORT possa atuar.

O primeiro passo para cumprir esta tarefa era obrigar toda requisição à rede externa realizada por qualquer uma das máquinas Servers da *honeynet* passar pela máquina Fogo através da interface eth0 10.1.2.3 (IP estático).

Os comandos estão descritos a seguir:

sudo IPTables -A FORWARD -i eth1 -o eth0 -s [ip rede]/24 -m conntrack --ctstate NEW -j ACCEPT

sudo IPTables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

sudo IPTables -A POSTROUTING -t nat -j MASQUERADE

Os três comandos acima em conjunto permitem a realização do Network Address Translate (NAT) para os pacotes que trafegam pela máquina Fogo.

Editando o arquivo /proc/sys/net/ipv4/ip_forward com o comando

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"

e adicionando as linhas abaixo ao arquivo /etc/sysctl.conf

net.ipv4.conf.default.forwarding=1

net.ipv4.conf.all.forwarding=1

é possível estabelecer a passagem dos pacotes de uma interface a outra.

O segundo passo era fazer com que as máquinas Servers fossem transparentes à rede externa por meio de IPs estáticos na máquina Fogo da seguinte maneira: criaram-se as interfaces eth0:1, eth0:2, eth0:3, eth0:4 às quais encaminham as requisições respectivamente às máquinas Server1, Server2, Server3 e Server4.

Desejando que toda requisição encaminhada a interface eth1:1 da máquina Fogo seja reencaminhada ao ip 10.1.2.4 (Server1):

IPTables -t nat -A PREROUTING -i eth1 -d [ip da interface eth1:1] -j DNAT --to 10.1.2.4

Desejando que toda requisição encaminhada a interface eth1:2 da máquina Fogo seja reencaminhada ao ip 10.1.2.5 (Server2):

IPTables -t nat -A PREROUTING -i eth1 -d [ip da interface eth1:1] -j DNAT --to 10.1.2.5

Desejando que toda requisição encaminhada a interface eth1:3 da máquina Fogo seja reencaminhada ao ip 10.1.2.6 (Server3):

IPTables -t nat -A PREROUTING -i eth1 -d [ip da interface eth1:1] -j DNAT --to 10.1.2.6

Desejando que toda requisição encaminhada a interface eth1:4 da máquina Fogo seja reencaminhada ao ip 10.1.2.7 (Server4):

IPTables -t nat -A PREROUTING -i eth1 -d [ip da interface eth1:1] -j DNAT --to 10.1.2.7

9.2 TUTORIAL DE INSTALAÇÃO DO VMWARE SERVER

Primeiramente deve-se instalar Ubuntu Server 32 bits na máquina real. Atentar para a conta e senha de administrador no momento da instalação. Será necessário mais adiante para configurar o VMware Server.

Baixar o VMware server (compatível com o Ubuntu server 32 bits). Site fonte:

http://downloads.vmware.com/d/

Supondo a utilização do Ubuntu Server, antes de iniciar a instalação:

#sudo su

Supondo que o arquivo VMware-server-2.0.1-156745.i386.tar encontra-se num periférico por exemplo, um pen drive.

Criar diretório para realizar o mount do drive:

#mkdir /tmp/pen

Realizar o mount do pen drive no novo diretório:

#mount /dev/sdb1 -t auto /tmp/pen

Crie um diretório em algum lugar no seu computador a seu critério (aqui vamos chamar de VMware):

#mkdir /VMware

Acessar os arquivos:

#cd /tmp/pen

Estando em /temp/pen , copiar para o diretório criado anteriormente :

#cp VMware-server-2.0.1-156745.i386.tar.gz /vmware

Estando dentro de /VMware descompacte o arquivo tar, será criada a pasta VMware-server-distrib :

#tar –zxvf VMware-server-2.0.1-156745.i386.tar.gz

Desmonte o pendrive:

#umount /dev/sdb1

Atualize para a última versão:

#apt-get update

Recomenda-se também:

#apt-get upgrade

Para saber a versão que se está utilizando

#uname –r

Agora se utiliza a versão nos próximos passos, neste exemplo temos que a versão é 2.6.28-11-server32 Instale os pacotes abaixo:

#apt-get install psmisc libxrender1 libxt6 build-essential linux-headers-2.6.28-11-server openssh-server

Para facilitar a vida e evitar problemas na instalação vamos renomear:

#mv linux-headers-2.6.28-11-server/ linux/

Vá na pasta VMware-server-distrib criada assim que houve o descompactamento do arquivo tar e execute o arquivo VMware-install.pl :

#./VMware-install

Durante a instalação, no momento que ele pedir a porta para conexão remota utilize uma conhecida default do tipo 80. É possível ocorrer problemas caso deixar a porta sugerida.

Durante a instalação, no momento em que é pedido “the current administrative user for VMware server is” “Would you like to specify a different administrator ? ” responda Yes e indique como administrador aquele criado no momento da instalação do Ubuntu Server.

Serial deve ser adquirido no site:

https://www.vmware.com/tryvmware/?p=server20&lp=1

Para adquirir o serial deve-se registrar em “Registrer for your FREE download”

Chaves utilizadas no presente trabalho:

VMware Server 2 for Windows AA10J-PHEDK-1DLGJ-4223M

VMware Server 2 for Linux AA421-PR96Q-0F2F2-4C5UE

Caso ocorra problema na instalação relativo ao vmmon, deve-se adicionar as linhas indicadas abaixo no arquivo /etc/init.d/vmware.

Provavelmente se está utilizando devfs. Vá em /etc/init.d/vmware

Logo abaixo das linhas que começam com ‘#’ insira as linhas indicadas abaixo. Utilize um editor de texto. Um editor recomendado é o nano. Exemplo de utilização: nano [arquivo]. Caso não tenha o nano instalado, execute:

apt-get install nano

Linhas:

[code]mknod -m 0600 /dev/vmmon c 10 165

mknod -m 0600 /dev/vmnet0 c 119 0

mknod -m 0600 /dev/vmnet1 c 119 1
mknod -m 0600 /dev/vmnet8 c 119 8[/code]

A instalação foi concluída. Agora as máquinas virtuais podem ser instaladas no VMware Server. O VMware Server funciona com browser gráfico. Como o Ubuntu Server trabalha sem interface gráfica, recomenda-se que o acesso ao VMware Server seja realizado por meio do browser de outra máquina real que esteja na mesma rede e consiga enxergar o Ubuntu Server com o VMware Server recém instalado.

Procedimento: descubra o ip da máquina instalado com o Ubuntu Server. Estando na própria máquina que contém o Ubuntu Server, utilize o comando ifconfig para esta tarefa. Agora por intermédio da máquina real (que consegue enxergar a máquina com o Ubuntu Server instalado) entre com o endereço de ip da máquina que contém o Ubuntu Server no browser. Exemplo:

https://[ip]/ui/#

Será requisitado um login e senha. A entrada correta será o login e senha configurados lá na instalação do Ubuntu Server.

Próximo passo é criar uma máquina virtual (*Create new Virtual Machine*). Assim que a máquina virtual for criada, vá em Console. Neste momento será requisitado a instalação do plug-in. Clique no link para a instalação. O normal é que a instalação proceda sem problemas em qualquer plataforma. A versão do Mozilla Firefox a qual o plug-in funciona é a 3.5.7.

9.3 TUTORIAL DE INSTALAÇÃO DO POSTFIX

O pacote *postfix* pode ser instalado através do comando abaixo:

#sudo apt-get install postfix

Perguntas durante a instalação:

-Função do servidor de e-mails.Opções:

a) Internet Site, cria um servidor, que envia e recebe os e-mails diretamente

b) Internet with smarthost, seu servidor recebe mensagens

c) Satellite System, servidor envia através de outra máquina e não recebe mensagens

d) Local Only, troca de email apenas com usuários logados.

-Nome da conta de usuário que se deseja enviar e receber emails;

-Domínio;

Se houve domínio registrado, pode ser utilizado, senão pode-se utilizar um domínio de teste. Este domínio será utilizado como endereço na hora do envio. Ex: Dominio: minha.maquina; Usuário: eu; Endereço de email: eu@minha.maquina.

Para criar usuários:

#sudo adduser aluno

Envio de emails:

O envio é realizado com comandos de texto. O servidor pode ser acessado via telnet pela porta 25. Tendo como exemplo o ip do servidor na rede interna: 192.168.1.40

telnet 192.168.1.40 25

Após o comando, o usuário estará obtendo acesso ao servidor SMTP. Abaixo temos um exemplo de envio de um email.

Trying 192.168.1.40...
Connected to 192.168.1.40.
Escape character is '^]'.
220 minha.maquina ESMTP Postfix (Ubuntu)
HELO smtp.eu.com
250 minha.maquina
MAIL From: eu@minha.maquina
250 Ok
RCPT to: joao@minha.maquina
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Testando!

.
250 Ok: queued as 0E92F195DB
QUIT
221 Bye
Connection closed by foreign host.

As linhas em negrito são os comandos executados no terminal, seguidos pelas respostas do servidor. Segue uma breve descrição desses comandos:

HELO: Inicia-se o processo de envio de email se identificando
MAIL from: Especifica o emissor;
RCPT to: Especifica o receptor;
DATA: Conteudo de texto do email.
Obs: “.” Após o conteúdo de texto determina o fim da mensagem.
QUIT: Fecha conexão.

Para que a mensagem seja visualizada no modo texto, pode-se utilizar o pacote MUTT:
#sudo apt-get install mutt
O usuário receberá a seguinte mensagem ao se logar da próxima vez:
You have new mail
Com o comando mutt é possível visualizar os emails.

Arquivo de configuração do Postfix trata-se do arquivo main.cf e se encontra em /etc/postfix/
Exemplo de configuração de main.cf:

myhostname = minha.maquina
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = minha.maquina, localhost.minha.maquina, localhost
relayhost =

mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128

mailbox_size_limit = 51200000

recipient_delimiter = +

inet_interfaces = all

Principais itens:

Myhostname: nome completo do servidor;

Mydestination: indica quais nomes e domínios serão considerados endereços locais pelo servidor;

Mynetworks: especifica os endereços ou faixas de endereços a partir de onde o servidor aceitará o envio de mensagens;

Mailbox_size_limit: Tamanho da caixa do usuário.

9.4 TUTORIAL DE INSTALAÇÃO DE FTP

Utilizando-se o modo GUI:

#sudo apt-get install proftpd gproftpd

Inicie o GUI e configure o apserver;

Nenhum suporte é oferecido para essa ferramenta.

9.5 TUTORIAL DE INSTALAÇÃO DO LAMP PARA UBUNTU

9.5.1 Instalação do Apache

Abra o Terminal (Applications > Accessories > Terminal).

Entre com a seguinte linha de comando no terminal:

#sudo apt-get install apache2

O Terminal irá pedir sua senha, forneça e conclua a instalação.

Testando o Apache

Para garantir que tudo foi instalado corretamente, procede-se a um teste.

Abra qualquer navegador web e entre com o seguinte código na barra de endereços:

http://localhost/

Você deverá ver uma pasta intitulada *apache2-default/*. Abra-a e veja a mensagem *“It works!”*.

9.5.2 Instalação do PHP

Novamente abra o Terminal (Applications > Accessories > Terminal).h

Entre com a seguinte linha de comando no terminal:

#sudo apt-get install php5 php5-common libapache2-mod-php5 php5-gd php5-dev curl php5-curl php-pear php5-mysql

Para que o PHP funcione e seja compatível com o Apache, nós devemos reiniciá-lo. Entre com o seguinte código no Terminal:

#sudo /etc/init.d/apache2 restart

Testando o PHP

Para garantir que não há problemas com o PHP, procede-se a um teste rápido.

No Terminal entre com o seguinte código:

#sudo gedit /var/www/testphp.php

Isso irá abrir um arquivo chamado *phptest.php*.

Entre com a seguinte linha dentro do arquivo de teste:

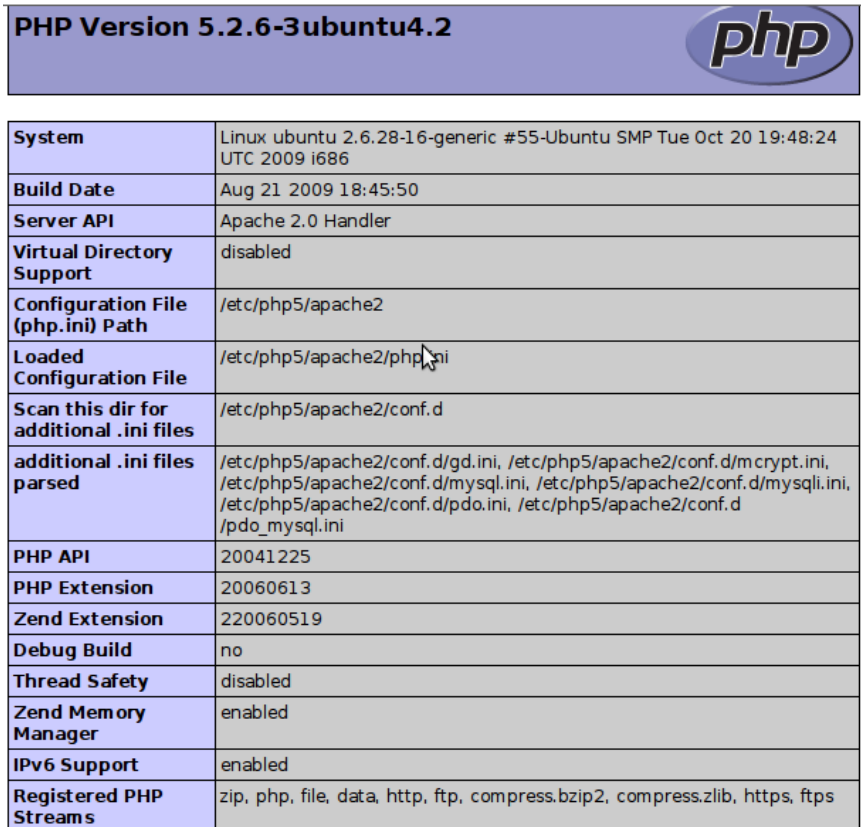
<?php phpinfo(); ?>

Salve e feche o arquivo.

Abra o seguinte código na barra de endereços:

http://localhost/testphp.php

Deverá aparecer a página ilustrada na FIG. 9.1.



PHP Version 5.2.6-3ubuntu4.2

System	Linux ubuntu 2.6.28-16-generic #55-Ubuntu SMP Tue Oct 20 19:48:24 UTC 2009 i686
Build Date	Aug 21 2009 18:45:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
additional .ini files parsed	/etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d /pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps

FIG. 9.1 – Página de configuração do PHP.

9.5.3 Instalação do Mysql

No Terminal entre com o seguinte código:

#sudo apt-get install mysql-server

Para que outros computadores da sua rede visualizem o servidor criado, você deve editar o *"Bind Address"*. Comecemos abrindo o Terminal para a editar o arquivo my.cnf:

#gksudo gedit /etc/mysql/my.cnf

Comente a seguinte linha (insira o caracter # no início):

bind-address = 127.0.0.1

Pelo Terminal, entre com a linha de código:

mysql -u root@localhost

Em seguida com:

mysql> SET PASSWORD FOR 'root@localhost' = PASSWORD('yourpassword')

Para sair, deve-se digitar:

/q

Recomenda-se instalar um programa chamado phpMyAdmin, que é uma ferramenta fácil para a edição do banco de dados. No Terminal entre com:

#sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin

Para que o PHP trabalhe com o MySQL edite do arquivo *php.in*. No Terminal entre com:

#gksudo gedit /etc/php5/apache2/php.ini

Agora descomentaremos a seguinte linha de código removendo o *';'*

;extension=mysql.so

Agora reiniciaremos o Apache:

sudo /etc/init.d/apache2 restart

9.5.4 Instalação do SSH

No terminal entre com a seguinte linha de comando:

sudo apt-get install openssh-server

9.5.5 Instalação do Tomcat

Primeiramente instalaremos o Java:

#sudo apt-get install sun-java6-jdk

Pegar a versão binária do TomCat do seguinte link:

#sudo wget http://linorg.usp.br/apache/tomcat/tomcat-6/v6.0.20/bin/apache-tomcat-6.0.20.tar.gz

No diretório do arquivo *apache-tomcat-6.0.20.tar.gz*, executar a seguinte linha de comando no Terminal:

tar xvzf apache-tomcat-6.0.20.tar.gz

Mover o arquivo descompactado para a pasta do Tomcat:

#sudo mv apache-tomcat-6.0.20 /usr/local/tomcat

9.5.6 Instalação do Postgres

Inicializar a instalação do postgres executando a seguinte linha de código no Terminal:

#sudo apt-get install postgresql-8.3 postgresql-client-8.3

Agora instalaremos o pgAdmin, uma ferramenta para administrar o postgres. No Terminal execute:

#sudo apt-get install pgadmin3 pgadmin3-data

O próximo passo é configurar uma senha para o usuário postgres. Comece chamando o utilitário psql com o usuário postgres e conecte no postgres especificamente no database postgres:

#sudo su postgres -c psql postgres

Agora altere a senha do usuário postgres com o seguinte comando:

ALTER USER postgres WITH PASSWORD ‘password’;

Finalize o psql:

\q

Disponibilizaremos o acesso ao banco para receber conexões de outras máquinas. Teremos que alterar dois arquivos para isso. Primeiramente editemos o arquivo *postgresql.conf*:

gksudo gedit /etc/postgresql/8.3/main/postgresql.conf

Na linha *listen_addresses*, troque o *localhost* por *, ficando a linha assim:

listen_addresses = ‘*’

Dessa forma seu postgres vai “escutar” não só conexões provenientes da sua própria máquina. A próxima configuração no mesmo arquivo é habilitar a encriptação de passwords, para fazer isso descomente a linha abaixo simplesmente removendo o # da frente dela:

password_encryption = on

Finalmente a próxima configuração é no arquivo *pg_hba.conf*. Neste arquivo você consegue restringir o acesso ao seu banco de dados por IP. Normalmente queremos liberar o acesso para todos os IP’s em uma faixa, por exemplo, liberando para todas as máquinas da rede 10.5.2.*, então adicionemos a seguinte linha no *pg_hba.conf*:

host all all 10.5.2.0 255.255.0.0 md5

Feito isso, basta reiniciar o postgres com o comando:

#sudo /etc/init.d/postgresql-8.3 restart

9.5.7 Instalação do SSL

Pegue o pacote de certificado ssl do apache:

#sudo wget http://www.theatons.com/test/Linux/apache2-ssl.tar.gz

Descomprima o arquivo:

#sudo tar xzvf apache2-ssl.tar.gz

Mova para a pasta do apache:

#sudo mv ssleay.cnf /usr/share/apache2/

Mover o certificado do apache para a pasta */usr/bin*:

#sudo mv apache2-ssl-certificate /usr/bin/

Crie o diretório onde o certificado será criado:

#sudo mkdir /etc/apache2/ssl

Executar o script do certificado:

#sudo apache2-ssl-certificate

Habilitar o modulo ssl:

#sudo a2enmod ssl

Criar link simbolico:

#sudo ln -s /etc/apache2/sites-available/ssl /etc/apache2/sites-enabled/ssl

Copiar diretório de certificados default para o de certificados do apache:

#sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-available/ssl

Abra o arquivo ssl:

#sudo gedit /etc/apache2/sites-available/ssl

Altere a linha <*VirtualHost *:80*> para <*VirtualHost *:443*> e logo abaixo adicione as linhas:

SSLEngine on

ServerSignature On

SSLCertificateFile /etc/apache2/ssl/apache.pem

Abra o arquivo default dentro de *sites-avaliable*:

#sudo gedit /etc/apache2/sites-available/default

O topo do arquivo deve ter uma estrutura parecida com a que segue:

NameVirtualHost *:80

<VirtualHost *:80>

Instale os arquivos PEM de certificados CA comuns:

#sudo apt-get install ca-certificates

Reinicialize o Apache:

#sudo /etc/init.d/apache2 force-reload

Caso ocorra algum erro, então consulte o arquivo de log do Apache 2. Este arquivo está localizado em */var/log/apache2/error.log.*

9.6 TUTORIAL DE INSTALAÇÃO DO SNORT

O aplicativo Snort é simples de ser instalado. Junto com o pacote vem as regras que esta ferramenta utiliza para gerar os alertas.

sudo apt-get install snort

9.7 TUTORIAL DE INSTALAÇÃO DO GUARDIAN

O pré-requisito para instalação do Guardian é que a ferramenta Snort já tenha sido instalada. Após isso, execute:

cd /etc/snort

wget -c http://www.chaotic.org/guardian/guardian-1.7.tar.gz

Vamos descompactar o nosso arquivo:

tar -xzvf guardian-1.7.tar.gz

cd guardian-1.7

Agora vamos copiar os arquivos para o /usr/local/bin. Pode-se notar que quando estamos copiando os arquivos já estamos mudando o nome deles, isso é preciso pois o arquivo guardian.pl, que é o arquivo que executa as ações do Guardian,

chama os arquivos de bloqueio e desbloqueio: guardian_block.sh e guardian_unblock.sh.

```
# cp scripts/IPTables_block.sh /usr/local/bin/guardian_block.sh
# cp scripts/IPTables_unblock.sh /usr/local/bin/guardian_unblock.sh
```

Agora vamos copiar o arquivo de configuração do Guardian para o diretório do Snort.

```
# cp guardian.conf /etc/snort/
```

E vamos criar um link simbólico do guardian.conf para o /etc, que é aonde que o guardian.pl vai procurar o arquivo.

```
# ln -s /etc/snort/guardian.conf /etc/guardian.conf
```

Vamos copiar o arquivo de gerenciamento do Guardian para o /usr/local/bin:

```
# cp guardian.pl /usr/local/bin/
```

Configurar as permissões dos nossos arquivos:

```
# chown root:root /usr/local/bin/guardian*
```

Configuração do Guardian:

```
# vim /etc/snort/guardian.conf
```

```
#Definir o endereço IP que será visto pela internet
HostIpAddr [IP]

#Definir a interface de rede que fica para a internet
Interface       [eth]

#Informar o ultimo octeto do endereço do Gateway
HostGatewayByte  1

#Arquivo de log
#LogFile         /var/log/guardian.log
LogFile          /var/log/snort/guardian.log

#Arquivo aonde estão sendo gerados os alertas do Snort
#AlertFile      /var/adm/secure
AlertFile       /var/log/snort/alert

#Lista de endereços de ip que serão ignorados pelo guardian
IgnoreFile      /etc/guardian.ignore
```

```
#Lista de endereços de ip que serão monitorados pelo guardian
#TargetFile      /etc/guardian.target
TargetFile      /etc/snort/guardian.target

#tempo em segundos para deixar um endereço ip bloqueado
TimeLimit       86400
```

Agora vamos criar os arquivos adicionais, necessários para o Guardian funcionar corretamente.

No arquivo guardian.ignore devem ser definidos os endereços IPs das máquinas que devem ser ignoradas,ou seja, as máquinas que não sofreram ações do Guardian caso estas geram algum tipo de alerta.

```
# touch /etc/snort/guardian.ignore
```

Criar e cadastrar os endereços IPs que devem ser monitorados em guardian.target. Dentro de guardian.target insira os IPs um a um aos quais se deseja que o Guardian execute uma ação caso o Snort gere um alerta advindo destes IPs.

```
# vim /etc/snort/guardian.target
```

Também precisamos criar o arquivo de log do Guardian.

```
# touch /var/log/snort/guardian.log
```

Agora só precisamos de um arquivo de inicialização para gerenciar o Guardian.

```
# vim /etc/init.d/guardian.sh
```

Inserir as seguintes linhas no arquivo:

```
#!/bin/sh

start()
{
  export PATH=$PATH:/usr/local/bin
  /usr/local/bin/guardian.pl -c /etc/snort/guardian.conf
}

stop()
{
  ps aux | grep 'guardian.pl *-c' 2>&1 > /dev/null
```

```
  if [ $? -eq 0 ];
  then
    kill `ps aux | grep 'guardian.pl *-c' | awk '{print $2}'`
  else
    echo "Guardian is not running ....."
  fi
}

status()
{
  ps aux | grep 'guardian.pl *-c' 2>&1 > /dev/null
  if [ $? -eq 0 ];
  then
    echo "Guardian is Running ....."
  else
    echo "Guardian is not Running ...."
  fi
}

case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    stop
    start
    ;;
  status)
    status;;
  *)
    echo $"Usage: $0 {start|stop|restart|status}"
esac
```

Vamos agora dar permissão de execução para o arquivo.

```
# chmod +x /etc/init.d/guardian.sh
```

E vamos adicionar o mesmo para que seja sempre carregado na inicialização.

```
# ln -s /etc/init.d/guardian.sh /etc/rc2.d/S21guardian.sh
```

Agora vamos reiniciar o serviço:

```
# /etc/init.d/guardian.sh stop
# /etc/init.d/guardian.sh start
```

Após isso, deve aparecer no terminal a seguinte mensagem:

> “My ip address and interface are: [IP] [eth]
> Loaded 0 addresses from /etc/snort/guardian.ignore
> Loaded 1 addresses from /etc/snort/guardian.target
> Becoming a daemon…”