UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDERSON BRUNEL MODOLON

ANÁLISE E IMPLEMENTAÇÃO DE HONEYPOT EM AMBIENTE LINUX

CRICIÚMA, JUNHO DE 2010

ANDERSON BRUNEL MODOLON

ANÁLISE E IMPLEMENTAÇÃO DE HONEYPOT EM AMBIENTE LINUX

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade do Extremo Sul Catarinense.

Orientador: Prof. MSc. Paulo João Martins

CRICIÚMA, JUNHO DE 2010

ANDERSON BRUNEL MODOLON

Análise e Implementação de *Honeypot* em Ambiente Linux

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paulo João Martins (UNESC)
Orientador

Prof. MSc. Rogério Antonio Casagrande (UNESC)

Prof. Esp. Fabrício Giordani (UNESC)

AGRADECIMENTOS

A Deus, pela vida, e pela Sua constante presença. “Tudo posso Naquele que me fortalece”.

Aos meus pais, por estarem sempre me apoiando em todos os momentos que necessito.

A minha família, amigos e todos aqueles que de alguma forma estiveram presentes para ajudar nos momentos difíceis.

Ao meu orientador, professores, e todos aqueles que contribuíram para a minha formação acadêmica.

A humildade é a única base sólida de todas as virtudes.

Confúcio

RESUMO

A segurança da informação vem se tornando um fator cada vez mais importante devido ao valor que elas podem ter para uma organização. É necessário sempre ter ferramentas atualizadas para a sua proteção, porém mesmo com as melhores ferramentas, é possível que exista um ou mais ataques que não seja possível a sua prevenção. O trabalho apresenta um estudo e implementação do *honeypot honeyd* em um ambiente *linux*, onde foram simulados ataques a *hosts* falsos que respondem como verdadeiros e analisado os *logs* gerados. Foram também instaladas máquinas virtuais com o programa *VirtualBox* afim de que fossem confundidas tanto com as máquinas reais como também com os *hosts* criados pelo *honeypot*, além de um estudo e implementação com um sistema de detecção de intrusão para monitorar o tráfego na rede, a fim de ajudar na monitoração contra ataques.

Palavras-Chaves: *Honeypot; Honeyd;* Sistema de Detecção de Intrusão; *Snort;* Segurança da Informação.

ABSTRACT

The security of information is becoming an increasingly important factor of value, that they can have for an organization. Is always necessary have updated tools for their protection, but even with the best tools, it is possible that exist one or more attack that couldn't be possible have protection. This report show a study and implementation of honeypot Honeyd in Linux, where the hosts were simulated fake hosts that reply like real logs generated and analyzed. We also installed a VirtualBox in order that they were both confused as real as the machine with the hosts also created by the honeypot, as well as a study and implementation with an intrusion detection system to monitor network traffic in order assist in monitoring attack.

Key Words: Honeypot; Honeyd; Intrusion Detection System; Snort; Security of Information.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As informações são de grande importância para que uma empresa possa conseguir o sucesso, como também para o processo de tomadas de decisões, onde esta possa definir o futuro de uma organização. Assim, torna-se necessário a segurança e integridade das informações, pois são descobertas novas ameaças a cada dia. As ameaças geralmente são encontradas na *Internet* ou por pessoas ligadas à própria organização, com o objetivo de roubá-las ou de destruí-las pelos mais variados motivos, como um ex-funcionário com motivos pessoais, outra organização com intenções que não são consideradas éticas ou por motivos financeiros em geral, seja por interesse de usuários ou de organizações.

O sigilo das informações para uma empresa é de vital importância, pois estas são de grandes utilidades, como por exemplo, de uma estratégia em resposta à concorrência, ou de planejamentos futuros para a mesma, e se fossem copiadas sem autorização, poderia gerar conseqüências a longo e médio prazo, como investimentos com pouco ou sem retorno, ou pelo fato da concorrência utilizar dessas informações antes mesmo da empresa proprietária.

Desta forma, torna-se importante a segurança das informações, prevenindo assim acessos não autorizados aos dados. Mas mesmo com toda a segurança que se possa ter disponível, ainda há a possibilidade de possíveis falhas. Uma forma de prevenção é a detecção das tentativas de invasão, para que seja gerada uma documentação e que possa servir de referência a mecanismos de segurança, procurando evitar que a mesma falha ocorrida possa ser repetida posteriormente. Para isso, utiliza-se um Sistema de Detecção de Intrusão.

Entre as principais funções de um Sistema de Detecção de Intrusão (IDS) uma delas é de monitorar e analisar as possíveis tentativas de invasões que possam ocorrer em um ambiente computacional. Sendo assim, é importante a análise e o estudo das tentativas para que possam ser documentadas e assim, surge a necessidade de ferramentas mais específicas

para isso, como exemplo os *honeypots*, que são ferramentas que têm como objetivo serem atacados e invadidos, e assim colher informações do atacante. Isso significa que não se trata de uma ferramenta de segurança contra os ataques em sí, mas sim de uma que pode ser usada como prevenção de futuros ataques e invasões, por meio de informações que pode fornecer sobre uma determinada invasão ocorrida.

Nesta pesquisa será realizado um estudo de caso na utilização de um sistema *honeypot* em ambiente linux.

1.1 OBJETIVO GERAL

Implantar e analisar o comportamento de sistema *honeypot* em ambiente *linux.*

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos fazem parte do conteúdo a se alcançar o objetivo geral, que são eles:

a) estudar e aplicar os conceitos sobre segurança da informação e *honeypots;*
b) analisar e descrever mecanismos de detecção de intrusão;
c) analisar e identificar ferramentas *honeypot;*
d) aplicar uma ferramenta *honeypot* ao ambiente *linux;*
e) simular tentativas de ataques ao *honeypot;*
f) analisar o comportamento de um *honeypot* diante de tentativas de ataques.

1.3 JUSTIFICATIVA

As informações podem ser consideradas entre os principais patrimônios de uma organização, por isso estas devem ser protegidas como qualquer outro bem valioso. Dessa forma, a segurança das informações tornou-se um ponto considerado crucial para a evolução de uma organização. Ao serem colocadas em computadores que têm acesso a Internet, ficam vulneráveis a alguém sem autorização que queira ter acesso as mesmas, dessa forma tornam-se necessário a utilização de mecanismos para a sua segurança, com o objetivo de protegê-las de acessos não autorizados (FERREIRA, 2003).

Quando um invasor conseguir atravessar as barreiras de segurança impostas pela organização, as informações ficarão a seu dispor. Com isso, se for utilizada uma ferramenta com fins de desviar o atacante de seu foco principal sem que o mesmo perceba, e se obter sucesso com sua utilização, as informações da organização poderão permanecer em segurança. Essa ferramenta chama-se *honeypot*, que tem como objetivo desorientar o atacante de seu foco principal e atraí-lo para si, onde o mesmo será monitorado a cada passo que realizar.

Os *honeypots* são ferramentas projetadas de forma que ela aparenta ser algo interessante para o atacante, desviando-o dessa forma de seu foco principal, com o objetivo de atrair o atacante para si, ou seja, com o objetivo de ser atacada, pois são altamente monitoradas e projetadas para parecerem ser algo que na realidade não são. Ao ser invadido, o *honeypot* gera avisos e *log's* sobre a invasão, permitindo assim saber quais os passos do atacante durante a invasão, e com essas informações, após ter ocorrido a invasão, garantir que recursos de segurança da rede possam ser alterados corretamente (THOMAS, 2007).

Além de ser um tema bem atual, os *honeypots* são ainda desconhecidos em algumas organizações, por isso torna-se necessário esta pesquisa, a fim de fazer uma análise de seu comportamento em ambiente *linux* através de testes e tentativas de ataques.

1.4 ESTRUTURA DO TRABALHO

O trabalho foi dividido em seis capítulos. No primeiro capítulo, encontram-se quais são os objetivos gerais e específicos, assim também a justificativa do mesmo.

O segundo capítulo, onde realmente começa o conteúdo desta pesquisa, é abordada sobre a segurança das informações, a sua importância, o motivo de ser um fator importante para uma organização e também sobre a política se segurança, o que é e o porquê se aplicar.

O Capítulo 3, apresenta o que é detecção de intrusão, quais os seus objetivos, as formas que existem e o posicionamento dos sensores de detecção de intrusão na rede.

O Capítulo quatro, tem como objetivo conceituar *honeypots*, o que são, onde são utilizados, os níveis de interação, a sua importância, o que são *honeynets*, *honeytokens* e apresenta também alguns projetos *honeypots* e *honeynets* existentes.

No quinto capítulo, mostra várias ferramentas *honeypots* disponíveis tanto em plataforma *windows* como em plataforma *linux*.

O Capítulo 6 apresenta alguns trabalhos correlatos que foram realizados na mesma linha de pesquisa que este trabalho.

No sétimo e último capítulo, onde é feita a implementação do *honeypot* no ambiente *linux*, mostra qual o *honeypot*, qual sistema de Detecção de Intrusão e qual distribuição *linux* foram utilizados no trabalho. Apresenta os testes realizados no *honeypot*, iniciando de testes básicos, depois testes com ferramentas *scanners* e identificação de *hosts* na

rede, assim como também tanto a resposta obtida dele como também os *logs* gerados e os alertas do sistema de detecção utilizado a cada teste realizado.

1.5 METODOLOGIA

Para a análise da ferramenta *honeypot* em ambiente *Linux* foi montado um ambiente de forma a se fazer a simulação de um ambiente real de produção. Foi montada uma rede virtual de computadores através do *Virtualbox,* um *software* dedicado à emulação de vários sistemas operacionais em uma única máquina. No ambiente foi utilizado um computador para fazer os testes e ataques ao *honeypot* com o sistema operacional *Windows XP*, e foi feito nele a intalação do *Ubuntu 9.04* através do *VirtualBox*. Foi utilizado também um *notebook* com o sistema operacional *Ubuntu 8.04*, onde foram instaladas três máquinas virtuais com o *Windows XP* e a ferramenta *honeyd,* que foi o *honeypot* utilizado para os testes.

No computador que tem o *Windows XP*, foi instalado o retina, um *software* utilizado para fazer a análise de vulnerabilidades de uma rede, incluindo as máquinas e serviços da mesma. Na máquina virtual do *Ubuntu 9.04* foi instalado o *nmap*, um programa para localizar as máquinas e serviços disponíveis em uma rede. Na ferramenta *honeypot* foram criados sete *hosts* de forma a se parecerem com as máquinas instaladas no ambiente.

1.6 RESULTADOS E DISCUSSÕES

Foram realizados testes básicos de comunicação no *honeypot*, como testes com *ftp, telnet* e *iis*, onde se obteve respostas em todos os testes básicos realizados, de forma como se fossem sistemas operacionais intalados. Depois foram realizados testes com o *nmap* e o retina,

que também se se obteve por parte deles respostas reais, sendo vistos pelas ferramentas como máquianas reais.

Em todos os testes realizados o *honeypot* armazenou registros, mostrando várias informações, como a data e hora da conexão, o *ip* de origem, e o protocolo utilizado. Também identificou qual o serviço utilizado e os comandos usados na tentativa de acesso ao *honeypot*. No caso dos *scanners*, obteve-se um extenso armazenamento dos logs, devido aos muitos testes que as ferramentas realizam para se ter as informações possíveis dos *hosts* disponíveis na rede.

Vários trabalhos buscam a implementação de um *honeypot* em um ambiente real, de forma a analisar as informações geradas por possíveis invasões que possam ocorrer à ferramenta. Armazenam e analisam os *log's* gerados, como também através dos resultados mostram a existência de iteratividade do atacante com a ferramenta, porém não são mostradas as respostas dadas a eles, e se de acordo com elas é possível que haja alguma desconfiança por parte do atacante que se trata de um *honeypot*. Nesta pesquisa são também analisadas as respostas que a ferramenta fornece através dos testes, como por exemplo, o comportamento de um falso servidor *iis* ou na identificação dos *host's* e serviços gerados pelo *honeypot* através de *scanners* de vulnerabilidades.

Com a implantação de um sistema de detecção de intrusão ao ambiente, é possível também se monitorar o ambiente a fim de registrar as atividades consideradas suspeitas, como *scanners* por exemplo, onde é possível obter conhecimentos dos *host's* escaneados e várias outras informações sobre a atividade suspeita, com o *ip* de origem do *scanner*, o protocolo utilizado entre outras.

A utilização do *honeypot* em conjunto de um sistema de detecção de intrusão faz com que os alertas gerados por ele em comparação aos *log's* gerados pelo *honeypot* aumentem as informações sobre as atividades consideradas suspeitas, bem como as

confirmações dos alertas gerados ao serem comparados com os *log's* sobre o tráfego malicioso destinado ao *honeypot*.

Nos testes de comunicações simples praticamente não foram gerados alertas pelo sistema de detecção de intrusão devido à situação proposta pelo ambiente já existir computadores com os mesmos serviços que são fornecidos pelo *honeypot* implementado, mas seria possível configurar o sistema de detecção de intrusão para a geração de alertas para estes serviços, caso fossem fornecidos somente pelo *honeypot*. Para os testes com os *scanners*, foram gerados vários alertas com informações sobre a varredura realizada, como também muitos *log's* na ferramenta de *honeypot*.

2 SEGURANÇA DAS INFORMAÇÕES

O homem sempre buscou ter o controle das informações de alguma forma, pois esta é considerada dependendo o caso de valor ou utilidade a alguém ou a alguma organização, pois atualmente não existe organização que não seja dependente de suas informações. Assim, torna-se necessário ter um nível de segurança que seja aceitável, limitando o acesso apenas a pessoas autorizadas e protegendo contra invasões e em relação ao acesso físico das mesmas (CARUSO; STEFFEN, 1999).

Elas são importantes para as organizações e podem ser essenciais para manter a competitividade com os seus concorrentes, como também para a sua própria imagem. Devido à dependência que as empresas têm da informação, é recomendável que se tenha um controle onde elas possam ser acessadas apenas por pessoas autorizadas, e que possa ser confiável em relação ao seu conteúdo, e que não seja alterada e nem apagada sem a autorização do dono. Em função disso, a Segurança da informação se torna útil, pois procura proteger este bem de vários tipos de ameaças, seja causada por fatores internos ou externos.

Esta segurança pode ser considerada como um fator de grande importância, pois trata do estado das informações. Para que esta segurança seja aplicada de forma correta, ela deve inicialmente ter seus objetivos pré-definidos, respondendo a uma série de perguntas, sobre o que se deve proteger, contra quem, de quais ameaças, sobre o grau de proteção desejado e os recursos utilizados para alcançá-lo, como também das conseqüências caso as informações sejam destruídas ou furtadas.

Os prejuízos que uma organização pode ter, caso suas informações fossem acessadas pela concorrência (perda de confienciabilidade), ou fossem corrompidas (falta de integridade) ou não pudessem ser acessadas (perda de disponibilidade) para o fechamento de

um grande negócio podem ser de grande escala, visto que também poderia afetar a sua reputação e sua confiança (FERREIRA, 2003).

A segurança da informação em si não pode ser vista como um estado a ser alcançado, onde quando alcançada não seja mais necessário se preocupar, mas sim como um processo, pois não existe algo completamente “seguro”, mas sim com um nível de segurança que seja considerado como aceitável. Por isso, não há como afirmar que algo considerado seguro permanecerá seguro sendo que a cada dia surgem novas ameaças e com isso o risco para a segurança aumenta, ainda mais com a evolução e uso da Internet (WADLOW, 2000).

Existem vários riscos que as informações de uma organização correm ao estarem de alguma forma conectada a rede, desde o simples acesso de um funcionário não autorizado, até o acesso de uma outra empresa externa, com objetivos de cópia ou destruição dos dados existentes. Em função da redução destes riscos é que se deve ter uma política interna de segurança, onde deva conter apenas regras básicas e aplicáveis. As regras mais detalhadas deverão ser relacionadas com cada setor específico (FERREIRA, 2003).

2.1 POLÍTICA DE SEGURANÇA

Ela trata de um conjunto de regras gerais para controlar e padronizar a proteção das informações de uma organização. Com o seu uso, os riscos a segurança devem ser de alguma forma prevista antes que possam causar algum dano as informações e consequentemente à organização. A política de segurança deve ser implantada e sempre divulgada a todos os funcionários ou qualquer outra pessoa que possa ter acesso a algum tipo de informação.

Para que a política possa ser implantada e mantida de forma correta, ela deve ser de acordo com a realidade da organização. Não apenas ser escrita de forma rígida, sem uma

análise para verificar se existem condições de ser colocada em prática, caso exija muitos recursos, tempo, esforços, se é muito extensa ou algo mais que possa prejudicar o dia-a-dia dos funcionários ou da administração da organização. Por exemplo, se exigir muitos recursos, a administração pode não concordar com a mesma, se for muito tempo, pode prejudicar a produção e se exigir muitos esforços, os funcionários podem não aplicá-la devidamente. Este tipo de política pode ser chamado de política sem imposição, ou seja, a política é escrita, mas não imposta, seja por algum motivo dos citados anteriormente ou algum outro motivo específico. Uma política desse tipo, dependendo o caso, pode até encorajar as pessoas a ignorarem ou modificarem suas regras, fazendo com que as informações fiquem de certa forma desprotegidas ou com uma segurança problemática, dependendo da situação (NORTHCUTT, 2002).

Algumas organizações encontram dificuldades ao implantar ou modificar a sua política de segurança. Isso devido à sua cultura interna, pois sua estrutura funciona de forma a não ter um controle rígido necessário para uma mudança em sua estrutura. Um dos motivos é o das organizações buscarem mão-de-obra mais barata e que possam realizar o mesmo serviço de alguém especializado, com o objetivo de reduzir custos. Sendo desta forma, a mão-de-obra menos qualificada implica em profissionais com menos disciplina e disposição a seguir regras de segurança com rigidez, seja pelo fato de não ter o conhecimento necessário a seguir da forma correta, ou pelo fato de atribuir pouca ou nenhuma importância a esse tipo de cultura. Este tipo de perfil profissional torna difícil a implantação de uma política de segurança que vise mudanças na estrutura de uma organização, a não ser que haja alguma forma de conscientização dos funcionários da mesma (CARUSO, 1999).

3 DETECÇÃO DE INTRUSÃO

Com a evolução e o crescimento da Internet nos últimos anos, obteve-se um aumento na comunicação entre as organizações, seja para oferecer soluções ou simplesmente mostrar sua atividade também no mundo *virtual*. Com essa evolução, os dados das organizações tendem a ficar mais vulneráveis, correndo riscos de acessos indevidos a determinadas informações importantes ou de conexões externas não autorizadas. Assim, torna-se necessário a implantação de mecanismos de segurança para tentar evitar os acessos indevidos aos dados, sejam internos ou externos, e também as conexões não-autorizadas.

Mesmo que estes mecanismos sejam dimensionados e implantados de forma correta com a situação da empresa, é ainda possível a existência de falhas na segurança, colocando em risco a confidenciabilidade dos dados. Uma forma de se prevenir é ter conhecimento das tentativas de invasão que possivelmente possam ser sofridas, e com isso, poder ser tomadas as possíveis providencias. No caso de uma invasão acontecer, os dados coletados com a mesma podem ser de utilidade em sua própria proteção mais adiante (MARCELO; PITANGA, 2003).

Os Sistemas de Detecção de Intrusão (IDS) têm como objetivo a detecção das tentativas de intrusão que possam ocorrer, sejam elas bem sucedidas ou não. Com ele é possível a identificação das tentativas e se algum ataque foi bem-sucedido, mesmo que não tenha danificado nada, é possível saber informações que poderão ajudar a evitar que esta invasão aconteça novamente.

Um IDS devidamente configurado pode desempenhar mais de uma função na identificação de ataques. Pode detectar *scans* de reconhecimento, notificando aos responsáveis pela segurança a ocorrência do evento e também gerar alerta de tentativas de violações de segurança em um *host*, por meio de vários métodos, onde alguns deles serão

vistos mais adiante. Pode também realizar a monitoração da rede e análise de tráfego, dependendo de qual IDS e versão está sendo utilizada. Estas outras funções podem ser muito valiosas para a identificação de violações que possam ocorrer em relação à política de segurança, como por exemplo, o uso de serviços não autorizados e de tráfego desconhecido.

Alguns produtos podem oferecer soluções onde há implantação de várias instalações do IDS em vários pontos da rede, onde estes funcionam como sensores. Eles transmitem todos os seus dados a um servidor centralizado, que poderá armazenar e comparar os resultados de todos os IDS, tomando a decisão ou não de gerar um alerta. Estas são chamadas de soluções em camadas, que envolve em sua maioria das vezes a implantação de vários sensores em vários segmentos da rede (NORTHCUTT, 2002).

Os possíveis estados que um IDS pode assumir em relação às suas falhas são quatro: os Falsos Positivos, onde uma atividade normal é considerada como suspeita devido a comportamentos imprevisíveis que os usuários e as redes podem ter, os Verdadeiros Positivos, que são quando uma atividade ilegal é detectada como tal, sendo este o objetivo de qualquer IDS. Os Falsos Negativos, que consiste em uma atividade intrusiva real que não seja detectada pelo IDS, sendo este estado o de maior perigo a uma organização, onde o atacante obteve sucesso com sua intrusão e o IDS não foi capaz de detectá-lo, ou quando o fez já era tarde demais. O estado de Verdadeiro Negativo é o estado dito como normal, onde o usuário está dentro de seu perfil de utilização dos recursos. A atividade não é ilegal, e não é considerada como tal. Portanto um IDS sempre estará em um destes estados, seja qual for seu tipo ou sua arquitetura (CARVALHO, 2005).

3.1. DETECÇÃO DE INTRUSÃO BASEADO EM ANOMALIAS

A detecção por anomalia busca a construção de perfis que podem ser de usuários, sistemas, conexões de redes ou de processos. Eles são comparados às atividades monitoradas com o objetivo de determinar se as atividades são diferentes do comportamento considerado normal, podendo dessa forma identificar um ataque (BORGES; COUTINHO, 2007).

Existem várias formas para implementar os *IDS*, como por exemplo os Sistemas de Detecção de Intrusão Baseado em *host (HIDS),* que são caracterizados por utilizarem em um determinado *host* a detecção por anomalias, onde é construído um perfil de utilização do usuário para ser monitorado. Isso é feito com base em vários fatores da máquina, como o uso do processador, processos em execução no sistema, caso tenha algum que leve a um desvio de comportamento, como também as mudanças no registro, como a adição de usuários, acesso a determinados arquivos confidenciais entre outros.

Algumas das vantagens são de não precisar de nenhum *hardware* adicional para a sua implantação, e de poder associar um usuário a um desvio de comportamento. As desvantagens são de dependerem do funcionamento do sistema operacional, assim como o *host* monitorado apresenta uma perda de desempenho (MARCELO; PITANGA, 2003).

3.2. DETECÇÃO DE INTRUSÃO BASEADO EM ASSINATURAS

Uma assinatura pode ser considerada como algo que está sendo buscado no tráfego de rede e que possa ser comparado a uma base de dados, para se saber se há coincidência entre elas. Havendo, então é tomada uma ação, como a geração de alertas ou eventos registrado como um possível ataque.

Essa base de dados necessita estar sempre atualizada, assim como a exemplo de um *software* de *antivírus*. Sempre que uma nova vulnerabilidade ou exploração que possa ser vista de forma generalizada for encontrada, os fabricantes rapidamente pesquisam essa nova vulnerabilidade ou exploração e criam uma assinatura nova para ela, ou mesmo informam que assinaturas existentes já correspondem a ela.

Uma assinatura pode ser considerada como específica ou geral. A específica pode ser precisa na detecção de um ataque específico, mas se forem rapidamente feitas mudanças nesse ataque, essa assinatura pode não detectar mais este ataque. Neste caso uma assinatura geral poderia identificá-lo, pois não importando o ataque, se este se encaixar em um perfil considerado malicioso, esta gerará um alerta de um possível ataque. A vantagem é que o atacante poderá fazer mudanças no ataque e sempre será impedido de obter sucesso enquanto se encaixar em algum perfil visto como perigoso. A desvantagem é o fato de gerar os falsos positivos, onde um usuário em condições normais pode ser considerado como invasor. Isso devido ao seu perfil de uso coincidir com alguma assinatura do IDS que seja considerada como perigosa.

Todo *IDS* gera falsos positivos, seja em grande ou pequena quantidade, de acordo com o ajuste dos sensores da rede. Por mais que os sensores sejam corretamente ajustados, sempre há a possibilidade de algum tráfego benigno coincidir com alguma assinatura. Se algum falso positivo for ignorado sem uma devida análise do mesmo, pode ser gerada uma espécie de falso negativo, onde o ataque é realizado, mas não detectado, pois o mesmo estava junto e foi confundido com um falso positivo, ao invés de ter sido analisado devidamente. Existem vários tipos de *IDS* baseados em assinaturas, como por exemplo, o baseado em rede, mas a escolha a ser utilizada dependerá da situação da empresa ou organização (NORTHCUTT, 2002).

Os *IDS* baseado em Rede visam a detecção por meio do monitoramento do tráfego de pacotes em um determinado segmento da rede, gerando alertas e armazenando *log's*, conforme a sua configuração. Ele examina o cabeçalho de cada pacote que circula em seu segmento à procura de sinais suspeitos, como por exemplo, um ataque de negação de serviço (DoS – *Denial of Service*). A sua utilização permite a rápida notificação sobre um possível ataque, como também guardar as sua evidências para tomar as devidas providências que possam melhorar o nível de segurança da organização. São independentes de sistemas operacionais, não tendo problemas em situações onde ocorrem anomalias com as máquinas em si, pois sua função é a monitoração da rede, independendo de arquiteturas de máquinas ou sistemas operacionais. Existem vários comerciais ou *opensource*, como o *Snort*, por exemplo, que é de uma ferramenta de código fonte aberto que pode ser utilizada por especialistas de segurança em redes (MARCELO; PITANGA, 2003).

3.3. TAREFAS ADICIONAIS NA DETECÇÃO DE INTRUSÃO EM REDE

Os IDS possuem vários papéis para a defesa em uma rede. Em alguns casos são exclusivos para a tarefa que realizam, em outros casos executam outras tarefas, como a identificação dos pontos fracos e vulnerabilidades na segurança, complementando outros componentes de defesa, realizando funções que não poderiam ser feitas sem os mesmos. É de grande utilidade a identificação dos pontos fracos da rede, pois senão será apenas questão de tempo até algum atacante possa encontrar e explorar para realizar um possível ataque. E se for bem sucedido e comprometer algum *host*, este poderá servir ao ataque de outros hosts. Encontrar vulnerabilidades e pontos fracos na rede permite a evitar os primeiros estágios de possíveis ataques. Algumas tarefas que os IDS podem realizar em uma rede são analisadas a seguir (NORTHCUTT, 2002):

a) detecções de violações da política podem ser identificadas por meio de alguns IDS. Quando algumas portas específicas ou protocolos estiverem sendo usados sem a devida permissão, este poderá gerar um alerta, como também na localização de sistemas mal configurados nas próprias redes, como por exemplo, um *host* que não está usando um determinado servidor *proxy* da *Web* e assim reduzindo o nível de segurança do ambiente. Podem também ajudar a identificar sistemas não autorizados que alguém esteja rodando, como algum servidor *Web* de algum funcionário específico para fins pessoais ou profissionais, mas que sejam fora parte da organização;

b) a detecção de ataques que saem dos próprios *hosts* a entidades externas pode ser considerado um relevante fator para a segurança, pois pode ser causado por alguns usuários ou algum tipo de verme que conseguiu se infiltrar e comprometer algumas máquinas para atacar outras. Esta detecção é valiosa em ambientes onde o acesso de saída é por parte ou total irrestrita. Onde os *firewalls* e os filtros de pacotes são configurados para permitir quase ou toda atividade para fora, os IDS podem ser o único método na identificação destes tipos de ataques;

c) comparar as atividades individuais dos *hosts*. Se por exemplo, 50 hosts registrarem uma tentativa falha de invasão, poderia passar despercebido, mas se um sensor IDS detectar essas 50 tentativas nos *hosts*, teria maiores chances de disparar um alerta;

d) a resposta ativa, onde se for detectado um possível ataque, ele pode mudar sua própria configuração, ou de algum outro dispositivo de rede. Uma resposta ativa pode incluir o envio de pacotes para o atacante, de modo parecer estar vindo do endereço de IP da vítima, a fim de terminar a conexão suspeita. Pode

também enviar mudanças no conjunto de regras do *firewall*, e monitorar todo o tráfego do atacante o tempo necessário para alguma resposta. Com a resposta ativa devem-se tomar alguns cuidados, como o de o atacante não estar realmente no endereço de *IP* que o sensor está identificando, ou com a geração de um falso positivo, sendo que o ataque não está realmente acontecendo, e uma resposta poderia causar a negação de serviço a algum setor ou usuário inocente, que poderia ficar incapacitado de usar seus serviços.

3.4. POSICIONAMENTO DOS SENSORES DE DETECÇÃO DE INTRUSÃO NA REDE

Em ambientes de redes, é aconselhável a implantação de vários sensores, onde cada sensor normalmente monitora um segmento da rede. Em empresas pequenas também se deve usar mais de um sensor, mesmo que apenas um possa parecer adequado. O uso de mais sensores pode produzir melhores resultados, onde cada um é ajustado para o tráfego do segmento a ser monitorado. Se para uma empresa pequena for implantado somente um sensor, mesmo sendo adequado, a quantidade de ajustes neste será mais limitada do que se tivesse mais de um nesta mesma rede. Um outro motivo é a tolerância à falhas, onde se o único sensor falhar, ou se o segmento da rede em que o sensor estiver não estará disponível, a rede ficará sem nenhuma detecção de intrusão até que a falha seja corrigida, mas se tiver mais de um sensor, apenas o segmento da rede será afetado, e não a rede toda (NORTHCUTT, 2002).

O posicionamento dos sensores de IDS normalmente pode ser colocado próximo a *firewalls* ou filtros de pacotes, ou em lugares perto de acesso à *Internet.* Ele pode estar tanto do lado externo, para identificar toda a atividade suspeita, como do lado interno, para identificar as atividades suspeitas que passe pelo *firewall*. Um melhor resultado é obtido colocando os sensores dos dois lados das *firewalls* e dos filtros de pacotes, mas se por algum

motivo é possível apenas de um lado, é recomendado colocar na rede externa, de modo a detectar todos os ataques, tanto os que passam como os que não passam pela filtragem.

Em alguns casos, a implantação do sensor na rede interna pode trazer uma série de vantagens. Quando estão na rede externa, eles terão mais chances de serem atacados, por isso processarão muito mais tráfego do que se estivessem na rede interna, e se a empresa ou organização não tiver muito tempo disponível para realizar a análise de intrusão, se preocupando apenas com as ameaças mais sérias, o sensor na rede interna fará apenas a detecção dos ataques que entram na rede, e ainda estarão ajudando a determinar se o dispositivo de filtragem pode estar mal configurado (NORTHCUTT, 2000).

Uma rede alternativa pode ser utilizada na comunicação entre os sensores de IDS e assim melhorar a segurança da rede. Desta forma, cada sensor da rede possui pelo menos duas placas de *interface* de rede, onde uma ou mais são utilizadas para farejar o tráfego das redes monitoradas, sendo esta como sua única função. Estas placas não transmitem tráfego, pois o mesmo é transmitido através de uma outra placa, que é conectada a uma rede alternativa, sendo esta específica para a transmissão dos dados dos sensores e atualizações de configurações dos mesmos (NORTHCUTT, 2002).

Uma outra solução atraente é o uso de honeypots que se trata de um recurso de segurança próprio para ser atacado ou comprometido por um atacante, que pode ser utilizado para distrair a atividade maliciosa, como mecanismo de alerta ou monitoração de ataques.

4 HONEYPOTS

Se for considerada a tradução da palavra, *honeypot* significa pote de mel, onde o próprio nome já sugere, pois o mel atrai abelhas, formigas, e outros animais que gostam desse sabor. Junto com esse conceito, pode-se definir como uma armadilha afim de atrair pessoas interessadas em algo específico.

A forma de utilização é feita através da Engenharia Social, que trata-se de um conjunto de técnicas para manipular os sentimentos, opiniões ou até atitudes através de formas específicas, como a curiosidade, por exemplo. A Engenharia social está relacionada ao *honeypot* de forma que este é utilizado para chamar a atenção do atacante para si, desviando-o de seu foco principal (ASSUNÇÃO, 2009).

Normalmente, os *honeypots* são utilizados para desorientar o atacante de seu foco, coletando informações sobre o ataque e o atacante, pois são ferramentas projetadas para se parecerem uma coisa aos olhos dos atacantes, enquanto na verdade é outra.

Alguns de suas utilidades podem ser citados como a sua capacidade de poder distrair o atacante dos seus objetivos, que normalmente são os recursos mais valiosos da rede, e ao mesmo tempo fornecer avisos sobre tal fato de forma a ser mais confiável em relação aos falsos positivos que os IDS, pois o foco de seu monitoramento é ele mesmo (THOMAS, 2007).

O surgimento do primeiro *honeypot* real foi em 1997, com o nome de *Deception Toolkit (DTK),* que se trata de um conjunto de scripts didáticos que simulam vários servidores, inclusive vulnerabilidades. É um software livre e ainda é utilizado atualmente. Em 1998 surgiu o *Sting,* o primeiro produto comercial, e foi desenvolvido para ambiente *Windows NT,* simulando uma rede inteira, além de emitir falsas respostas para os atacantes e simular vários ambientes operacionais. No mesmo ano *Martin Roesch,* que é o criador do *Snort,* um

IDS muito famoso e utilizado, desenvolveu um *honeypot* que era capaz de simular uma rede de classe C com sete sistemas operacionais e vários serviços.

A criação do Projeto *Honeynet* em 1999 por *Lance Spitzner* e mais 30 especialistas de segurança se tornou com o passar do tempo uma referência, pois se trata de uma série de metodologias para a implementação de *honeypots.* Em 2001 eles começaram a se popularizarem em grande escala. Em 2002 foi lançado o *Honeyd*, *por Niels Povos*, que é uma ferramenta *open source* amplamente utilizado nos dias atuais (MARCELO; PITANGA, 2003).

A utilização de um *honeypot* pode trazer melhorias nos Sistemas de Detecção de Intrusão, assim como pode também ser utilizado para atacar outras redes, caso seja comprometido. Em relação ao aspecto jurídico, existem argumentos alegando que ele induz alguém a fazer algo errado, porém está apenas disponível na rede, se o invasor chegou até ele, foi por conta dele que se confundiu na invasão graças ao *honeypot*. Assim como também no aspecto de privacidade, pois a monitoração está sendo feita no próprio sistema, onde o correto seria que o atacante não estivesse iteragindo com o sistema. Pode-se ser comparada a instalação de câmeras de segurança e uma casa, onde o ladrão não poderia reclamar delas.

A localização pode ser em três locais (ASSUNÇÃO, 2009):

a) antes do primeiro *Firewall,* onde a finalidade é a exposição ao máximo sem proteções, para fins de pesquisa;
b) na zona Desmilitarizada (DMZ), onde ele ficará no mesmo nível dos servidores da rede, podendo levar um possível atacante durante a varredura de portas a acreditar que se trata de mais um servidor;
c) junto à rede interna, onde pode ser utilizado na detecção de funcionários a acessar/buscar dados indevidos na rede.

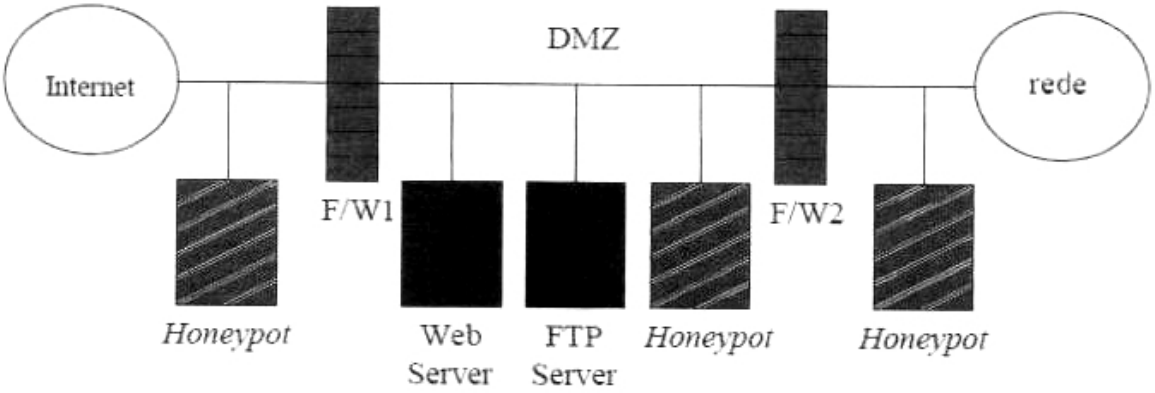


Figura 1: Localização do *Honeypot*

Fonte: ASSUNÇÃO 2009.

4.1 TIPOS DE HONEYPOT'S

Existem dois tipos*:* os de pesquisa e de produção. Os de pesquisa têm um nível de comprometimento alto, pois o seu objetivo é de atuar depois que o *hacker* conseguiu realizar a invasão. A principal função desse tipo é a coleta de informações sobre as ferramentas utilizadas para o ataque e como foram utilizadas. É utilizado em redes externas com pouca ligação com a rede principal. Também podem ser considerados como IDS passivos.

Quando o seu tipo é de produção, o seu foco é de distrair o atacante, tirando-o de seu foco sem que o mesmo possa perceber. Não traz nenhuma vantagem na estrutura de segurança, pois seu foco é outro.

Como o principal objetivo dos *honeypots* é de analisar os ataques depois dos mesmos serem comprometidos, logo todos os dados que passam por eles podem ser considerados como informação para análise e estudo (MARCELO; PITANGA, 2003).

4.2 NÍVEIS DE INTERAÇÃO DE UM HONEYPOT

Existem três níveis de interação para um *honeypot:*

a) alta interação: ocorre quando é utilizado um sistema operacional real com serviços reais como *honeypot*, tendo como vantagem de que o invasor acreditará que é um computador normal de produção. Existe o perigo de ele realmente conseguir comprometer afim de utilizá-lo a acesso a outro computador da rede (ASSUNÇÃO, 2009);

b) média interação: nesses é utilizado um sistema operacional falso, onde em nenhum momento o atacante terá contato com o sistema operacional verdadeiro. As respostas que o atacante terá serão de um sistema operacional falso com serviços falsos, onde é possível até a simulação de falhas de segurança. Caso o atacante descubra que se trata de um *honeypot* este poderá se direcionar à invasão do sistema operacional real, onde se encontra o *honeypot* (MARCELO; PITANGA, 2003);

c) baixa interação: quando a interação é baixa, não é permitido a interação do atacante com o sistema. O atacante é limitado a interagir com os serviços do sistema. Como exemplo, poderia ter a emulação de um sistema operacional Unix rodando um serviço de *telnet.* O atacante poderia cair na tela de *prompt* de *login*, onde ele apenas conseguiria fazer tentativas de validações, enquanto o sistema armazenaria as tentativas, não tendo na verdade nenhum sistema operacional para ele se logar. A interação estaria limitada nas tentativas de login (AMARAL, 2006).

4.3 A IMPORTÂNCIA DOS HONEYPOT'S

A grande quantidade de dados que pode ser capturados diariamente, como *log's* de *firewall's* e alertas de IDS pode ser até prejudicial para uma organização, pois é possível que existam vários falsos positivos e por isso sejam ignorados. Talvez ataques reais sejam ignorados sendo tratados pela organização como falsos positivos, e dessa forma se transformando em algo equivalente a falsos negativos, pois a atividade maliciosa não é detectada pela organização. Não por falha do IDS na não detecção da atividade ilegal, mas sim por um possível ataque real ser ignorado ao ser cofundido com um falso positivo pelo fato dos alertas não serem testados um-a-um.

Os *honeypot's* coletam apenas informações reais destinadas a eles, onde normalmente são poucas, se comparado ao resto, mas de muita importância, pois são de comprometimentos reais e são considerados maliciosos. Podem ser utilizados para estatísticas, detecção de ataques, ou até mesmo para pesquisas (SPITZNER, 2003, Tradução nossa).

Um desafio enfrentado é onde a maioria dos sensores de IDS tem dificuldade na monitoração de redes de *gigabit's* de velocidade, onde pacotes são perdidos e possíveis tentativas de ataques podem deixar de serem registradas. Um *honeypot* montado nessas circunstâncias não passaria por esse problema, pois registraria apenas o tráfego direcionado a ele, onde além da redução e confiança das informações nele registrado, não há necessidade de grandes investimentos financeiros. *Honeypot's* não exigem uma grande quantidade de memória ou processamento (AMARAL, 2006).

4.4 HONEYNETS

Uma *honeynet* pode ser definida como uma rede de *honeypots*, onde todo o tráfego que entra e sai dela pode ser considerado malicioso, pois ninguém conhece os computadores dessa rede e nem essa rede, logo todo pacote que se destine a ela pode ser considerado um possível ataque.

Existem hoje duas gerações de *honeynet's:* a *Gen I*, e a *Gen II*. A *Gen I* é a primeira geração, onde são utilizadas tecnologias básicas para capturar um atacante, como a não utilização de IDS, por exemplo. A *Gen II* surgiu com a visão de um computador que age como *firewall* e IDS ao mesmo tempo. Já se tem a idéia da utilização da *Gen III*, que deve incluir recursos de virtualização mais complexos (ASSUNÇÃO, 2009).

Existem as *honeynets* reais, onde a rede em si é real, com componentes físicos reais e diversos mecanismos para o alerta e informação sobre o ataque. Como vantagem pode-se citar a sua tolerância a falhas, pois o ambiente é distribuído como também possibilitar um detalhamento mais aprofundado dos ataques. Como desvantagem o custo em si dos equipamentos, como também o espaço físico e com isso gerando uma manutenção mais trabalhosa.

Para a redução de custos, é possível a virtualização de várias máquinas em apenas uma, gerando assim o conceito de *honeynet* virtual, onde é utilizado apenas um *honeypot* físico para emular os demais. Nesse tipo, todo o ambiente é preparado para a emulação de vários sistemas operacionais e serviços através de uma única máquina. Além da redução de custos, de espaço e de energia para a sua implementação, também tem como vantagem a facilidade de instalação. Mas o fato de todos *os honeypots* estarem em uma única máquina, o atacante pode perceber alguma relação entre elas, como por exemplo, a queda de desempenho

do servidor pode ocasionar o mesmo em todas as máquinas (WEISSHEIMER JÚNIOR, 2007).

4.5 HONEYNET PROJECT

É uma organização sem fins lucrativos focando a melhoria da segurança na *internet,* dedicando gratuitamente os desenvolvimentos mais atuais em investigações. Desde a sua fundação, em 1999 a *Honeynet Project* disponibiliza serviços como de aumentar a percepção sobre as ameaças e vulnerabilidades existentes na internet hoje, conscientizando indivíduos e empresas que não têm conhecimentos de que podem ser alvos de atacantes e quem são esses atacantes. Para as empresas que têm interesse em continuar as investigações por conta própria a *Honeynet Project* disponibiliza técnicas e ferramentas para auxílio.

Essa organização tem contribuído para a criação de avisos, correções e novas ferramentas, auxiliando no combate de ações ilegais realizadas pelo mundo inteiro (NUNES, 2006).

4.6 PROJETO HONEYNET.BR

O projeto *Honeynet.BR* constitui o *Brazilian Honeypots Alliance*, ou Projeto *Honeypots* Distribuídos que é uma associação de *honeypots* distribuídos espalhados pelo país. Tem como objetivo o aumento da capacidade de detecção de incidentes e também determinar tendências de ataques na Internet brasileira.

Para isso, são feitas diversas redes de *honeypots* de baixa interatividade com o objetivo de formar uma base de dados a ser reunida e analisada (SUZIKI; DELPHINO, 2007).

4.7 PROJETO HONEYPOTBR

O surgimento da *honeypotBR* foi pelas mãos de um grupo de especialistas em segurança e também de pesquisadores independentes que formaram uma entidade independente, sem nenhuma ligação governamental ou mesmo com o Projeto *Honeynet.* As ferramentas criadas são servidores nulos (*fake servers*), que emulam serviços de *Proxy, Telnet, SMTP e HTTP*. De acordo com Marcelo e Pitanga (2003) as ferramentas são:

a) *Fakesquid,* que responde a conexões *telnet* e armazena *o IP* de origem do atacante, data, hora e todas as informações de *whois.* Alertas em tempo real;

b) *Fakehttp,* que responde a conexões *telnet* na porta 80, emulando um servidor *web apache* versão 1.3.27;

c) *Faketelnet,* onde se trata de um servidor falso que responde a conexões *telnet* a porta 23 e simula um servidor *telnet;*

d) *Fakesmtp,* onde é um servidor de médio envolvimento, simulando um servidor de correio eletrônico como o *SendMail, Excharge* entre outros. Também responde a conexões *telnet* à porta 25. Armazena o *IP* do atacante, data, hora e comandos. Os alertas são em tempo real.

4.8 HONEYTOKENS

Trata-se de um trecho de informações falsas colocadas dentro de um *honeypot* que tem como objetivo chamar a atenção do atacante, como um arquivo com o nome “senhas.txt”, “salários.xls”, ou até mesmo uma conta de usuário com nomes provocativos, como “diretor”

ou “presidente”, por exemplo. Pode-se criar regras no IDS para a monitoração dos *honeytokens* que foram criados (ASSUNÇÃO, 2009).

5 FERRAMENTAS HONEYPOT'S

Os *honeypots* podem ser baseados em *softwares* livres, gratuitos ou comerciais, construídos com o objetivo de estudos ou de uso profissional. Serão descritos alguns dos principais *honeypots* existentes no mercado.

5.1 DECEPTION TOOLKIT (DTK)

Este *honeypot* pode simular os serviços do sistema. Ele pode estar sendo executado em um ambiente *Linux,* mas pode simular ao atacante um outro ambiente, como o *Solaris*, por exemplo (MARCELO; PITANGA, 2003). Trata-se de um *honeypot* de baixa iteração lançado em 1997, onde este é o primeiro *honeypot* baseado em *software* (ASSUNÇÃO, 2009).

5.2 KFSENSOR

O *KFSensor* é um *honeypot* de licença comercial desenvolvido para o *windows.* É de baixa interatividade, que pode simular respostas como se fossem dados reais. Com ele é possível a instalação em locais estratégicos da rede, onde se pode monitorar o tráfego e enviar respostas para o computador que está rodando o *KFSensor*.

Os *log's* também são divididos para cada cenário, a fim de facilitar não somente a visualização, mas também a busca de eventos que ocorreram naquele determinado local. É guardado um registro completo para cada detecção feita. Como está mostrado na figura 2 com apenas com um duplo clique é possível ter várias informações em sua janela com quatro abas, que são (ASSUNÇÃO, 2009):

a) *summary:* Mostra informações gerais, como o sensor, id, porta, sensor que detectou o ataque, data, hora, etc.;

b) *details:* Detalhes técnicos sobre o evento, como as interações que o invasor realizou;

c) *signature:* Apresenta a assinatura do evento, que pode ser alterada conforme a necessidade;

d) *data:* Outros dados recolhidos durante a invasão.

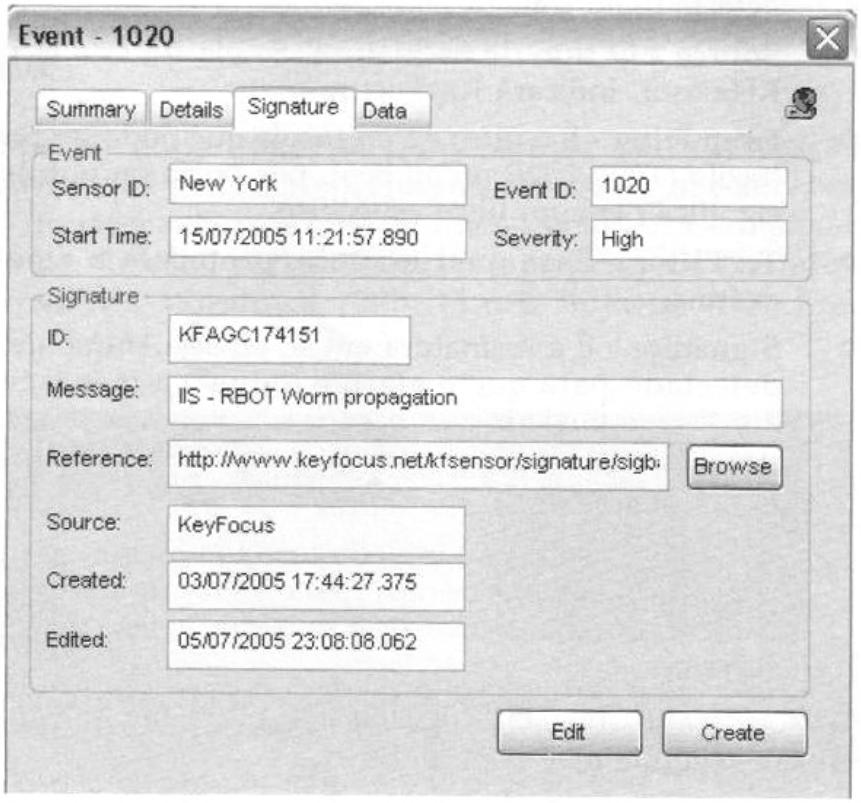


Figura 2: Detalhe da aba "*Signature" do KFSensor*

Fonte: ASSUNÇÃO (2009)

5.3 BACKOFFICER FRIENDLY (BOF)

É um pequeno *honeypot* para ambientes *windows* de baixa interatividade. Ele é muito simples, é grátis e ideal para iniciantes, onde também seus serviços, instalação, configuração e manutenção são simples. O número de serviços é pequeno, assim como também a sua capacidade de emulação (AMARAL, 2006).

5.4 HONEYPERL

É um *honeypot* desenvolvido pelo grupo *HoneypotBR,* da família *FakeSquid, FakeHttp, Faketelnet e Fakesmtp.* Escrito em *perl* que responde a conexões *telnet* nas portas 23, 80, 3128 ou a qualquer porta a escolha do usuário. Ele armazena as tentativas de conexões com o *IP* de origem, data, hora e outras informações (MARCELO; PITANGA, 2003).

5.5 NETBAIT

O *NetBait* é capaz de redirecionar ataques contra faixas de *IP* não autorizadas para "fazendas de *honeypots*", criando uma grande ilusão ao atacante. É criado um desvio da rede real, onde o intruso fica cercado de vários nós falsos com configurações diferentes (MARINHO, 2004).

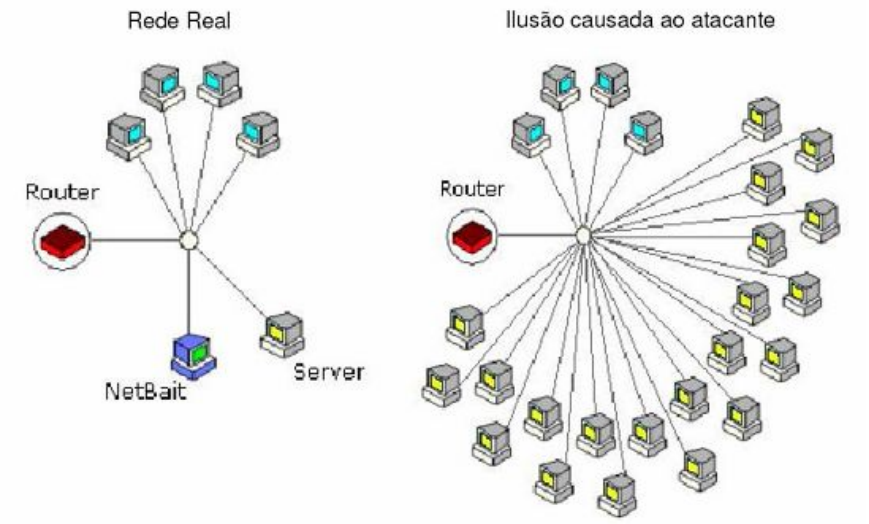


Figura 3 – Ilusão causada no atacante pelo *NetBait*.

Fonte: Marinho (2004)

5.6 SPECTER

É um *honeypot* para *windows* que pode monitorar até quatorze portas *TCP*, sendo sete para armadilhas e sete para serviços. As portas de serviços interagem com o invasor emulando o aplicativo de acordo com o serviço a ser usado, e as armadilhas registram e bloqueiam as tentativas de ataques.

É um *honeypot* de baixa interação e que possui grande quantidade de configurações, características e de facilidade de uso. Pode emular também até quatorze sistemas operacionais diferentes, mas não consegue emular uma pilha IP (ROJAS, 2003).

5.7 VALHALA HONEYPOT

Foi desenvolvido para sistemas *Windows*, embora funcione em *linux* por meio de *software* de emulação, como o *Wine*. O *honeypot* possui serviços de baixa e também de alta

interação, e tem como objetivo fornecer a funcionalidade da detecção de intrusos de forma gratuita, simples e sem muita complexidade de implementação.

Os serviços que o *honeypot* suporta são os de *HTTP, FTP, SMTP, POP3, TELNET, TFTP, FINGER,* e *PROXY,* embora ainda não tenha suporte para a utilização de captura de *ip's* disponíveis na rede para a criação de *hosts* virtuais, por isso deve ser utilizado o endereço de *ip* atual da máquina para a utilização do programa (ASSUNÇÃO, 2009).

5.8 HONEYD

É um *honeypot* de baixa interatividade, que pode ser utilizado como *honeypot* de produção ou de pesquisa, pois além de visar a detecção de ataques possui aplicações específicas de pesquisa. É baseado em *software* livre que através de *scripts* é capaz de simular vários sistemas operacionais e são suportados os protocolos *UDP, TCP* e *ICMP* (SUZIKI; DELPHINO, 2007).

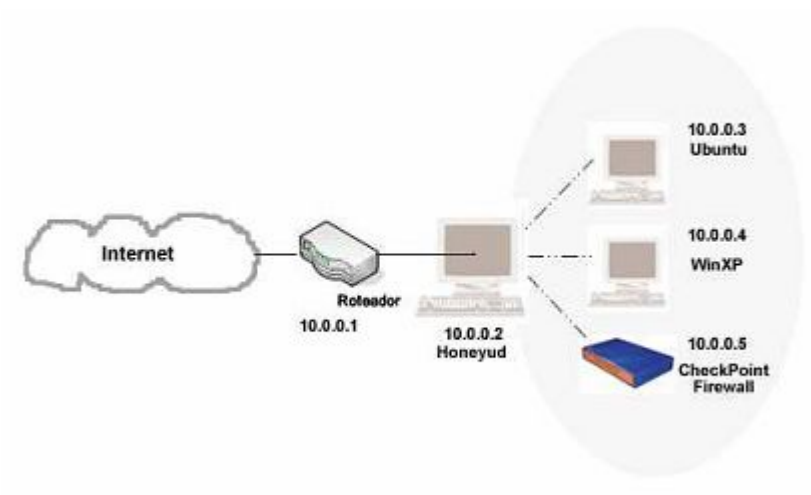


Figura 4 – Ilustração de um ambiente criado pelo *Honeyd*

Fonte: SUZIKI, DELPHINO (2007)

Ele além da emulação de sistemas operacionais, emula também serviços, inclusive pode enganar *scanners*, como o *nmap*, por exemplo. A utilização do *honeyd* deve ser feita com um programa chamado *arpd*, onde como o *honeypot* não pode direcionar a si os ataques nos *ip's* disponíveis na rede, o *arpd* os falsifica como endereços válidos, e o *honeyd* respondem por eles. Em uma suposição onde em uma rede se encontre quatro endereços livres para a utilização, iniciando de 192.168.1.4, 192.168.1.5, 192.168.1.10, e o endereço 192.168.1.12, ao ser instalado o *honeyd* em uma máquina, ele poderá virtualizar os endereços como se fossem válidos para o atacante. Poderá ser configurado para responder para estes endereços com serviços e servidores falsos, por exemplo, se um intruso fizesse um scanner no *ip* 192.18.1.5, poderia existir uma configuração para dar a resposta de como se fosse um *windows* 2000 com *IIS* 5.0 (MARCELO; PITANGA, 2003).

Deverá ser feita a verificação se a *libcap, libevent* e *libdnet* estão instaladas no sistema, pois o *honeyd* necessita para o seu funcionamento. Existem versões tanto para *linux* como também para *windows* (ASSUNÇÃO, 2009).

Com o *honeyd* é possível também a criação de redes virtuais ou *honeynets* que possuam servidores, roteadores e clientes onde cada um pode ter a sua característica própria, podendo assim ser formada uma topologia de rede (WEISSHEIMER JÚNIOR, 2007).

5.9 PATRIOTBOX

Trata-se de um *honeypot* para *Windows* com licença comercial, que possui vários recursos interessantes, como o de aumentar a sua funcionabilidade baixando *scripts* e *plugins* novos e também de possuir suporte para a utilização de scripts do *honeyd.* Com ele é possível selecionar o sistema operacional a ser emulado, junto com os serviços disponíveis ao

sistema escolhido. Podem ser utilizados vários serviços, desde *FTP* a cavalos de tróia, como o *Netbus*, por exemplo (ASSUNÇÃO, 2009).

6 TRABALHOS CORRELATOS

Os trabalhos que aqui estão descritos foram feitos na mesma linha de pesquisa que este, onde foi de auxílio também para algumas questões sobre a utilização da ferramenta *honeyd*.

6.1. IMPLANTAÇÃO DE UM HONEYPOT E PROPOSTA PARA METODOLOGIA DE GERENCIAMENTO DE PROJETOS DE HONEYNETS

Este trabalho foi desenvolvido em Brasília no ano de 2007, onde teve como objetivo a implantação d um honeypot de baixa interatividade para a análise de ataques e também propor um modelo de metodologia para o gerenciamento de projetos honeynets, afim de auxiliar os administradores de redes a desenvolverem projetos com essa tecnologia. Devido a problemas no desenvolvimento do trabalho, como a indisponibilidade do cpd da universidade para efetuar a parte prática a pesquisa foi implementada em uma residência.

Embora as proporções para a implantação do honeypot se tornaram menores, com o estudo e a exploração dos recursos do honeypot, pode-se concluir que com o fato de um bom planejamento em sua implantação o planejamento tornam-se importante para a exploração futura do potencial de uma honeynet com honeypots de alta interatividade.

6.2. ANÁLISE DE INTRUSÕES ATRAVÉS DE HONEYPOTS E HONEYNETS

Nesta pesquisa, onde são abordados de forma profunda os conceitos sobre honeypot e honeynet, assim como os seus funcionamentos, técnicas para a análise, projetos sobre esse tema que estão sendo realizados, e algumas ferramentas comerciais. São

apresentados também outros níveis de honeypots, como honeytokens, honeyfarm, e honeypots dinâmicos.

6.3. HONEYNET – ESTUDO TEÓRICO E PRÁTICO

São abordados a importância da segurança das informações em ambientes corporativos, a descrição da ferramenta honeyd bem como a sua implementação real dentro de uma universidade. A proposta inicial era implementar a ferramenta em sua versão para windows e para linux, mas devido a problemas apenas a implementação na versão para linux foi feita.

6.4. ESTUDO E IMPLANTAÇÃO DE FERRAMENTAS HONEYPOTS NA REDE DA UNIMONTES

Este trabalho tem como objetivo o estudo de ferramentas honeypots bem como a sua implantação na rede da Universidade Estadual de Montes Claros, com o objetivo de registrar o tráfego malicioso que possa existir. Por causa de problemas encontrados, foram implementados e registrados apenas o tráfego interno da rede local de computadores, onde por meio dos *log's* foi visto que existe tráfego autônomo na rede, com pouco fluxo malicioso.

7 IMPLEMENTAÇÃO DO HONEYPOT PARA REGISTRO DE LOGS

A implementação do *honeypot* em ambiente *Linux* foi realizada com uma simulação muito próxima do que poderia acontecer em um determinado caso de um ambiente real especifico. A simulação foi realizada com a ferramenta *honeyd* e com o *IDS snort*, onde para a visualização foi utilizada uma interface chamada *BASE (Basic Analysis and Security Engine),* que necessita do servidor *apache*, do *php*, que se trata de uma linguagem de programação para a *web*, e do banco de dados *mysql* para registrar os *log's* no banco de dados.

Os registros do *honeyd* foram feitos localmente em arquivos texto puro (*txt*). A visualização para análise dos *log`s* foi realizada de duas formas, uma por meio do próprio arquivo de texto e a outra com a ferramenta *honeydsum*, que é uma ferramenta para a visualização dos *log's* em forma de gráficos. Na figura 5 pode-se verificar alguns gráficos gerados pela ferramenta *honeydsum*.

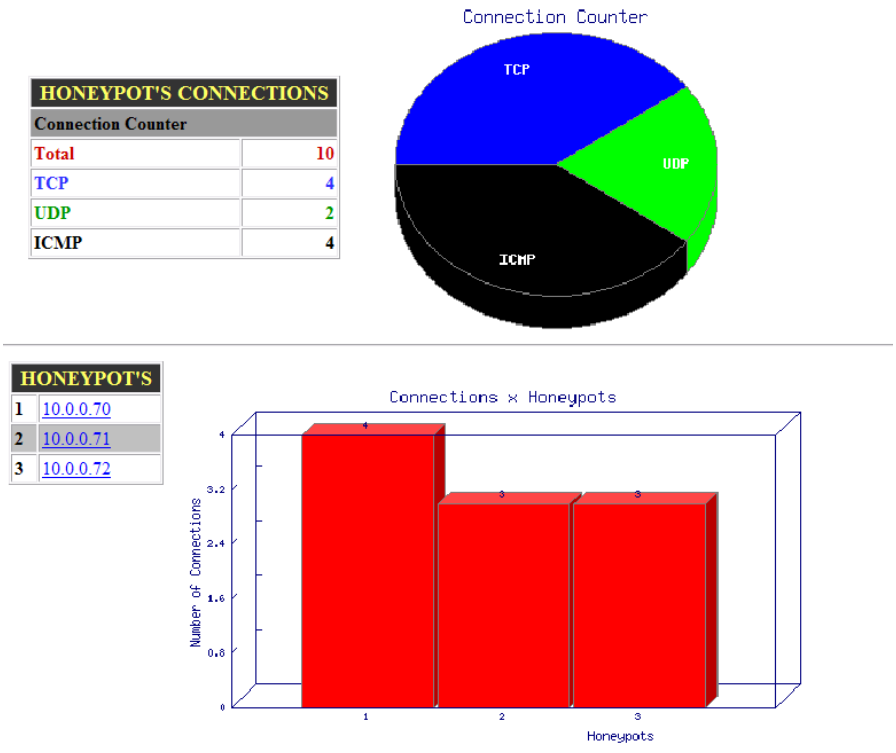


Figura 5 - Exemplos de gráficos gerados pelo *HoneydSum.*

7.1 FERRAMENTAS PARA O AMBIENTE OPERACIONAL

Existem vários motivos para a escolha de o *Linux* ser feita, tal como o preço, por exemplo, onde não apenas um sistema operacional gratuito, mas também o desenvolvimento de forma transparente, e o código fonte livre.

O projeto *GNU* tem como objetivo o desenvolvimento de um sistema operacional do tipo *unix* que seja totalmente livre. A utilização do *linux* como *kernel* em variantes do sistema *GNU* ocorre onde são chamados por "*linux*", onde na verdade são "*GNU/Linux*" (NORTON, GRIFFITH, 2000).

O *VirtualBox* é um programa utilizado para a virtualização de sistemas operacionais, criado pela *Sun Microsystens* foi utilizado para o ambiente de testes ser implementado. No computador onde foi instalado o *honeyd*, ferramenta *honeypot* selecionada para a análise, é um *notebook* com o sistema operacional *Ubuntu* 8.04, foi instalado o *VirtualBox* e nele 3 máquinas virtuais com o sistema operacional *Windows XP*, a fim de se fazer um ambiente o mais próximo possível do real, onde as máquinas são representadas como máquinas normais de produção a uma determinada situação, assim como também no computador de onde foram feitos os ataques, que tem o sistema operacional *Windows XP* e foi instalado o *Ubuntu 9.04* pelo *VirtualBox*. O fato de o ambiente ser feito através de máquinas virtuais e não máquinas reais, devem-se a não ser necessário investimentos de *hardware* para se ter uma rede de computadores devido as facilidades que as máquinas virtuais disponibilizam, como também a situação física atual do acadêmico a realizar este trabalho, que tornaria inviável a locomoção até a universidade para a construção do ambiente e a realização dos testes durante o semestre. Foi também utilizado um modem roteador e um *hub*, onde o ambiente para a análise do *honeypot* pode ser visualizado na figura 6.

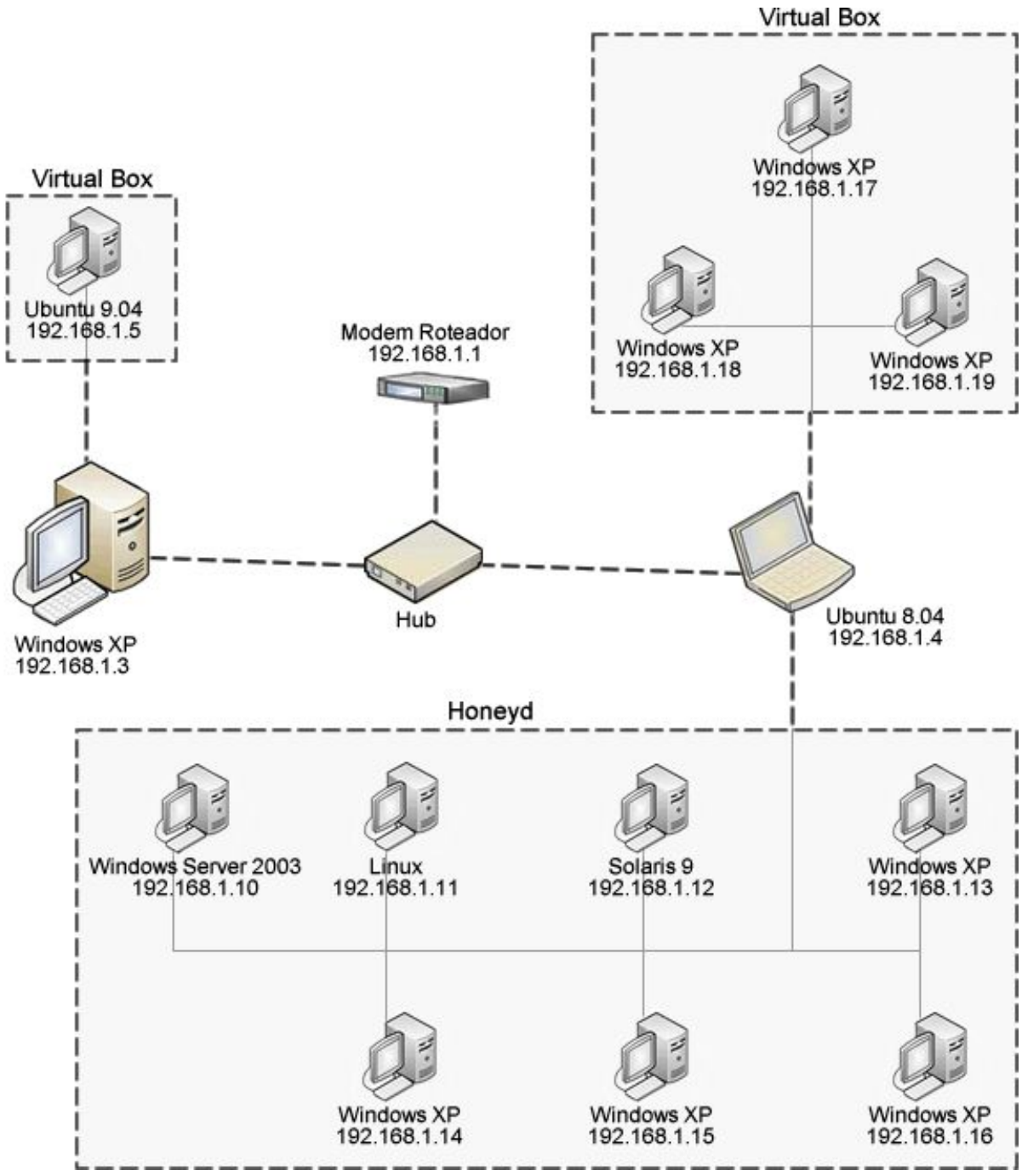


Figura 6 - Ambiente montado para a realização dos testes com o *honeypot.*

Para o correto funcionamento das máquinas virtuais no *Ubuntu* foram encontradas algumas dificuldades, onde a instalação é explicada com detalhes através do Anexo A. Após ter sido instalado o *VirtualBox* assim como as máquinas virtuais, quando foram feitos testes para a comunicação com a rede da máquina hospedeira, observou-se que não havia

comunicação. Devido a isso, foram realizadas pesquisas em fóruns na internet, que se obteve como resultado a configuração de acesso à rede hospedeira via *brigde*, que tem como objetivo a ligação entre duas redes distintas, servindo assim de ponte entre elas. Para que seja feita a configuração, deve-se inicialmente fazer a instalação do pacote de utilitários *brigde* para o *Ubuntu (brigde-utils)*, que é realizado através do comando: *sudo apt-get install bridge-utils*. Depois deve-se acrescentar no arquivo */etc/network/interfaces* as linhas vistas na figura a seguir para criar um *brigde* que será chamado de *br0.*

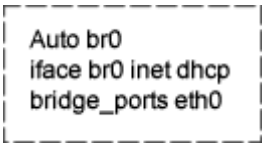


Figura 7 - Adição de comandos para criação de um *brigde.*

Logo se reinicia a rede do hospedeiro com o comando *sudo etc/init.d/networking restart*, e assim será feita a criação automática de um *brigde* em cada *boot* do hospedeiro. O próximo passo foi criar uma i*nterface* permanente no *host*, que será chamada de *vbox1*, *vbox2* e *vbox3*, onde cada uma é referente a cada máquina virtual criada. A criação é feita com o comando *sudo VBoxAddIF <interface permanente> <usuário> br0*, onde <*usuário*> é referente ao nome do usuário hospedeiro que deverá utilizar a máquina virtual e <*interface permanente*> é a interface a ser criada, nas quais foram descritas anteriormente. Por último deve-se ir para a configuração da máquina virtual, e escolher a opção de rede de acordo com a Figura 8, onde a placa de rede deve ser ligada a *interface* do hospedeiro e o nome da placa de rede são as *interfaces* anteriormente criadas, *vbox1*, *vbox2* e *vbox3*.

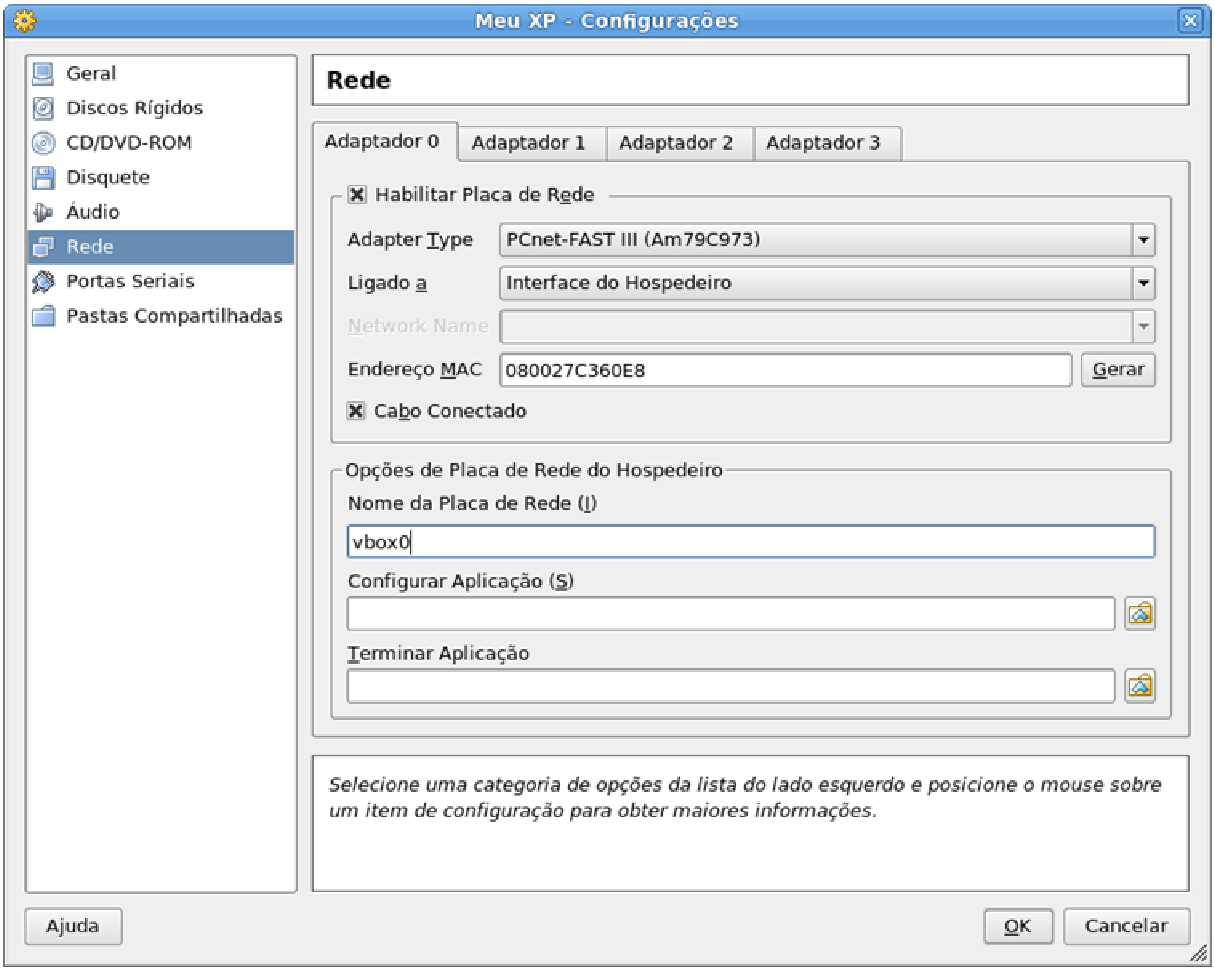


Figura 8 - Configuração realizada após o hospedeiro estar configurado.

7.2 FERRAMENTA HONEYPOT SELECIONADA PARA A ANÁLISE

O *honeyd* foi a ferramenta selecionada para os testes. Com ele é possível a criação de *hosts* virtuais onde através de *scripts* que são criados pelo administrador podem conter vários serviços capazes de interagir em determinadas portas. Possui seu código-fonte livre e é o mais conhecido entre os *honeypots*. Existem versões para *Windows* e para *Linux*, porém será abordada apenas a versão para *Linux*. Para que seus *scripts* sejam utilizados na rede, é necessário ter endereços de *ip's* não utilizados, para que sejam definidos os endereços a serem utilizados pelo software (ASSUNÇÃO, 2009).

No funcionamento são criados de *hosts* virtuais que podem emular vários serviços, como por exemplo, *ftp, telne*t ou *email.* São utilizados vários endereços de *ip's* não utilizados na rede associados a cada *host* virtual um *ip* diferente, mas necessita do *proxy arp(arpd)* para poder alocar os endereços não utilizados para o *honeyd*. O *arpd* é um programa com o objetivo de alocar os endereços não utilizados para o *honeyd*, pois sem este, o *honeyd* por si não seria capaz de fazer este serviço sozinho (WEISSHEIMER JÚNIOR, 2009).

Quando algum pacote *tcp, udp*¸ou *icmp* é endereçado para algum *honeypot* virtual que pertence ao *honeyd*, é verificado qual o tamanho do pacote *ip* e o seu *checksum*, ou seja, se existe algum erro. Os pacotes de outros protocolos são ignorados. Existe uma central de envio de pacotes para que sejam processados os pacotes por ele recebidos, e é verificado na base de dados de configurações se tem algum *honeypot* correspondente ao endereço de *ip* solicitado. Caso não exista, um *template* padrão pode ser opcionalmente utilizado, que significa que um *host* virtual do *honeyd* pode responder pelos endereços que não são utilizados pelo *honeypot* e nem pela rede. Em seguida a central de envio de pacotes chama a parte específica que cuida do protocolo do pacote recebido e também a configuração que correspondente do *honeypot*. No protocolo *icmp* o único pacote suportado é o *ICMP_ECHO request* (AMARAL, 2006). Na figura 9 é mostrada a arquitetura básica do *honeyd*.

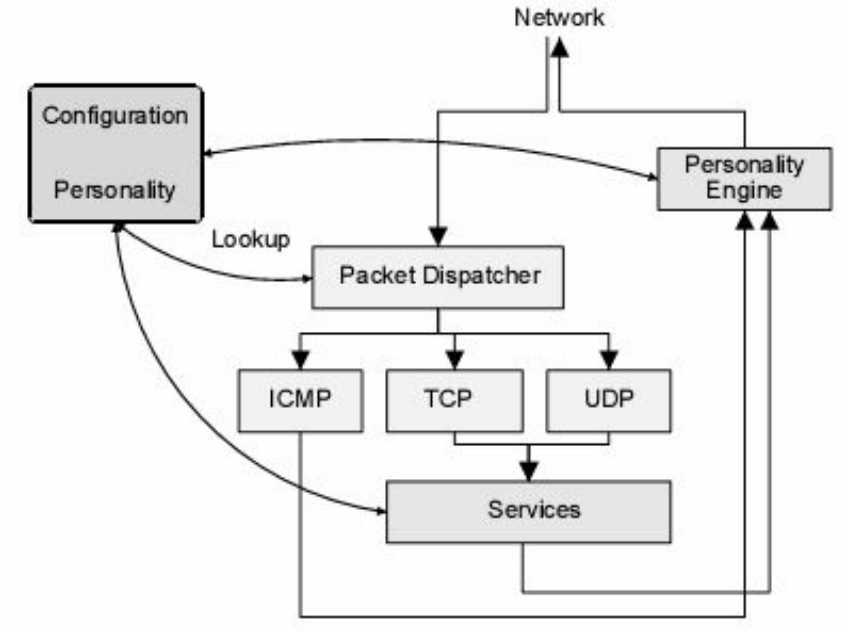


Figura 9 - Arquitetura básica do *honeyd*.

Fonte: AMARAL (2006)

O fato de associar cada *host* a um *ip* livre da rede, dessa forma consegue-se enganar *scanners*, pois ao *scannear* um endereço que na realidade está livre, mas sendo utilizado pelo *honeyd*, um atacante poderia encontrar como resposta um *Windows* 2000 com o *IIS* 5.0 ou um outro sistema operacional com outros serviços, conforme a configuração feita no *honeyd*. (MARCELO; PITANGA, 2003).

Na instalação, são necessárias as seguintes bibliotecas: *libevent,* que se trata de uma biblioteca que fornece mecanismos para identificar quando um evento ocorrer, sendo assíncrona para a notificação dos mesmos. A biblioteca *libdnet* que contém vários *scripts* para auxiliar o tratamento dos pacotes como também disponibiliza um *framework* para trabalhar de baixo nível à recepção e tratamento de pacotes, como a injeção de novos na rede. A biblioteca *libpcap* tem por objetivo capturar todos os pacotes que passam por ela, que trabalha na camada *ethernet,* e o *arpd,* explicada anteriormente (WEISSHEIMER JÚNIOR, 2009).

Como o sistema operacional utilizado foi o *Ubuntu* na sua versão 8.04, o seguinte comando instala o *honeyd* junto com todas as suas dependências: *apt-get install honeyd,* e o *arpd,* que na realidade trata-se do *fake-arpd,* é instalado com o seguinte comando: *apt-get install farpd.*

A configuração do *honeyd* é feita através do arquivo *honeyd.conf* que após a instalação no *Ubuntu* pode ser localizado no caminho */etc/honeypo/honeyd.conf*. Caso exista alguma dificuldade para a sua localização, pode ser encontrado através do comando *find / -name honeyd.conf.* Esse arquivo contém *scripts* gerados pelo administrador para enganar o invasor e deve ser configurado com cuidado, pois senão os resultados apresentados podem ser comprometidos.

A utilização do arquivo *honeyd.conf* é baseado na criação de *script's* que emulam os sistemas operacionais e serviços dos mesmos, onde os principais comandos para a configuração do arquivo é mostrado no Apêndice A, e no Apêndice B se encontra o arquivo de configuração *honeyd.conf* com os *scripts* utilizados para esse trabalho. Além desse arquivo, existem outros arquivos que podem ser alterados, que são os *scripts* específicos dos serviços a serem emulados, onde podem ser feitas modificações, como o local independente onde armazenam os *log`s.*

Foram criados um total de 7 (sete) *hosts* virtuais com o *honeyd*. O primeiro *host* criado foi o *Windows Server 2003* e foi colocado para ele o *ip* 192.168.1.10, com os serviços de *ftp* na porta *tcp* 21, serviço de *smtp* na porta 25, servidor *iis* na porta 80 e serviço de *pop3* na porta 110. A porta 20 foi deixada aberta. As outras portas *tpc* e *udp* foram deixadas fechadas. No *script* do serviço para emular um servidor *iis* é possível colocar uma página personalizável em linguagem *html,* que ao ser acessado o *ip* da máquina por um *browser* de algum outro computador da rede, será apresentado a página *html* editada no *script*, de forma que aparente ser um servidor real.

No segundo *host* virtual foi dado o *ip* 192.168.1.11 e criado o sistema Operacional *Linux*, onde foi utilizado a versão do seu *kernel*, como 2.4.7 com os serviços de *telnet* na porta 23, *sendmail* na porta 25, *pop3* na porta 110 e *squid* na porta 3128. Foram deixadas as portas *udp* 97 e tcp 150 abertas. As outras portas *tcp* e *udp* foram deixadas flechadas.

O sistema operacional *Solaris* em sua versão 9 foi o terceiro *host* virtual a ser criado com o *ip* 192.168.1.12, e as portas *tcp* 3321, 4431 e 5321 deixadas abertas. Os próximos 4 *hosts* que foram criados são com o sistema operacional *windows XP*, afim de serem confundidos por um invasor com os computadores criados no *VirtualBox*, afim de simular computadores de produção. Os *ip's* referentes a estes *host* são 192.168.1.13, 192.168.1.14, 192.168.1.15 e 192.168.1.16 com as portas *tcp's* 135, 139, 445 e portas *udp's* 123, 137, 138, 445 e 500 deixadas abertas. As outras portas foram fechadas. Em todos os *hosts* e máquinas virtuais criadas foram também especificados endereços de *mac's* simulando endereços físicos individuais para as placas de rede.

No computador utilizado para a realização dos testes, que contém o *Windows XP* foi também instalado o *VirtualBox* e nele instalado como máquina virtual o *Ubuntu* em sua versão 9.04, onde o sistema operacional *Windows XP* tem o *ip* 192.168.1.3 e a máquina virtual o endereço 192.168.1.4.

Para inicializar o *honeyd*, deve-se antes inicializar o *arpd*, por meio do comando *sudo farpd* 192.168.1.0/24 onde o comando indica que o *farpd* redirecionará ao *honeyd* os endereços referentes a rede 192.168.1.0, ou especificar os endereços individuais deixando-se um espaço entre eles, com o comando *sudo farpd* 192.168.1.10 192.168.1.11 192.168.1.12 192.168.1.13 192.168.1.14 192.168.1.15 192.168.1.16 . Depois na pasta */etc/honeypot* foi executado o seguinte comando:

sudo honeyd –p nmap.prints –f honeyd.conf -l

/home/anderson/Documentos/honeyd/logs_honeyd.log 192.168.1.10-192.168.1.16

A seguir, tem-se a completa explicação dos comandos (Assunção 2009):

a) *honeyd* é o programa em si;

b) a opção *–p nmap.prints* tem como objetivo fazer o *nmap* identificar o *host* virtual da forma que foi configurado no *script;*

c) a opção *–f* indica a localização do arquivo que contém os *scripts* dos *hosts* virtuais;

d) a opção *–l* indica a localização do arquivo de *logs*, porém além desse arquivo, existem os arquivos individuais referentes aos *logs* de cada serviço utilizado.

e) a faixa de endereços indica quais *ip's* serão reservados para que seja utilizado pelo *honeyd*.

Além do arquivo de *log* que contém todas as conexões realizadas ao *honeypot* também existe o registro dos *log's* individuais de cada serviço específico. Se por exemplo, em uma tentativa de conexão *telnet* além das conexões feitas ao *honeypot* estariam também registrados os comandos utilizados na tentativa de invasão, que pode ser armazenado em um outro arquivo independente. Para cada serviço é possível configurar um arquivo individual de *log*, em seu *arquivo.sh*, mas em todos os serviços utilizados pelo *honeyd* nesta pesquisa foram configurados os serviços individuais para registrarem os *log's* em um único arquivo, que se localiza em */home/anderson/Documentos/honeyd/logs_honeyd.log*.

Em alguns arquivos de serviços específicos não constava o comando referente ao caminho de armazenamento de *log*, onde nesses casos foram adicionados. O comando foi copiado no mesmo formato dos arquivos *.sh* que já tinham o comando e obteve-se sucesso com a tentativa. Aqueles que já tinham apenas foram modificados os caminhos.

Uma dificuldade encontrada para fazer os serviços funcionarem foi que na tentativa de inicializar o *honeypot* era retornado uma mensagem de erro, onde na mensagem o arquivo *base.sh* não era encontrado. Em cada arquivo de cada serviço específico é necessário a localização para este arquivo, e estava como

. *scripts/misc/base.sh*, mas o arquivo *base.sh* se localiza em outro local, onde no *script* foi realizado a mudança de local no mesmo formato que estava anteriormente, ficando assim:

. *scripts/base.sh*, onde também não se obteve sucesso na mudança. Depois com a tentativa de colocar o caminho completo /*usr/share/honeyd/scripts/base.sh* nos arquivos específicos obteve-se a eliminação do erro. Essa correção foi realizada em todos os arquivos de serviços utilizados no trabalho.

A pasta *scripts* onde contém os scripts dos serviços utilizados localizada em */usr/share/honeyd* não veio junto com a instalação do *honeyd*, onde se obteve com a instalação do *honeyd-common* no gerenciador de pacotes do *Symaptic*. Assim foi selecionado o pacote e aplicado para a instalação. Foi selecionado também o pacote *issemulator*, que fornece o serviço de emulação *iis*.

Para uma melhor visualização dos *log's* foi utilizada a ferramenta *HoneydSum* em sua versão *0.3*. A ferramenta pode ser baixada através do endereço *http://www.honeynet.org.br/tools/* que é do projeto *honeynet.BR*. De acordo com o arquivo *readme.txt* que vem junto com a ferramenta, ela é utilizada para mostrar os *log's* de forma resumida utilizando parâmetros para filtrar as informações, como portas, protocolos, endereços de *ip* ou de redes. Ele mostra as conexões por hora, podendo-se utilizar vários arquivos de *log's* para se poder ter uma estatística deles. A sua visualização é feita através da geração de páginas na linguagem *html* com vários gráficos ilustrando as informações mostradas.

Para que a ferramenta funcione, no arquivo *readme.txt* também indica que são necessários alguns pacotes instalados, que são:

a) *Perl*: que se trata de uma linguagem de programação;

b) *Net:Netmask*: que é um módulo para o *perl*

c) módulos *GD*: são pacotes específicos para o *perl*, como o *GD::Graph::pie, GD::Graph::bars* e o *GD::Graph::bars3d.*

A instalação do *HoneydSum* não é necessária, basta executar o comando correto para que a geração dos gráficos e as páginas *html* necessárias sejam criadas. O comando utilizado nesse trabalho foi o seguinte:

./honeydsum.pl –c honeydsum.conf –w /home/anderson/Documentos/honeyd/logs_honeyd.logs

O comando feito já dentro da pasta onde se encontram os arquivos da ferramenta, e a pasta foi colocada no endereço */etc/honeypot*, dentro da pasta que se encontra o arquivo *honeyd.conf*. A explicação do comando é a seguinte:

a) *./honeydsum.pl*: este é o comando para inicializar o *honeydsum*;

b) –c *honeydsum.conf*: o parâmetro *–c* indica o arquivo de configuração *honeydsum.conf;*

c) –*w*: com este parâmetro será gerado os gráficos e páginas em *html* necessários para a visualização dos *log's*.

d) /home/anderson/Documentos/honeyd/logs_honeyd.logs: esta é a localização do arquivo de *log* que servirá para que as informações possam ser mostradas de forma gráfica. O *honeydsum* permite que vários arquivos sejam incluídos, colocando a localização de cada um, e deixando um espaço entre um e outro.

Ainda no arquivo *readme* da ferramenta, existem explicações sobre a configuração da mesma. Algumas opções foram configuradas no arquivo *honeydsum.conf* antes de executar a ferramenta, que são as seguintes:

a) *honeyd_conf = ../honeyd.conf* : deve-se colocar a localização do arquivo *honeyd.conf*, onde no caso está dentro de uma pasta onde se localiza o arquivo;

b) *honeypot_list* = 192.168.1.10, 192.168.1.11, 192.168.1.12, 192.168.1.13, 192.168.1.14, 192.168.1.15, 192.168.1.16 : nessa opção são colocados os *ip's* dos *host's* criados pelo *honeyd*;

c) *net_lis = 192.168.1.0/24*: essa opção corresponde a rede onde se localizam os *host's*;

d) *dest_port* = 20, 21, 23, 25, 80, 97, 110, 139, 150, 3128, 3321, 4431, 5321: lista de portas utilizadas pelos *host's* do *honeyd*;

e) *proto_list = tcp. udp. icmp*: protocolos utilizados pelo *honeypot*.

Assim, cada vez que se desejar visualizar os *log's* através da ferramenta *honeydsum* deve-se antes executar o comando descrito anteriormente para que ela faça a geração das páginas e gráficos com os registros do *honeyd* atualizados.

7.3 FERRAMENTA DE SISTEMA DE DETECÇÃO DE INTRUSÃO SELECIONADA PARA A ANÁLISE

O *Snort* é um sistema de detecção de intrusão baseado em assinaturas, onde compara o tráfego da rede com as suas assinaturas de anomalias, que é a forma utilizada na detecção de ataques. Ele é de código fonte aberto e um dos *IDS* mais famosos do mundo (SANTOS, 2005).

Ele pode selecionar em tempo real registros de pacotes e análises de tráfego unindo os métodos baseados em assinatura de ataques, anomalias e protocolos (GASPAR; JESUS; SILVA, 2008).

Com o S*nort* é possível que qualquer pessoa possa escrever as suas próprias assinaturas para a utilização, pois a sintaxe para isso é flexível e precisa. Pode ser utilizado como *sniffer*, *logger* ou *IDS*, como também existem vários *plugins* e módulos para ser utilizados. Ele pode funcionar em arquiteturas *x86* como em *amd64* e em vários sistemas operacionais, como *Linux*, *FreeBsd*, *OpenBsd* ou *Windows* (XAVIER, 2007).

Pode detectar quando uma invasão está acontecendo, baseado em suas regras que podem identificar pelas características do ataque, alterar as configurações de acordo com a necessidade, como também alertar o administrador sobre o ataque. Como toda boa ferramenta, deve ser bem implementada para bons resultados, no *Snort*, deve ser aplicadas somente ás regras que entram na realidade da rede, afim de evitar alertas irrelevantes, como a monitoração de um banco de dados M*ysql* onde não se tem o banco (SANTOS, 2005).

Quando se vai escrever uma nova regra, devem-se observar algumas definições, como se vai gerar um alerta ou apenas um *log*, de forma que a regra seja especifica e clara, alertando as ameaças com informações úteis ao administrador. Pode-se dizer que e segurança do *Snort* é boa em uma relação à qualidade das regras que estão sendo utilizadas, onde a construção de uma nova regra deve iniciar com o descobrimento de uma nova vulnerabilidade de segurança. A regra deve ser testada em um ambiente que seja controlado, fazendo-se ajustes a ela. Quando estiver concluída deve-se documenta-la, para utilização da mesma por um outro administrador (XAVIER, 2007).

Para melhor visualização dos *logs* do S*nort*, existe o *Analysis Console for Intrusion Detection (ACID*), evoluído hoje para *Basic Analysis and Security Engine (BASE*), que se trata de uma ferramenta para melhor análise de *logs* desenvolvida em *PHP*, que é utilizada para busca e também uma ferramenta de *front-end* para análise dos *logs* (XAVIER, 2007).

O S*nort* usa a biblioteca *libcap* que é independente de plataforma, onde para o *Windows* tem como nome *wincap* e é utilizada para que a captura dos pacotes possa ser realizada. Também pode ser utilizada para a recepção de pacotes, dentro de uma parte da rede para qual for destinado (BORGES; COUTINHO, 2007).

O pacote capturado pelo *Snort,* através da biblioteca *libcap* que coloca a placa de rede em modo de operação promíscuo, que se trata de um modo onde ela poderá receber todos os pacotes que passam pelo segmento de rede. Depois da recepção, os pacotes necessitam ser decodificados. O decodificador de pacotes é responsável por esta operação, onde nesta etapa o S*nort* pode assumir três formas de funcionamento, como a de *snifer*, onde apenas captura os pacotes e exibe as informações na tela do monitor, a de *Packet logger*, onde o codificador deixará os dados no formato binário ou *ASCII38* e arquivará no disco rígido. Na forma de *Network intrusion detection system* permite que o *Snort* analise o tráfego da rede e compare com as assinaturas definidas pelo usuário, onde realizará as ações que forem definidas de acordo com a regra que for ativada.

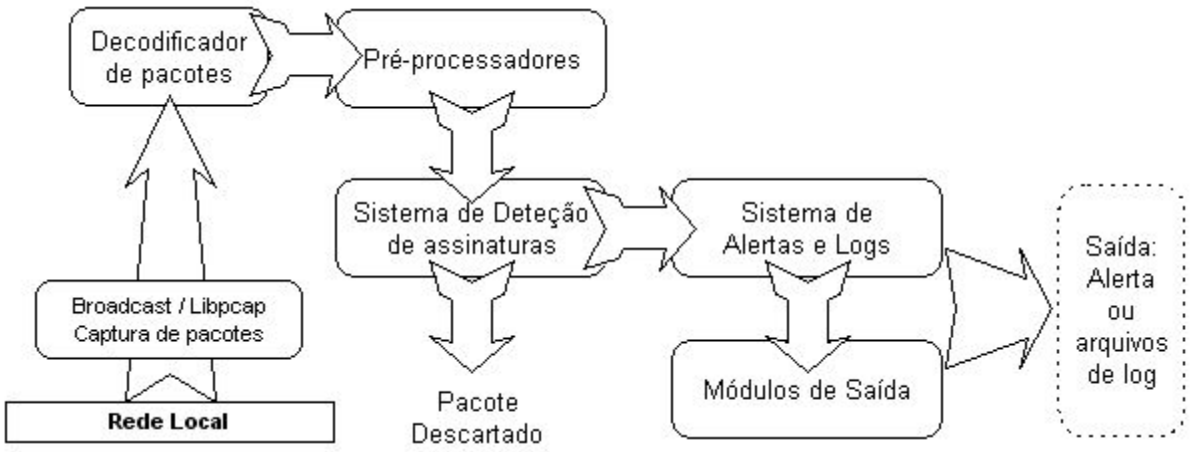


Figura 10 - Funcionamento do s*nort*.

Fonte: GASPAR; JESUS; SILVA (2008)

Após a decodificação, os dados são enviados aos pré-processadores que farão com que as regras sejam aplicadas de forma otimizada, através de reagrupamentos e ajustes dos pacotes. Depois os dados são comparados com o banco de dados das regras selecionadas no arquivo de configuração do *Snort*. Caso a comparação realizada seja positiva, outros componentes (módulos de saída) irão fazer o que está na regra a se fazer, como gerarem alertas e fazerem registros no arquivo de *log*. Se a comparação não for positiva, o pacote então é descartado (GASPAR; JESUS; SILVA, 2008).

Determinar a forma de funcionamento do *Snort* depende dos parâmetros que serão inseridos na sua inicialização. Para a utilização em modo *sniffer*, os seguintes comandos deverão ser observados:

a) *./snort –v*: utilizado para capturar e imprimir os cabeçalhos dos pacotes *tcp/ip;*

b) *./snort –vd*: com este comando é feita a analise do cabeçalho *ip, tcp/udp/icmp* e os dados existentes na camada de aplicação;

c) *./snort –vde*: mostra os cabeçalhos *ip, tcp/udp/icmp*, os dados do pacote e o cabeçalho da camada de *enlace* (LARI; AMARAL, 2004).

Utilizando-se o *Snort* como registrador de pacotes, que captura e grava no disco rígido, deve ser adicionado o parâmetro *–l* seguido do caminho de onde será armazenado, ficando o comando assim*: ./snort –vde –l /var/log/snort*, onde no comando os pacotes serão armazenados no caminho descrito. Utilizando o parâmetro *–h* seguido de uma rede, como por exemplo a rede *192.168.1.0/24*, é utilizado para armazenar como nomes de subdiretórios os *ip's* de computadores remotos, pois o *Snort* não tem critério para armazenar os registros em diretórios, usando o endereço do computador remoto ou da rede inteira como nome para o diretório. O comando completo ficaria assim: *./snort –vde –l /var/log/snort –h 192.168.1.0/24*.

Quando se deseja utilizar o *Snort* para a detecção de intrusão, basta adicionar o parâmetro *–c* e o caminho completo para o arquivo *snort.conf*, que contém quais as regras

serão utilizadas, ficando como exemplo o comando com todos os parâmetros mostrados: *./snort –vde –l /var/log/snort –h 192.168.1.0/24 –b –c snort.conf* (LARI; AMARAL, 2004).

Para este trabalho foi utilizado apenas o comando *sudo snort –c snort.conf*, onde o arquivo *snort.conf* encontra-se na pasta */etc/snort* e, dentro dessa pasta foi dado o comando. Para a visualização dos alertas, que é feito através do *BASE*, é visualizado pelo *browser* através do endereço *localhost/web/base-1.4.4.*

A instalação do *Snort* no *Ubuntu* está no Anexo B junto também com o *BASE* e todas as dependências necessárias para a instalação e funcionamento.

7.4 TESTES REALIZADOS COM O HONEYPOT

Os testes que foram realizados para análise do *honeyd* também serão analisados para o *snort*, onde os alertas gerados pelo mesmo podem ser comparados com os *log's* gerados a fim de confirmar por meio do *honeypot* o que foi também detectado pelo *snort* onde assim se terá a certeza de que não haverá falsos positivos. Inicialmente, serão realizados testes básicos de comunicação com os serviços, como comunicações via *telnet, ftp,* e também o *ip* do servidor *iis* no navegador com o objetivo de observar o resultado tanto no navegador como no *log* gerado para este evento. Em alguns testes foram feitos comparações com as respostas de uma máquina real. Na segunda etapa serão feitos testes com *scanners*, como o *nmap* e o retina, observando-se as respostas obtidas pelos programas como também os *log's* do *honeypot*, tanto no arquivo dos *log's* de conexões para com ele como também nos arquivos de *log's* específicos a cada serviço e também os alertas gerados a cada teste.

7.4.1 Forma de leitura dos *log's* gerados pelo *honeyd*

Com o *honeyd* é possível se ter dois tipos de *log's*, os de conexões e os de utilização dos serviços específicos. Eles podem ser colocados para a geração em um único arquivo ou em arquivos separados, de acordo com o serviço e o *host*, mas se terá apenas um arquivo para todos os log's gerados pelo *honeypot*, que a sua localização será em */home/anderson/Documentos/logs_honeyd.logs*. A leitura dos *log's* das conexões realizadas, ao *honeyd* independente se foi ou não interagido com algum serviço é simples, onde será explicado através da figura 11.

```
2010-05-19-18:16:05.7506 tcp(6) S 192.168.1.3 2120 192.168.1.11 3128
2010-05-19-18:16:22.5945 tcp(6) S 192.168.1.4 57167 192.168.1.11 3128 [Linux 2.6 .1-7]
2010-05-19-18:17:15.0405 tcp(6) E 192.168.1.3 2120 192.168.1.11 3128: 8 1043
2010-05-19-18:17:26.1165 tcp(6) E 192.168.1.4 57167 192.168.1.11 3128: 2 1043
```

Figura 11 – *Logs* gerais de conexões geradas pelo *honeyd*.

No primeiro campo, está a data e a hora em que ocorreu a iteração com o *honeypot*, incluindo os milisegundos. A especificação do protocolo utilizado está no segundo campo, no terceiro campo poderá ter a letra "E" que significa o final de uma conexão ou a letra "S", que indica o início de uma conexão. No quarto e o quinto campo, são indicados o *ip* de origem, porta de origem e o *ip* de destino, que seria algum *host* do *honeyd* junto com a porta de destino. Quando os pacotes *tcp* não fazem parte de uma conexão, são adicionados ao final da linha registros do tamanho do pacote e os sinalizadores *tcp*, como também comentários como a identificação do sistema operacional (AMARAL, 2006).

Para os *logs* dos serviços específicos de cada *host*, são mostrados os dados de início de conexão, todas as tentativas de digitação de comandos e o fechamento da conexão, como se pode observar na figura 12.

```
--MARK--,"Qui Mai 13 19:10:10 BRT 2010","exchange/SMTP","","",,,
"helo
auth login
ehlo
adm
",
--ENDMARK--
```

Figura 12 - *Log* de tentativa de conexão a um serviço específico.

7.4.2 – Testes de comunicação Simples

Os testes foram realizados primeiro no *host* do *Windows Server 2003*, que contêm o *ip* 192.168.1.10 e os serviços de *ftp, smtp, iis* e *pop3.* A máquina onde foram realizados os testes, estava com o sistema operacional *Windows XP SP3*, com o *ip* 192.168.1.3. O teste do serviço *ftp* foi realizado através do *prompt* de comando, através do navegador e também por uma ferramenta para *ftp* no ambiente *windows.* Ao entrar inicialmente no *prompt* de comando e ter digitado *ftp 192.168.1.10*, ele atendeu como se realmente tivesse um serviço de *ftp* funcionando, mas ficando apenas na tela de *login*. Ao fazer o teste com a ferramenta de *ftp*, o *Filezilla* na sua versão 3.3.2.1 aparece a mensagem de que a conexão é estabelecida, depois uma mensagem mostrando a versão do *ftp* e, logo em seguida a mensagem de que não é possível conectar ao servidor. Ao se tentar com o navegador, não se obteve resultado. Com esse teste o *snort* registrou alertas identificando o uso do *ftp* no *prompt* quando o *pwd*, utilizado para mostrar o caminho do diretório atual era utilizado. Na figura 13 pode-se ver os log's gerados com este teste, e na figura 14 está a visualização do alerta gerado pelo snort com este teste. Na figura 15 pode-se ver a resposta de uma máquina real e a resposta do *honeypot* para o teste de *ftp* através do prompt de comando, onde na parte de cima é a resposta de uma máquina com o *Windows XP* instalado e na parte de baixo com o *honeyd* através do serviço *iis.*

```
2010-06-18-09:33:26.2464 tcp(6) S 192.168.1.3 2148 192.168.1.10 21
--MARK--,"Sex Jun 18 09:33:26 BRT 2010","MSFTP/FTP","","",,,
"USER anonymous
2010-06-18-09:34:13.9256 tcp(6) S 192.168.1.3 2150 192.168.1.10 21
--MARK--,"Sex Jun 18 09:34:13 BRT 2010","MSFTP/FTP","","",,,
"",
--ENDMARK--
2010-06-18-09:34:13.9935 tcp(6) E 192.168.1.3 2150 192.168.1.10 21: 0 53
2010-06-18-09:34:19.0081 tcp(6) S 192.168.1.3 2151 192.168.1.10 21
--MARK--,"Sex Jun 18 09:34:19 BRT 2010","MSFTP/FTP","","",,,
"",
--ENDMARK--
2010-06-18-09:34:19.0132 tcp(6) E 192.168.1.3 2151 192.168.1.10 21: 0 53
XPWD
QUIT
",
--ENDMARK--
2010-06-18-09:38:09.1223 tcp(6) E 192.168.1.3 2148 192.168.1.10 21: 28 142
```

Figura 13 - *Log's* gerados com o teste de *ftp* no *host* do *Windows Server 2003*.

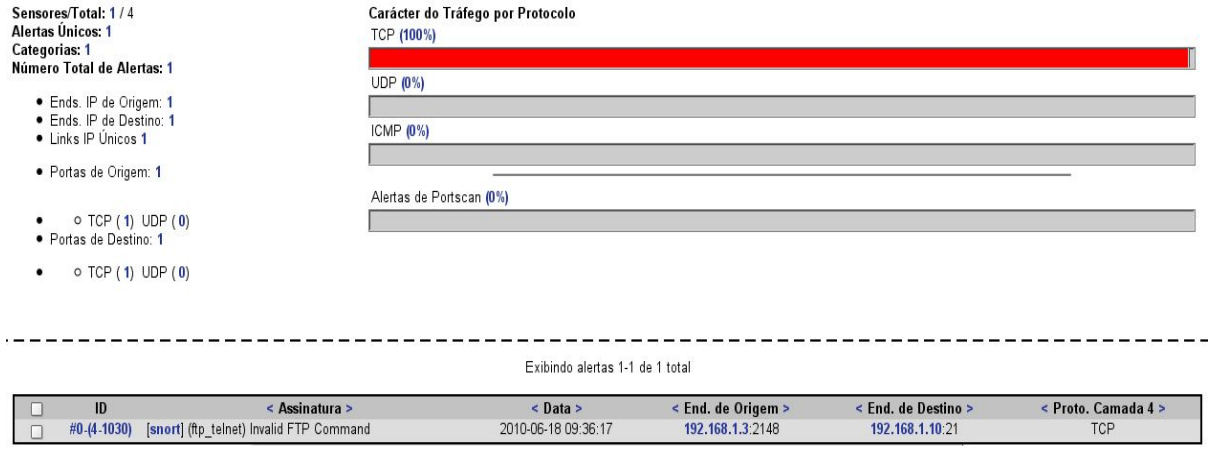


Figura 14 - Alerta gerado pelo *snort* no teste de *ftp*.

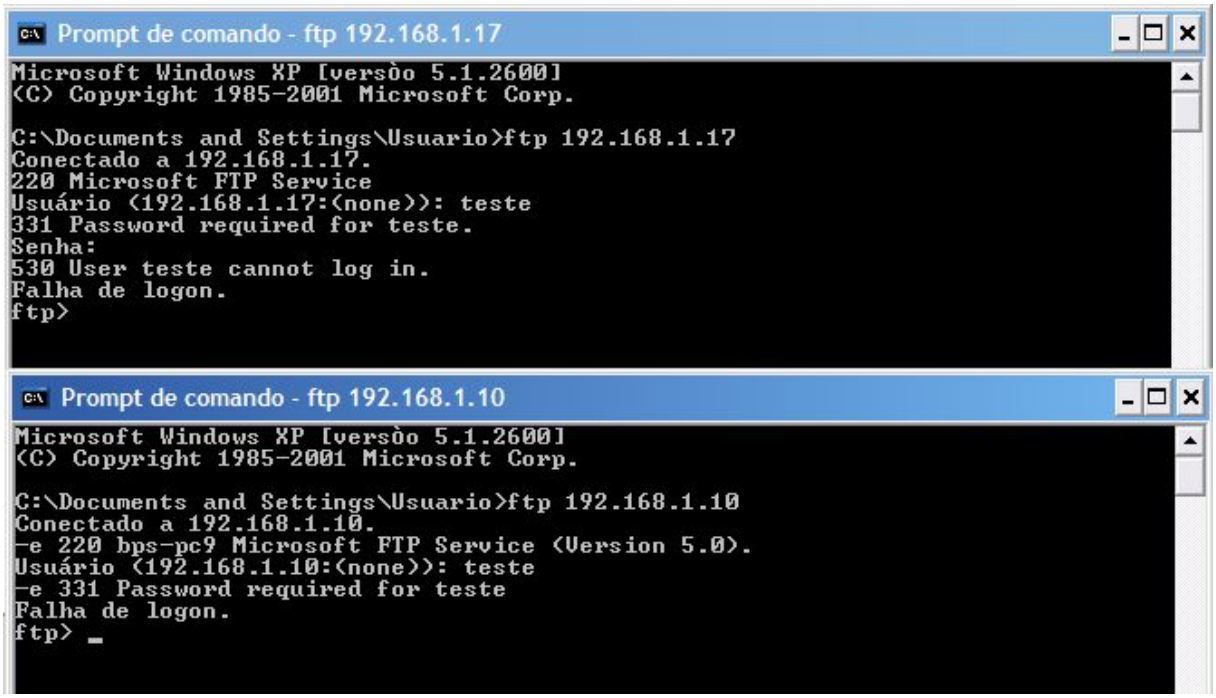


Figura 15 – Comparação das respostas para o teste com *ftp*.

O próximo teste foi realizado com o serviço *smtp* através de tentativas de resposta por *telnet.* Ao se digitar o comando *telnet 192.168.1.10 25* no *prompt* de comando, obteve-se respostas sobre o serviço, onde pode-se também identificar informações a respeito do *host* virtual. Foram executados os comandos *"helo"*, e *"ehlo"*, onde ambos servem para a identificação do emissor da mensagem para o receptor, e depois apenas digitado *"adm"* e depois *"quit"*. A figura 16 referente aos *log's* gerados com o teste do serviço *smtp*, onde pode-se também perceber a interação do usuário com o *honeypot* bem como os comandos que lá foram digitados. Na figura 17 é possível ver através do teste que a resposta dada pelo *honeypot* é da mesma forma como se fosse um servidor real, onde na parte de cima trata-se de um servidor real, e na parte de baixo da resposta dada pelo *honeypot*.

```
2010-06-18-09:38:49.1101 tcp(6) S 192.168.1.3 2153 192.168.1.10 25
--MARK--,"Sex Jun 18 09:38:49 BRT 2010","exchange/SMTP","","",,,
"helo
ehlo
adm
",
--ENDMARK--
```

Figura 16 - *Log's* gerados com o teste de *smtp* no *host* do *Windows Server 2003*.

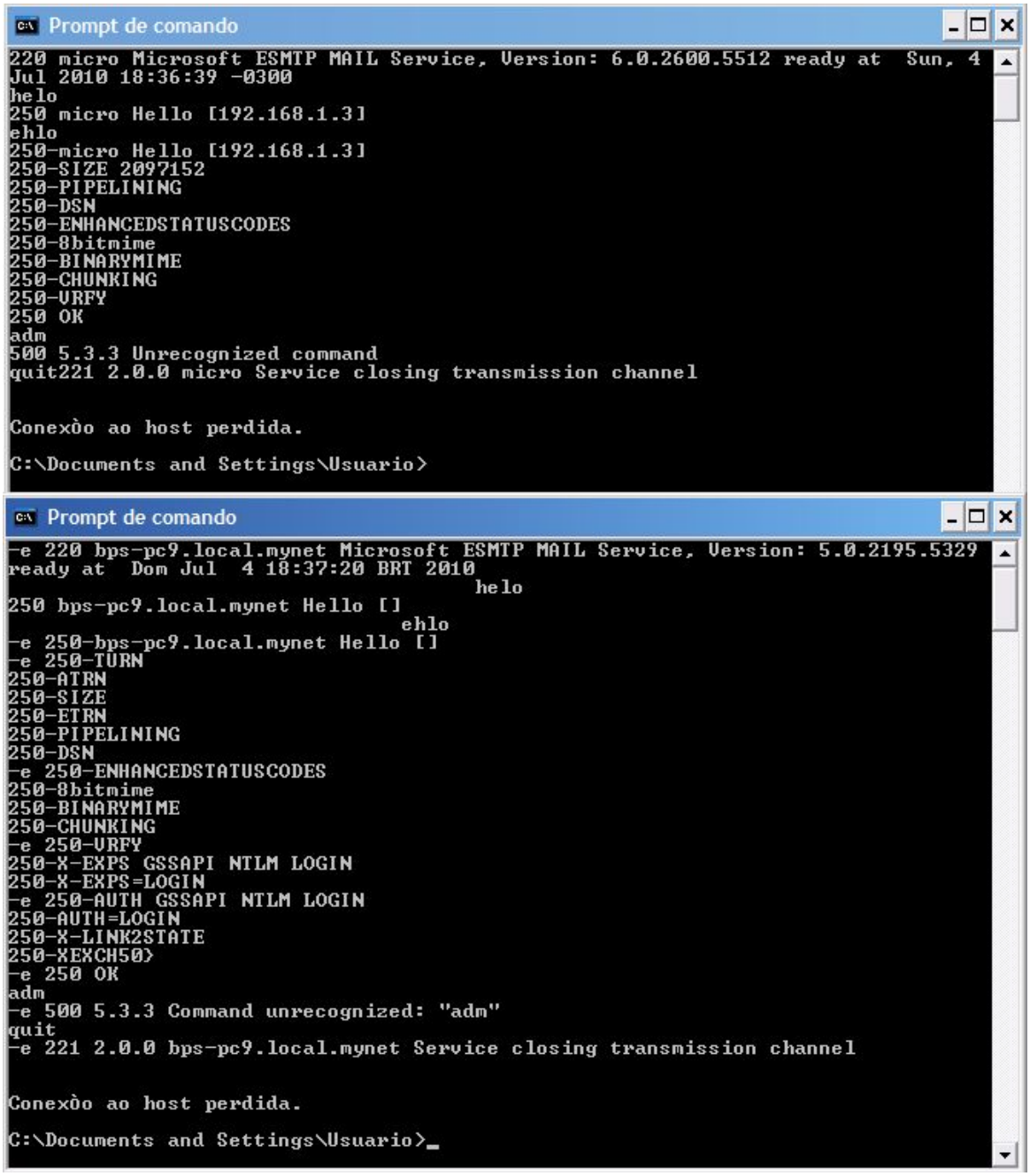


Figura 17 – Comparação da resposta no teste de *smtp* no *host* do *Windows Server 2003.*

Da mesma forma que o serviço *smtp* foi testado, o serviço *pop3* também foi testado, ou seja, através do *telnet*, com o comando *telnet 192.168.1.10 110* no *prompt* de comando. Tiveram-se como respostas informações de identificação do serviço, assim como também interação entre o usuário e o *honeypot*. Foram efetuados os comandos *"user teste", "pass123",* e *"list"*. Tanto no serviço de *smtp* como no *pop3,* também ficaram limitados na tela de boas vindas e respostas simples, e nos testes realizados não se obteve alertas no *snort*.

```
2010-06-18-09:40:28.3230 tcp(6) S 192.168.1.3 2154 192.168.1.10 110
--MARK--,"Sex Jun 18 09:40:28 BRT 2010","exchange/POP3","","",,,
"-e user teste
-e pass 123
-e list
2010-06-18-09:41:48.1509 tcp(6) E 192.168.1.3 2154 192.168.1.10 110: 34 312
```

Figura 18 - *Log's* gerados do serviço de *pop3* no *host* do *Windows Server 2003*.

Para os testes realizados no serviço *iis*, que se trata de um servidor *web* para a tecnologia *asp*, o primeiro teste foi feito por meio de um navegador, ao digitar o *ip* do *host* no mesmo e obtendo-se como resposta uma página *web* do *honeypot*. No arquivo *iis.sh* que se localiza em */usr/share/honeyd/scripts/win32/win2k/iis.sh* é possível editar uma página, através de comandos *html* para que seja vista pelo navegador. Foi colocado apenas uma parte dos *logs* gerados, devido a sua extensão neste teste, que se encontram na Figura 19. Também foram feito testes pelo *telnet*, digitando-se o comando *telnet 192.168.1.10 80* no *prompt* de comando, onde obteve-se como resposta informações sobre o serviço. Para estes testes também não se obteve alertas gerados pelo *snort.* Na figura 20 pode-se ver a página em *html* que o *honeypot* fornece como resposta para o teste e na figura 21 um comparativo utilizando *telnet* entre a resposta que um servidor real *iis* fornece, que é a parte de cima da imagem e a resposta do *honeypot*, que está na parte de baixo, através do prompt e a resposta do *honeyd.*

Na figura 22 observa-se através da ferramenta *honeydsum* um gráfico com os testes realizados neste *host*. No campo "*Source IP"* é mostra o *ip* de origem da possível tentativa de invasão.

```
2010-06-18-09:42:44.9322 tcp(6) S 192.168.1.3 2155 192.168.1.10 80
--MARK--,"Sex Jun 18 09:42:44 BRT 2010","IIS/HTTP","","",,,
"GET / HTTP/1.1
Host: 192.168.1.10
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US)
AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.70 Safari/533.4
Accept:
application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,
image/png,*/*;q=0.5
Accept-Encoding: gzip,deflate,sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
",
--ENDMARK--
```

Figura 19 – Parte dos *log's* gerados de *iis* no *host* do *Windows Server 2003*.

Figura 20 – Página *html* fornecida pelo *honeypot* através do serviço de *iis*.

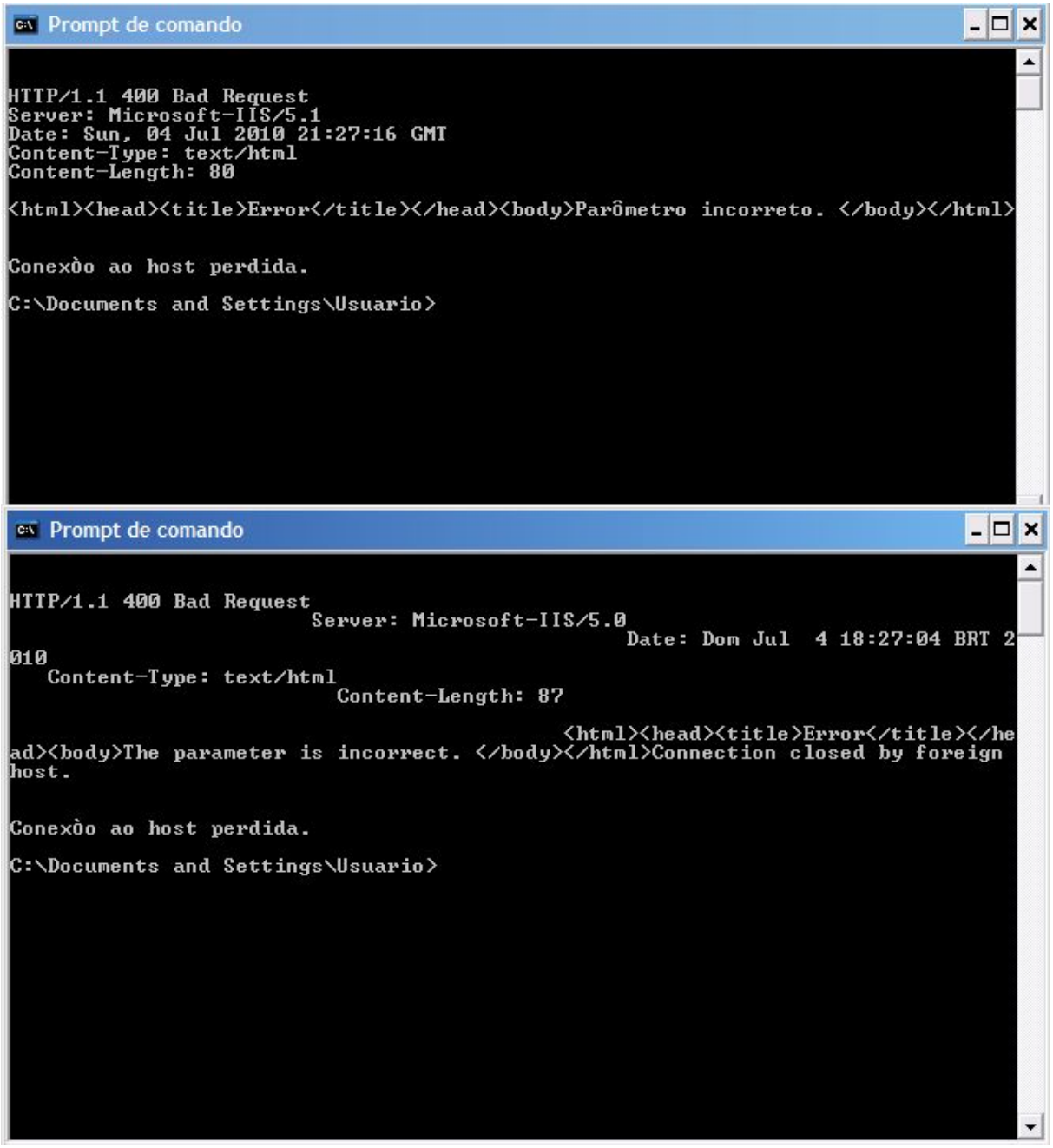
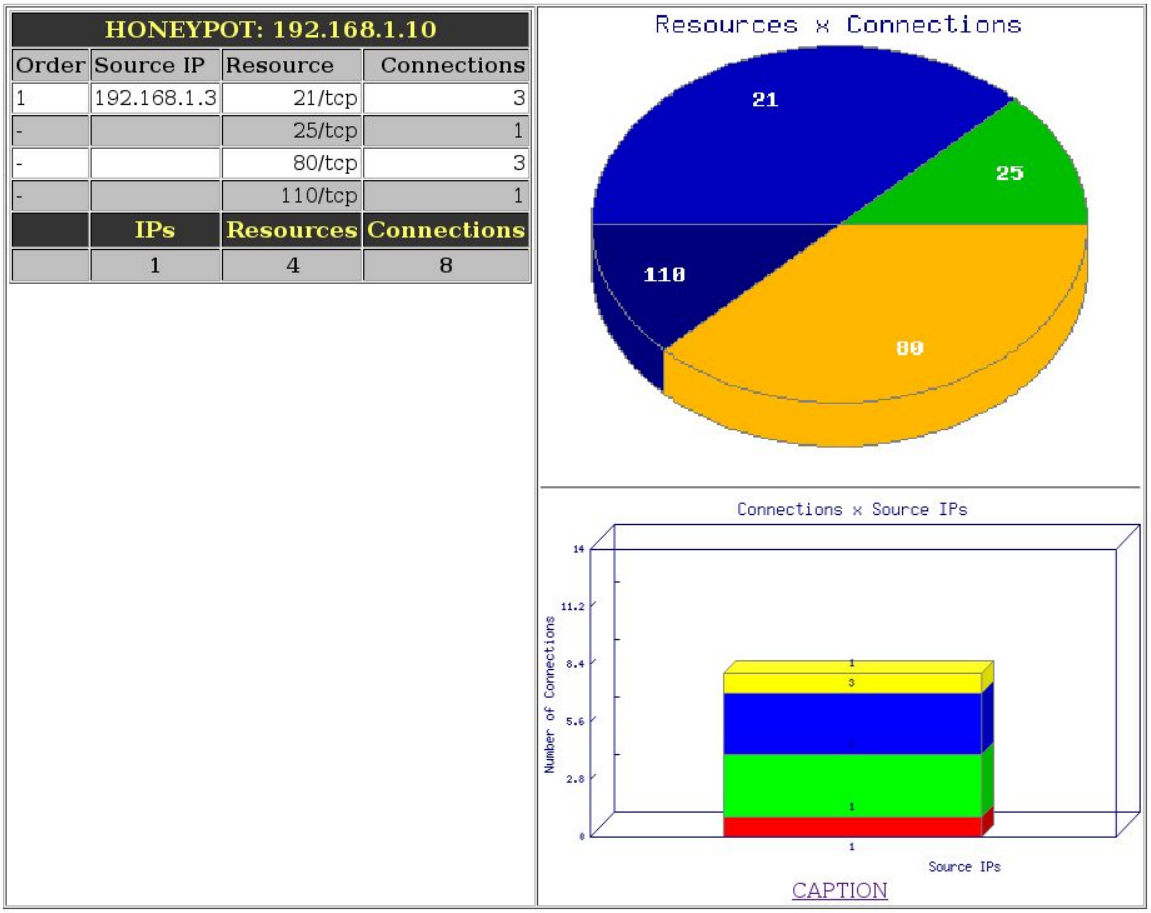


Figura 21 – comparativo utilizando *telnet* para o serviço de *iis*.



HONEYPOT: 192.168.1.10			
Order	Source IP	Resource	Connections
1	192.168.1.3	21/tcp	3
-		25/tcp	1
-		80/tcp	3
-		110/tcp	1
	IPs	Resources	Connections
	1	4	8

Figura 22 - Gráfico gerado com os testes no *host* do *Windows Server 2003*.

Os próximos testes foram realizados no *host* virtual do *Linux*, que contêm o endereço de *ip 192.168.1.11* e os serviços de *telnet, sendmail, pop3* e *squid*. O primeiro teste realizado foi com o *telnet*. No *prompt* de comando do *windows*, foi digitado o seguinte comando: *telnet 192.168.1.11*, e obteve-se como resposta uma tela de *login* e senha, mostrando as informações do sistema operacional. Foi digitado como *login* a palavra *"teste"* e como senha *"123"*, onde a iteração não passou da tela de *login*. Os *log's* gerados com este teste estão na figura 23 e na figura 24 um comparativo das respostas entre um servidor de *telnet* real, que se encontra na parte de cima da figura, e a resposta fornecida pelo *honeypot*.

```
2010-06-18-10:03:38.7229 tcp(6) S 192.168.1.3 2165 192.168.1.11 23
--MARK--,"Sex Jun 18 10:03:38 BRT 2010","telnet","","",,,
"teste
123


2010-06-18-10:03:57.7977 tcp(6) E 192.168.1.3 2165 192.168.1.11 23: 16 171
",
--ENDMARK--
```

Figura 23 - *Log's* gerados com o teste de *telnet* no *host* virtual do *linux*.

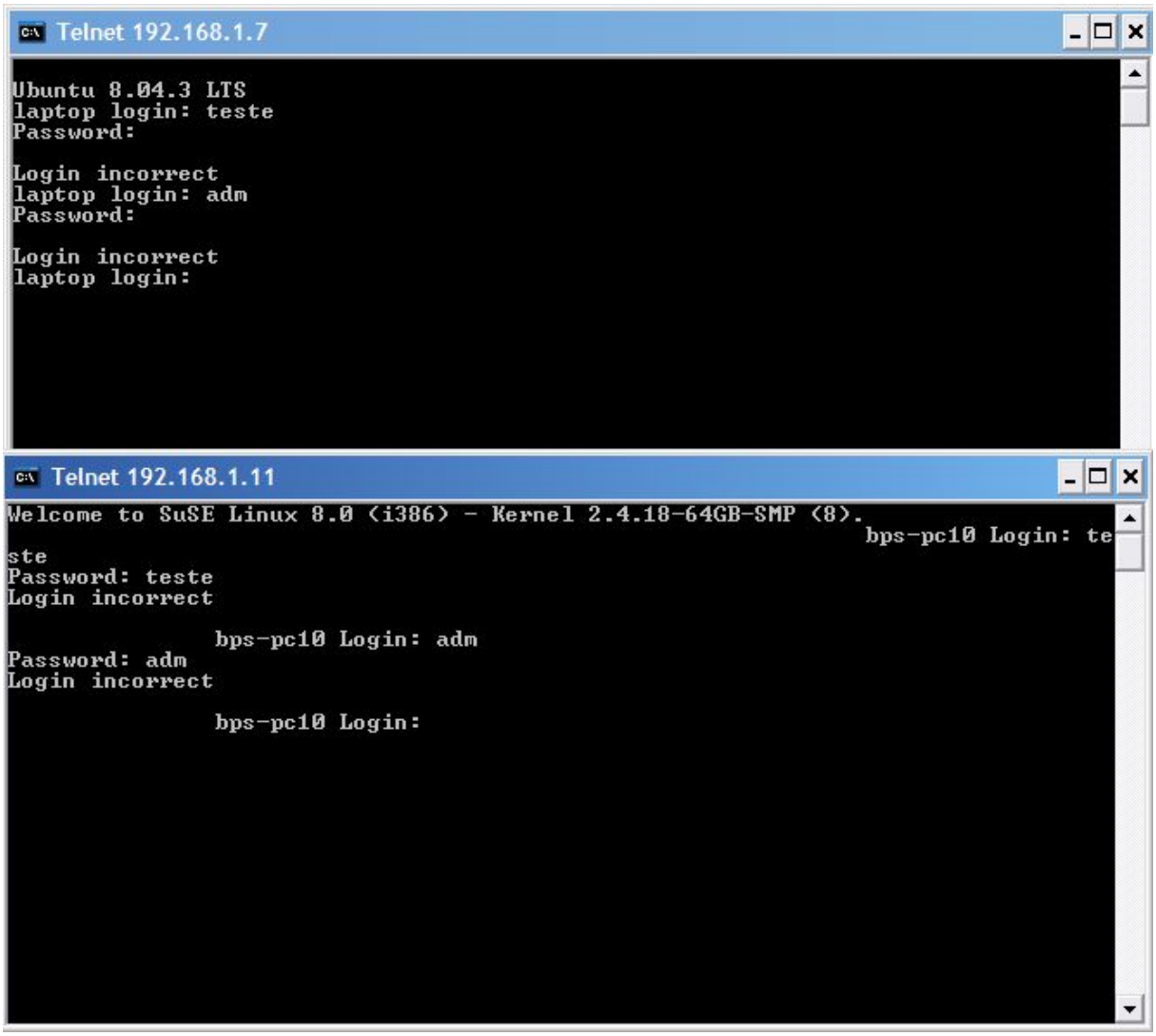


Figura 24 – comparação da respostas do teste *telnet* ao *host* Linux do *honeyd.*

O próximo teste a ser realizado foi com o serviço *sendmail*, e como se trata de um serviço *smtp*, foram realizados testes com o *telnet* através do comando: *telnet 192.168.1.11 25*, onde após ser realizada a conexão, os comandos *"ehlo"*, que se obteve como resposta o pedido de um endereço domínio, *"ehlo 192.168.1.3"* e *"helo 192.168.1.3"* foram feitos e obtidos respostas. Foi feito também o comando *"auth login"*, ao ser realizado, obteve-se a resposta *"mechanism not available"*, que indica que o serviço não está disponível.

```
2010-06-18-10:04:13.0586 tcp(6) S 192.168.1.3 2166 192.168.1.11 25
--MARK--,"Sex Jun 18 10:04:13 BRT 2010","sendmail/SMTP","","",,,
"ehlo
ehlo 192.168.1.3
helo 192.168.1.3


auth login
",
--ENDMARK--
```

Figura 25 - *Log's* gerados do serviço de *sendmail* no *host* virtual do *linux.*

Para o serviço *pop3*, assim como o *sendmail*, foi testado através do *telnet* pelo comando *telnet 192.168.1.11 110*. Após ser feita a conexão, os comandos *"list"*, *"user adm" e "pass root"* foram executados.

```
2010-06-18-10:05:01.8658 tcp(6) S 192.168.1.3 2167 192.168.1.11 110
--MARK--,"Sex Jun 18 10:05:01 BRT 2010","qpop/POP3","","",,,
"-e list
-e user adm
-e pass root
-e
-e
",
--ENDMARK--
```

Figura 26 - *Log's* gerados do serviço de *pop3* no *host* virtual do *linux*

Para o teste com o serviço *squid*, o teste também foi feito via *telnet*, digitando-se o comando *telnet 192.1687.1.11 3128*, onde foi apenas a conexão.

```
2010-06-18-10:05:31.9748 tcp(6) S 192.168.1.3 2168 192.168.1.11 3128
--MARK--,"Sex Jun 18 10:05:31 BRT 2010","squid/PROXY","","",,,
"",
--ENDMARK--
2010-06-18-10:05:49.8429 tcp(6) E 192.168.1.3 2166 192.168.1.11 25: 64 514
```

Figura 27 - *Log's* gerados do serviço *squid* no *host* virtual do *linux*.

Através do arquivo de *log's* representado na figura 27, onde estão os *log's* do *squid*. O simples fato de se fazer uma conexão sem nenhuma tentativa também fica registrado. Em nenhum teste realizado no *host* do linux não foi gerado alerta pelo *snort*. Na figura 28 pode ver os gráficos gerados pelo honeydsum dos testes realizados no host do linux, onde no campo "*Source IP"* é referente ao endereço de origem do possível ataque.

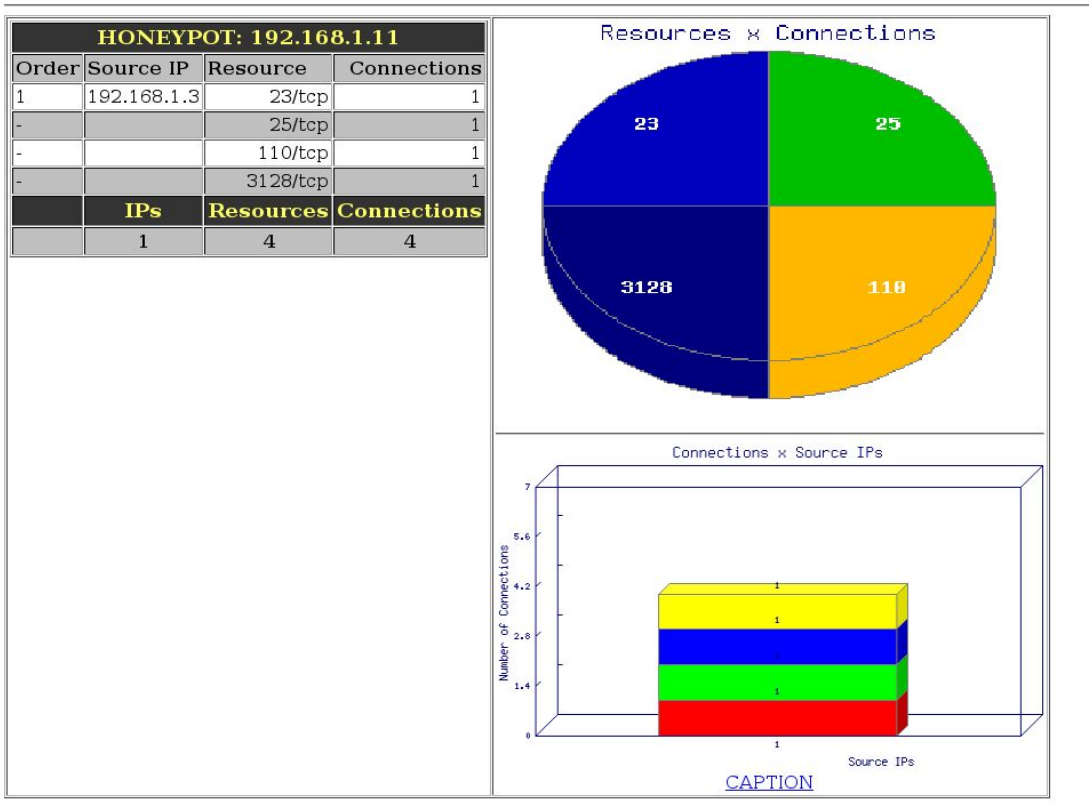


Figura 28 - Gráfico gerado pelo *honeydsum* nos testes com o *host* do *linux*.

Por fim, nos *hosts* que contem os sistemas operacionais *Solaris 9* e os vários *Windows XP* não foi colocado nenhum serviço neles, devido a isso não foi realizado nenhum teste específico. O motivo de não ser colocado nenhum serviço no *host* do *Solaris 9* e apenas portas com valor alto deve-se ao fato de que a um possível atacante encontrar este sistema operacional com portas que não são normalmente utilizadas, ser atraído pela curiosidade de querer saber mais informações sobre o sistema. Os vários *host's* com o *Windows XP* foram colocados a fim de que ao ser feita uma varredura na rede sejam cofundidos com máquinas de produção, no caso do ambiente com as máquinas virtuais que tem este sistema operacional instalado.

7.4.3 Testes com *scanners*

Os *scanners* utilizados para os testes foram o *nmap*, em sua versão 5.0 no sistema operacional *linux* e o retina, e sua versão 5.10.0 que é um *software* para o sistema operacional *windows*. Como o retina é um *software* pago, foi utilizado a sua versão *trial*.

O *nmap* trata-se de um *software open-source* para identificar quais *host's* estão ativos na rede, junto com quais portas estão abertas, quais serviços estão disponibilizando, quais sistemas operacionais estão sendo utilizados. Utilizado na exploração de redes e também para auxílio para a detecção de falhas de segurança (XAVIER, 2007).

O Retina é um *scanner* para a realização de testes de vulnerabilidades de redes, fazendo um scanner em todas as máquinas da rede, independente o sistema operacional que possa ser encontrado, dispositivos de redes, como *switches*, roteadores, e também bancos de dados. Depois de ter feito o escaneamento, um relatório detalhado com todas as vulnerabilidades e ações corretoras é fornecido pelo *software* (THOMAS, 2007).

O primeiro teste foi feito com o *nmap*, onde foi testado com o comando *nmap 192.168.1.** utilizado para verificar quais máquinas estão ativas na rede. Na figura 29 o *ip* 192.168.1.1 é o modem roteador, as máquinas com *ip's* 192.168.1.3 é a máquina com *windows XP* utilizada para testes. Os endereços 192.168.1.4 e 192.168.1.7 são referentes a máquina onde está o *honeypot* e o *VirtualBox*, mas devido às configurações realizadas para que o virtualBox funcione corretamente, a interface *eth0* é referente ao ip 192.168.1.7 e a interface criada para o virtualbox, que é a *br0* tem o ip 192.168.1.4. O endereço 192.168.1.5 é a máquina onde está instalado o *nmap* para a realização do teste, onde estas duas máquinas estão com o *linux*, sendo utilizada a distribuição *Ubuntu* 9.10, os *host's* virtuais do ip 192.168.1.10 ao 192.168.1.16 são referentes ao *honeyd* e os endereços do 192.168.1.17 ao 1921.168.1.19 são do *virtualbox*. Para cada máquina encontrada, pode-se também perceber que são listadas as portas e serviços de cada uma em específico, onde no final o *nmap* mostra a quantidade de máquinas encontradas como também o tempo que ele levou para realizar o *scanner*, como se pode ver na figura 29.

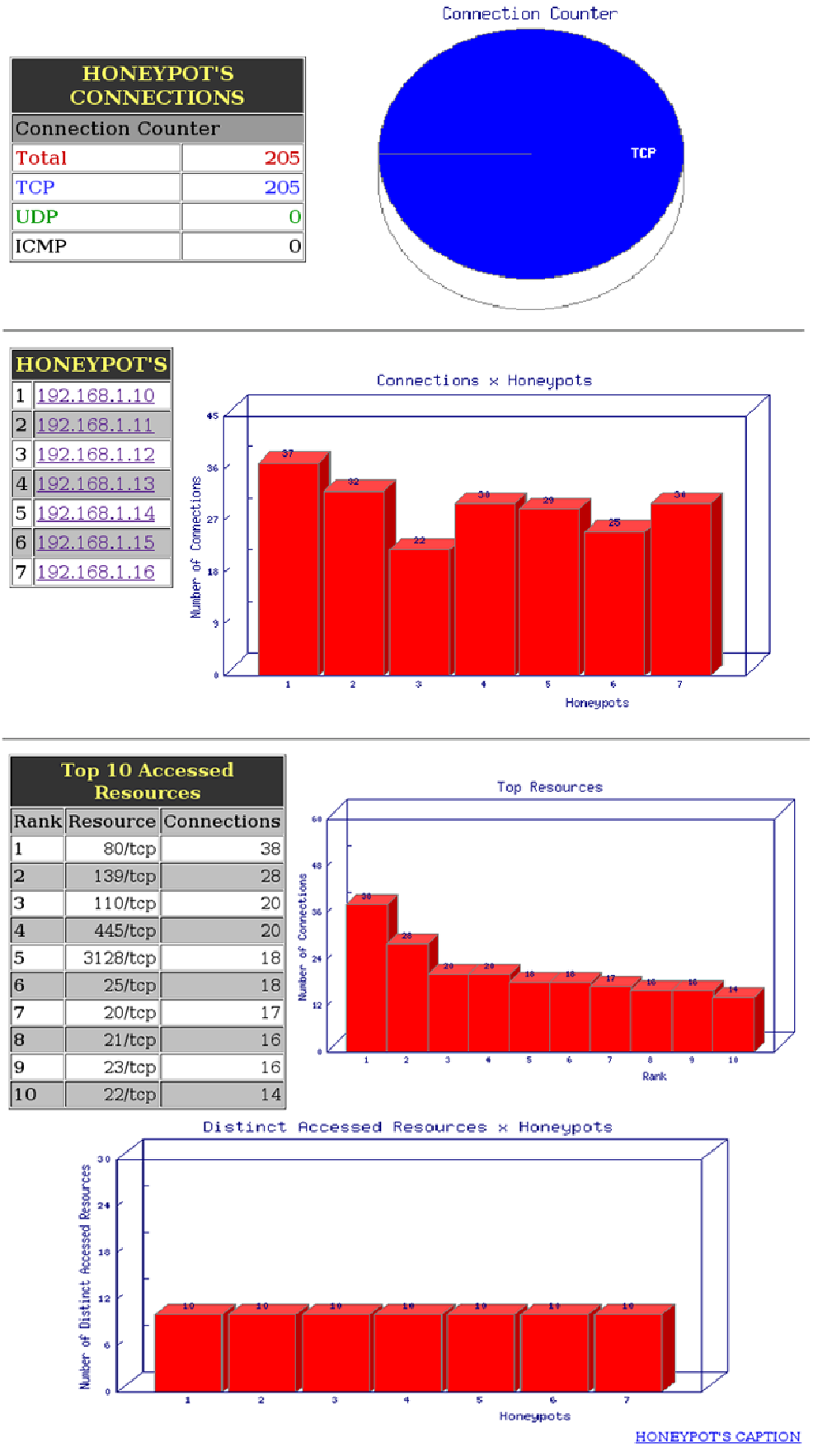
```
ubuntu@ubuntu2:~$ nmap 192.168.1.*

Starting Nmap 5.00 ( http://nmap.org ) at 2010-06-18 08:17 BRT
Interesting ports on 192.168.1.1:
Not shown: 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
80/tcp    open  http
8701/tcp open  unknown

Interesting ports on 192.168.1.3:
Not shown: 996 closed ports
PORT       STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
2869/tcp   open  unknown
50300/tcp open  unknown

Interesting ports on 192.168.1.4:
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Interesting ports on 192.168.1.5:
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5900/tcp open  vnc

Interesting ports on 192.168.1.7:
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Interesting ports on 192.168.1.10:
Not shown: 995 closed ports
PORT     STATE SERVICE
20/tcp   open  ftp-data
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
110/tcp open  pop3

Interesting ports on 192.168.1.11:
Not shown: 996 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
25/tcp    open  smtp
110/tcp   open  pop3
3128/tcp open  squid-http

All 1000 scanned ports on 192.168.1.12 are closed

Interesting ports on 192.168.1.13:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.14:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.15:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.16:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.17:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.18:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Interesting ports on 192.168.1.19:
Not shown: 997 closed ports
PORT     STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 256 IP addresses (15 hosts up) scanned in 37.16 seconds
```

Figura 29 – Resultado retornado pelo teste com o *nmap.*

Os *log's* de conexões gerados pelo *honeyd* podem ser vistos na figura 30, onde foi colocado apenas uma parte dos *log's* gerados, pois devido ao *scanner* a lista completa de *log's* gerados seria extensa demais. Na figura 31 pode-se ver alguns gráficos gerados pelo *honeydsum* de acordo com os *log's* gerados.

```
2010-06-18-11:17:22.1321 tcp(6) S 192.168.1.5 36540 192.168.1.10 80 [Linux
2.6 .1-7]
2010-06-18-11:17:22.1323 tcp(6) - 192.168.1.5 46231 192.168.1.13 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:22.1323 tcp(6) - 192.168.1.5 50355 192.168.1.14 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:22.1449 tcp(6) E 192.168.1.5 36540 192.168.1.10 80: 0 0
2010-06-18-11:17:22.4371 tcp(6) - 192.168.1.5 52370 192.168.1.11 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:22.4372 tcp(6) - 192.168.1.5 40379 192.168.1.15 80: 60 S
[Linux 2.6 .1-7]
--MARK--,"Sex Jun 18 11:17:22 BRT 2010","IIS/HTTP","","",,,
"
",
--ENDMARK--
2010-06-18-11:17:22.6452 tcp(6) - 192.168.1.5 39494 192.168.1.12 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:22.6453 tcp(6) - 192.168.1.5 47633 192.168.1.16 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:23.2296 tcp(6) - 192.168.1.5 46565 192.168.1.13 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:23.2297 tcp(6) - 192.168.1.5 50689 192.168.1.14 80: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:23.2297 tcp(6) - 192.168.1.5 43529 192.168.1.13 443: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:23.2298 tcp(6) - 192.168.1.5 33105 192.168.1.14 443: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:24.5331 tcp(6) - 192.168.1.5 33586 192.168.1.14 443: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:50.9479 tcp(6) - 192.168.1.5 59152 192.168.1.10 256: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:50.9480 tcp(6) - 192.168.1.5 40456 192.168.1.11 256: 60 S
[Linux 2.6 .1-7]
2010-06-18-11:17:50.9481 tcp(6) - 192.168.1.5 40119 192.168.1.12 256: 60 S
[Linux 2.6 .1-7]
```

Figura 30 – Parte dos *log's* de conexões geradas do teste com o *nmap.*



HONEYPOT'S CONNECTIONS	
Connection Counter	
Total	205
TCP	205
UDP	0
ICMP	0

HONEYPOT'S	
1	192.168.1.10
2	192.168.1.11
3	192.168.1.12
4	192.168.1.13
5	192.168.1.14
6	192.168.1.15
7	192.168.1.16

Top 10 Accessed Resources		
Rank	Resource	Connections
1	80/tcp	38
2	139/tcp	28
3	110/tcp	20
4	445/tcp	20
5	3128/tcp	18
6	25/tcp	18
7	20/tcp	17
8	21/tcp	16
9	23/tcp	16
10	22/tcp	14

Figura 31 – Gráficos gerados pelo *honeydsum* para o teste com o *nmap*.

Através da figura 31 pode-se perceber quais protocolos foram utilizados para o ataque, quais os *honeypot's* onde houve o scaneamento, e quais os serviços que mais foram acessados. Na figura 32 pode-se ver em "*Source IP"* qual é o *ip* de origem do scaneamento e gráficos referentes aos endereços de origem e aos *honeypot's* mais utilizados. Na figura 33 são mostradas informações de ataques a um *host* específico, como portas, protocolos, número de conecções, *ip's* de origem e um gráfico comparatico sobre os serviços acessados, que no caso foi utilizado informações apenas sobre o *host* do *Wndows Server 2003* com o objetivo de mostrar que é possível ver estas informações a respeito de qualquer *host.*

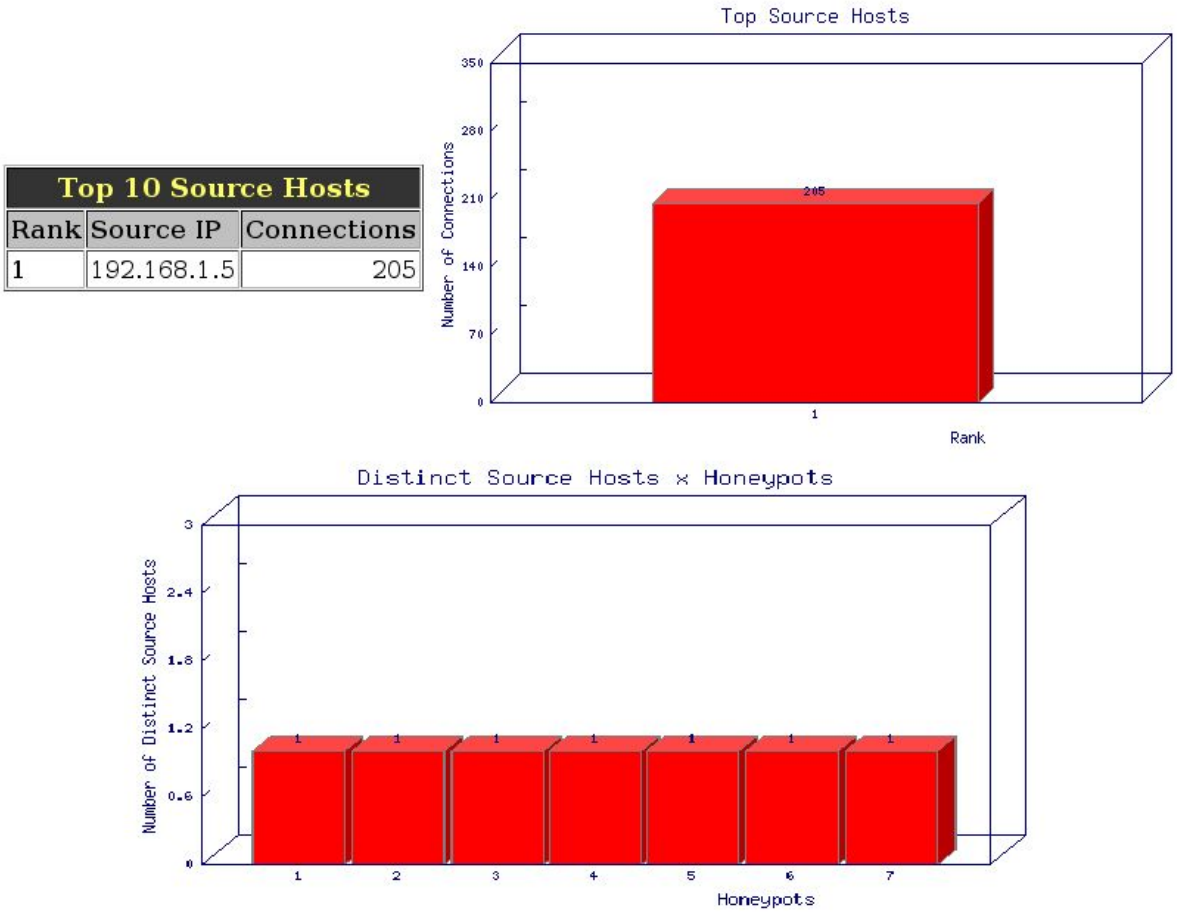


Figura 32 – Endereço de origem e gráficos das conecções para o teste com o *nmap*.

HONEYPOT'S CONFIGURATION					
Operating System	Default TCP Action	Default UDP Action	Default ICMP Action	Ports	IP Address
Microsoft Windows Server 2003	reset	reset	-	20/tcp 21/tcp 25/tcp 80/tcp 110/tcp	192.168.1.10

HONEYPOT: 192.168.1.10			
Order	Source IP	Resource	Connections
1	192.168.1.5	20/tcp	5
-		21/tcp	4
-		22/tcp	2
-		23/tcp	2
-		25/tcp	4
-		80/tcp	8
-		110/tcp	6
-		139/tcp	2
-		445/tcp	2
-		3128/tcp	2
	IPs	Resources	Connections
	1	10	37

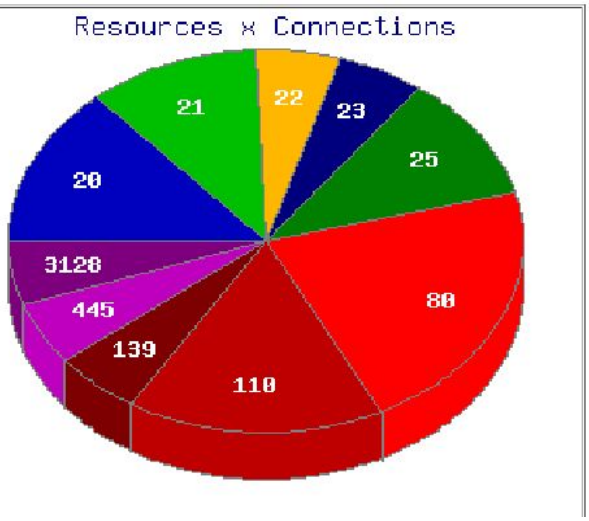


Figura 33 – Informações de um *host* específico do *honeypot* do teste com o *nmap*.

Foram capturados 43 alertas de *portscan* pelo *snort,* onde deles o *BASE* identificou três tipos alertas únicos. Isso significa que todos os 43 alertas gerados são classificados em três tipos. A figura 34 mostra a quantidade de alertas gerados, a figura 35 mostra os alertas únicos gerados e a figura 36 mostra uma parte dos 53 alertas gerados, pois ao ser colocado todos, a lista ficaria muito extensa. Na figura 37 pode-se ver por meio da detecção de intrusão, onde de acordo com o ambiente, o sensor do IDS apenas pode fazer as verificações nos 12 endereços de máquinas que estão no notebook, mas se pode perceber nesse ponto a eficácia de se ter um IDS para a monitoração da rede.

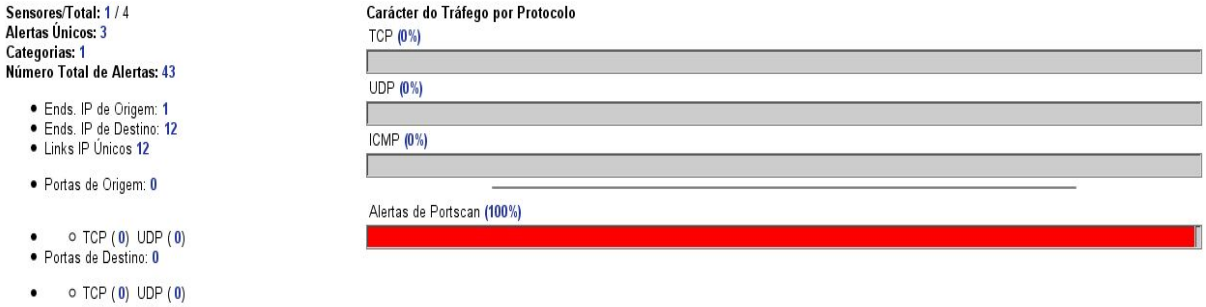


Figura 34 – Visualização geral dos alertas gerados com o teste do *nmap*.

< Assinatura >	< Classificação >	< Total # >	Sensor #	< End. de Origem >	< End. de Destino >	< Primeiro >	< Último >
[snort] (portscan) TCP Portscan	*não-classificado*	12(28%)	1	1	12	2010-06-18 11:17:50	2010-06-18 11:17:50
[snort] (portscan) Open Port	*não-classificado*	30(70%)	1	1	11	2010-06-18 11:17:50	2010-06-18 11:17:52
[snort] (portscan) TCP Portsweep	*não-classificado*	1(2%)	1	1	1	2010-06-18 11:17:22	2010-06-18 11:17:22

Figura 35 – Visualização dos alertas únicos gerados do teste com o *nmap*.

ID	< Assinatura >	< Data >	< End. de Origem >	< End. de Destino >	< Proto. Camada 4 >
#0-(4-1216)	[snort] (portscan) Open Port: 20	2010-06-18 11:17:52	192.168.1.5	192.168.1.10	Raw IP
#1-(4-1214)	[snort] (portscan) Open Port: 3128	2010-06-18 11:17:52	192.168.1.5	192.168.1.11	Raw IP
#2-(4-1215)	[snort] (portscan) Open Port: 20	2010-06-18 11:17:52	192.168.1.5	192.168.1.10	Raw IP
#3-(4-1213)	[snort] (portscan) Open Port: 3128	2010-06-18 11:17:52	192.168.1.5	192.168.1.11	Raw IP
#4-(4-1211)	[snort] (portscan) Open Port: 23	2010-06-18 11:17:50	192.168.1.5	192.168.1.11	Raw IP
#5-(4-1212)	[snort] (portscan) Open Port: 23	2010-06-18 11:17:50	192.168.1.5	192.168.1.11	Raw IP
#6-(4-1210)	[snort] (portscan) Open Port: 80	2010-06-18 11:17:50	192.168.1.5	192.168.1.10	Raw IP
#7-(4-1208)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.16	Raw IP
#8-(4-1209)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.16	Raw IP
#9-(4-1207)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.13	Raw IP
#10-(4-1205)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.19	Raw IP
#11-(4-1206)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.13	Raw IP
#12-(4-1203)	[snort] (portscan) Open Port: 80	2010-06-18 11:17:50	192.168.1.5	192.168.1.4	Raw IP
#13-(4-1204)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.19	Raw IP
#14-(4-1201)	[snort] (portscan) TCP Portscan: 25:1723	2010-06-18 11:17:50	192.168.1.5	192.168.1.19	Raw IP
#15-(4-1200)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.17	Raw IP
#16-(4-1199)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.17	Raw IP
#17-(4-1198)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.15	Raw IP
#18-(4-1197)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.15	Raw IP
#19-(4-1196)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.14	Raw IP
#20-(4-1195)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.14	Raw IP
#21-(4-1194)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.18	Raw IP
#22-(4-1193)	[snort] (portscan) Open Port: 135	2010-06-18 11:17:50	192.168.1.5	192.168.1.18	Raw IP

Figura 36 – parte da visualização do total de alertas gerados com o teste do *nmap*.

< Ends. IP de Destino >	Sensor #	< Total # >	< Alertas Únicos >	< Ends. de Origem >
192.168.1.4	1	3	2	1
192.168.1.7	1	3	2	1
192.168.1.10	1	7	2	1
192.168.1.11	1	7	2	1
192.168.1.12	1	2	2	1
192.168.1.13	1	3	2	1
192.168.1.14	1	3	2	1
192.168.1.15	1	3	2	1
192.168.1.16	1	3	2	1
192.168.1.17	1	3	2	1
192.168.1.18	1	3	2	1
192.168.1.19	1	3	2	1

Figura 37 – Endereços *scaneados* detectados pelo *snort*.

Com o retina um teste com uma faixa de *ip's*, do endereço 192.168.1.1 até o endereço 192.168.1.19, que é o *host* com o *ip* de número mais elevado. Ao se colocar a faixa de *ip's* no retina, obteve-se a seguinte mensagem: *"Warning: Your license only supports scanning 1 ip address. you requested to scan 21. this scan may exceed your license."*, que indica que a licença suporta somente uma varredura de endereços de *ip,* e que ao pedir a varredura de mais endereços pode exceder a licença, que no caso é a versão de demonstração. Mesmo assim a ferramenta permitiu que a varredura acontecesse. Os *host's* criados pelo *honeypot* foram tratados pela ferramenta como se fossem *host's* reais, onde se obteve várias informações a respeito, como o sistema operacional, portas abertas, serviços que estão no host entre outras informações. Na figura 38 pode-se ver os *ip's* que foram vistos como máquinas reais, onde com uma comparação entre um dos *host's* criados pelo *honeyd* são vistos da mesma forma que uma das máquinas virtuais criadas pelo *virtualbox*, como se pode verificar na figura 39, onde o endereço 192.168.1.16 é do honeyd *e o endereço 192.168.1.17 pertence a uma máquina virtual do virtualbox.* na figura 40 está uma parte dos log's gerados com este teste, pois devido a sua extensão não foi possível colocar todos, e na figura 41 se observa alguns gráficos gerados pela ferramnta *honeydsum*.

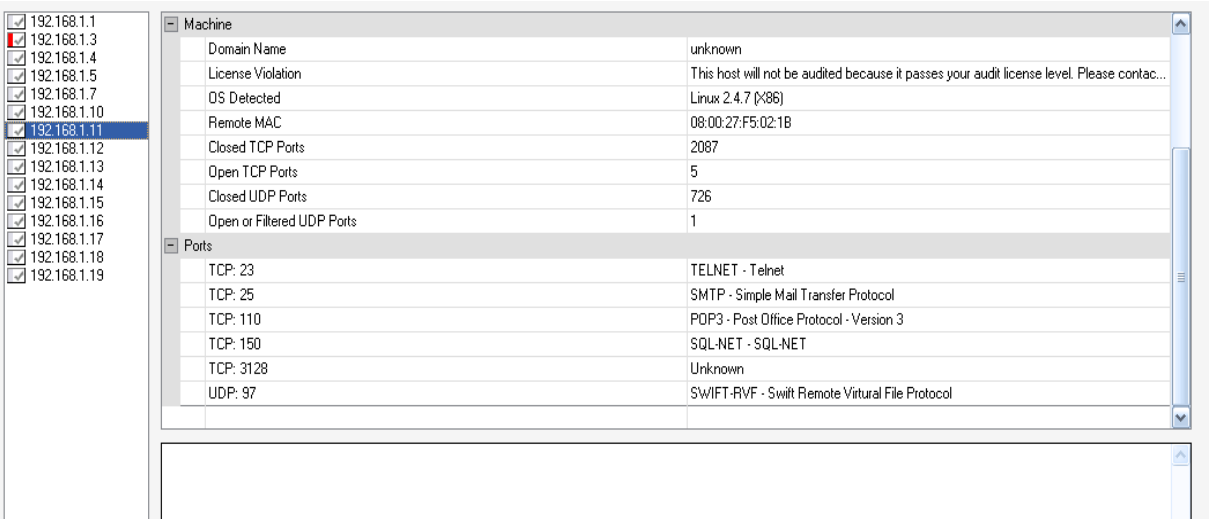


Figura 38 – Visualização dos *hosts* e máquinas virtuais pelo retina.

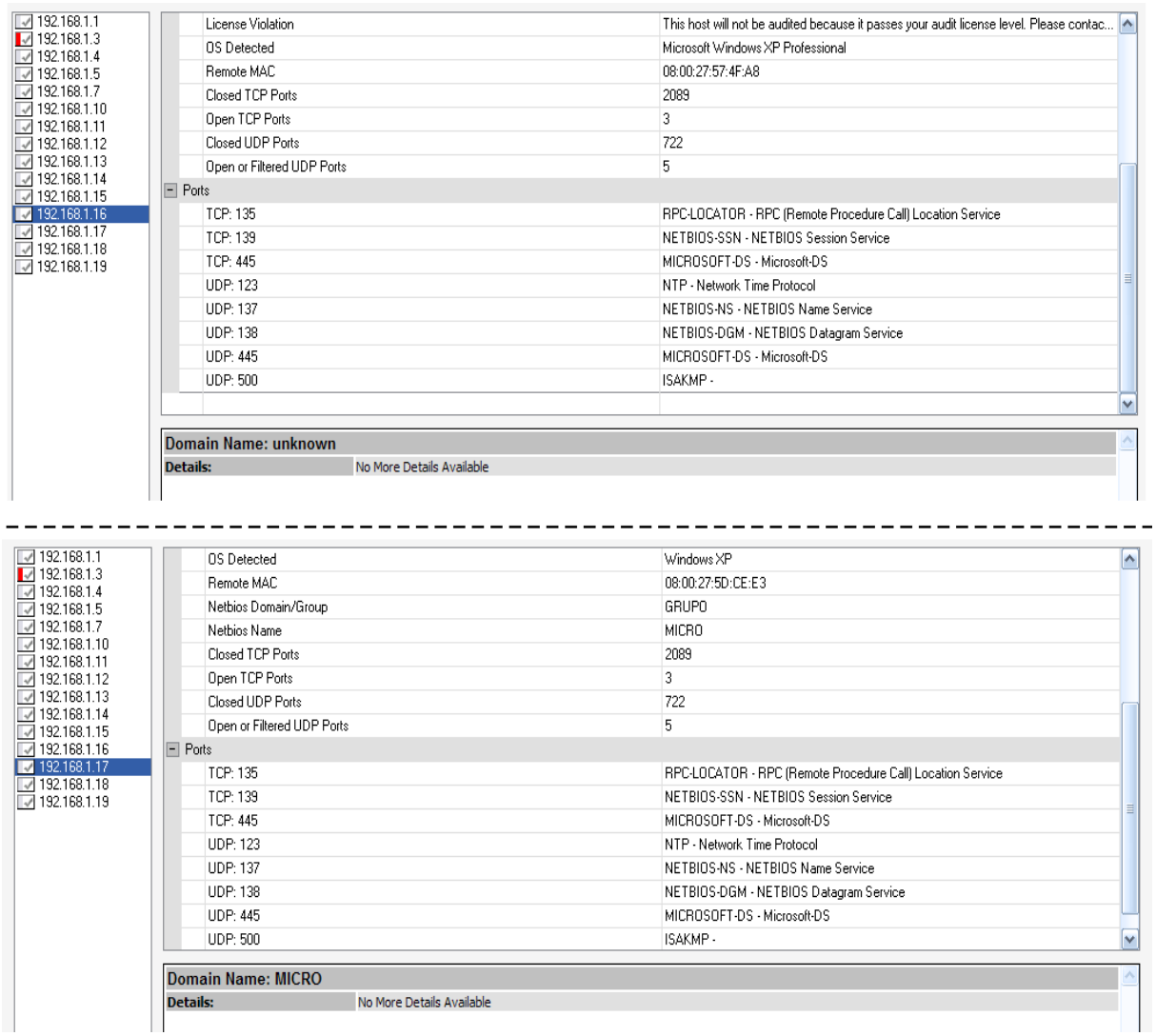
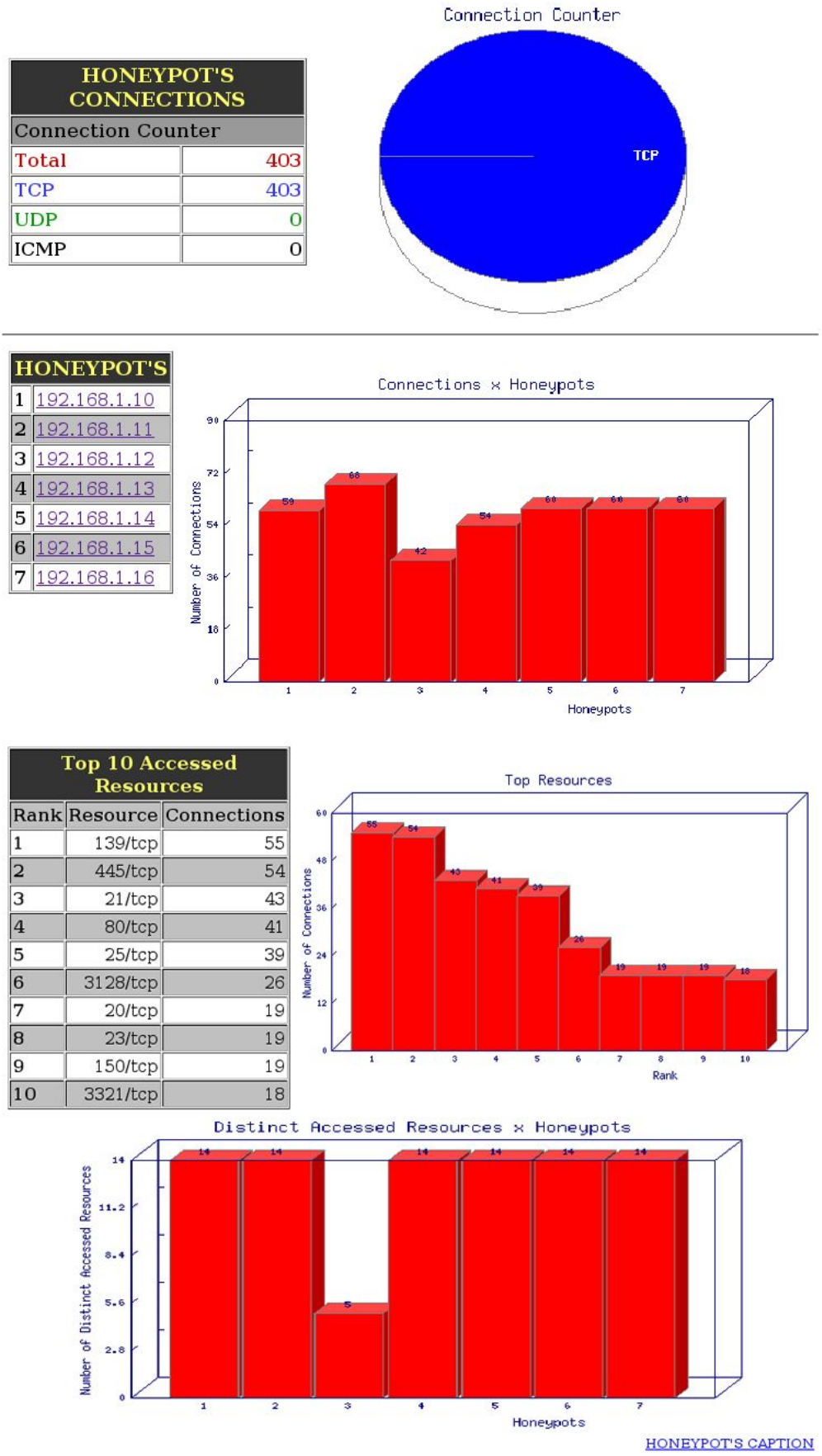


Figura 39 – Comparação de um *host* criado pelo *honeyd* com uma máquina virtual.

```
2010-06-18-11:53:33.4974 icmp(1) - 192.168.1.3 192.168.1.12: 8(0): 60
2010-06-18-11:53:33.4989 tcp(6) - 192.168.1.3 2875 192.168.1.12 139: 52 S
2010-06-18-11:53:33.4994 tcp(6) - 192.168.1.3 2876 192.168.1.12 445: 52 S
2010-06-18-11:53:33.5054 tcp(6) - 192.168.1.3 2880 192.168.1.12 80: 52 S
2010-06-18-11:53:33.5055 tcp(6) - 192.168.1.3 2881 192.168.1.12 25: 52 S
2010-06-18-11:53:33.5380 tcp(6) - 192.168.1.3 2890 192.168.1.12 21: 52 S
2010-06-18-11:53:33.5381 tcp(6) - 192.168.1.3 2891 192.168.1.12 139: 52 S
2010-06-18-11:53:33.5458 tcp(6) - 192.168.1.3 2893 192.168.1.12 445: 52 S
2010-06-18-11:53:33.9741 tcp(6) - 192.168.1.3 2880 192.168.1.12 80: 52 S
2010-06-18-11:53:33.9742 tcp(6) - 192.168.1.3 2890 192.168.1.12 21: 52 S
2010-06-18-11:53:33.9742 tcp(6) - 192.168.1.3 2875 192.168.1.12 139: 52 S
```

Figura 40 – Parte dos *log's* gerados do teste com a ferramenta retina.



HONEYPOT'S CONNECTIONS	
Connection Counter	
Total	403
TCP	403
UDP	0
ICMP	0

HONEYPOT'S	
1	192.168.1.10
2	192.168.1.11
3	192.168.1.12
4	192.168.1.13
5	192.168.1.14
6	192.168.1.15
7	192.168.1.16

Top 10 Accessed Resources		
Rank	Resource	Connections
1	139/tcp	55
2	445/tcp	54
3	21/tcp	43
4	80/tcp	41
5	25/tcp	39
6	3128/tcp	26
7	20/tcp	19
8	23/tcp	19
9	150/tcp	19
10	3321/tcp	18

Figura 41 – Gráficos gerados pelo *honeydsum* com os logs do teste com o retina.

Na figura 42 pode-se observar através do campo *"Source IP"* o endereço de origem utilizado para o teste, bem como gráficos mostrando os *ip's* de origem como os *honeypot's* que interagiram com a ferramenta de *scanner*. Na figura 43 pode-se observar informações específicas de um *hots* do *honeyd,* mostrando o endereço de origem, a porta que foi requisitada no *honeypot* e o número de conexões realizadas para o teste com o retina.

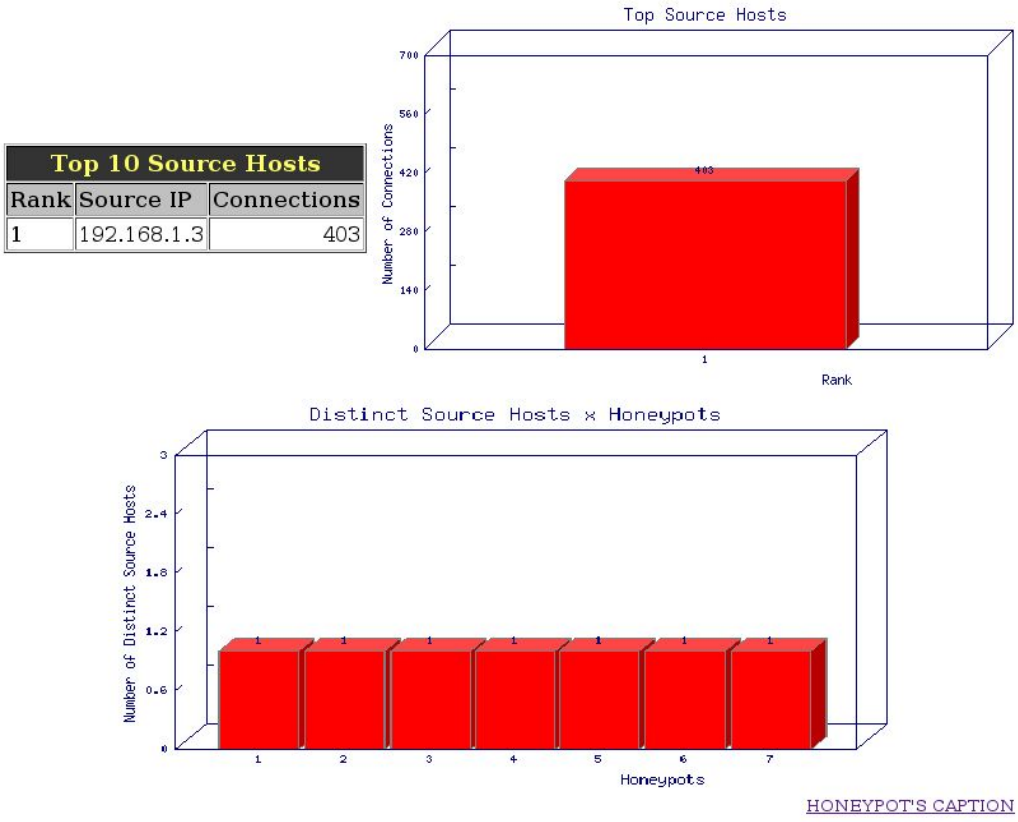


Figura 42 – Informações sobre a origem do ataque no teste com o retina.

HONEYPOT'S CONFIGURATION					
Operating System	Default TCP Action	Default UDP Action	Default ICMP Action	Ports	IP Address
Microsoft Windows Server 2003	reset	reset	-	20/tcp 21/tcp 25/tcp 80/tcp 110/tcp	192.168.1.10

HONEYPOT: 192.168.1.10			
Order	Source IP	Resource	Connections
1	192.168.1.3	20/tcp	4
-		21/tcp	6
-		22/tcp	3
-		23/tcp	3
-		25/tcp	4
-		80/tcp	4
-		97/tcp	3
-		110/tcp	2
-		139/tcp	9
-		150/tcp	3
-		445/tcp	9
-		3128/tcp	3
-		3321/tcp	3
-		4431/tcp	3
	IPs	Resources	Connections
	1	14	59

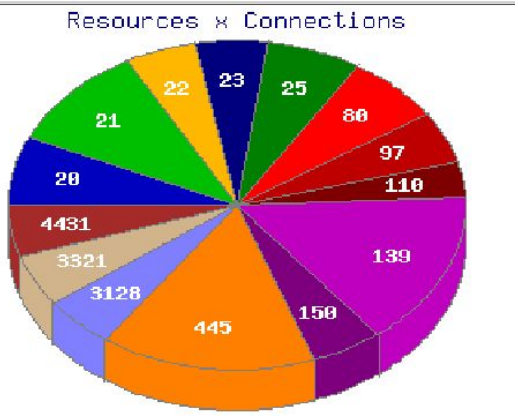


Figura 43 – Informações específicas sobre um *host* após o teste com o retina.

Neste teste foram gerados um total de 155 alertas pelo *snort*, onde são divididos em 7 tipos de alertas únicos. Na figura 44 pode-se ver a indicação dos alertas gerados, que mostra 17% dos alertas do protocolo TCP e 83% vindos de *portscan*. Na figura 45 uma parte da listagem de todos os alertas gerados, onde não foi colocado toda a listagem devido a sua extensão, e na figura 46 a listagem dos 7 alertas únicos gerados. Na figura 47 estão os hosts que o ids detectou onde estava sendo feito uma varredura de rede, que foi apenas para os hosts que estão no *notebook*.

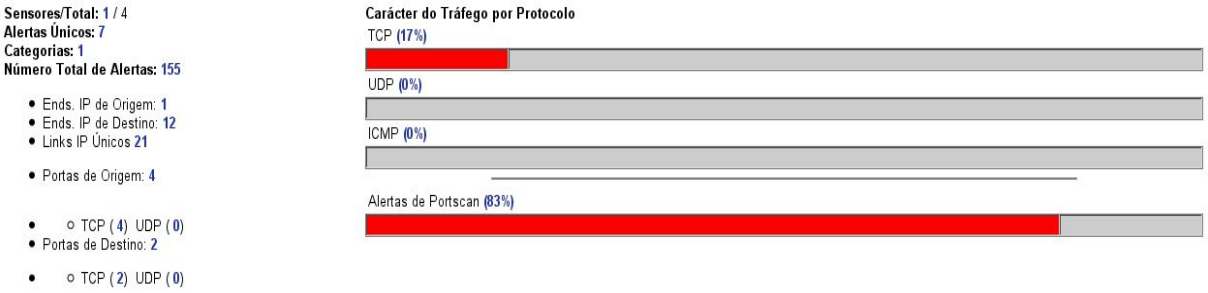


Figura 44 – Indicação pelo *BASE* dos alertas gerados no primeiro teste com o retina.

ID	< Assinatura >	< Data >	< End. de Origem >	< End. de Destino >	< Proto. Camada 4 >
#0-(4-1332)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#1-(4-1331)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#2-(4-1330)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#3-(4-1329)	[snort] (snort_decoder): Tcp Window Scale Option found with length > 14	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	TCP
#4-(4-1328)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#5-(4-1327)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#6-(4-1326)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#7-(4-1325)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#8-(4-1324)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#9-(4-1323)	[snort] (portscan) TCP Portscan: 80:445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#10-(4-1322)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#11-(4-1321)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP
#12-(4-1320)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#13-(4-1319)	[snort] (snort_decoder): Tcp Window Scale Option found with length > 14	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	TCP
#14-(4-1318)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.15	Raw IP
#15-(4-1317)	[snort] (portscan) Open Port: 445	2010-06-18 11:56:16	192.168.1.3	192.168.1.13	Raw IP

Figura 45 – Parte da listagem de alertas gerados com o teste inicial do retina.

FQDN de Origem	< IP de Origem >	Direção	< IP de Destino >	FQDN de Destino	Protocolo	Portas de Destino Únicas	Eventos Únicos	Total de Eventos
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.15		Raw IP	0	2	30
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.13		Raw IP	0	2	28
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.15		TCP	0	1	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.13		TCP	0	1	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.14		Raw IP	0	3	11
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.12		TCP	0	1	8
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.14		TCP	0	1	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.11		TCP	1	2	3
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.10		TCP	1	3	5
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.12		Raw IP	0	2	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.4		TCP	0	1	1
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.7		TCP	0	1	1
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.11		Raw IP	0	2	6
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.16		Raw IP	0	2	30
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.10		Raw IP	0	2	6
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.19		Raw IP	0	2	4
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.18		Raw IP	0	2	4
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.17		Raw IP	0	2	4
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.16		TCP	0	1	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.7		Raw IP	0	2	2
nenhuma consulta DNS realizada	192.168.1.3	-->	192.168.1.4		Raw IP	0	2	2

Figura 46 – Alertas únicos gerados do primeiro teste com o retina.

< Ends. IP de Destino >	Sensor #	< Total # >	< Alertas Únicos >	< Ends. de Origem >
192.168.1.4	1	3	3	1
192.168.1.7	1	3	3	1
192.168.1.10	1	11	5	1
192.168.1.11	1	9	4	1
192.168.1.12	1	10	3	1
192.168.1.13	1	30	3	1
192.168.1.14	1	13	4	1
192.168.1.15	1	32	3	1
192.168.1.16	1	32	3	1
192.168.1.17	1	4	2	1
192.168.1.18	1	4	2	1
192.168.1.19	1	4	2	1

Figura 47 – *Hosts* foram detectados pelo *ids* que esta sendo *scaneado*.

Através dos testes básicos e dos testes com ferramentas de *scanner* pode-se perceber claramente que o as ferramentas *honeypot* podem passar de forma segura a alguém não autorizado que deseje ter conhecimentos sobre a rede, que são máquinas reais dentro da rede, onde também através dos alertas gerados pelo *IDS* é possível se ter informações preciosas em relação á segurança da rede. Dessa forma também se terá informações se existe alguma máquina comprometida ou atividades suspeitas na rede, tal como a origem das mesmas.

Na ocorrência de um possível ataque, se um *honeypot* estiver colocado a fim de causar cofunsão ao invasor, este não ficará apenas na geração de *log's*, mas também estará a mostrar ao atacante que existem outras máquinas para um outro possível ataque, ou até mesmo a anulação do ataque atual, onde seus esforços estarão voltados aos *honeypots.* Além dos registros de *log's* que ele faz, onde todo tráfego que é destinado a ele pode ser considerado suspeito ou malicioso devido a ninguém da rede ter o conhecimento da existência dele, a distração que pode causar a um possível atacante torna esta ferramenta de grande importância.

CONCLUSÃO

As ameaças que estão expostas as informações consideradas importantes para uma organização são inúmeras, mesmo que aparentemente não se haja percepção. Através dos *honeypot's* que são destinados a serem vistos como servidores importantes para possíveis intrusos e através da análise de seus *log's* para permitir visualizar ataques que podem não ser detectados pelos dispositivos de segurança.

Quando estas ferramentas são colocadas em um meio de produção, sejam para atuarem como falsos servidores ou falsas estações de trabalho é assim podendo ser simulado os ataques, como também *scanners* feitos por usuários mau-intencionados, como também feito por computadores de usuários que já estejam comprometidos por algum programa malicioso.

O *honeyd*, um *honeypot* de baixa iteração utilizado nesta pesquisa no ambiente *Linux* pode perfeitamente se fazer passar por um ou mais *hosts* utilizando os endereços de *ip* que estão livres na rede, onde consegue responder desde serviços simples, como até criar um endereço físico diferente para cada estação falsa criada por ele.

No ambiente em que foi realizado os testes, desde os testes de comunicação simples até as ferramentas confiáveis e conhecidas, como o *nmap* rodando no sistema operacional *Linux* e o Retina, para o *Windows* teve-se a visualização do *honeypot* como se fossem máquinas reais de produção, o que indica o quanto as ferramentas *honeypot's*, em particular o *honeyd* podem ser úteis.

No decorrer do trabalho, foram encontradas algumas dificuldades, como na instalação bo *BASE*, pois os vários tutoriais encontrados na internet não relatavam a instalação com os devidos detalhes, como também para o *honeyd*, devido a algumas alterações feitas nos arquivos de serviços utilizados, que foram a localização do *base.sh* e a localização para

armazenamento dos *log's*. Na utilização do *farpd*, as descrições encontradas apenas indicavam a sua utilização para os endereços disponíveis em uma rede inteira, mas após algum tempo foi descoberto que tem como especificar endereços individuais. Para o *VirtualBox* em ambiente *Linux* foi encontrado um problema, onde no após uma máquina virtual ter sido instalada não era possível a sua comunicação com a rede da máquina hospedeira, que com a criação de uma ponte entre a rede das máquinas virtuais com a rede da máquina hospedeira foi possível resolver também este problema.

A implantação de um *honeypot* em ambiente *Linux* com a análise de seus resultados foi realizada, e através dos testes comprovada a importância que um *honeypot* pode ter quando aplicado e configurado de forma correta, pois podem detectar atividades onde outros tipos de ferramentas em determinada situação não seriam capazes de detectar, sejam pela má configuração delas ou realmente por uma invasão que consiga passar por elas sem ser detectada.

Através desta pesquisa é possível dar continuidade para se fazer a análise e implementação *honeypot's* em redes *wirelles*, e um estudo estatístico com *honeypot's* sobre tentativas de invasões e de tráfego considerados suspeitos em algumas organizações, em casos onde as ferramentas de segurança utilizadas pelas mesmas não conseguem realizar a detecção destas informações. Este estudo teria como objetivo de trazer informações se há tráfego nas organizaçãções que não é detectado, e de fazer uma comparação entre elas, desde as tentativas que são detectadas antes do *firewall*, as tentativas que são detectadas na DMZ e que são detectadas dentro da rede de produção a fim de se ter o controle de todo o tráfego que há na organização.

REFERÊNCIAS

AMARAL, Eduardo Diniz. **Estudo E Implantação De Ferramentas Honeypots Na Rede Da Unimontes.** 2006. 117 f. Monografia (Sistemas de Informação) - Universidade Estadual de Montes Claros, Monte Carlos, 2006. Disponível em: <http://www.ccet.unimontes.br/arquivos/monografias/235.pdf> Acesso em 2 Nov. 2009, 20:15.

ASSUNÇÃO, Marcos F. A. **Honeypots e honeynets – Aprenda a detectar e enganar os invasores.** Santa Catarina: Visual Books, 2009. 122p.

BORGES, Pedro C.; COUTINHO, Rodrigo T. **Análise de Sistemas de Detecção de Intrusos em Redes de Computadores.** 2007. 133 f. Trabalho de Conclusão de Curso (Ciência da Computação) - Universidade de Franca – Franca, 2007.

CARUSO, Carlos A. A.; STEFFEN, Flávio Deny. **Segurança em informática e de informações**. 2.ed. São Paulo: SENAC/SP, 1999. 367 p.

CARVALHO, Luciano G. **Segurança de Redes**. Rio de Janeiro: Ed. Ciência Moderna, 2005. 73p.

FERREIRA, Fernando N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003. 160p.

GASPAR, Antônio E. O.; JESUS, Karla L. S.; SILVA, Michele C. **Um Estudo sobre Detecção de Intrusão.** 2008. 155 f. Monografia (Pós-Graduação em Suporte a Redes de Computadores e Tecnologia Internet) - Universidade Federal do Pará - Belém-PA, 2008.

Intrusion Detection: Snort, Base, MySQL, And Apache2 On Ubuntu 7.10 (Gutsy Gibbon). Disponível em: <http://www.howtoforge.com/intrusion-detection-with-snort-mysql-apache2-on-ubuntu-7.10> Acessado em 25 Nov. de 2009.

JÚNIOR, Carlos A. W.; **HONEYNET – Estudo Teórico E Prático.** 2007. 91 f. Trabalho de Conclusão de Curso (Ciência da Computação) - Centro Universitário Feevale, Novo Hamburgo, 2007. Disponível em: <http://tconline.feevale.br/tc/files/1044.pdf> Acesso: 07 Out. 2009.

LARI, Paulo A. M. ; AMARAL, Dino M. SNORT, MYSQL, APACHE E ACID. Rio de Janeiro: Brasport, 2004 103P.

MARCELO, Antonio; PITANGA, Marcos. **Honeypots: A arte de iludir hackers**.Rio de Janeiro: Brasport, 2003 100 p.

MARINHO, Renato Rodrigues. **Honeypots – Acompanhando os passos de uma invasão em tempo real**. 2004. 78 f. Monografia (Curso de Informática) - Universidade de Fortaleza, Fortaleza, 2004. Disponível em:

<http://www.dnocs.gov.br/php/util/downloads_file.php?&dir=&file=/home/util/livres/ebooks/monografias/honeypots_acompanhando_uma_invasao_em_tempo_real.pdf&> Acesso: 01 Nov. 2009 as 19:30.

NORTHCUTT, Stephen. **Como Detectar Invasão em Rede: Um guia para analistas**. Rio de Janeiro: Ciência Moderna, 2000. 277 p.

_____ et al. **Desvendando : segurança em redes**. Rio de Janeiro: Campus, 2002. 650 p.

NORTON, Peter; GRIFFITH, Artur. **Guia completo do Linux**. Tradução Sérgio Facchim. São Paulo: Berley. 2000.

NUNES, Bruno G.; DE SA, Diego P. M.; CID, Rafael M. **Honeypot.** 2006. 31 f. Trabalho apresentado para a disciplina de Tópicos Avançados em Redes do (Ciência da Computação) - Universidade Veiga de Almeida, Rio de Janeiro, 2006.

ROJAS, Gislaine Aparecida. **Análise de Intrusões através de Honeypots e Honeynets.** 2003. 67 f. Trabalho de graduação (Tecnólogo em Processamento de Dados) - Faculdade de Tecnologia de Americana. Americana – SP. 2003.

SANTOS, Bruno Ribeiro. **Detecção de Intrusos Utilizando o Snort.** 2005. 91 f. Monografia (Pós Graduação em Latu Senso em Administração de Rede Linux) – Universidade Federal de Lavras – Minas Gerais, 2005.

SPITZNER, Lance. Honeypots - Definitions and Value of Honeypots. **Disponível em:** <http://www.tracking-hackers.com/papers/honeypots.html> **Acessado em 02 Nov. 2009.**

SUZUKI, Luiz H.; DELPHINO, Alexandre D. **Implantação de um Honeypot e proposta para metodologia de Gerenciamento de Projetos de Honeynets.** 2007. 80 f. Monografia (Ciência da Computação) - Universidade de Brasília – UnB, Brasília, 2007. Disponível em: <http://monografias.cic.unb.br/dspace/bitstream/123456789/102/1/LuizAlexandre.pdf> Acesso 05 Out. 2009.

THOMAS, Tom. **Segurança de redes – Primeiros passos.** Rio de Janeiro: Ed Ciência Moderna, 2007. 395p.

VirtualBox: WindowsXP e Ubuntu Juntos. Guia de Instalação. Disponível em: <http://www.tecnoclasta.com/2008/07/03/virtualbox-windowsxp-e-ubuntu-juntos-guia-de-instalacao/#more-749> Acessado em 05 Jun. 2010.

WADLOW, Thomas A. **Segurança de redes: projeto e gerenciamento de redes seguras**. Rio de Janeiro: Campus, 2000. 269 p.

XAVIER, Victor Hugo Canali. **Tolerância a Intrusões: Uma Análise do uso do Snort.** 2007. 96 f. Monografia (Ciência da Computação) - Universidade de Passo Fundo – Passo Fundo, 2007.

APÊNDICE A – COMANDOS PARA A CRIAÇÃO DE HOSTS DO HONEYD

Neste apêndice serão mostrados os comandos principais para a criação de *hosts* no *honeyd*, como também a criação de serviços para o mesmo. Os comandos mostrados foram utilizados na configuração do arquivo *honeyd.conf* que se encontra no endereço */etc/honeypot.* Os comandos utilizados foram:

a) *create*: este comando indica que será criado um perfil de *host* virtual que terá como referencia o nome indicado após o comando. Ex: *create windows*;

b) *set* (perfil criado) *personality*: esta opção define o sistema operacional criado. O nome colocado será consultado no arquivo *nmap.prints*, por isso deve estar nesse arquivo a forma exata que for colocado. Ex: *set windows personality "Windows XP Professional"*;

c) *set* (perfil criado) *default tcp action reset* e *set* (perfil criado) *default udp action reset:* estes comandos indicarão que as portas *tcp/udp* estarão sendo simuladas como portas fechadas. Estas opções são importantes para enganar um *scanner*;

d) *add* (perfil criado) *tcp port* 80 *"sh /usr/share/honeyd/scripts/win32/win2k/iis.sh"*: com este comando é possível a criação de serviços para o *host*. Nesse comando foi criado na porta *tcp* 80 um servidor *iis*, onde nos parênteses vai o endereço do script utilizado;

e) *add* (perfil criado) *tcp port* 20 *open*: A opção de deixar portas abertas individuais é realizada através deste comando, que no exemplo foi deixado a porta *tcp* 20 aberta, mas poderia ser qualquer outra porta,como por exemplo: *add* (perfil criado) *udp port* 132 *open*;

f) *set* (perfil criado) *uptime*: a indicação de tempo que o sistema está ativo ser dar *boot* em segundos é mostrado por este comando. Ex: *set* (perfil criado) *uptime* 325691;

g) *bind* (*ip*) (perfil criado): define qual endereço de *ip* o *host* virtual *simulará* junto com o *arpd* para responder ao atacante como em real sistema. Ex: *bind* 192.168.1.10 (perfil criado) (MARCELO; PITANGA, 2003).

h) *add* (perfil criado*) tcp port* 21

"*sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh*": assim como no item "d" deste apêndice, que foi colocado um servidor *iis*, da mesma forma apenas mudando o numero da porta e o endereço do *script* foi colocado o serviço de *ftp* da *microsoft*. No caso para *Linux* deve ser um outro script em um outro endereço, como segue no comando a seguir: *add* (perfil criado) *tcp port* 21 *"sh /usr/share/honeyd/scripts/unix/general/ftp.sh"*. Para saber a localização dos serviços a serem utilizados, devem ser observados a localização dos *scripts* correspondentes.

i) *set* (perfil criado*) ethernet "08:00:27:B8:4D:2B"*: com este comando é definido um endereço de *mac* para a placa de rede, onde este é um endereço físico da placa.

APÊNDICE B – ARQUIVO *HONEYD.CONF* COM OS SCRIPTS UTILIZADOS PARA ESTE TRABALHO

O arquivo *honeyd.conf* que se encontra em */etc/honeypot* serve para criar os *hosts* virtuais com os seus devidos serviços. O *script* abaixo mostra o arquivo utilizado no trabalho.

```
################## windows server 2003 ######################
create windows
set windows personality "Microsoft Windows Server 2003"
set windows uptime 1508509
set windows default tcp action reset
set windows default udp action reset
set windows ethernet "08:00:27:B8:4D:2B"
set windows uid 1000 gid 1000
add windows tcp port 20 open
add windows tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh
"
add windows tcp port 25 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-
smtp.sh "
add windows tcp port 80 "sh /usr/share/honeyd/scripts/win32/win2k/iis.sh"
add windows tcp port 110 "sh
/usr/share/honeyd/scripts/win32/win2k/exchange-pop3.sh "
bind 192.168.1.10 windows


############ linux ######################
create linux
set linux personality "Linux 2.4.7 (X86)"
set linux uptime 3205540
set linux default tcp action reset
set linux default udp action reset
set linux ethernet "08:00:27:7B:39:76"
set linux uid 2000 gid 2000
add linux udp port 97 open
add linux tcp port 150 open
add linux tcp port 23 "sh
/usr/share/honeyd/scripts/unix/linux/suse8.0/telnetd.sh"
add linux tcp port 25 "sh
/usr/share/honeyd/scripts/unix/linux/suse8.0/sendmail.sh"
add linux tcp port 110 "sh
/usr/share/honeyd/scripts/unix/linux/suse8.0/qpop.sh"
add linux tcp port 3128 "sh
/usr/share/honeyd/scripts/unix/linux/suse8.0/squid.sh"
bind 192.168.1.11 linux


##### solaris 9 ###########
create solaris
set solaris personality "Sun Solaris 9 with TCP_STRONG_ISS set to 2"
set solaris uptime 2435789
set solaris default tcp action reset
set solaris default udp action reset
```

```
set solaris ethernet "08:00:27:9A:3D:EA"
add solaris udp port 3321 open
add solaris tcp port 4431 open
add solaris tcp port 5321 open
bind 192.168.1.12 solaris


################## estacoes windows XP ######################
create estacao_windows
set estacao_windows personality "Microsoft Windows XP Professional"
set estacao_windows uptime 1324578
set estacao_windows default tcp action reset
set estacao_windows default udp action reset
set estacao_windows ethernet "08:00:27:49:A2:18"
add estacao_windows tcp port 135 open
add estacao_windows tcp port 139 open
add estacao_windows tcp port 445 open
add estacao_windows udp port 123 open
add estacao_windows udp port 137 open
add estacao_windows udp port 138 open
add estacao_windows udp port 445 open
add estacao_windows udp port 500 open
bind 192.168.1.13 estacao_windows


create estacao2_windows
set estacao2_windows personality "Microsoft Windows XP Professional"
set estacao2_windows uptime 1324578
set estacao2_windows default tcp action reset
set estacao2_windows default udp action reset
set estacao2_windows ethernet "08:00:27:6B:41:D5"
add estacao2_windows tcp port 135 open
add estacao2_windows tcp port 139 open
add estacao2_windows tcp port 445 open
add estacao2_windows udp port 123 open
add estacao2_windows udp port 137 open
add estacao2_windows udp port 138 open
add estacao2_windows udp port 445 open
add estacao2_windows udp port 500 open
bind 192.168.1.14 estacao2_windows


create estacao3_windows
set estacao3_windows personality "Microsoft Windows XP Professional"
set estacao3_windows uptime 1324578
set estacao3_windows default tcp action reset
set estacao3_windows default udp action reset
set estacao3_windows ethernet "08:00:27:B8:CC:DF"
add estacao3_windows tcp port 135 open
add estacao3_windows tcp port 139 open
add estacao3_windows tcp port 445 open
add estacao3_windows udp port 123 open
add estacao3_windows udp port 137 open
add estacao3_windows udp port 138 open
add estacao3_windows udp port 445 open
add estacao3_windows udp port 500 open
bind 192.168.1.15 estacao3_windows
create estacao4_windows
```

```
set estacao4_windows personality "Microsoft Windows XP Professional"
set estacao4_windows uptime 1324578
set estacao4_windows default tcp action reset
set estacao4_windows default udp action reset
set estacao4_windows ethernet "08:00:27:28:41:23"
add estacao4_windows tcp port 135 open
add estacao4_windows tcp port 139 open
add estacao4_windows tcp port 445 open
add estacao4_windows udp port 123 open
add estacao4_windows udp port 137 open
add estacao4_windows udp port 138 open
add estacao4_windows udp port 445 open
add estacao4_windows udp port 500 open
bind 192.168.1.16 estacao4_windows
```

ANEXO A – INSTALAÇÃO DO *VIRTUALBOX*

Este anexo foi retirado do endereço *http://www.gdhpress.com.br/blog/virtualbox-no-ubuntu/*

O *VirtualBox* está disponível nos repositórios do *Ubuntu* desde a versão 7.10. Se você está usando o *Ubuntu*, *Kubuntu*, *Xubuntu* ou outra distribuição derivada deles, você pode instalá-lo diretamente via *apt-get*, a partir dos repositórios principais:

$ sudo apt-get install virtualbox-ose

Em seguida, use o comando *"uname -r"* para verificar qual é a versão do *Kernel* em uso e instale a versão correspondente do pacote *"virtualbox-ose-modules"*, como em:

$ uname -r
2.6.24-16-generic

$ sudo apt-get install virtualbox-ose-modules-2.6.24-16-generic

Se, por acaso, você estiver usando um *Kernel* personalizado, ou não houver um módulo com a versão correta, você pode fazer a instalação do módulo usando o *module-assistant*, que se encarregará de compilar um módulo sob medida. Comece instalando o pacote, juntamente com o *build-essential*, que contém os compiladores básicos:

$ sudo apt-get install module-assistant build-essential

Em seguida, gere o módulo usando o comando *"m-a a-i"* (versão abreviada do comando *"module-assistant auto-install*"), como em:

$ sudo m-a a-i virtualbox-ose-source

Isso compilará e instalará o módulo automaticamente. Se houver algum erro nesse ponto, verifique se o pacote *"build-essential"* foi corretamente instalado e se você não deletou acidentalmente os *headers* do *Kernel*, que são armazenados na pasta *"/usr/src"*.

O ícone para o *VirtualBox* é criado no "Aplicações > Ferramentas do Sistema", mas você pode chamá-lo diretamente via terminal. Entretanto, da primeira vez que tentar executar alguma máquina virtual, receberá uma mensagem de erro avisando que o *driver* não está acessível para o usuário atual.

Este é um velho *bug* de usabilidade, que ocorre devido ao uso de um grupo de acesso ao módulo do *VirtualBox*. Por padrão, apenas o *root* faz parte do grupo, de forma que nenhum usuário do sistema, incluindo a conta administrativa que foi criada durante a instalação tem permissão para usar o programa. Felizmente, ele é fácil de resolver, basta adicionar seu *login* de usuário ao grupo "*vboxusers*", como em:

$ sudo adduser gdh vboxusers

Para que a alteração entre em vigor, e necessário fazer *logout* (ou reiniciar o micro, o que preferir). A partir daí, é só abrir o *VirtualBox*, criar a máquina virtual.

ANEXO B – INSTALAÇÃO DO SNORT JUNTO COM O BASE

Este apêndice é uma tradução de um tutorial que foi encontrado do seguinte endereço: *http://www.howtoforge.com/intrusion-detection-with-snort-mysql-apache2-on-ubuntu-7.10* que explica com detalhes a instalação do *Snort* junto com o *BASE*. Explica as configurações necessárias a se fazer, e também os arquivos dependentes necessários para a instalação. Embora a descrição seja completa, como no manual se trata do *Ubuntu* em sua versão 7.10, as versões dos pacotes e ferramentas estão desatualizadas. Basta apenas substituir as versões descritas pelas versões mais atualizadas dos pacotes a serem instalados. Uma observação no passo 12 pode ser notada que há duas indicações diferentes que se referem como passo 12. devido a ser uma tradução ao documento original, foi desta forma mantida.

Para instalar o *BASE* que é utilizado para a visualização dos *log's* do *snort* através de um *browser*, são necessário também a instalação do *Mysql*, do *Apache2* e de vários outros arquivos que serão mostrado nesse apêndice. A instalação mostrada será através de passos no *Ubuntu*, pois este é o sistema utilizado neste trabalho.

Passo 1: Pré-requisitos

Inicialmente deve-se baixar vários arquivos dependentes. Através do *Synaptic*, que está localizado a partir da área de trabalho em: Sistema->Administração->Gerenciador de Pacotes S*ymaptic*. Digite o *password* (caso peça) e pesquise os seguintes pacotes para instalar:

a) *Libpcap0.8-dev*
b) *libmysqlclient15-dev*
c) *mysql-client-5.0*
d) *mysql-server-5.0*

e) *bison*

f) *flex*

g) *apache2*

h) *libapache2-mod-php5*

i) *php5-gd*

j) *php5-mysql*

k) *libphp-adodb*

l) *php-pear*

Passo 2: Adquirir privilégios

Após a instalação dos pacotes, vá até o terminal e digite:

sudo –i

Agora digite a sua senha.

Passo 3: Fazer download e descompactar arquivos

Verifique se o pacote referenciado pelo comando abaixo está instalado. Caso não esteja baixe também este pacote:

apt-get install libc6-dev g++ gcc.

Deve ser criada uma pasta *temp* para que seja feito o *download* e a descompactação dos arquivos. O nome da pasta será *snorttmp*, que é criado com os seguintes comandos:

cd /root

mkdir snorttmp

cd /root/snorttmp

será iniciada a instalação do *snort*, que de acordo com a indicação da época desse *tutorial* se encontra na versão 2.8.0. vá até o site *http://www.snort.org/downloads* e escolha a versão mais recente, ou digite no terminal:

wget http://www.snort.org/dl/current/snort-2.8.0.tar.gz

mudando apenas a versão do *snort* pela versão mais atual. Descompacte e remova o arquivo *.tar*:

tar -xzvf /root/snorttmp/snort-2.8.0.tar.gz
rm /root/snorttmp/snort-2.8.0.tar.gz

Passo 4: Obter regras do *Snort*

Vá até a pasta *snort-2.8.0*:

cd /root/snorttmp/snort-2.8.0

Vá até o site onde estão as regras para fazer o *download* das mesmas, mas antes é necessário ter registro no site. O registro é grátis e você terá acesso somente às regras após um período de 30 dias após ela for liberada. Para ter acesso logo após a sua liberação deve-se se tornar assinante do site através de um pagamento que é feito anualmente. O site que se encontram as regras é o seguinte: *http://www.snort.org/snort-rules*, ou após estar logado no site, digite no terminal:

wget http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz

onde o "2.4" é referente as regras atualizadas na época que foi feito esse tutorial. Descompacte e remova o arquivo *.tar:*

tar -xzvf /root/snorttmp/snort-2.8.0/snortrules-pr-2.4.tar.gz

rm /root/snorttmp/snort-2.8.0/snortrules-pr-2.4.tar.gz

Passo 5: Obter PCRE - Expressões Regulares Compatível com *Perl*

Vá para a pasta *snorttmp*:

cd /root/snorttmp

Abra um navegador web e vá para *http://www.pcre.org,* busque a versão mais atual e clique para baixa-la. No momento desse manual era a versão *pcre-7.4*. No terminal digite:

wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.4.tar.gz.

Descompacte e remova o arquivo *PCRE*:

tar -xzvf /root/snorttmp/pcre-7.4.tar.gz

rm /root/snorttmp/pcre-7.4.tar.gz

Passo 6: Obter o *BASE (Basic Analysis and Security Engine).*

Vá para a pasta temporária:

cd /root/snorttmp,

abra o navegador *web* e vá para o seguinte endereço:
http://sourceforge.net/project/showfiles.php?group_id=103348, busque o pacote mais recente, que no momento era o *base-1.3.8* ou digite no terminal o seguinte comando, apenas substituindo a versão pela versão atualizada:

wget http://downloads.sourceforge.net/secureideas/base-1.3.8.tar.gz?modtime=1183896336&big_mirror=0.

Descompacte e remova o arquivo *BASE*:

tar -xzvf /root/snorttmp/base-1.3.8.tar.gz

rm /root/snorttmp/base-1.3.8.tar.gz

Passo 7: Obter ADOdb (biblioteca de abstração de banco de dados para PHP)

Vá para a pasta *snorttmp*: *cd /root/snorttmp*

Abra um navegador web e vá para o site: *http://sourceforge.net/project/showfiles.php?group_id=42718*. Clique no link para *download* do *adodb-php5* e baixe o pacote *adodb502a.tgz*, que era a versão atual quando foi feito este tutorial, ou digite no terminal:

wget http://downloads.sourceforge.net/adodb/adodb502a.tgz?modtime=1191343792&big_mirror=0

Descompacte e remova o arquivo.

tar -xzvf /root/snorttmp/adodb502a.tgz

rm /root/snorttmp/adodb502a.tgz

Digite o comando *ls* para ter a certeza que estão todos os pacotes, onde você deverá ter as seguintes pastas referentes as versões instaladas:

adodb5

base-1.3.8

pcre-7.4

snort-2.8.0

Passo 8: Instalação

A instalação será feita em duas partes:

a) Instalação do *PCRE*: vá para a pasta específica do *PCRE* com o comando

cd /root/snorttmp/pcre-7.4

e faça a instalação através dos comandos:

./configure

make

make install

b) Instalação do *Snort*: Vá para o diretório do *snort*, com o comando

cd /root/snorttmp/snort-2.8.0

e faça a instalação:

./configure -enable-dynamicplugin --with-mysql

make

make install

Passo 9: Compilando os arquivos

Deve ser criada algumas pastas no diretório */etc* afim de copiar alguns arquivos:

mkdir /etc/snort /etc/snort/rules /var/log/snort .

Depois mova os arquivos:

cd /root/snorttmp/snort-2.8.0/rules

*cp * /etc/snort/rules/*

mova também:

cd /root/snorttmp/snort-2.8.0/etc

*cp * /etc/snort/*

e também essa arquivo:

cp /usr/local/lib/libpcre.so.0 /usr/lib

Passo 10: Configuração do *Snort*

Abra o arquivo */etc/snort.conf* com um editor de textos para que nele sejam feitas alterações. *vim /etc/snort/snort.conf.* Mude onde está *"var HOME_NET any"* para a rede que será utilizada, que no caso deste trabalho ficaria: *"var HOME_NET 192.168.1.0/24".*
Mude *"var EXTERNAL_NET any"* para *"var EXTERNAL_NET !$HOME_NET"* contando que *$HOME_NET* não seja como rede externa. Mude também *"var RULE_PATE ../rules"* para *"var RULE_PATH /etc/snort/rules". o*nde está escrito "# output database: log, mysql, user=", retire o *"#"* da frente da linha. Deixe onde está *"user=root"* e altere onde está escrito *"password=password"* para *"password=sua senha",* e altere também o *"dbname"* para *"snort",* ficando desta forma: *"dbname=snort".* Salve o arquivo e saia.

Passo 11: Configurar o banco de dados Mysql

Faça o *login* no servidor *mysql*:

mysql -u root –p.

As vezes não há uma senha definida, apenas pressione *enter*. Se você receber uma falha de *logon*, tente o comando acima novamente e digite a sua senha, mas se ainda não foi definida, você precisará definir para a conta *root*, que é feita através do comando:

mysql> SET PASSWORD FOR root@localhost=PASSWORD('SUA SENHA');

Agora você deve criar o banco de dados *snort*:

mysql> create database snort;

mysql> exit

Digite também o comando:

mysql -D snort -u root -p < /root/snorttmp/snort-2.8.0/schemas/create_mysql.

É necessário comentar algumas linhas nas regras da *web*, por isso abra o arquivo */etc/snort/rules/web-misc.rules* com um editor de texto. Comente as linhas 97,98 e 452 com um "#" (sem aspas) logo no início da linha.

PASSO 12: Testar o Snort

No terminal digite:

snort -c /etc/snort/snort.conf.

Se tudo correr bem você deverá ver o desenho de um porco em *ascii*, onde *ascii* é um código que represente textos em computadores e outros dispositivos que trabalham com texto. Para encerrar o programa pressione *ctrl+c*.

Passo 12: Base e Apache2

Contando que o *Apache2* e o *BASE* já estejam corretamente instalados, deve-se mover alguns arquivos e modificar um arquivo de configuração. Crie um arquivo chamado *teste.php* em */var/www/* com um editor de textos:

vim /var/www/test.php

e escreva nele:

<?php

phpinfo();

?>

Salve e feche o arquivo. Agora é necessário editar o arquivo */etc/php5/apache2/php.ini* com o seguinte comando:

vim /etc/php5/apache2/php.ini.

Você precisa adicionar o seguinte em *"Dynamic Extensions":*

extension=mysql.so

extension=gd.so

Reinicie o *apache2*:

/etc/init.d/apache2 restart.

Digite o comando *ifconfig –a* para obter o endereço *IP* da máquina que você está trabalhando.

Abra um navegador web e digite:

http://SEU ENDEREÇO DE IP/test.php.

Se tudo estiver correto aparecerá informações sobre o *php*.

Passo 13: Movendo mais arquivos

Mova o *ADOdb* para o diretório */var/www*:

mv /root/snorttmp/adodb5 /var/www.

Agora faça uma pasta em */var/ww* e mova o *BASE* para lá:

mkdir /var/www/web

mv /root/snorttmp/base-1.3.8 /var/www/web/

É necessário permissão de escrita na pasta *base-1.3.8* para realizar a sua instalação:

chmod 757 /var/www/web/base-1.3.8.

Também é necessário modificar a configuração do *PHP* através de um editor de texto:

vim /var/www/web/base-1.3.8/setup/setup1.php.

Encontre a linha onde está *"base_header"* e modifique para *"header"*. Salve e saia.

Faça as instalações a seguir para que possa ser gerado gráficos com os comandos:

pear install Image_Color

pear install Image_Canvas-alpha

pear install Image_Graph-alpha

Passo 14: Configuração do *BASE* para a *web*

Abra em seu navegador o endereço: *http://SEU ENDEREÇO DE IP/web/base-1.3.8/setup* e siga os passos:

Clique em continuar na primeira página

Etapa 1 de 5: Digite o caminho para ADODB. Isto é / var/www/adodb5.

Etapa 2 de 5: faça as seguintes definições:

Database type = MySQL, *Database name* = snort, *Database Host* = localhost, *Database username* = root, *Database Password* = sua senha

Passo 3 de 5: Se você quiser usar a autenticação digite um nome de usuário e senha aqui.

Etapa 4 de 5: Clique em *Create BASE AG.*

Etapa 5 de 5: Uma etapa 4 é feito na parte inferior clique em Continue agora para o passo 5.

Guarde esta página e altere novamente as permissões da pasta *var/www/web/base-1.3.8/*.

chmod 775 /var/www/web/base-1.3.8

Está terminado, agora faça os testes

Insira o seguinte comando no terminal para inicializar o *Snort:*

snort -c /etc/snort/snort.conf -i eth0 -D

Este comando executa o *snort usando a interface eth0* em segundo plano. Para se certificar de que está funcionando, digite o seguinte comando:

ps aux | grep snort.

Se o *snort* estiver funcionando será mostrado algo semelhante a *snort -c /etc/snort/snort.conf -i eth0 –D.*