Centro Universitário de Brasília
Instituto CEUB de Pesquisa e Desenvolvimento - ICPD

GILDÁSIO ANTONIO DE OLIVEIRA JÚNIOR

DESENVOLVIMENTO DE UM AMBIENTE HONEYNET VIRTUAL DE AUTOCONTENÇÃO PARA ÓRGÃO GOVERNAMENTAL

Brasília
2015

GILDÁSIO ANTONIO DE OLIVEIRA JÚNIOR

DESENVOLVIMENTO DE UM AMBIENTE HONEYNET VIRTUAL DE AUTOCONTENÇÃO PARA ÓRGÃO GOVERNAMENTAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para obtenção de Certificado de Conclusão de Curso de Pós-graduação *Lato Sensu* em Rede de Computadores com Ênfase em Segurança

Orientador: Prof. Dr. José Eduardo Malta de Sá Brandão

Brasília
2015

GILDÁSIO ANTONIO DE OLIVEIRA JÚNIOR

DESENVOLVIMENTO DE UM AMBIENTE HONEYNET VIRTUAL DE AUTOCONTENÇÃO PARA ÓRGÃO GOVERNAMENTAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Pós-graduação *Lato Sensu* em Rede de Computadores com Ênfase em Segurança

Orientador: Prof. Dr. José Eduardo Malta de Sá Brandão

Brasília, 30 de maio de 2015.

Banca Examinadora

_____

Prof. Esp. Silas Rodrigues Mendes

_____

Prof. Dr. Gilson Ciarallo

A Deus, aos meus pais, a minha esposa e aos pesquisadores deste país que, apesar das dificuldades continuam trilhando esse caminho.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me ajudar nos momentos difíceis e me mostrar novas oportunidades que contribuirão para o meu crescimento pessoal e profissional. Aos meus pais pela excelente formação e educação, a minha esposa, Sirrame pela motivação, apoio e incentivo nas horas difíceis.

Ao meu professor e orientador José Eduardo Malta de Sá Brandão pelas idéias e ensinamentos.

Aos pesquisadores deste país que de certa forma contribuíram para realização deste trabalho.

O propósito do aprendizado é crescer, e nossas mentes, diferentes de nossos corpos, podem continuar crescendo enquanto continuamos a viver.

Mortimer Adler

RESUMO

Recentemente, técnicas de detecção de intrusos têm sido aplicadas para detectarem tráfego malicioso. Por conta do extenso número de vulnerabilidades em sistemas de informação e da grande criatividade dos invasores, torna-se cada vez mais difícil a proteção de ativos de redes apenas com estes dispositivos tradicionais de segurança. Portanto é crucial ter um ambiente cibernético preparado para ser invadido e comprometido com a finalidade de analisar e verificar a evolução dos diversos tipos de ataques e vulnerabilidades exploradas por invasores. Nesse contexto, este trabalho apresenta uma solução de segurança projetada especificamente para pesquisa e obtenção de informações de intrusos. A arquitetura do trabalho foi dividida em três fases distintas: construção de um ambiente *honeynet* virtual de autocontenção, validação do ambiente e um estudo de caso.

Palavras-chave: *Honeynets. Honeypots. Intrusion Detection Systems (IDS)*. Controle de Dados de Intrusões. Captura de Dados de Intrusões.

ABSTRACT

Recently, intrusion detection techniques have been applied to detect malicious traffic. Because of the large number of vulnerabilities in information systems and the great creativity of the invaders, it becomes increasingly difficult to network asset protection only with these traditional security devices. Therefore it is crucial to have a cyber environment prepared to be invaded and committed in order to analyze and verify the evolution of the various types of attacks and vulnerabilities exploited by attackers. In this context, this paper presents a security solution specifically designed for research and obtaining information from intruders. The architecture work has been divided into three distinct phases: construction of a virtual honeynet environment of self-restraint, environmental validation and a case study.

Key words: *Honeynets. Honeypots. Intrusion Detection Systems (IDS). Intrusion Data Control. Intrusion Data Capture.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

ASCII	*American Standard Code for Information Interchange*
BSD	*Berkeley Software Distribution*
CIDR	Classless Inter-Domain Routing
DNS	*Domain Name System*
DTK	*Deception Toolkit*
FTP	*File Transfer Protocol*
GUI	*Graphical User Interface*
HIDS	*Host Intrusion Detection System*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
IRC	*Internet Relay Chat*
MAC	*Media Access Control*
NIDS	*Network Intrusion Detection System*
NTP	*Network Time Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
RAID	*Redundant Array of Independent Disks*
SCP	*Session Control Protocol*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
TCP	*Transmission Control Protocol*

SUMÁRIO

INTRODUÇÃO

Atualmente um dos principais problemas de segurança enfrentados no ciberespaço é a invasão de redes de computadores. Conforme estatísticas apresentadas pelo CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil), em 2014 houve 467.621 notificações tentativas de fraudes e um aumento de 54% a ataques de servidores *Web* em relação ao ano anterior (CERT.BR, 2015). O rápido avanço de informações através da internet aumentou a possibilidade de vulnerabilidades e gerou novas formas de atividades intrusivas. Por conta desse crescimento exagerado de informações o ciberespaço se tornou um campo de guerra cibernética, uma guerra invisível e interminável. Desta forma torna-se fundamental a aplicação de camadas de segurança em alto nível para proteger os ativos de rede contra ameaças e garantir a integridade, a confidencialidade e a disponibilidade dos dados trafegados.

Grande parte das ferramentas utilizadas para segurança da informação com a finalidade de combater ataques, tais como *firewall*, aplicações de criptografia, *hash*, assinatura digital, *antivírus*, entre outros, não são suficientes para assegurar a segurança de redes e sistemas.

As tecnologias de detecção de intrusão trabalham em conjunto com outros mecanismos de segurança, buscando sempre incidentes de ataques. Segundo Scarfone e Mell (2007), detecção de intrusão é o processo de monitoramento de eventos que ocorrem em um sistema de computador ou rede para detectar sinais de possíveis incidentes.

Estes sistemas são classificados de acordo com as técnicas utilizadas para procurar comportamento anormal nos seus registros. A arquitetura de um IDS (*Intrusion Detection System*) depende de sua localização e da forma como os dados são coletados. Podem ser classificados em duas categorias: baseado em host HIDS (*Host Intrusion Detection System*) e baseado em rede NIDS (*Network Intrusion Detection System*).

Além disso, o método utilizado para detecção de intrusos em um IDS é o ponto mais importante desses sistemas, pois definem como os dados são

analisados e qual a forma de atuação de acordo com o ataque. Os principais métodos utilizados são: detecção por assinatura e detecção por anomalia.

O grande problema da implementação de um IDS é que ele garante apenas um certo nível de segurança. Como coletar e analisar os dados de entrada com os possíveis ataques desconhecidos na rede de uma organização? A solução seria construir um recurso de segurança chamado *honeynet*.

Honeynet é uma ferramenta projetada especificamente para pesquisa e obtenção de informações de intrusos (PROJECTS, 2002). Em uma *honeynet* o sistema operacional e serviços não são emulados como em uma *honeypot* tradicional. Esta rede quando comprometida pode ser usada para verificar e analisar os diversos tipos de ataques e vulnerabilidades exploradas pelos invasores.

Este trabalho tem como finalidade a contribuição de conhecimento com Organizações Governamentais sobre segurança em redes de computadores, mostrando uma solução nova e promissora como fonte de pesquisa.

O objetivo do trabalho é desenvolver um ambiente *honeynet* virtual de autocontenção dentro de uma Organização Governamental para ser invadido e comprometido, para isso, foi utilizado um laboratório de pesquisa do Governo Federal no qual foram realizadas coletas dentro da *honeynet* para demonstrar através de pesquisa e estudo as formas de ataques encontradas.

A metodologia aplicada para o desenvolvimento deste trabalho foi composta por pesquisa, validação e comparação de técnicas aplicadas para o desenvolvimento de um ambiente *honeynet virtual* de autocontenção. Portanto, foram feitas pesquisas das diversas fontes da literatura sobre Segurança de Redes, Sistemas de Detecção de Intrusos, *Honeypots* e *Honeynets*.

Este trabalho está organizado da seguinte forma: No capítulo 1 apresentamos os conceitos relacionados a *honeynets*. É detalhada no capítulo 2 a arquitetura proposta do ambiente *honeynet* virtual de autocontenção para ser invadido e comprometido. No capítulo 3 validamos o ambiente através dos requisitos de controle, captura e análise dos dados. No capítulo 4 simulamos dois ataques de

força bruta, para demonstrar de forma detalhada o funcionamento do ambiente *honeynet*.

Espera-se demonstrar com este trabalho a importância de ter um ambiente cibernético dentro de um Órgão Governamental para acompanhar a evolução dos diversos tipos de ataques a fim de impedir intrusões maliciosas nos sistemas, garantindo desta forma, o funcionamento dos serviços essenciais à população.

1 HONEYNETS

Conforme Project (2002), *honeynet* é um conjunto de *honeypots* de alta interatividade projetada especificamente para ser invadida e comprometida. Diferentemente dos *honeypots* de baixa interatividade que emulam sistemas operacionais e serviços os *honeypots* de alta interatividade fornecem sistemas operacionais e aplicações reais para que os intrusos possam interagir. Esta interatividade faz com que pesquisadores observem o comportamento de um intruso em um sistema real fornecendo desta forma, oportunidades para descobrir novas ferramentas, identificar novas vulnerabilidades e aprender como estes intrusos se comunicam.

Referências sobre mecanismos para acompanhamentos de atividades intrusivas surgiram no final da década de 1980. Em agosto de 1986 um intruso atacou computadores dos laboratórios da LBL (Lawrence Berkeley Laboratory) para roubar dados. Com a finalidade de monitorar estes intrusos, Clifford Stoll criou um projeto governamental para rastrear com detalhes os ataques até sua origem (STOLL, 1988).

Em 1990, Bill Cheswick descreveu o acompanhamento de uma invasão no laboratório da AT&T. Nesta invasão, foram exploradas falhas no Serviço *Sendmail* obtendo-se acesso ao *gateway* do laboratório. A finalidade desta experiência era localizar e aprender sobre as técnicas que foram utilizadas pelos intrusos. Uma gaiola *chroot* foi construída para observar todas as atividades que o intruso queria fazer (CHESWICK, 1990).

Cohen (1998) desenvolveu o DTK (*Deception Toolkit*), uma coleção de *scripts Perl* e código C, que emulava várias vulnerabilidades conhecidas do *Unix,* com o propósito de obter informações e enganar atacantes. Este *toolkit* pode ser utilizado também para alertar e aprender sobre vulnerabilidades conhecidas.

Em 1999 Lance Spitzer liderou um grupo de 30 profissionais sem fins lucrativos, da área de segurança dedicados a aprender técnicas, táticas e motivações de intrusos. Em 2001, os membros do projeto lançaram o livro “*Know Your Enemy*”, baseado em dois anos de pesquisas que descrevia em detalhes as tecnologias *Honeynet* (SPITZER, 2002). Ainda em dezembro de 2001, o *Honeynet*

Project anunciou a "*Research Alliance Honeynet*" com o objetivo de melhorar as pesquisas e desenvolvimento de *honeynets*. Depois do *Honeynet Project* muitos autores criaram definições e classificações que serão descritas nas próximas seções.

1.1 Arquitetura de uma Honeynet

Um projeto *honeynet* só terá sucesso se sua arquitetura estiver bem definida. Portanto, a construção e manutenção de uma *honeynet* dependem de três requisitos críticos: controle de dados, captura de dados e coleta de dados (SPITZNER, 2002). O controle e a captura dos dados são os requisitos mais importantes da arquitetura. O terceiro requisito só será utilizado em organizações que tenham vários *honeypots* em ambientes distribuídos.

1.1.1 Controle de Dados

Requisito muito crítico, o controle de dados tem como finalidade controlar os dados de entrada e saída para reduzir os riscos dentro da *honeynet*. Isto garante que sistemas comprometidos não sejam usados para atacar sistemas de produção de outras redes. O tráfego de dados deve ser controlado de modo automático e transparente. O primeiro reduz de forma rápida qualquer dano no sistema, e o segundo garante que intrusos não percebam que suas atividades estão sendo controladas.

Segundo Spitzner (2002), existem oito requisitos específicos essenciais de controle de dados que podem ser implementados para reduzir os riscos em uma *honeynet*:

- O controle de dados deve ser implementado de forma automática e manual;
- Pelo menos duas camadas de controle de dados para proteção de falhas;
- Capacidade de manter o estado de todas as conexões de entrada e saída;
- Capacidade de controlar qualquer atividade não autorizada;
- Capacidade de configurar o controle de dados em qualquer momento;

- Controle de conexões para dificultar a detecção do sistema por intrusos;
- Pelo menos dois métodos de alerta para *honeypots* comprometidos; e
- Administração remota do controle de dados.

Desta forma, o controle de dados deve ser utilizado para separar a *honeynet* das outras redes, tais como: *Internet*, administrativa e produção. Isso faz com que cada pacote seja controlado e inspecionado quando entram e saem da *honeynet*. Geralmente os ambientes permitem que qualquer sistema iniciem conexões com a *honeynet,* permitindo que intrusos sondem, identifiquem e explorem os sistemas vulneráveis dentro da *honeynet*.

1.1.2 Captura de Dados

Segundo requisito da arquitetura de uma *honeynet*, a captura de dados tem como objetivo coletar todas as atividades dos intrusos dentro da *honeynet,* incluindo conexões de entrada, atividades de rede e sistema. Conforme Project (2002), a captura adequada dos dados é crítica para o sucesso do projeto, portanto quanto mais métodos de captura de dados o projeto tiver melhor.

Entretanto, nenhum dado capturado deve ser armazenado localmente nos *honeypots*. Estes dados devem ser armazenados em um local seguro e confiável. Dados armazenados localmente podem ser detectados por intrusos e utilizados para comprometer o sistema. Além disso, estes dados podem ser modificados e destruídos.

1.1.3 Coleta de Dados

Organizações com apenas uma *honeynet* não precisam utilizar este requisito. A coleta de dados só será aplicada em organizações que possuem várias *honeynets* em ambientes distribuídos. Neste caso, todos os dados capturados deverão ser armazenados em um local central para poderem ser correlacionados e aumentar o valor das *honeynets* de pesquisa.

Conforme Spitzner (2002), se a *honeynet* faz parte de um ambiente distribuído, então quatro requisitos específicos para coleta de dados devem ser aplicados:

- Cada *honeynet* deverá ter um identificador único;
- Os dados deverão ser transmitidos através de sensores para um coletor de forma segura, garantido confidencialidade, integridade e autenticidade dos dados;
- O anonimato dos dados deverá ser garantido; e
- Um servidor NTP deverá ser utilizado para garantir que os dados capturados na *honeynet* distribuída estejam devidamente sincronizados.

1.2 Honeynets Reais

As *honeynets* reais fornecem sistemas operacionais reais para que os intrusos possam interagir. O objetivo dessa interação é mostrar como eles invadem os sistemas, como se comunicam e qual a finalidade do ataque (PROJECTS ; SPITZNER, 2002). Estas informações podem ser de extrema importância para que Órgãos Governamentais compreendam e protejam seus sistemas contra ameaças e ataques.

Neste tipo de *honeynet* todos os dispositivos e mecanismos de segurança (*honeypots*, contenção, alerta e coleta de informações) são físicos (PROJECTS ; SPITZNER, 2002). O Quadro 1 apresenta as principais vantagens e desvantagens das *honeynets* reais.

Quadro 1 - Vantagens e desvantagens das *honeynets* reais.

Vantagens	Desvantagens
Intrusos interagem com dispositivos físicos reais	Custo de implementação e espaço físico
Ambiente distribuído (tolerante a falhas)	Dificuldade de instalação e administração
	Complexidade de manutenção

Fonte: Elaborado pelo autor do trabalho.

1.3 Honeynets Virtuais

Conforme seção anterior as *honeynets* tradicionais são difíceis e complexas de construir, além disso, sua implementação exige uma variedade de sistemas físicos e mecanismos de segurança. Por outro lado as *honeynets* virtuais permitem executar todos os sistemas operacionais, aplicações e serviços no mesmo hardware através de um software de virtualização (SPITZNER, 2002) e (THE HONEYNET PROJETCT, 2003). O Quadro 2 apresenta as principais vantagens e desvantagens das *honeynets* virtuais.

Quadro 2 - Vantagens e desvantagens das *honeynets* reais.

Vantagens	Desvantagens
Custo e espaço físico reduzido	Limitação e comprometimento do software de virtualização. Neste caso o intruso poderá controlar toda a *honeynet*
Facilidade de manutenção e administração	Risco de *fingerpriting*. Os intrusos poderão detectar se os sistemas estão sendo executados em um software de virtualização

Fonte: Elaborado pelo autor do trabalho.

As *honeynets* virtuais estão divididas ainda em duas categorias: autocontenção e híbridas. Na primeira, todos os dispositivos, incluindo captura e coleta de dados, geração de alertas e os *honeypots* estão implementados em um único computador. A segunda é uma combinação das *honeynets* tradicionais e virtuais. Nesta categoria, captura, controle de dados e sistemas de *logs* são implementados em dispositivos físicos distintos. Os *honeypots* são configurados em um único computador através de um software de virtualização.

2 PROPOSTA DO AMBIENTE HONEYNET VIRTUAL DE AUTOCONTENÇÃO

Neste capítulo serão abordados os aspectos relacionados ao desenvolvimento de um ambiente *honeynet* virtual de autocontenção para ser invadido e comprometido com a finalidade de detectar e capturar ataques novos e desconhecidos em Órgãos Governamentais. Uma das grandes vantagens deste ambiente é que grande parte do tráfego capturado terá origem ilícita ou maliciosa, ou seja, a maioria do tráfego estará comprometido por códigos maliciosos.

2.1 Arquitetura proposta e modelo de solução

Nesta fase a arquitetura foi desenvolvida com o *Hypervisor XenServer*, uma solução de virtualização de código aberto que possibilita gerenciar infraestruturas virtuais. Toda a estrutura do projeto será feita em apenas um servidor físico. Recomenda-se a utilização de hardware dedicado para que as máquinas virtuais possam ser executadas corretamente. O Quadro 3, apresenta as características do servidor e plataforma de virtualização utilizada.

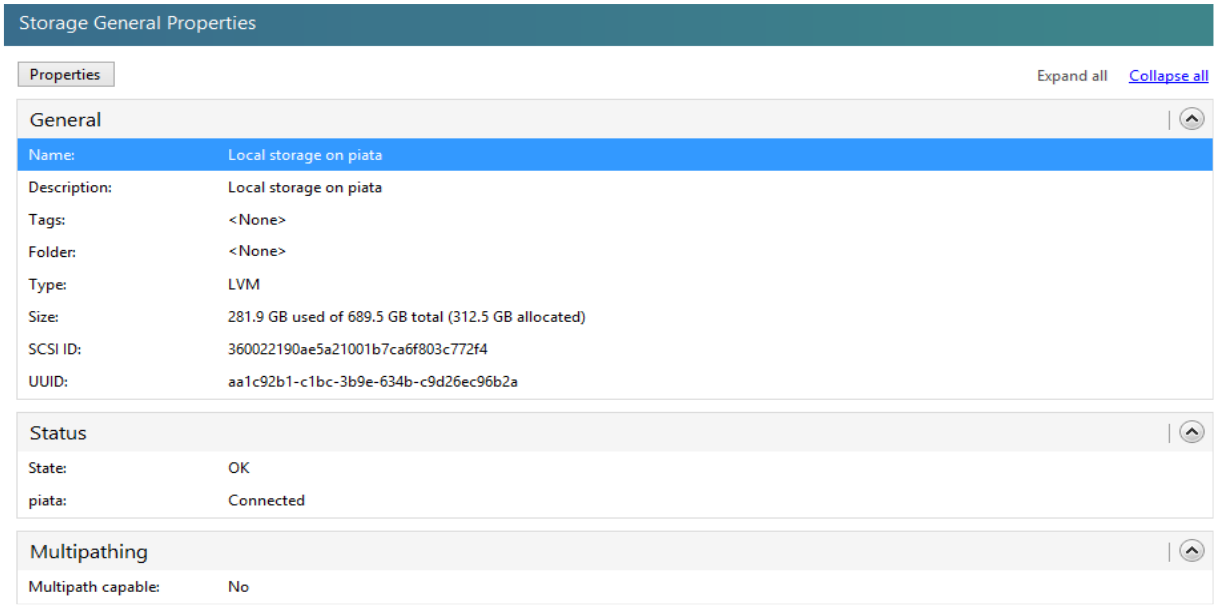
Quadro 3 - Características do servidor e plataforma de virtualização utilizada.

Servidor Dell PowerEdge 2950	Configuração
Hypervisor XenServer	Processador Intel *Xeon* E5410 2.33 GHz 8 núcleos com tecnologia Intel VT , 8 GB de memória RAM, 2 HDs de 250 GB, 2 HDs de 500 GB e 2 placas de rede 10/100/1000
	Plataforma *XenServer* 6.2 com as *features* XS62ESP1, XS62E001, XS62E002, XS62E004, XS62E005, XS62E007, XS62E008, XS62E009, XS62E010, XS62E011, XS62E012, XS62E013

Fonte: Elaborado pelo autor do trabalho.

O servidor foi configurado com RAID 10 para garantir desempenho e redundância dos dados. Portanto, foi utilizado os quatro HDs (2 HDs de 250 GB e 2 HDs de 500 GB) para realizar esta configuração. Isso permitiu utilizar o próprio servidor como *storage* para armazenar as máquinas virtuais. A Figura 1 apresenta o tamanho total do *storage* local (689.5 GB) após a configuração RAID 10 no servidor.

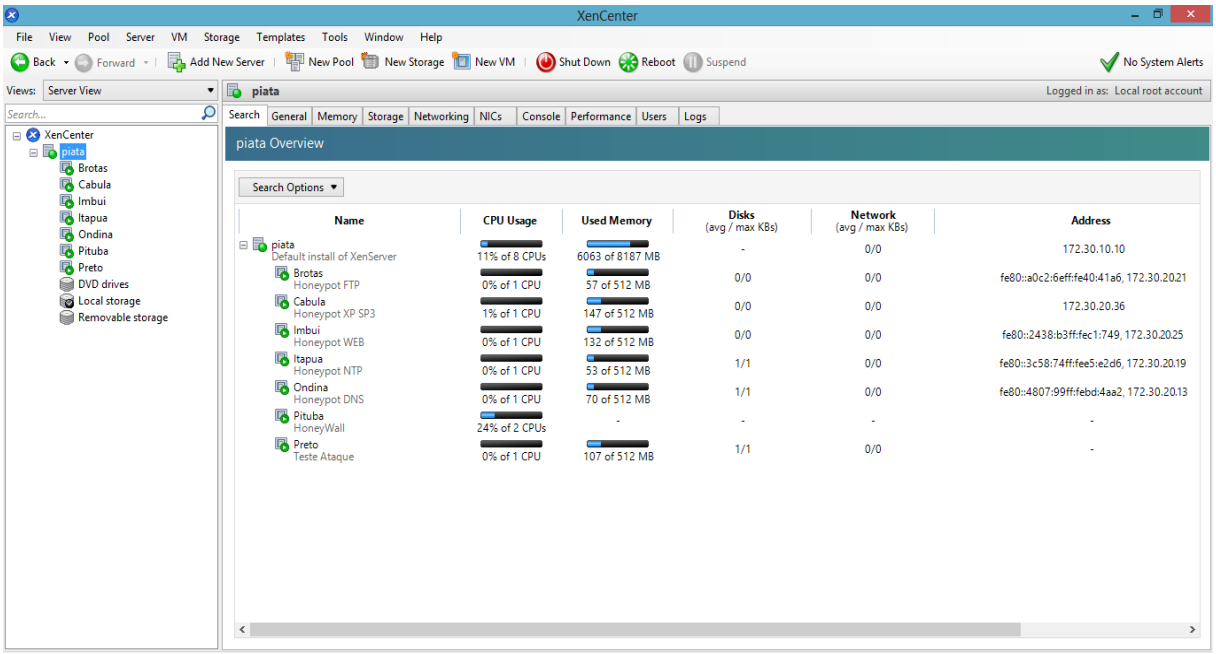
Figura 1 - Tamanho do *storage* local.



Fonte: Elaborado pelo autor do trabalho.

O *Hypervisor XenServer* foi implementado no servidor físico para criar a estrutura da *honeynet* virtual de autocontenção. A configuração e gerência das máquinas virtuais no *XenServer* foi feita através do *XenCenter 6.2*, por ser uma ferramenta de código aberto sob licença BSD (*Berkeley Software Distribution*). Abaixo segue tela inicial do *XenCenter*. A Figura 2 mostra os *honeypots* virtuais que foram criados pelo *XenCenter.*

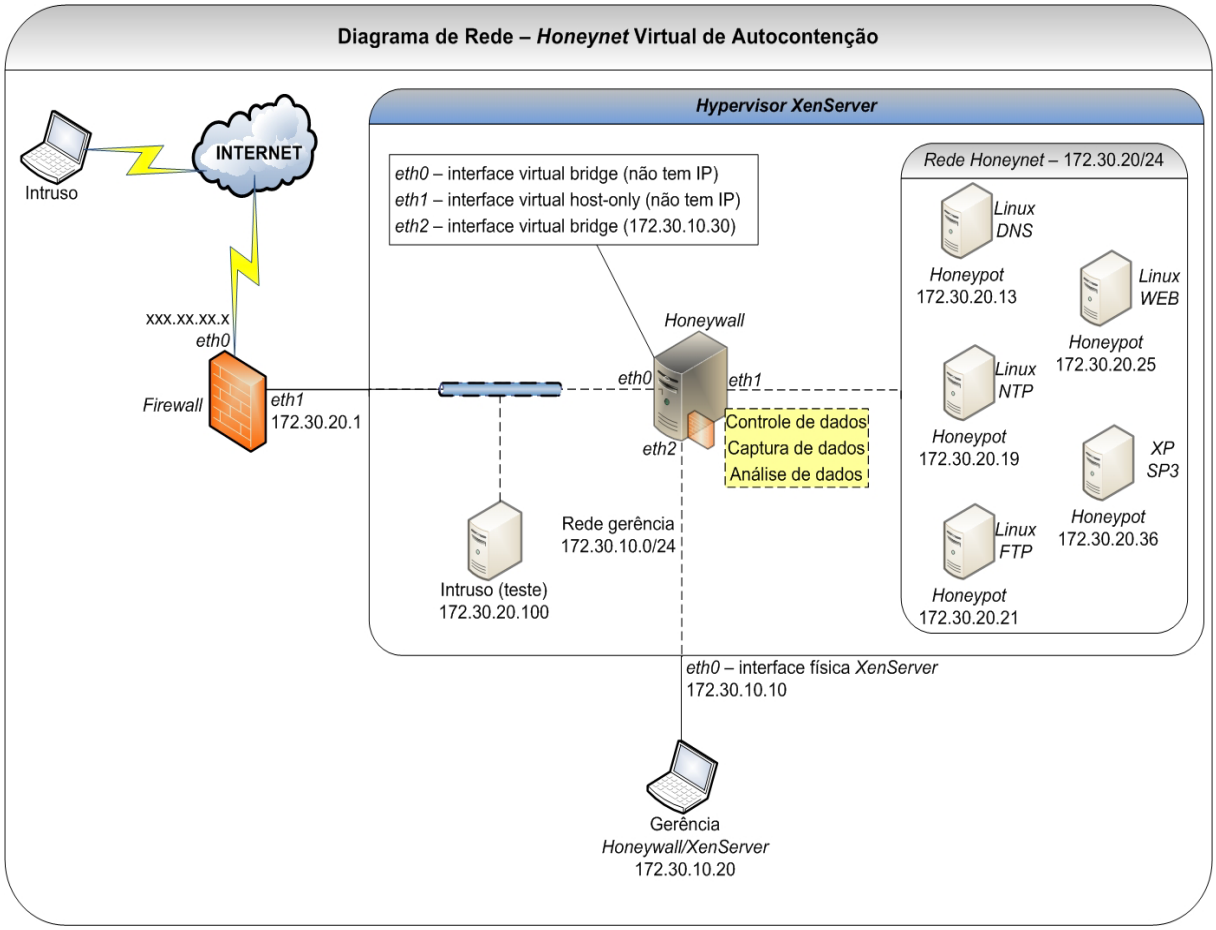
Figura 2 - Gerência do *XenServer* com *XenCenter*.



Fonte: Elaborado pelo autor do trabalho.

A arquitetura do ambiente *Honeynet* Virtual de Autocontenção dispõe de um servidor físico (*Dell PowerEdge 2950*), sete servidores virtuais e um *firewall (CISCO ASA* 5520). A Figura 3 apresenta a configuração do ambiente utilizado.

Figura 3 - Arquitetura do *Honeypot* de Alta Interatividade.



Fonte: Elaborado pelo autor do trabalho.

Vale ressaltar que todos os sistemas foram testados e rodaram com sucesso dentro do *XenServer.* As máquinas virtuais foram configuradas de acordo com o Quadro 4.

Quadro 4 - Máquinas virtuais e suas configurações.

Máquinas Virtuais	Configuração
Honeywall	Processador com 2 núcleos, 2 GB de memória RAM, 200 GB de HD e 3 interfaces virtuais de rede (*eth0, eth1 e eth2*)
	Versão *Roo-1.4* baseado no Sistema Operacional *CentOS release* 5 (final) com os serviços *snort, iptables* e *sebek-3.0.3-6*

Intruso (teste)	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Linux Kali 1.0.9* com ferramentas para testes de penetração e forense digital
Honeypot DNS	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Linux Debian Wheezy 7.2* com os serviços *bind9 e OpenSSH-6.0-p1*
Honeypot NTP	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Linux Debian Wheezy 7.2* com os serviços *ntpdate-4.2.6-p5 e OpenSSH-6.0-p1*
Honeypot FTP	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Linux Debian Wheezy 7.2* com os serviços *proftpd-1.3.4a e OpenSSH-6.0-p1*
Honeypot WEB	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Linux Debian Wheezy 7.2* com os serviços *apache2.2.22, php5-5.4.4-14, mysql-server-5.5.31 e OpenSSH-6.0-p1*
Honeypot XP	Processador com 1 núcleo, 512 MB de memória RAM, 8 GB de HD e uma interface virtual de rede (*eth0*)
	Sistema Operacional *Windows XP SP3* com instalação padrão. Não foram instalados outros serviços

Fonte: Elaborado pelo autor do trabalho.

Conforme a Figura 3, o ambiente possui três redes distintas: A *Internet*, uma rede não confiável (lugar de onde vêm os ataques); a rede *honeynet* 172.30.20.0/24, utilizada por um conjunto de *honeypots* para serem comprometidos; e a rede gerência 172.30.10.0/24, para gerência do *honeywall e XenServer*.

O *honeywall* foi configurado com três interfaces virtuais. A primeira interface virtual *eth0* se comunica com o *firewall CISCO ASA* (IP 172.30.20.1). A segunda interface virtual *eth1* é utilizada para se comunicar com a rede *honeynet*. As duas interfaces virtuais (*eth0* e *eth1*) estão configuradas como *bridge* (camada 2),

portanto não possuem endereço IP. Por fim, a terceira interface *eth2* (IP 172.30.10.30) é utilizada para gerência do *honeywall*.

Todos os *honeypots* estão configurados na rede *honeynet*. Esta rede foi configurada como *host-only* (rede virtual privada) para fazer a comunicação entre os *honeypots e* a interface virutal *eth1* do *honeywall*. No link da rede externa, existe ainda uma máquina virtual (IP 172.30.20.100) configurada para testar a configuração do ambiente *honeynet* virtual de autocontenção.

A rede de gerência é uma rede confiável que será usada para coletar e analisar remotamente os dados. Esta rede deverá ser usada ainda para administrar o *honeywall* (IP 172.30.10.30) e o *XenServer* (IP 172.30.10.10*). A* gerência será feita por um *host* dedicado exclusivo para esta finalidade. Um cabo *crossover* foi utilizado para fazer link entre o *host* de gerência e o servidor físico.

Todos os *honeypots* foram configurados com a instalação padrão do Linux *Debian Wheezy 7.2* e *Windows* XP SP3. Foram feitas instalações e configurações dos serviços de DNS, FTP, NTP e WEB. Vale ressaltar que não foi aplicado nenhum procedimento de *hardening* para manter os sistemas mais seguros.

2.1.1 Controle de dados no ambiente proposto

O controle de dados recebidos e enviados no ambiente *honeynet* virtual de autocontenção tem como finalidade controlar quais dados podem ir para qual destino. Este controle será feito pelo *firewall* (elemento que tem como finalidade separar as duas redes: internet e *honeynet*) e pelo *iptables* configurado no *honeywall*. Neste requisito foram definidas três regras para controlar o fluxo do tráfego:

- Qualquer individuo poderá realizar uma conexão da internet com o ambiente *honeynet* virtual de autocontenção. Isso permite que um intruso explore os *honeypots*;

- O *firewall* controlará conexões feitas da rede *honeynet* com a internet para evitar que os intrusos usem os *honeypots* para atacar outros sistemas de produção em redes confiáveis. Esta regra será replicada também no *firewall*

iptables implementado no *honeywall* para que haja uma redundância de controle de fluxo;

- A rede *honeynet* e a rede gerência não poderão se comunicar. Isso garante que os *honeypots* comprometidos não modifiquem ou destruam os dados coletados.

Ao mesmo tempo, o *script rc.firewall* (implementado no *iptables* do *honeywall*) é utilizado com a mesma finalidade, ou seja, prevenir ataques de dentro da rede *honeynet* para outros sistemas. O principal objetivo deste *script* é limitar o número de conexões (UDP, TCP, ICMP) que podem ser feitas para fora da rede *honeynet* através de uma escala de tempo (mensal ou diária).

Conforme apresentado acima, o controle de dados foi configurado em dois dispositivos de segurança para diminuir o impacto de falha durante o controle do tráfego de dados.

2.1.2 Captura de dados no ambiente proposto

A captura de dados tem como finalidade coletar todas as atividades que ocorrem dentro da rede *honeynet*. Quanto maior o número de camadas (métodos de captura), melhor será o sucesso do projeto. O *script rc.firewall* (implementado no *honeywall*) registrará todas as conexões de entrada e saída em */var/log/messages* para indicar o início de um ataque.

Neste projeto o *snort* (implementado no *honeywall*) foi configurado com regras atualizadas e utilizado para capturar todo o tráfego da interface virtual *eth1* do *honeywall*. Isto foi feito para registrar apenas o tráfego de entrada e saída da rede *honeynet*.

Finalmente, a última camada de captura de dados fica por conta da ferramenta *sebek.* Esta ferramenta tem como objetivo principal recriar com precisão ataques em um *honeypot* (The *Honeynet* Project, 2003). Em cada *honeypot* foi instalado e configurado o cliente *sebek* para ser executado no *kernel,* com a finalidade de capturar todas as atividades dos invasores (teclas pressionadas, *upload* de arquivos, senhas). Estas atividades serão enviadas por um canal seguro para o servidor *sebek* instalado no *honeywall* através do protocolo UDP (porta 1101).

2.2 Implantação do Honeywall

Em 2003 foi apresentado o primeiro *honeywall* chamado de *Igor*. O objetivo era implementar todas as ferramentas e requisitos para automatizar as *honeynets GenII* (Segunda Geração). Em 2004, uma equipe de arquitetos e desenvolvedores projetou e desenvolveu uma nova solução chamada atualmente de *Roo*[1]. Várias mudanças foram feitas para melhorar o sistema fazendo com que esta nova versão fosse considerada como tecnologia *GenIII* (Terceira Geração). Todo o projeto foi baseado na *honeynet GenIII* (VIECCO, 2005), desta forma, decidiu-se então optar pelo sistema *honeywall*. Ela contém as seguintes funcionalidades:

- Controle e captura de dados da *GenII*;
- Administração remota através de uma Interface Gráfica com Usuário (GUI);
- Suporte para *Sebek 3.x*[2];
- Base robusta do Sistema Operacional;
- Análise de dados integrada e atualização automatizada.

Atualmente o *honeywall* está na versão *roo-1.4*. Esta versão é baseada no *Linux CentOS release 5* (final). A finalidade deste dispositivo é funcionar como *bridge* camada 2 para capturar, analisar e controlar todo o tráfego de entrada e saída da rede *honeyne*t. Neste projeto o *CentOS* foi modificado para ter apenas os serviços necessários para o funcionamento da *honeywall.* O sistema deverá ter 3 interfaces virtuais de rede:

- Interface virtual *eth0*: funciona na camada 2 (*bridge* sem endereço IP) e interage com o *firewall*;
- Interface virtual *eth1:* funciona na camada 2 (*bridge* sem endereço IP) e interage com a rede *honeynet* e;
- Interface virtual *eth2:* única interface que possui endereço IP com o propósito de gerenciar e coletar dados do *honeywall*.

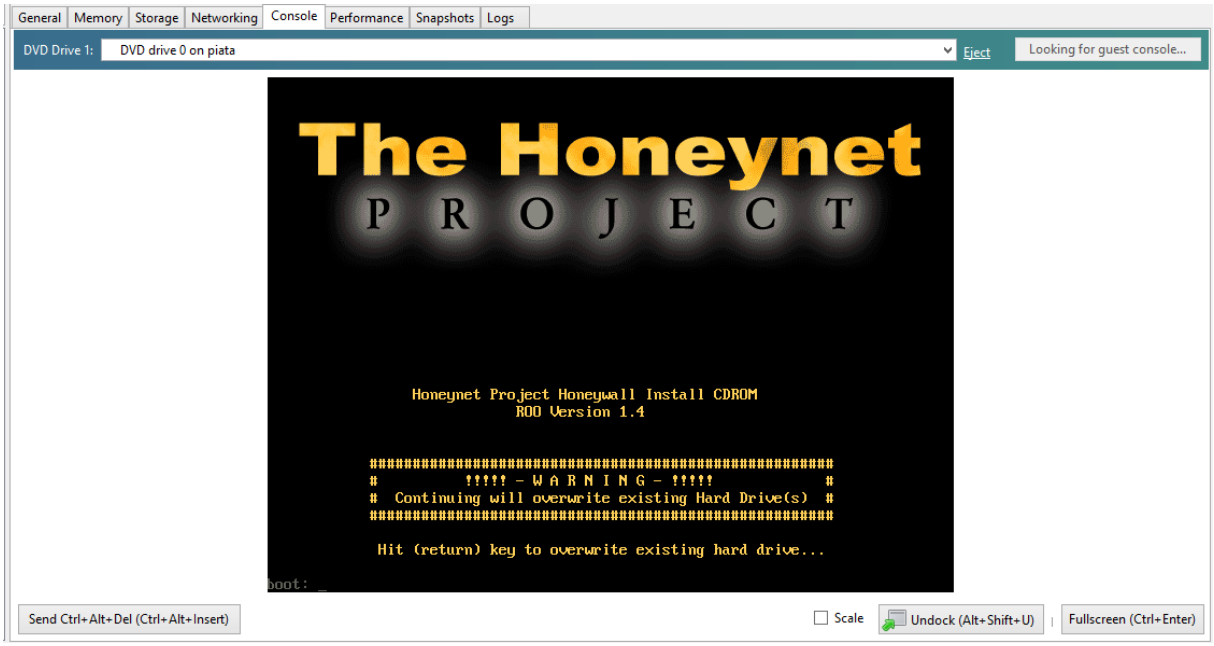
[1] http://old.honeynet.org/papers/cdrom/roo/
[2] http://www.honeynet.org/papers/sebek

O funcionamento deste dispositivo na camada 2 apresenta duas grandes vantagens: a primeira é que não há *hops* de roteamento e decremento do TTL (*Time To Live)* no cabeçalho IP; a segunda vantagem é a dificuldade por parte dos intrusos em detectar o ambiente.

Para exemplificar a instalação do *honeywall* serão mostrados neste trabalho apenas alguns tópicos. O objetivo deste processo é atribuir valores em todas as variáveis do *honeywall* para o seu perfeito funcionamento. Desta forma, a criação e configuração da máquina virtual no *XenServer* será feita através do *XenCenter*. A máquina deverá ser configurada de acordo com o Quadro 4.

O processo de instalação do *honeywall roo-1.4* (Figura 4) é totalmente automatizado.

Figura 4 - Tela de instalação do *honeywall roo-1.4.*



Fonte: Elaborado pelo autor do trabalho.

Depois da instalação o sistema reiniciará automaticamente e apresentará um *prompt* de comando. Neste ponto, deverá ser feita autenticação para iniciar o processo de configuração do *honeywall*. O sistema cria duas contas, *roo* (*uid:* 501) e *root* (uid 0), ambas com a mesma senha (*honey*), para configuração e administração. Em uma instalação padrão, a autenticação deverá ser feita com a conta *roo*.

Durante a primeira autenticação o *honeywall* apresentará um alerta informando que o sistema não está configurado. Desta forma, deve-se selecionar a opção *"honeywall configuration"* no menu principal. A configuração do *honeywall* poderá ser feita através de três métodos:

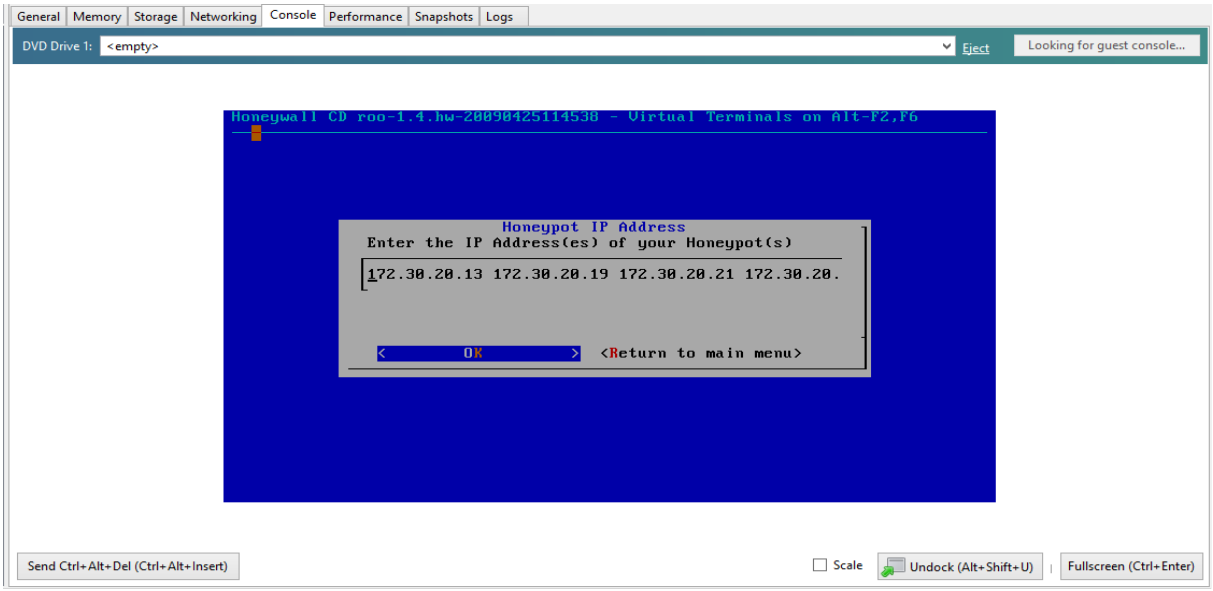
- *Floppy*: configura o *honeywall* (*honeywall.conf*) através de um disquete. Geralmente é utilizado para agilizar a implantação de um grande número de *honeywalls*;

- *Default:* restaura a configuração padrão do *honeywall.* Durante a primeira instalação, uma cópia do *honeywall.conf* será feita para o arquivo */etc/honeywall.conf.orig*;

- *Interview*: configura todos os parâmetros do *honeywall* através de perguntas. Neste método, o Órgão deverá ter todas as informações necessárias do projeto para realizar a configuração do *honeywall*.

O método "*Interview*" deverá ser utilizado durante a primeira instalação. Após isso será executado um *script* de instalação para configurar todas as opções do sistema. Deverão ser informados quais são os endereços IPs do *honeypots* (Figura 5), endereço de rede (172.30.20.0/24) e endereço de *broadcast* (172.30.20.255).

Figura 5 - Configuração dos endereços IPs dos *honeypots*.



Fonte: Elaborado pelo autor do trabalho.

2.2.1 Administração do honeywall

Uma terceira interface (*eth2*) deverá ser informada para administração remota do *honeywall*. Esta administração poderá ser feita remotamente (*ssh*) ou pelo *GUI Walleye*[3]. Nesta etapa (Figura 6), foram informados ainda os seguintes parâmetros: endereço IP (172.30.10.30) da interface de administração; a máscara de rede (255.255.255.0); o *gateway* padrão da interface de administração (172.30.10.1); o *hostname* (pituba); o domínio e o endereço IP do DNS.

Figura 6 - Parâmetros de configuração da interface de administração.

```
# Define a interface de administração
# [Argumento válido: eth* | br* | ppp* | none]
HwMANAGE_IFACE=eth2

# Endereço IP da interface de administração
# [argumento válido: Endereço IP]
HwMANAGE_IP=172.30.10.30

# Máscara de rede da interface de administração
# [Argumeto válido: Máscara de rede]
HwMANAGE_NETMASK=255.255.255.0

# Gateway da interface de administração
# [Argumento válido: endereço IP]
HwMANAGE_GATEWAY=172.30.10.1
```

Fonte: Elaborado pelo autor do trabalho.

O Serviço SSH foi configurado para utilizar a porta 22. Esta é a porta padrão utilizada por este serviço, porém ela pode ser alterada de acordo com as necessidades de cada Órgão. Uma conta (*admin*) foi criada para administrar o *honeywall*. As senhas das contas *root* e *roo* foram alteradas nesta etapa.

A interface *eth2* limita o número de portas e endereços IPs ou redes que poderão ser utilizados para realizar conexões de entrada para administração do *honeywall*. Para garantir mais segurança no sistema, foi configurado apenas um endereço IP (172.30.10.10) e duas portas: a porta 22 para gerenciamento remoto através do *ssh* e a porta 443 utilizada pelo GUI *Walleye* (administração e análise dos dados da *honeywall*). Nesta etapa, o *Iptables* foi configurado e executado através do *script rc.firewall* para restringir conexões de saída da interface de administração (*eth2*). A Figura 7 mostra as portas TCP e UDP utilizadas:

[3] http://old.honeynet.org/tools/cdrom/roo/manual/6-maintain.html

Figura 7 - Conexões de saída TCP e UDP da interface de administração.

```
# Conexões de entrada TCP da interface de admnistração.
# [Argumento válido: Espaço delimitado de portas TCP]
HwALLOWED_TCP_IN=22 443

# Restringe conexões de saída
# [Argumento válido: yes | no]
HwRESTRICT=yes

# Conexões de saída TCP da interface de administração
# [Argumento válido: Espaço delimitado de portas TCP]
HwALLOWED_TCP_OUT=22 25 43 80 443

# Conexões de saída UDP da interface de administração
# [Argumento válido: Espaço delimitado de portas UDP]
HwALLOWED_UDP_OUT=53 123
```

Fonte: Elaborado pelo autor do trabalho.

2.2.2 Limites de conexões de saída com Iptables

O sistema permite limitar as conexões de saída que um intruso pode realizar a partir de um *honeypot*. Esta medida evita que os *honeypots* comprometidos ataquem ou prejudiquem outros sistemas externos, evitando desta forma, escaneamento em massa e ataques de negação de serviço, atividade esta que requer muitas conexões de saída. O *script rc.firewall* foi utilizado novamente para definir quantas vezes um intruso pode iniciar uma conexão de saída TCP, UDP ou ICMP. A quantidade de limite de conexões irá depender do risco que o Órgão está disposto a assumir. O *honeywall* possui cinco escalas de limites de conexões: segundos, minutos, hora, dia, semana, mês e ano. A Figura 8 mostra a configuração feita no projeto:

Figura 8 - Limites de conexões de saída.

```
# Define escalas para limites de conexões de saída
# [Argumento valido: second, minute, hour, day, week, month, year]
HwSCALE=hour

# Número de conexões TCP por (HwSCALE)
# [Argumento valido: inteiro]
HwTCPRATE=15

# Número de conexões UDP por (HwSCALE)
# [Argumento valido: inteiro]
HwUDPRATE=15

# Número de conexões ICMP por (HwSCALE)
# [Argumento valido: inteiro]
HwICMPRATE=30

# Outras conexões IP por (HwSCALE)
# [Argumento valido: inteiro]
HwOTHERRATE=10
```

Fonte: Elaborado pelo autor do trabalho.

Esta configuração define o limite para 15 conexões TCP de saída por hora. Desta forma, quando um *honeypot* for comprometido o intruso poderá realizar apenas 15 conexões TCP de saída. Neste caso, o limite de conexões será redefinido após uma hora.

2.2.3 Snort-inline

O *snort-inline*[4] é uma versão modificada do *Snort* que tem como finalidade identificar e bloquear ataques conhecidos. Pode ser considerado um IPS (*Intrusion Prevention Systems*) que utiliza as assinaturas do IDS para tomar decisões. Este serviço utiliza regras (*drop, reject e replace*) do *iptables* para verificar se o pacote deverá ser negado, rejeitado ou substituído. Tem como vantagem reduzir o risco de um ataque de saída ser bem sucedido. Desta forma, o *snort-inline* bloqueia ou desativa qualquer ataque conhecido durante as 15 primeiras conexões TCP de saída. Durante este processo o *iptables* carrega o módulo *ip_queue* no *kernel* do sistema operacional fazendo com que os pacotes sejam roteados para o *snort_inline* com a finalidade de serem analisados. Vale ressaltar que apesar o *snort_inline* e o *iptables* trabalharem juntos, o *iptables* sempre contará as conexões de saída independente do *snort_inline* bloquear ou não os pacotes.

Neste projeto, o *iptables* foi configurado para enviar os pacotes para serem analisados pelo *snort_inline*. Foi utilizada a opção *drop* para fazer com que o *snort_inline* negasse pacotes de acordo com as regras configuradas.

2.2.4 Filtro de pacotes

O *honeywall* possui várias funções para realizar a filtragem de pacotes com a finalidade de realizar controle de dados, a saber:

- *Black list*: bloqueia endereços IPs e blocos CIDR[5] sem registros;
- *White list*: permite endereços IPs e blocos CIDR sem registros;
- *Fence list*: protege endereços IPs e blocos CIDR de qualquer *honeypot;*
- *Roach motel*: não permite todo o tráfego de saída do *honeypot*.

[4] http://snort-inline.sourceforge.net/oldhome.html
[5] https://tools.ietf.org/html/rfc4632

Durante a configuração foram criados quatro arquivos (Figura 9): */etc/blacklist*, */etc/whitelist* e */etc/fencelist*.

Figura 9 - Parâmetros de filtragem.

```
# Funções Blacklist, Whitelist, e Fencelist.
# [Argumento válido: string ]
HwFWBLACK=/etc/blacklist.txt

# [Argumento válido: string ]
HwFWWHITE=/etc/whitelist.txt

# [Argumento válido: string ]
HwFWFENCE=/etc/fencelist.txt

# Halilita o filtro Blacklist e Whitelist
# [Argumento válido: yes | no]
HwBWLIST_ENABLE=yes

# Habilita o filtro Fencelist
# [Argumento válido: yes | no]
HwFENCELIST ENABLE=no
```

Fonte: Elaborado pelo autor do trabalho.

2.2.5 DNS

A configuração deste item foi feita para que os *honeypots* não tivessem acesso ilimitado para outros DNS, ou seja, o *honeywall* irá bloquear conexões dos *honeypots* para outros servidores DNS externos. Neste projeto, um servidor DNS com IP (172.30.20.13) e domínio (*raeoo.com.br*) foi configurado dentro da infraestrutura da *honeynet* para esta finalidade. Portanto, todas as consultas externas serão feitas através do DNS (172.30.20.13) na porta 53 (TCP/UDP).

2.2.6 Alerta remoto

A finalidade desta opção é criar um procedimento automatizado para alertar o administrador através de *e-mails* caso haja ataques na *honeynet*. Este processo é fundamental para Órgãos Governamentais que não podem ter equipes trabalhando 24/7.

A ferramenta *Swatch* (Figura 10) foi configurada para realizar esta tarefa. Esta ferramenta monitora atividades do sistema, alertando os administradores da *honeynet* sobre possíveis ataques bem sucedidos.

Figura 10 - Monitoramento de atividades do sistema.

```
# Habilita o Swatch para enviar alertas através de e-mails
# [Argumento válido: yes | no]
HwALERT=yes

# Endereço de e-mail para enviar alertas
# [Argumento válido: qualquer endereço de e-mail]
HwALERT_EMAIL=raeoo2014@raeoo.com.br
```

Fonte: Elaborado pelo autor do trabalho.

Vale ressaltar que mesmo com esta ferramenta automatizada a *honeynet* deverá sempre que possível ser monitorada pelo administrador.

2.2.7 Sebek

Sebek[6] é uma ferramenta de captura de dados que tem como finalidade recriar com precisão os eventos de um *honeypot*. A intenção é verificar informações como: os passos de um ataque bem sucedido em um determinado *honeypot* e o que foi feito depois deste ataque. O *sebek* possui as seguintes funcionalidades:

- Capacidade de registrar pressionamentos de teclas em sessões criptografadas;
- Recuperar arquivos copiados com protocolo SCP e;
- Capturar senhas durante uma autenticação no sistema remoto.

Atualmente os intrusos estão utilizando criptografia para proteger seus canais de comunicação durante os ataques. Esta ferramenta captura os dados através de um módulo oculto que é executado no *kernel* do sistema operacional do *honeypot,* registrando qualquer atividade e/ou dados da função *read()*. Todos os pacotes gerados pelo cliente *sebek* instalado no *honeypot* são enviados através do protocolo UDP para a interface de rede do servidor no *honeywall*. Desta forma, mesmo com a utilização de criptografia será possível verificar quais são as intenções e motivações do intruso para realizar um ataque.

O *sebek* deve ser compilado de acordo com a versão do *kernel* do sistema operacional instalado em cada *honeypot*. Neste projeto, essa ferramenta foi compilada e instalada em apenas dois servidores: um servidor *Linux* com *kernel*

[6] http://www.honeynet.org/papers/sebek

2.6.26 e *Windows* XP. Abaixo segue alguns parâmetros de configuração do cliente *sebek*:

- *Destination_Port*: porta UDP de destino (1101) do servidor *sebek* configurado no *honeywall*;
- *Keystroke_Only*: este parâmetro deve ser definido como “0” para coletar todas as chamadas de sistema. O *honeywall* deve ter espaço suficiente em disco para salvar grande quantidade de informações, visto que este parâmetro recupera arquivos que são enviados e recebidos, por exemplo, através do protocolo SCP;
- *Module_Name*: define um nome para o módulo *sebek*. Este campo deve ser definido com nome diferente para não ser detectado pelo intruso. Na dúvida poderá ser deixado em branco.

No *honeywall* o servidor *sebek* foi configurado para detectar automaticamente os pacotes da interface. Conforme a Figura 11, informamos ainda: o endereço IP do *gateway*; o endereço IP de destino dos pacotes *sebek* e a porta UDP utilizada.

Figura 11 - Parâmetros de configuração do *Sebek*.

```
# Habilita o serviço Sebek para coletar eventos do honeypots
# [Argumento válido: yes | no]
HwSEBEK=yes

# Habilita a interface Web Walleye
#[Argumento válido: yes | no]
HwWALLEYE=yes

# Descartar pacotes Sebek ou permitir que sejam enviados para
# fora da honeynet
# [Argumento válido: ACCEPT | DROP]
HwSEBEK_FATE=ACCEPT

# Endereço IP de destino do servidor Sebek
# [Argumento válido: IP address]
HwSEBEK_DST_IP=172.30.20.1

# Porta UDP do servidor Sebek
# [Argumento válido: port]
HwSEBEK_DST_PORT=1101

# Habilitar registros Sebek no log do firewall do honeywall
# [Argumento válido: yes | no]
HwSEBEK_LOG=yes
```

Fonte: Elaborado pelo autor do trabalho.

Depois da configuração do *sebek* deverá ser informado o nome do *honeywall* para finalizar o processo de instalação.

2.3 Opções de configuração e manutenção

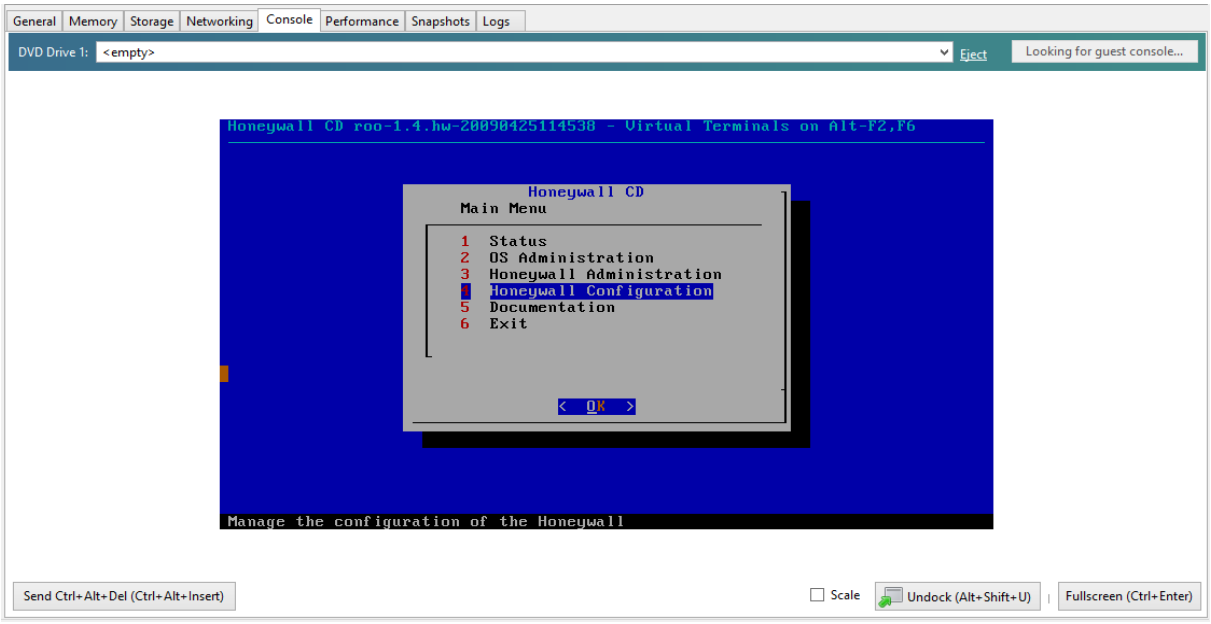
Após a instalação e configuração do ambiente, será necessário mantê-lo sempre atualizado e funcionando adequadamente. Todos os dados configurados no sistema (*interfaces*, IPs, portas, *e-mails*, etc) são armazenados como variáveis dentro do diretório de configuração */HW/conf/.* Por exemplo, o arquivo */HW/conf/HwMANAGE_IP* contém o endereço IP da interface de administração. Vale ressaltar que este parâmetro foi configurado em */etc/honeywall.conf*. Portanto, sempre que for feita uma configuração ou modificação no sistema os valores destas variáveis serão alterados.

A configuração do arquivo */etc/honeywall.conf* deverá ser feita através de três opções de configuração e manutenção oferecidas pelo *honeywall*.

2.3.1 Menu de Diálogo

É uma interface clássica de administração do *honeywall* (Figura 12). Esta versão implementa novos recursos adicionais sobre a versão antiga. Não é uma interface amigável, mas tem a vantagem de trabalhar localmente no sistema.

Figura 12 - Interface do Menu Diálogo.



Fonte: Elaborado pelo autor do trabalho.

Pode ter várias instâncias em execução e fica localizado em */usr/bin/Menu*. Este recurso foi utilizado anteriormente durante o processo de configuração do *honeywall*. Para abrir esta interface deve-se executar o comando:

 # menu

2.3.2 Utilitário HWCTL

A administração e manutenção do *honeywall* poderá ser feita também através do utilitário *HWCTL*[7]. Este é o único método que interage com o *honeywall* através de linha de comando. Permite ainda configurar parâmetros de sistema utilizados por vários serviços, realizar atualizações e backups do arquivo */etc/honeywall.conf*.

Esta ferramenta tem como vantagem realizar manutenção e administração localmente ou remotamente através do SSH. Abaixo segue alguns exemplos de comandos utilizados:

Imprime a saída do valor do parâmetro *HwSEBEK_DST_PORT*.

 # hwctl -n HwSEBEK_DST_PORT

Altera para 30 o valor do parâmetro *HwUDPRATE*.

 # hwctl HwSEBEK_DST_PORT = "30"

Verifica se as variáveis do sistema foram alteradas.

 # hwctl -r

Mostra as variáveis que estão sendo utilizadas atualmente.

 # hwctl -a

2.3.3 Interface Walleye[8]

Interface baseada em GUI que pode ser utilizada para acessar o servidor web do *honeywall* através de uma conexão SSL na interface de administração (THE *HONEYNET* PROJECT, 2005). Possui todas as funcionalidades do método Menu Diálogo e pode ser utilizada para administração e análise de todos dos dados do

[7] http://old.honeynet.org/tools/cdrom/roo/manual/6-maintain.html
[8] http://old.honeynet.org/tools/cdrom/roo/manual/6-maintain.html

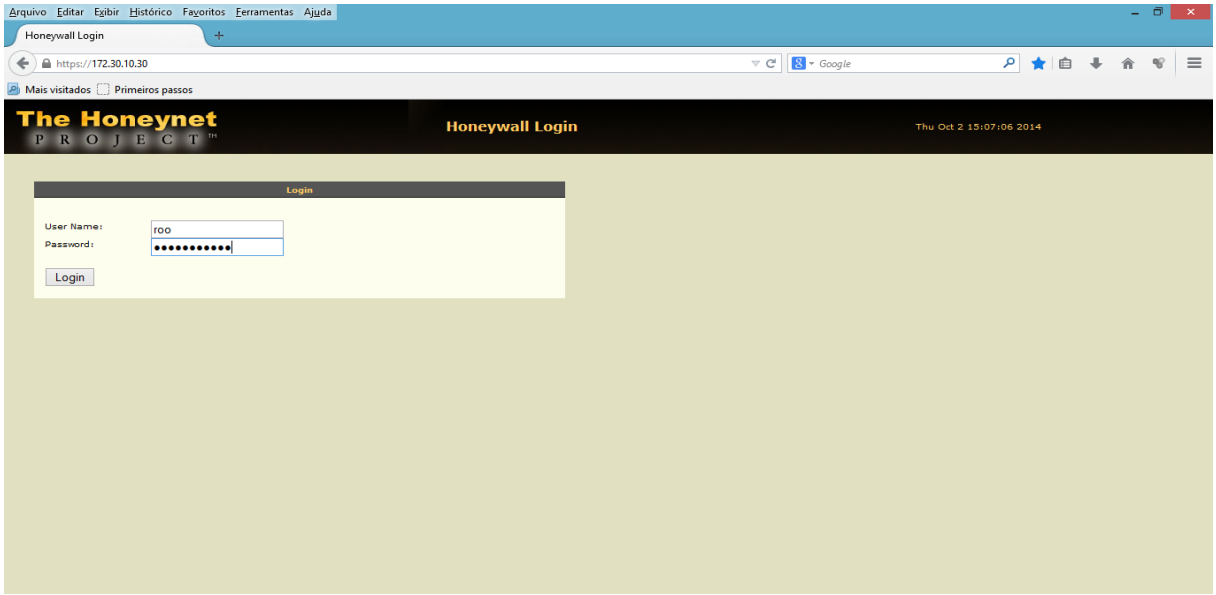
sistema. A sua principal vantagem é a facilidade de utilização (menu expansível, explicações detalhadas e controle de acesso para usuários). A desvantagem é que não pode ser utilizado localmente (precisa de uma terceira interface para realizar o acesso remoto).

Por padrão o servidor web do *honeywall* escuta na porta 443 (HTTPS com SSL), portanto será necessário configurar um certificado digital para garantir a autenticidade e a integridade do sistema. Durante o processo de configuração do *honeywall* foram definidos dois parâmetros de entrada na interface de administração:

- *HwALLOWED_TCP_IN*: define a porta de entrada da interface de administração. As portas definidas foram 22 e 443;
- *HwMANGE_IP*: define o endereço IP (172.30.10.30) da interface de administração.

Portanto, para acessar o sistema através do *walleye* o administrador deverá abrir o *browser* e informar a URL *https://172.30.10.30* (Figura 13).

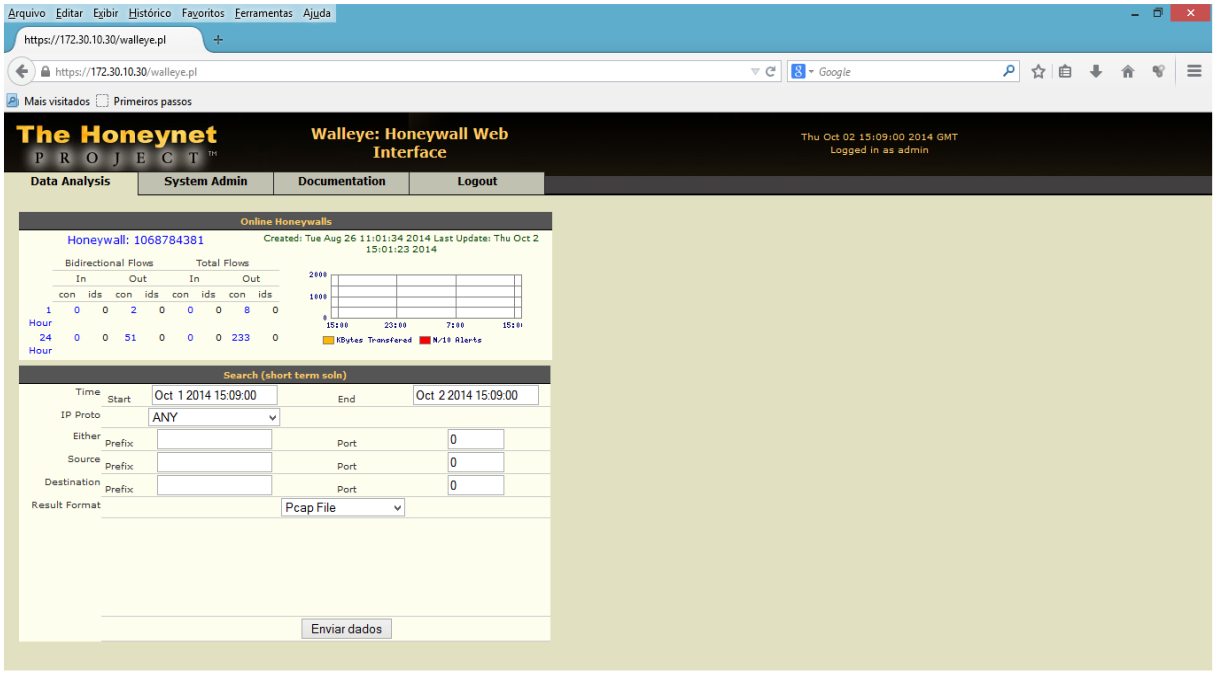
Figura 13 - Interface de autenticação do *Walleye.*



Fonte: Elaborado pelo autor do trabalho.

Assim como o *honeywall*, a conta *roo* e a senha *honey* deverão ser informados para ter acesso ao *walleye*. Depois da autenticação será exibida a interface *"Data Analysis"* (Figura 14) para análise dos dados.

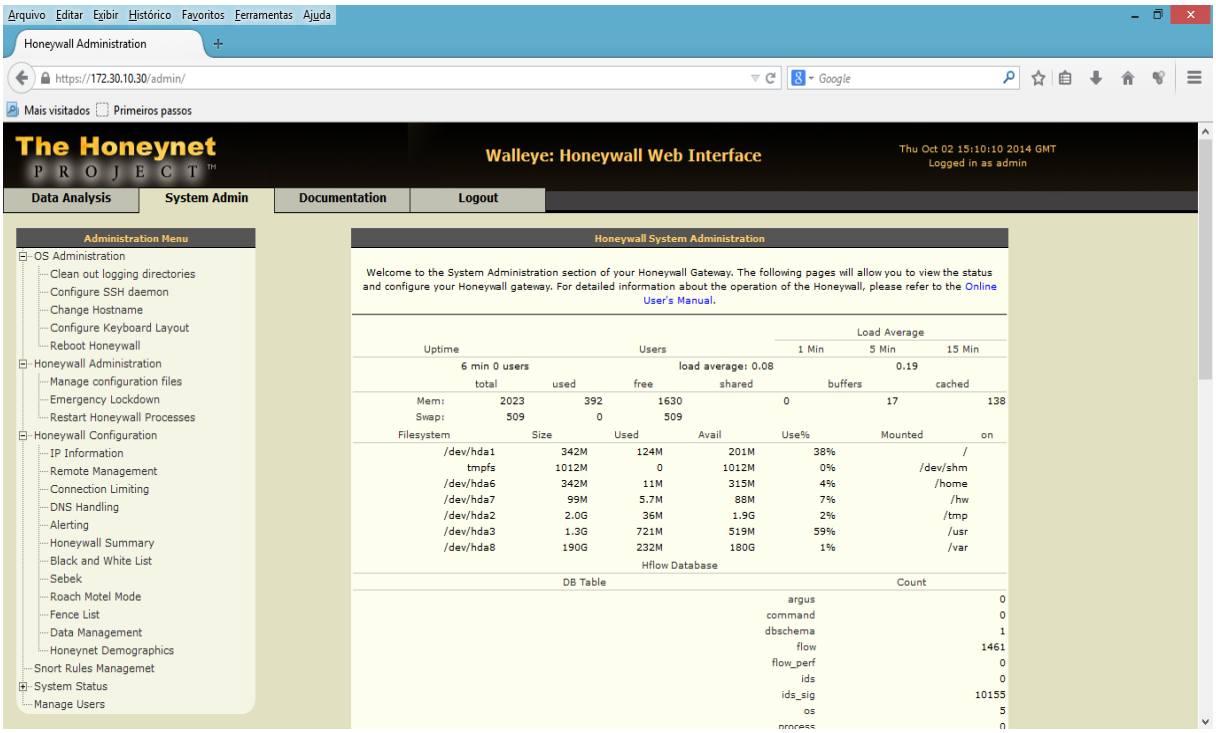
Figura 14 - Interface *Data Analysis* do *Walleye.*



Fonte: Elaborado pelo autor do trabalho.

A interface *"System Admin"* (Figura 15) deverá ser utilizada para administração do sistema. Esta opção é muito semelhante ao Menu Diálogo. A configuração do *honeywall* deverá ser feita nesta interface.

Figura 15 - Interface *System Admin* do *Walleye.*



Fonte: Elaborado pelo autor do trabalho.

3 VALIDAÇÃO DO AMBIENTE

A validação do ambiente foi feita através de vários testes na *honeynet* virtual de autocontenção para verificar se o sistema estava realmente funcionando. Desta forma, foi criada uma máquina virtual com o sistema operacional *Linux Kali* (IP 172.30.20.100) na rede externa para simular alguns ataques (Figura 3).

O primeiro teste foi realizado no controle de dados do ambiente para verificar se o *honeywall* estava coletando todos os dados de entrada e saída. Portanto, foi feito um *ping* da máquina virtual *Linux Kali* (172.30.20.100) para o *honeypot* DNS (172.30.20.13). Com base na configuração feita no conjunto de regras do *snort-inline*, foi possível capturar os pacotes ICMP (Figura 16). A interface de administração *walleye* foi utilizada para fazer a consulta.

Figura 16 - Captura do pacote ICMP.

Fonte: Elaborado pelo autor do trabalho.

Ainda neste teste, o *snort* mostrou algumas assinaturas relacionadas ao pacote ICMP que são de extrema importância para determinar o sistema operacional utilizado pelo intruso para realizar o ataque (Figura 17):

- **TTL:** observa-se que o campo TTL (IP utilizado pelo intruso) está configurado como 64. Com base nestas informações foi possível deduzir que este pacote foi enviado por um computador executando o *Linux*;

- **Tamanho do datagrama:** requisições de eco ICMP geradas através do utilitário *ping* terão 84 *bytes* de comprimento em sistemas operacionais UNIX e semelhantes ao UNIX;

- **Conteúdo da carga útil:** dados de uma requisição eco ICMP enviados através do utilitário *ping* em sistemas operacionais UNIX, ou semelhante ao UNIX serão compostos exclusivamente por números e símbolos.

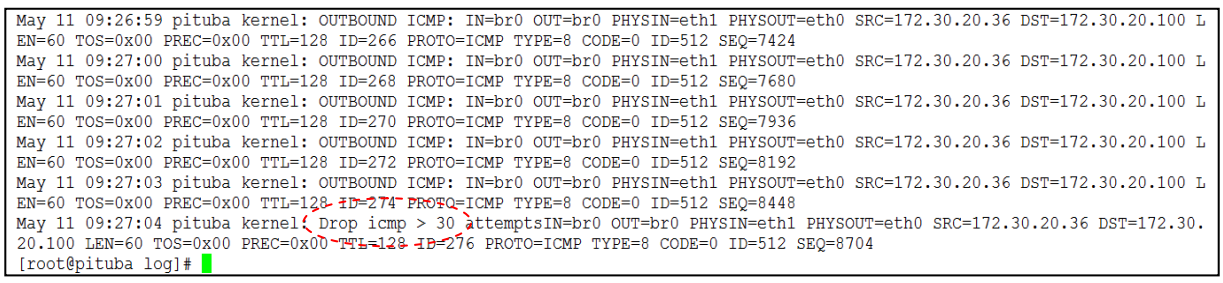
Figura 17 - Captura do pacote ICMP em formato binário.

Fonte: Elaborado pelo autor do trabalho.

O *snort* foi configurado também para converter quaisquer informações ASCII encontradas no *payload* do pacote para o arquivo *snort.log*. Este procedimento é fundamental para analisar rapidamente as seções de texto simples, tais como as seções de FTP, TELNET ou IRCs.

O segundo teste teve como propósito verificar os limites de conexões de saída para o protocolo ICMP. Para realizá-lo, foi executado um *ping* do *Honeypot* XP (172.30.20.36) para a máquina virtual de teste *Linux Kali* (172.30.20.100). Este procedimento indicará que o *Honeypot* XP foi comprometido e que um intruso está tentando realizar conexões para fora do ambiente, podendo ser um ataque. Quando o limite de conexão de saída ICMP (*HwICMPRATE=30*) configurado no *honeywall.conf* for atingido, o *script rc.firewall* executará uma entrada “DROP ICMP” e bloqueará durante uma hora estas conexões de saída (Figura 18). Todas as conexões serão registradas pelo *iptables* no *honeywall* em */var/log/iptables*.

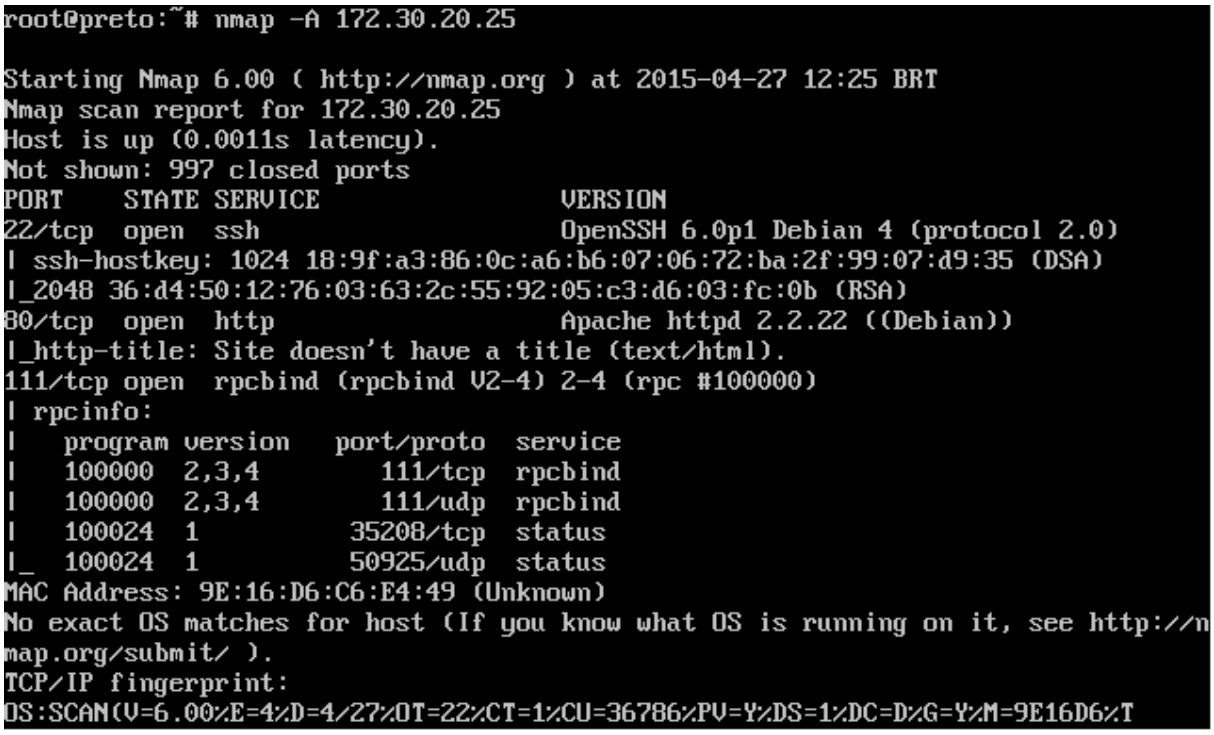
Figura 18 - Limites de conexões de saída do protocolo ICMP.



Fonte: Elaborado pelo autor do trabalho.

O terceiro e último teste (requisito captura de dados) teve como objetivo verificar se a base de assinaturas do *snort* no *honeywall* estava atualizada e configurada para detectar ataques. Primeiro foi feito um *portscan* com o *nmap* da máquina atacante (172.30.20.100) para o *Honeypot* WEB (172.30.20.25) com a finalidade de sondar e verificar quais eram os serviços que estavam sendo executados *no honeypot* (Figura 19).

Figura 19 - Ataque *portscan* com *nmap* no *Honeypot* WEB.



Fonte: Elaborado pelo autor do trabalho.

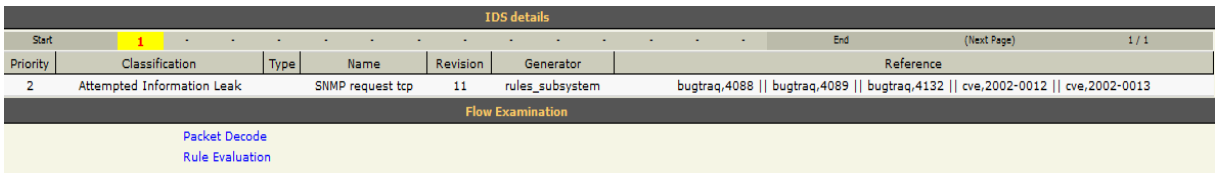
Este ataque mostrou que o *Honeypot* WEB estava executando os seguintes serviços: *OpenSSH 6.0p1* (porta 22/*tcp*), *Apache httpd 2.2.22* (porta 80/*tcp*) e *rpcbind v2-4* (porta 111/*tcp*). O intruso conseguiu ainda verificar o endereço *MAC* (9E:16:D6:C6:E4:49) e versão do sistema operacional (*Linux Debian*) utilizado.

Depois do ataque, o *walleye* foi utilizado para fazer a consulta. Conforme Figura 20, o *snort* conseguiu detectar dois ataques (em vermelho) *portscan* executado pelo *nmap* como uma tentativa de obter informações do *Honeypot* WEB através do protocolo SNMP direcionado para a porta 161/TCP. Neste caso, o administrador do *honeywall* poderá analisar com detalhes (Figura 21) os registros dos pacotes.

Figura 20 - Captura do ataque *portscan* no *Honeypot* WEB.

Fonte: Elaborado pelo autor do trabalho.

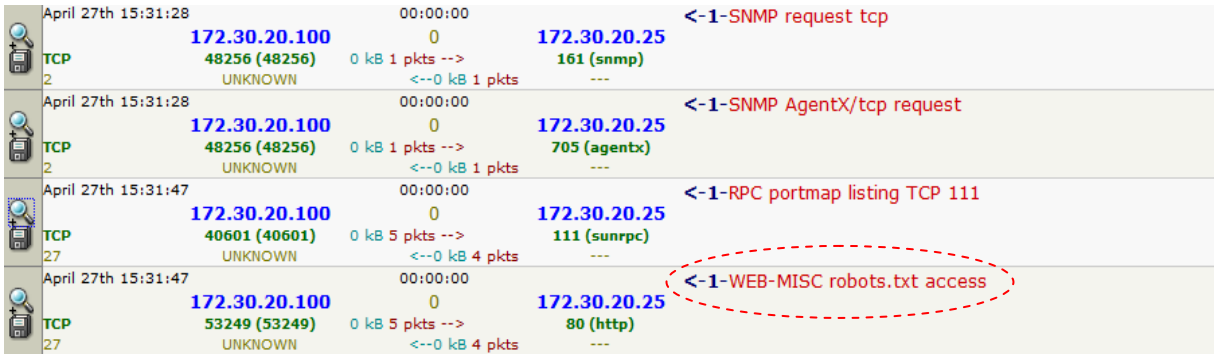
Figura 21 - Detalhes da captura do ataque *portscan* no *Honeypot* WEB.



Fonte: Elaborado pelo autor do trabalho.

Neste mesmo cenário foi feito outro *portscan* com o *nmap*, agora com a finalidade de verificar informações sobre o sistema operacional do *Honeypot* WEB. Desta vez o *snort* conseguiu detectar quatro tentativas de ataques (em vermelho) no *honeypot* (Figura 22).

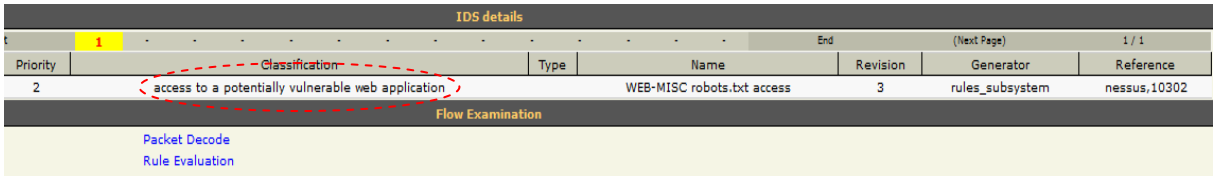
Figura 22 - Captura do segundo ataque *portscan* no *Honeypot* WEB.



Fonte: Elaborado pelo autor do trabalho.

Conforme a Figura 23, o ataque *WEB-MISC robots.txt Access*[9] detectado pelo *snort* informa que houve uma tentativa de coleta de informações a uma aplicação *web* potencialmente vulnerável.

Figura 23 - Tentativa de coleta a uma aplicação *web* vulnerável.



Fonte: Elaborado pelo autor do trabalho.

[9] http://manual.snort.org/node35.html

Este mesmo ataque pode ser visto ainda de uma forma mais detalhada pelo administrador através da *payload* do pacote em dois formatos diferentes. O primeiro formato é dado em hexadecimal (coluna da esquerda). O segundo formato é a conversão em ASCII (coluna da direita). A Figura 24 informa que foi executado um *portscan* através do *nmap*.

Figura 24 - Captura do ataque *portscan* em formato hexadecimal e ASCII.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/27-15:26:05.399945 26:0:78:DC:4E:F8 -> 9E:16:D6:C6:E4:49 type:0x800 len:0xE1
172.30.20.100:53225 -> 172.30.20.25:80 TCP TTL:64 TOS:0x0 ID:46481 IpLen:20 DgmLen:211 DF
***AP*** Seq: 0xA1C4F1ED  Ack: 0x500508FE  Win: 0x721  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1000550 66684
47 45 54 20 2F 72 6F 62 6F 74 73 2E 74 78 74 20  GET /robots.txt
48 54 54 50 2F 31 2E 31 0D 0A 43 6F 6E 6E 65 63  HTTP/1.1..Connec
74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 55 73 65  tion: close..Use
72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61  r-Agent: Mozilla
2F 35 2E 30 20 28 63 6F 6D 70 61 74 69 62 6C 65  /5.0 (compatible
3B 20 4E 6D 61 70 20 53 63 72 69 70 74 69 6E 67  ; Nmap Scripting
20 45 6E 67 69 6E 65 3B 20 68 74 74 70 3A 2F 2F   Engine; http://
6E 6D 61 70 2E 6F 72 67 2F 62 6F 6F 6B 2F 6E 73  nmap.org/book/ns
65 2E 68 74 6D 6C 29 0D 0A 48 6F 73 74 3A 20 31  e.html)..Host: 1
37 32 2E 33 30 2E 32 30 2E 32 35 0D 0A 0D 0A     72.30.20.25....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/27-15:26:05.419056 9E:16:D6:C6:E4:49 -> 26:0:78:DC:4E:F8 type:0x800 len:0x22B
172.30.20.25:80 -> 172.30.20.100:53225 TCP TTL:64 TOS:0x0 ID:3201 IpLen:20 DgmLen:541 DF
***AP*** Seq: 0x500508FE  Ack: 0xA1C4F28C  Win: 0x798  TcpLen: 32
TCP Options (3) => NOP NOP TS: 66687 1000550
48 54 54 50 2F 31 2E 31 20 34 30 34 20 4E 6F 74  HTTP/1.1 404 Not
20 46 6F 75 6E 64 0D 0A 44 61 74 65 3A 20 4D 6F   Found..Date: Mo
6E 2C 20 32 37 20 41 70 72 20 32 30 31 35 20 31  n, 27 Apr 2015 1
38 3A 33 34 3A 34 36 20 47 4D 54 0D 0A 53 65 72  8:34:46 GMT..Ser
76 65 72 3A 20 41 70 61 63 68 65 2F 32 2E 32 2E  ver: Apache/2.2.
32 32 20 28 44 65 62 69 61 6E 29 0D 0A 56 61 72  22 (Debian)..Var
79 3A 20 41 63 63 65 70 74 2D 45 6E 63 6F 64 69  y: Accept-Encodi
6E 67 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67  ng..Content-Leng
74 68 3A 20 32 38 36 0D 0A 43 6F 6E 6E 65 63 74  th: 286..Connect
69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 43 6F 6E 74  ion: close..Cont
65 6E 74 2D 54 79 70 65 3A 20 74 65 78 74 2F 68  ent-Type: text/h
74 6D 6C 3B 20 63 68 61 72 73 65 74 3D 69 73 6F  tml; charset=iso
2D 38 38 35 39 2D 31 0D 0A 0D 0A 3C 21 44 4F 43  -8859-1....<!DOC
54 59 50 45 20 48 54 4D 4C 20 50 55 42 4C 49 43  TYPE HTML PUBLIC
20 22 2D 2F 2F 49 45 54 46 2F 2F 44 54 44 20 48   "-//IETF//DTD H
54 4D 4C 20 32 2E 30 2F 2F 45 4E 22 3E 0A 3C 68  TML 2.0//EN">.<h
74 6D 6C 3E 3C 68 65 61 64 3E 0A 3C 74 69 74 6C  tml><head>.<titl
65 3E 34 30 34 20 4E 6F 74 20 46 6F 75 6E 64 3C  e>404 Not Found<
2F 74 69 74 6C 65 3E 0A 3C 2F 68 65 61 64 3E 3C  /title>.</head><
62 6F 64 79 3E 0A 3C 68 31 3E 4E 6F 74 20 46 6F  body>.<h1>Not Fo
75 6E 64 3C 2F 68 31 3E 0A 3C 70 3E 54 68 65 20  und</h1>.<p>The
72 65 71 75 65 73 74 65 64 20 55 52 4C 20 2F 72  requested URL /r
6F 62 6F 74 73 2E 74 78 74 20 77 61 73 20 6E 6F  obots.txt was no
74 20 66 6F 75 6E 64 20 6F 6E 20 74 68 69 73 20  t found on this
73 65 72 76 65 72 2E 3C 2F 70 3E 0A 3C 68 72 3E  server.</p>.<hr>
0A 3C 61 64 64 72 65 73 73 3E 41 70 61 63 68 65  .<address>Apache
2F 32 2E 32 2E 32 32 20 28 44 65 62 69 61 6E 29  /2.2.22 (Debian)
20 53 65 72 76 65 72 20 61 74 20 31 37 32 2E 33   Server at 172.3
30 2E 32 30 2E 32 35 20 50 6F 72 74 20 38 30 3C  0.20.25 Port 80<
2F 61 64 64 72 65 73 73 3E 0A 3C 2F 62 6F 64 79  /address>.</body
3E 3C 2F 68 74 6D 6C 3E 0A                       ></html>.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Fonte: Elaborado pelo autor do trabalho.

Este capítulo mostrou o funcionamento do ambiente *honeynet* virtual de autocontenção através de vários testes. Os requisitos de captura e coleta de dados foram utilizados e mostraram que a arquitetura proposta estava configurada corretamente.

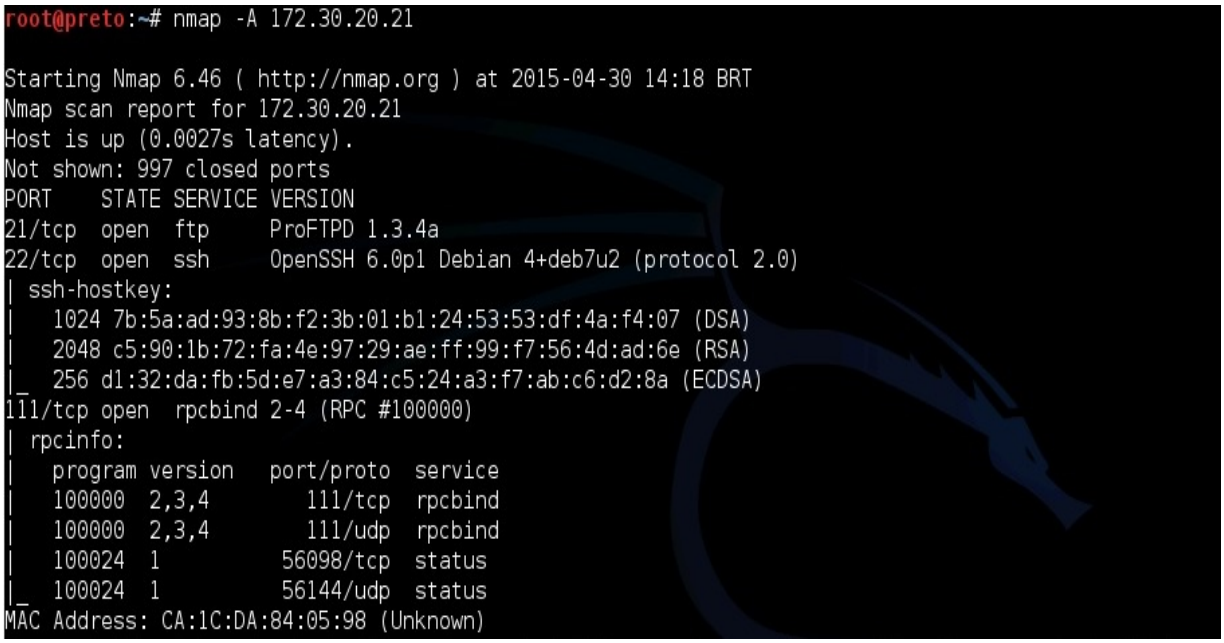
4 SIMULAÇÕES E RESULTADOS OBTIDOS

O objetivo deste estudo de caso é mostrar como os recursos apresentados neste trabalho podem ser utilizados por Órgãos Governamentais como fonte de pesquisa para coletar, analisar e estudar ataques e vulnerabilidades exploradas por invasores.

4.1 O ataque

Neste estudo de caso realizamos dois ataques de força bruta que geralmente são utilizados para comprometer severamente um sistema. O ataque foi feito da máquina virtual *Linux Kali* (172.30.20.100) para o *Honeypot* FTP (172.30.20.21). O serviço *proftpd-1.3.4a* do *honeypot* foi configurado para aceitar apenas conexões com autenticação. Para esta simulação o intruso será representado pela máquina virtual *Linux Kali* (172.30.20.100).

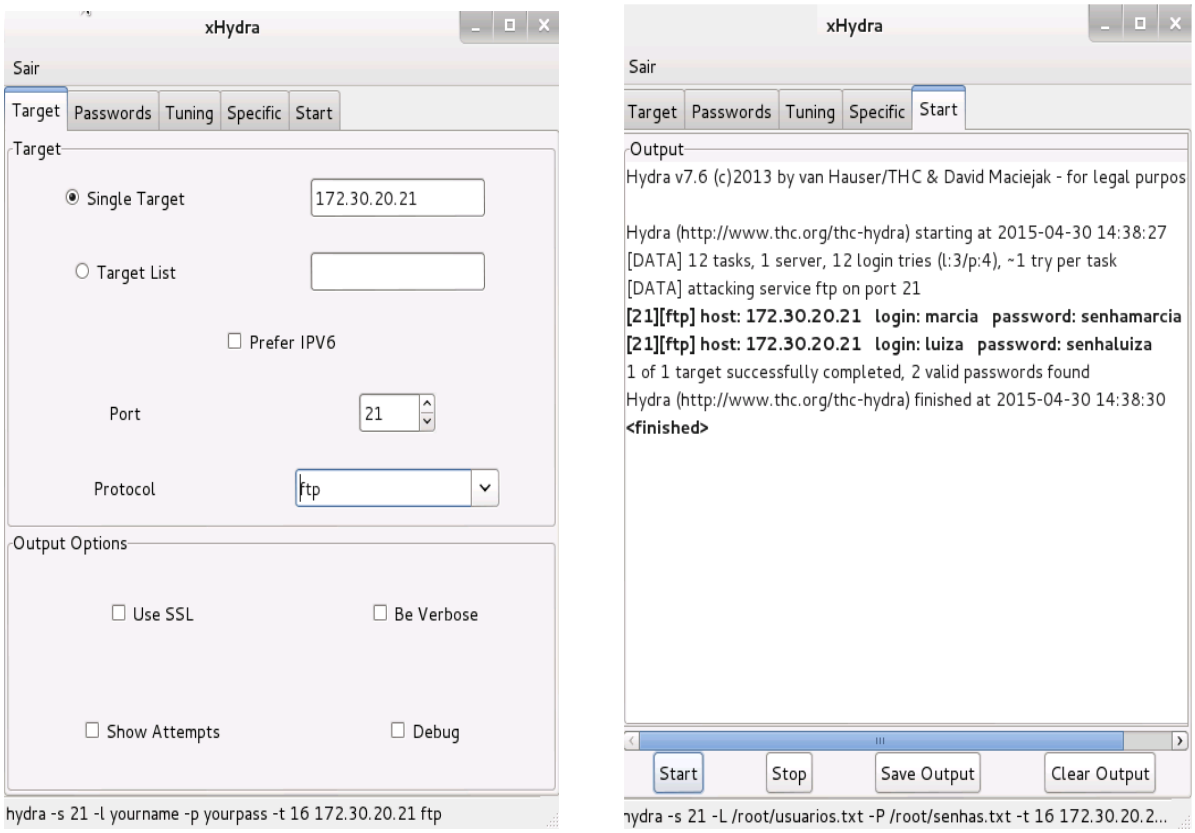
Figura 25 - Ataque *portscan* com o *nmap* no *Honeypot* FTP.



Fonte: Elaborado pelo autor do trabalho.

Primeiramente foi executado um *portscan* com o *nmap* pelo intruso (Figura 25). Após a varredura, verificou-se que vários serviços estavam com estado OPEN, inclusive o FTP e SSH.

Figura 26 - Ataque de força bruta com *xHydra*.



Fonte: Elaborado pelo autor do trabalho.

O primeiro ataque foi realizado no protocolo FTP através da ferramenta *xHydra*. Esta ferramenta faz escalação de privilégios através de quebra de senha online. A Figura 26 apresenta os usuários (luiza e marcia) e as senhas (senhaluiza e senhamarcia) que foram encontrados pelo *xHydra* durante o ataque.

Figura 27 - Ataque de força bruta com *medusa*.

```
root@preto:~# nc 172.30.20.21 22
SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
^C
root@preto:~# medusa -M ssh BANNER:SSH-2.0-OpenSSH_6.0p1 -h 172.30.20.21 -U /root/usuarios.txt -P senhas.txt | grep SUCCESS
ACCOUNT FOUND: [ssh] Host: 172.30.20.21 User: luiza Password: senhaluiza [SUCCESS]
ACCOUNT FOUND: [ssh] Host: 172.30.20.21 User: marcia Password: senhamarcia [SUCCESS]
root@preto:~#
```

Fonte: Elaborado pelo autor do trabalho.

Utilizamos as ferramentas *netcat* e *medusa* para realizar o segundo ataque no protocolo SSH. Primeiro foi executado o *netcat* para levantar o *banner* do serviço SSH. Depois executamos a ferramenta *medusa* para realizar o ataque de força bruta como mostrado na Figura 27. Após o ataque, a ferramenta retorna os usuários (luiza e marcia) e as senhas (senhaluiza e senhamarcia) que foram encontrados.

Figura 28 - Conexão FTP realizada pelo intruso.

Fonte: Elaborado pelo autor do trabalho.

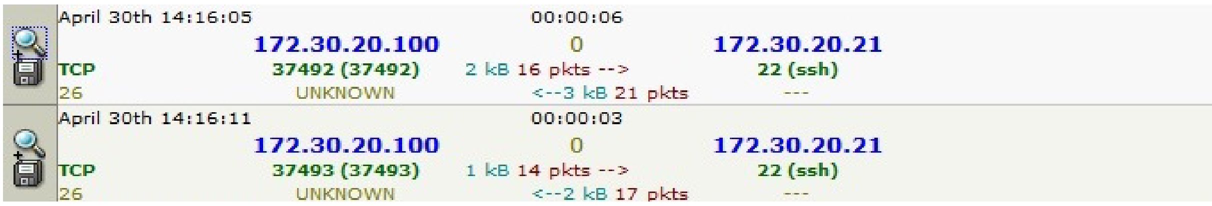
Em uma última etapa (Figura 28), o intruso realizou uma conexão FTP com o *honeypot* (172.30.20.21). Nesta conexão foram executados os seguintes comandos: *ls, cd Documentos, get seminario.txt, delete seminario.txt e exit.*

4.2 Analisando os ataques no ambiente

Normalmente os usuários maliciosos iniciam um ataque com a coleta de informações. Eles precisam explorar quais vulnerabilidades e *backdoors* existem nos

sistemas. Em 30 de abril, o *snort* detectou um ataque *portscan* no *Honeypot* FTP. Neste ataque, o intruso tentou explorar quais eram os serviços que estavam sendo executados no sistema.

Figura 29 - Captura do ataque de força bruta com *medusa.*



Fonte: Elaborado pelo autor do trabalho.

Figura 30 - Informações de cabeçalho do ataque de força bruta.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:16:05.155091 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x56
172.30.20.100:37492 -> 172.30.20.21:22 TCP TTL:64 TOS:0x0 ID:19439 IpLen:20 DgmLen:72 DF
***AP*** Seq: 0xA834A7ED  Ack: 0xC8892E29  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3730433 119684
```

Fonte: Elaborado pelo autor do trabalho.

No mesmo dia, o *snort* alertou que um dos *honeypots* havia sido comprometido. Neste caso, o ataque foi detectado e registrado conforme Figura 29. Este alerta do *snort* nos informou sobre uma tentativa de conexão SSH com um dos nossos *honeypots*. A seguir apresentamos as informações de cabeçalho do primeiro pacote (Figura 30):

- O pacote foi capturado em 30 de abril às 14h16min;
- O pacote foi enviado da porta 37492 da máquina 172.30.20.100 para porta 22 do *honeypot* 172.30.20.21;
- Esse pacote encapsula o protocolo TCP com TTL (*Time to Live*) 64, TOS (*Type of Service*) igual a zero, ID 19439 e comprimento de cabeçalho IP de '20 *bytes*;
- Número de seqüência 0xA834A7ED, número de confirmação *Ack* 0xC8892E29, *Win* (tamanho da janela) 0xE5 e *TcpLen* (Comprimento de cabeçalho TCP) 32 bytes;
- As opções TCP com dois NOPs e um TS (*timestamp*).

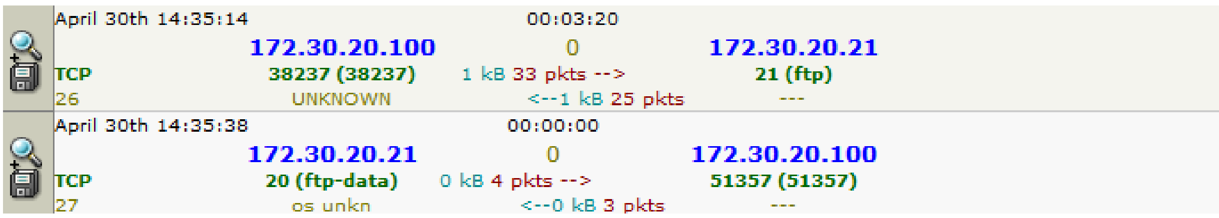
Figura 31 - Captura do ataque de força bruta em hexadecimal e ASCII.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:16:05.155091 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x56
172.30.20.100:37492 -> 172.30.20.21:22 TCP TTL:64 TOS:0x0 ID:19439 IpLen:20 DgmLen:72 DF
***AP*** Seq: 0xA834A7ED  Ack: 0xC8892E29  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3730433 119684
53 53 48 2D 32 2E 30 2D 4D 45 44 55 53 41 5F 31  SSH-2.0-MEDUSA_1
2E 30 0D 0A                                      .0..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:16:05.155468 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x42
172.30.20.21:22 -> 172.30.20.100:37492 TCP TTL:64 TOS:0x0 ID:64941 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xC8892E29  Ack: 0xA834A801  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 119685 3730433

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:16:05.164615 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x69
172.30.20.21:22 -> 172.30.20.100:37492 TCP TTL:64 TOS:0x0 ID:64942 IpLen:20 DgmLen:91 DF
***AP*** Seq: 0xC8892E29  Ack: 0xA834A801  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 119687 3730433
53 53 48 2D 32 2E 30 2D 4F 70 65 6E 53 53 48 5F  SSH-2.0-OpenSSH_
36 2E 30 70 31 20 44 65 62 69 61 6E 2D 34 2B 64  6.0p1 Debian-4+d
65 62 37 75 32 0D 0A                             eb7u2..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Fonte: Elaborado pelo autor do trabalho.

Para obter informações detalhadas sobre os pacotes enviados, analisamos os dados que foram detectados no *payload* do pacote. Conforme a Figura 31, confirmamos que o intruso realizou um ataque de força bruta através da ferramenta *Medusa* no serviço SSH.

Figura 32 - Captura do ataque no *Honeypot* FTP.



Fonte: Elaborado pelo autor do trabalho.

Ainda no mesmo dia, às 14h 35min recebemos outro alerta referente a uma conexão FTP no mesmo *honeypot* (Figura 32). Verificamos que a conexão foi feita da mesma máquina que realizou o ataque anterior. Este alerta do *snort* nos informou que a máquina 172.30.20.100:38237 estava tentando realizar uma conexão FTP com o *honeypot* (172.30.20.21:21).

Figura 33 - Captura do ataque em hexadecimal e ASCII no *Honeypot* FTP.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:14.334126 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x7F
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57841 IpLen:20 DgmLen:113 DF
***AP*** Seq: 0xC9C4D410  Ack: 0x9CE3029C  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 407010 4017585
32 32 30 20 50 72 6F 46 54 50 44 20 31 2E 33 2E  220 ProFTPD 1.3.
34 61 20 53 65 72 76 65 72 20 28 46 54 50 20 52  4a Server (FTP R
41 45 4F 4F 29 20 5B 3A 3A 66 66 66 66 3A 31 37  AEOO) [::ffff:17
32 2E 33 30 2E 32 30 2E 32 31 5D 0D 0A           2.30.20.21]..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:14.336021 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x42
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20269 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x9CE3029C  Ack: 0xC9C4D44D  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4017587 407010

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:31.025767 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x4E
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20270 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x9CE3029C  Ack: 0xC9C4D44D  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4021758 407010
55 53 45 52 20 6C 75 69 7A 61 0D 0A              USER luiza..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:31.027044 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x63
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57843 IpLen:20 DgmLen:85 DF
***AP*** Seq: 0xC9C4D44D  Ack: 0x9CE302A8  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 411184 4021758
33 33 31 20 50 61 73 73 77 6F 72 64 20 72 65 71  331 Password req
75 69 72 65 64 20 66 6F 72 20 6C 75 69 7A 61 0D  uired for luiza.
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:31.028818 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x42
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20271 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x9CE302A8  Ack: 0xC9C4D46E  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4021759 411184

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:34.355120 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x53
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20272 IpLen:20 DgmLen:69 DF
***AP*** Seq: 0x9CE302A8  Ack: 0xC9C4D46E  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4022591 411184
50 41 53 53 20 73 65 6E 68 61 6C 75 69 7A 61 0D  PASS senhaluiza.
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:35:34.369601 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x73
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57844 IpLen:20 DgmLen:101 DF
***AP*** Seq: 0xC9C4D46E  Ack: 0x9CE302B9  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 412020 4022591
32 33 30 2D 57 65 6C 63 6F 6D 65 2C 20 61 72 63  230-Welcome, arc
68 69 76 65 20 75 73 65 72 20 6C 75 69 7A 61 40  hive user luiza@
31 37 32 2E 33 30 2E 32 30 2E 31 30 30 20 21 0D  172.30.20.100 !.
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Fonte: Elaborado pelo autor do trabalho.

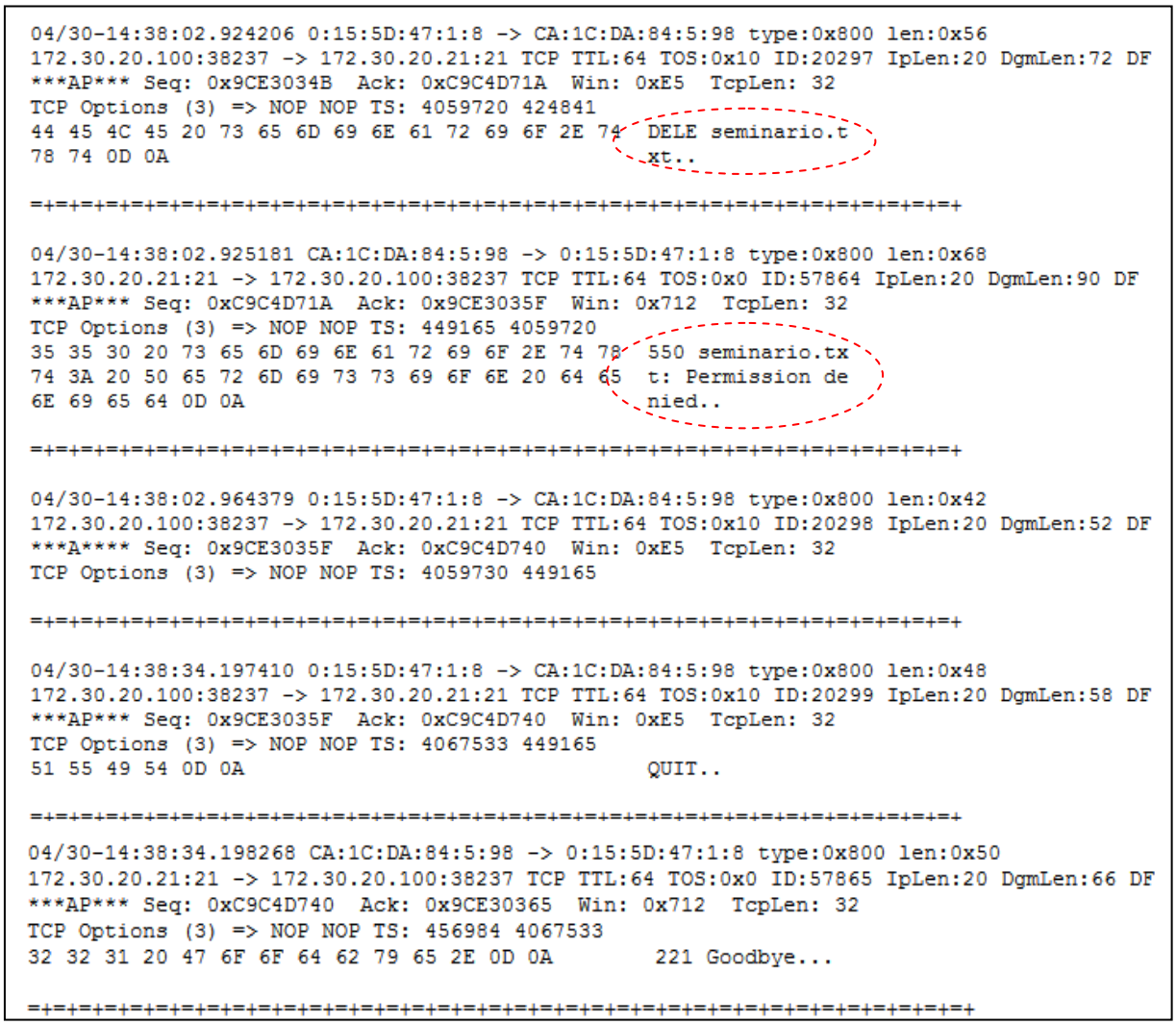
Analisando novamente o *payload* do pacote, verificamos que a conexão FTP foi feita em texto simples, os dados não foram criptografados. Isso significa que podemos decodificar os dados e capturar todas as teclas digitadas. Portanto, foi

possível verificar detalhes do ataque em formato hexadecimal e ASCII durante a conexão da máquina atacante com o *honeypot* FTP (Figura 33).

Dentre as informações obtidas pode-se verificar: endereço MAC (00:15:5D:47:01:08), endereço IP (172.30.20.100) e porta de origem (38237) da máquina que estava atacando; a porta de destino (21); a versão do servidor FTP (*ProFTPD 1.3.4a*), o usuário (*luiza*) e a senha (senhaluiza) utilizados pelo intruso para realizar a autenticação no servidor.

Figura 34 - Exploração de ataque no *Honeypot* FTP.

```
04/30-14:36:10.290380 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x52
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20287 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x9CE302E1  Ack: 0xC9C4D607  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4031573 413126
43 57 44 20 44 6F 63 75 6D 65 6E 74 6F 73 0D 0A  CWD Documentos..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:36:10.291440 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x5E
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57856 IpLen:20 DgmLen:80 DF
***AP*** Seq: 0xC9C4D607  Ack: 0x9CE302F1  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 421002 4031573
32 35 30 20 43 57 44 20 63 6F 6D 6D 61 6E 64 20  250 CWD command
73 75 63 63 65 73 73 66 75 6C 0D 0A              successful..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/30-14:36:25.640892 0:15:5D:47:1:8 -> CA:1C:DA:84:5:98 type:0x800 len:0x56
172.30.20.100:38237 -> 172.30.20.21:21 TCP TTL:64 TOS:0x10 ID:20295 IpLen:20 DgmLen:72 DF
***AP*** Seq: 0x9CE30337  Ack: 0xC9C4D6BD  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4035411 424840
52 45 54 52 20 73 65 6D 69 6E 61 72 69 6F 2E 74  RETR seminario.t
78 74 0D 0A                                      xt..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:36:25.643903 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x88
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57862 IpLen:20 DgmLen:122 DF
***AP*** Seq: 0xC9C4D703  Ack: 0x9CE3034B  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 424841 4035411
31 35 30 20 4F 70 65 6E 69 6E 67 20 42 49 4E 41  150 Opening BINA
52 59 20 6D 6F 64 65 20 64 61 74 61 20 63 6F 6E  RY mode data con
6E 65 63 74 69 6F 6E 20 66 6F 72 20 73 65 6D 69  nection for semi
6E 61 72 69 6F 2E 74 78 74 20 28 37 36 20 62 79  nario.txt (76 by
74 65 73 29 0D 0A                                tes)..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

04/30-14:36:25.667775 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x59
172.30.20.21:21 -> 172.30.20.100:38237 TCP TTL:64 TOS:0x0 ID:57863 IpLen:20 DgmLen:75 DF
***AP*** Seq: 0xC9C4D703  Ack: 0x9CE3034B  Win: 0x712  TcpLen: 32
TCP Options (3) => NOP NOP TS: 424847 4035411
32 32 36 20 54 72 61 6E 73 66 65 72 20 63 6F 6D  226 Transfer com
70 6C 65 74 65 0D 0A                             plete..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/30-14:36:25.649063 CA:1C:DA:84:5:98 -> 0:15:5D:47:1:8 type:0x800 len:0x8E
172.30.20.21:20 -> 172.30.20.100:33393 TCP TTL:64 TOS:0x8 ID:20157 IpLen:20 DgmLen:128 DF
***AP*** Seq: 0x18243534  Ack: 0x1E4A0B84  Win: 0x721  TcpLen: 32
TCP Options (3) => NOP NOP TS: 424842 4035412
50 61 6C 65 73 74 72 61 73 20 65 20 61 70 72 65  Palestras e apre
73 65 6E 74 61 63 6F 65 73 20 64 6F 20 73 65 6D  sentacoes do sem
69 6E 61 72 69 6F 0A 4C 69 73 74 61 20 64 6F 73  inario.Lista dos
20 50 61 6C 65 73 74 72 61 6E 74 65 73 3A 20 0A   Palestrantes: .
4D 61 72 63 69 61 0A 4C 75 69 7A 61              Marcia.Luiza

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```



Fonte: Elaborado pelo autor do trabalho.

Neste mesmo cenário, continuando a análise do ataque no *Honeypot* FTP, a Figura 34 mostra ainda que o intruso conseguiu entrar no diretório */Documentos* do servidor e fazer *upload* do arquivo *seminário.txt*. O intruso tentou também remover o arquivo *seminario.txt*, mas a captura do pacote mostra que ele não teve sucesso (*Permission denied)*.

CONCLUSÃO

Neste trabalho, um ambiente *honeynet* virtual de autocontenção foi desenvolvido como solução de pesquisa para analisar vulnerabilidades e acompanhar novas formas de atividades intrusivas. O ambiente proposto foi dividido em três fases buscando de certa forma uma otimização dos processos. Na primeira fase mostramos a arquitetura proposta, o *hypervisor* utilizado e como os *honeypots* foram configurados. Posteriormente, na segunda fase, foi feita a implantação e configuração do *honeywall*. Nesta fase, um *firewall* e um *script* implementado no *iptables* do *honeywall* foram utilizados para controlar o fluxo de dados. Na terceira fase foram implementadas três camadas para coletar as atividades dentro da *honeynet*: um *script* a fim de registrar conexões de entrada e saída; o *snort* configurado com regras atualizadas para capturar todo o tráfego; e a ferramenta *sebek,* utilizada para recriar com precisão os ataques sofridos nos *honeypots*.

Com o objetivo de validar o ambiente, vários testes foram feitos. O primeiro teste foi realizado no requisito controle de dados para verificar se o *honeywall* estava coletando todos os dados de entrada e saída. O propósito do segundo teste foi verificar os limites de conexões de saída do protocolo ICMP*.* O terceiro e último teste teve como finalidade verificar se a base de assinaturas do *snort* no *honeywall* estava configurada e atualizada para detectar os ataques. Por fim, fizemos um estudo de caso através da simulação de dois ataques de força bruta para mostrar o funcionamento do ambiente e obter os resultados.

Como trabalhos futuros, nesse contexto, sugerimos: testar outras formas de ataques; executar este ambiente por um período determinado de tempo e verificar os ataques reais oriundos da Internet para analisá-los e descrever o que aconteceu, o que foi aprendido; criar imagens de *honeypots* comprometidos para uma análise forense mais detalhada; extrair e analisar dados a partir de *dumps* de memória e incluir *honeypots* com sistemas operacionais utilizados por *smartphones.*

REFERÊNCIAS

CANSIAN, Adriano Mauro. **Desenvolvimento de um Sistema Adaptativo de Detecção de Intrusos em Redes de computadores.** Tese de Doutorado. Instituto de Fisica de Sao Carlos, São Carlos, SP, 1997.

CERT.BR. **Incidentes Reportados ao CERT.br – Janeiro a Dezembro de 2014**. Disponível em: <http://www.cert.br/stats/incidentes/2014-jan-dec/analise.html>. Acesso em: 12 abr. 2015.

CERT.BR. **Honeypots e Honeynets: Definições e Aplicações**. Disponível em: <http://www.cert.br/docs/whitepapers/honeypots-honeynets/>. Acesso em: 01 fev. 2014.

CHESWICK, B. **An evening with berferd in which a cracker is lured, endured, and studied**. In: In Proc. Winter USENIX Conference. [S.l.: s.n.], 1990. p. 163–174.

COHEN F. **A Note on the Role of Deception in Information Protection - 1998**. Disponível em: <http://all.net/journal/deception/deception.html/>. Acesso em: 29 nov. 2014.

FULLER, V., Li T. **RFC 4632 - Classes Inter-Domain Routing (CIDR):** The Internet Address Assignment and Aggregation Plan. Agosto de 2006. Disponível em: <https://tools.ietf.org/html/rfc4632/>. Acesso em: 21 jan. 2014.

KUROSE, James ; ROSS, Keith. **Redes de Computadores e a Internet:** uma nova abordagem. São Paulo: Addison Wesley, 2003.

HEADY, R.; LUGER, G.; MACCABE A. ; SERVILLA M. **The Architecture of a Network Level Intrusion Detection System, Technical Report**. Departament of Computer Science, University of New Mexico, USA, 1990.

HONEYNET.BR Team. **Honeynet.BR: Desenvolvimento e Implantação de um Sistema para Avaliação de Atividades Hostis na Internet Brasileira.** Instituto Nacional de Pesquisas Espaciais - IMPE.

OLIVEIRA, Vladimir Bezerra. **HoneypotLabsac:** Um Framework de Honeypot Virtual para o Android. Dissertação de Mestrado. Universidade Federal do Maranhão, São Luis, MA, 2012.

PROJECT, Honeynet. **Conheça seu inimigo - O Projeto Honeynet.** São Paulo: Pearson Education do Brasil, 2002.

ROCHA, Daniel Lira. **Utilização de um ambiente de honeynet no treinamento de redes neurais artificiais para detecção de intrusão.** Dissertação de Mestrado. Departamento de Engenharia Elétrica - Universidade de Brasília, Brasília, DF, 2006.

RUFINO, N. M. de O. **Segurança Nacional:** Técnicas e Ferramentas de Ataque e Defesa de Redes de Computadores. São Paulo: Novatec Editora Ltda, 2002.

SCARFONE, K. ; MELL, P. **Guide to Intrusion Detection and Prevention Systems (IDPS).** Recommendations of the National Institute of Standards and Technology, Gaithersburg, 2007.

SILVA, R. M. **Redes Neurais Aplicadas à Detecção de Intrusos em Redes TCP/IP**. Dissertação de Mestrado. Pontifica Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 2005.

SPITZNER, L. **Honeypots – Tracking Hackers**. Indianapolis, IN: AddisonWesley , 2002.

SNORT INLINE. **Snort-Inline.** Disponível em: <http://snort-inline.sourceforge.net/oldhome.html>. Acesso em: 01 dez. 2014.

STOLL, C. **Stalking the wily hacker**. Commun. ACM, ACM, New York, NY, USA, v. 31, n. 5, p. 484–497, 1988. ISSN 0001-0782.

SUNDARAM, Aurobindo. **An introduction to intrusion detection**. Crossroads: The ACM Student Magazine, 2 ed. Abril, 1996.

TANENBAUM, Andrew S. **Redes de computadores.** Rio de Janeiro: Editora Campos, 4ª ed, 2003.

THE HONEYNET PROJECT. **Honeynet Definitions, Requirements, and Standard.** Outubro de 2004. Disponível em: <http://old.honeynet.org/alliance/requirements.html/>. Acesso em: 16 fev. 2015.

THE HONEYNET PROJECT. **Know Your Enemy: Defining Virtual Honeynets.** Janeiro de 2003. Disponível em: <http:// http://old.honeynet.org/papers/virtual/>. Acesso em: 12 mar. 2015.

THE HONEYNET PROJECT. **Know Your Enemy: Sebek.** Novembro de 2003. Disponível em: <http://www.honeynet.org/papers/sebek/>. Acesso em: 30 out. 2014.

THE HONEYNET PROJECT. **Know Your Enemy: GenII Honeynets.** Maio de 2005. Disponível em: <http://old.honeynet.org/papers/gen2/>. Acesso em: 29 jan. 2014.

THE HONEYNET PROJECT. **Know Your Enemy: Honeywall CDROM Roo.** Agosto de 2005. Disponível em: <http://old.honeynet.org/papers/cdrom/roo/>. Acesso em: 29 dez. 2014.

THE HONEYNET PROJECT. **Roo CDROM User's Manual.** Maio de 2007. Disponível em: <http://old.honeynet.org/tools/cdrom/roo/manual/6-maintain.html/>. Acesso em: 29 dez. 2014.

THE HONEYNET PROJECT. **The Honeynet Projet.** Disponível em: <http://www.honeynet.org>. Acesso em: 25 dez. 2014.

VIECCO, E. B. C. **Towards a Third Generation Data Capture Architecture for Honeynets**. Advanced Network Management Lab, Indiana University, 2005.