DOI: 10.5769/C2015009 or http://dx.doi.org/10.5769/C2015009

Desenvolvimento de um Ambiente Honeynet Virtual para Aplicação Governamental

Gildásio Antonio de Oliveira Júnior, Rafael Timóteo de Sousa Júnior e Danilo Fernandes Tenório

Departamento de Engenharia Elétrica, Universidade de Brasília (UnB)
Campus Universitário Darcy Ribeiro – Asa Norte – 70910-900 – Brasília-DF – Brasil

jrgildasio@oi.com.br, desousa@unb.br, daniloftenorio@gmail.com

***Resumo*—Constitui uma prática comum aplicar técnicas de detecção de intrusos para detectar tráfego malicioso. Por conta do extenso número de vulnerabilidades em sistemas de informação e da grande criatividade dos invasores, torna-se cada vez mais necessário atualizar permanentemente as técnicas de detecção utilizadas. Portanto, é crucial operacionalizar um ambiente cibernético que propositadamente esteja preparado para ser invadido e comprometido, com a finalidade de permitir ao profissional de segurança analisar e verificar a evolução dos diversos tipos de ataques e vulnerabilidades exploradas por invasores. Este trabalho apresenta uma solução de segurança projetada especificamente para a pesquisa e a obtenção de informações de intrusos. Esse mesmo ambiente pode ser utilizado para a preservação de evidências de ataques para efeito forense.**

Palavras-Chave*—*Honeynets, Honeypots, Intrusion Detection Systems (IDS), Controle de Dados de Intrusões, Captura de Dados de Intrusões.

***Abstract*—It constitutes a common practice to apply intrusion detection techniques to detect malicious traffic. Because of the large number of vulnerabilities in information systems and the great creativity of the intruders, it becomes increasingly necessary to continuously update the employed detection techniques. Therefore, it is crucial to operationalize a cyber-environment that purposely is prepared to be invaded and compromised, in order to allow the security professional to analyze and verify the evolution of the several types of attacks as well as the vulnerabilities exploited by attackers. This paper presents a security solution projected specifically for research and for obtaining information from intruders. This same cyber-environment can be used for the preservation of attack evidences to forensic effect.**

Keywords*—*Honeynets, Honeypots, Intrusion Detection Systems (IDS), Intrusion Data Control, Intrusion Data Capture.

I. Introdução

Atualmente um dos principais problemas de segurança enfrentados no ciberespaço é a invasão de redes de computadores. Conforme estatísticas apresentadas em [1], no ano de 2014 houve 467.621 notificações de tentativas de fraudes e um aumento de 54% de ataques em servidores *Web*, em relação ao ano anterior. A rápida expansão do volume de informações acessadas através da Internet aumentou o interesse por novas formas de atividades intrusivas. Por conta desse crescimento, o ciberespaço se tornou um campo de guerra cibernética, uma guerra invisível e interminável. Desta forma, torna-se fundamental proteger os ativos de rede contra ameaças e garantir a integridade, a confidencialidade e a disponibilidade dos dados trafegados.

Verifica-se, entretanto, que grande parte das ferramentas e técnicas utilizadas para segurança da informação com a finalidade de combater ataques, tais como *firewall*, sistemas de criptografia, *hash*, assinatura digital, *antivírus*, dentre 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 indícios da ocorrência de ataques. Segundo [2], detecção de intrusão é o processo de monitoramento de eventos que ocorre em um sistema de computador ou rede para detectar sinais de possíveis incidentes. Tais tecnologias são classificadas de acordo com as técnicas utilizadas para investigar os citados indícios. Por exemplo, a arquitetura de um IDS (*Intrusion Detection System*) depende da localização do sistema e da forma como os dados são coletados, havendo duas categorias: baseado em host ou HIDS (*Host Intrusion Detection System*) e baseado em rede ou NIDS (*Network Intrusion Detection System*).

O grande problema da operacionalização de um IDS é que tal sistema limita-se a garantir certo nível de segurança, sobretudo no que se refere à detecção de ataques que sejam conhecidos previamente ou que se possa prever de alguma forma, ou seja, ataques que tenham alguma assinatura verificável previamente conhecida. Coloca-se por consequência a questão complementar de como coletar e analisar os dados de tráfego que incluam possíveis ataques ainda desconhecidos. Para tanto, prepara-se o IDS para detectar não somente ataques por assinatura, mas também indícios de ataques que correspondam a anomalias no tráfego ou na operação de sistemas. A detecção por anomalia, ainda que efetiva como método para bloquear os ataques, deixa em aberto a necessidade de analisar detalhadamente a anomalia para descrever o ataque e deste obter a respectiva assinatura.

A solução comum para a questão da análise dos ataques, tanto daqueles conhecidos quanto daqueles detectados por anomalia, consiste em construir um recurso de segurança chamado *honeynet* que, de acordo com [3], é uma ferramenta projetada especificamente para pesquisa e obtenção de informações de intrusos. Em uma *honeynet*, sistemas operacionais e serviços são emulados, de modo a parecer com sistemas reais funcionais, atraindo assim a atenção dos atacantes. Esta rede particular, deixada propositadamente ao

alcance dos intrusos, pode então ser usada para verificar e analisar os diversos tipos de ataques e as correspondentes vulnerabilidades exploradas nesses ataques.

Na gerência da segurança das redes, sistemas e aplicações governamentais, considera-se que o gestor deve aplicar toda medida a seu alcance, no sentido de proteger a informação e os sistemas, sendo então as medidas voltadas à captura e análise de novos ataques um importante pilar da gestão da segurança.

Outra característica de relevo das *honeynets* é a sua possível aplicação na área forense, visto que uma *honeynet* tem a possibilidade de capturar o *modus operandi* das operações de intrusão, bem como preservar evidências dos atos que indiquem ocorrência de crimes e também dos meios tecnológicos empregados em tais crimes.

Tais possibilidades das *honeynets* constituem uma justificativa do presente trabalho, pois se considera 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. Por outro lado, foram considerados os temas de pesquisa correlatos no domínio da informática forense e segurança da informação. Diante disso, a estratégia do trabalho foi dividida em três fases distintas: construção de um ambiente *honeynet* virtual de autocontenção, validação do ambiente e um estudo de caso.

Este artigo está organizado da seguinte forma. Na Seção 2 apresentamos os conceitos relacionados a *honeynets*. É detalhada na Seção 3 a arquitetura do ambiente de *honeynet* virtual de autocontenção proposto propositadamente para ser invadido e comprometido. Na Seção 4 validamos o ambiente, à luz dos requisitos de controle, captura e análise dos dados. Na Seção 5 simulamos dois ataques de força bruta, para demonstrar de forma detalhada o funcionamento do ambiente *honeynet*. Por fim, a Seção 6 apresenta as considerações finais e propostas de trabalhos futuros.

II. HONEYNETS

De acordo com [3], usando o conceito de *honeypot* como sendo um sistema, serviço ou aplicação emulada propositadamente para tornar-se alvo de um ataque, denomina-se *honeynet* um conjunto de *honeypots* de alta interatividade, integrados em uma solução projetada especificamente para ser invadida e comprometida. Diferentemente dos *honeypots* de baixa interatividade, que apenas emulam sistemas operacionais e serviços, os *honeypots* de alta interatividade fornecem sistemas operacionais e aplicações reais com as quais os intrusos possam interagir. Essa interatividade faz com que pesquisadores possam observar o comportamento de um intruso em um sistema real, a fim de descobrir novas técnicas de invasão, identificar novas vulnerabilidades e aprender como esses intrusos se comunicam.

Referências sobre mecanismos para monitoração e análise de atividades intrusivas surgiram em meados da década de 1980. Em agosto de 1986, um usuário malicioso atacou computadores dos laboratórios do LBL (Lawrence Berkeley Laboratory) para roubar dados. Com a finalidade de monitorar esses tipos de usuários, Clifford Stoll criou um projeto governamental para rastrear com detalhes os ataques até sua origem [4].

Em 1990, Bill Cheswick descreveu o acompanhamento de uma invasão no laboratório da AT&T, invasão esta em que foram exploradas falhas no serviço *Sendmail*, obtendo-se acesso ao *gateway* do laboratório. A finalidade desta experiência consistiu em localizar e aprender sobre as técnicas que foram utilizadas pelos intrusos. Uma técnica de mudança de diretório *root*, *chroot*, foi construída para observar todas as atividades que o intruso queria fazer [5].

A primeira solução de *honeypot* baseada em *software* foi chamada de DTK (*Deception Toolkit*) [6]. Ela foi desenvolvida em 1998 por Fred Cohen. Esta ferramenta tinha 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, sem fins lucrativos, de 30 profissionais 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 e descrevendo em detalhes as tecnologias *Honeynet* [3] e [7]. 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.

Já em [8] e [9], é apresentada uma comunidade de agentes de *software* que captura ataques e redireciona o tráfego para uma *honeynet*, de modo a permitir a análise dos detalhes dos ataques e a criação de proteções, em solução predecessora à deste trabalho.

A. Arquitetura de uma Honeynet

O sucesso de um projeto *honeynet* depende da correta definição da arquitetura, verificando-se que a construção e a manutenção de uma *honeynet* dependem de três requisitos críticos: controle de dados, captura de dados e coleta de dados [10]. O controle e a captura dos dados são os requisitos mais importantes da arquitetura. O terceiro requisito se aplica nas configurações que tenham vários *honeypots* em ambientes distribuídos.

1) Controle de Dados

Trata-se de um requisito muito crítico, cuja finalidade é a de 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 [3] e [10]. O tráfego de dados deve ser controlado de modo automático, para reduzir de forma rápida qualquer dano no sistema, e transparente, visando garantir que intrusos não percebam que suas atividades estão sendo controladas.

Ou seja, o controle de dados deve ser utilizado para separar a *honeynet* das outras redes, tais como: *Internet*, administrativa e produção. Para tanto, cada pacote deve ser controlado e inspecionado quando entra ou sai da *honeynet*. Geralmente, os ambientes permitem apenas que qualquer sistema inicie conexões com a *honeynet*, consentindo que intrusos sondem, identifiquem e explorem os sistemas vulneráveis dentro da *honeynet*.

2) Captura de Dados

Tratam-se das operações de captura de dados relativos a todas as atividades dos intrusos dentro da *honeynet*, incluindo as conexões de entrada, as atividades de rede e de sistema. Conforme [3] e [10], tais operações são tão críticas para o sucesso do projeto, que é melhor ter múltiplos métodos de captura de dados operantes.

Entretanto, nenhum dado capturado deve ser armazenado localmente nos *honeypots*, visto que dados armazenados localmente podem ser detectados por intrusos e utilizados para

comprometer o sistema. Além disso, estes dados podem ser modificados e destruídos. Em consequência, tais dados devem ser armazenados em outro local que seja seguro e confiável.

3) *Coleta de Dados*

A coleta de dados é um requisito aplicado em organizações que possuam várias *honeynets* em ambientes distribuídos. Neste caso, todos os dados capturados deverão ser transferidos a uma coletora central, para armazenamento e para poderem ser correlacionados e aumentar a efetividade das *honeynets* de captura.

Conforme [10], 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 dos sensores para uma coletora de forma segura, garantido sua confidencialidade, integridade e autenticidade;
- O anonimato dos dados deverá ser garantido; e
- Um serviço de sincronização de relógios, como o *Network Time Protocol* (NTP) deverá ser utilizado para garantir que os dados capturados na *honeynet* distribuída estejam devidamente sincronizados.

B. *Honeynets Reais*

As *honeynets* reais fornecem sistemas operacionais reais com quem os intrusos possam interagir. O objetivo dessa interação é aprender como os intrusos invadem os sistemas, como se comunicam e qual a finalidade do ataque [3], [10] e [11]. 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 [3], [7] e [10]. A Tabela I apresenta as principais vantagens e desvantagens das *honeynets* reais.

TABELA I. VANTAGENS E DESVANTAGENS DAS *HONEYNETS* REAIS

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

C. *Honeynets Virtuais*

Por suas características, as *honeynets* reais 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 [10] e [11]. A Tabela II apresenta as principais vantagens e desvantagens das *honeynets* virtuais.

TABELA II. VANTAGENS E DESVANTAGENS DAS *HONEYNETS* VIRTUAIS

Vantagens	Desvantagens
• Custo e espaço físico reduzidos • Facilidade de manutenção e administração	• Limitação e risco de comprometimento do software de virtualização (neste caso, o intruso poderá controlar toda a *honeynet*) • Risco de *fingerpriting* (os intrusos poderão detectar se os sistemas estão sendo executados em um software de virtualização)

As *honeynets* virtuais estão divididas ainda em duas categorias: autocontenção e híbridas. Na primeira, todos os dispositivos, incluindo os de captura e coleta de dados, geração de alertas e *honeypots*, estão implementados em um único computador. Já as híbridas representam uma combinação entre *honeynets* reais e virtuais. Nesta categoria, por exemplo, operações de captura, controle de dados e sistemas de *logs* são implementados em dispositivos físicos distintos, enquanto os *honeypots* são configurados em um único computador através de um *software* de virtualização.

III. PROPOSTA DE UM AMBIENTE HONEYNET VIRTUAL DE AUTOCONTENÇÃO

Esta seção descreve os aspectos relacionados ao desenvolvimento de um ambiente *honeynet* virtual de autocontenção para ser invadido e comprometido. A arquitetura proposta neste trabalho tem como objetivo detectar e capturar ataques novos e desconhecidos em Órgãos Governamentais. Neste ambiente, uma parte substancial do tráfego capturado terá origem ilícita ou maliciosa, ou seja, estará comprometida por códigos maliciosos.

Para atingir este objetivo, o desenvolvimento da arquitetura foi dividido em três fases distintas [3], [10]: arquitetura proposta e modelo de solução (fase 1), controle de dados (fase 2) e captura de dados (fase 3). Dessa forma, por um lado é possível atender os requisitos definidos na Seção anterior e, por outro lado, trata-se de uma solução em camadas e verifica-se que, quanto mais camadas de informações o ambiente tiver, mais fácil será analisar e aprender com os intrusos.

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 1 apresenta a estrutura do ambiente utilizado.

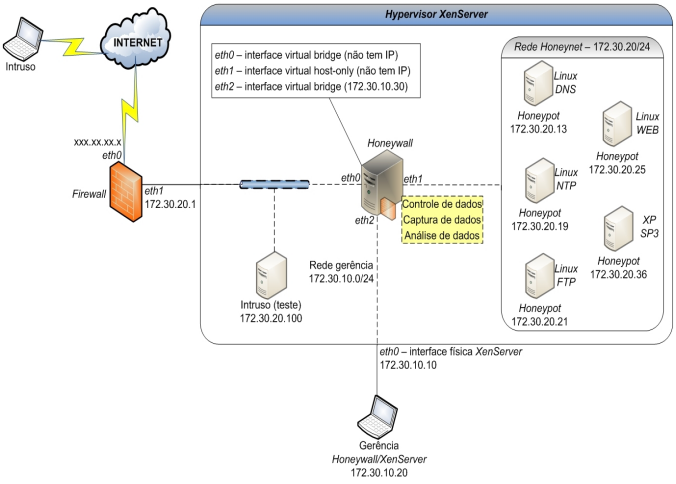

Fig. 1. Arquitetura proposta e modelo de solução.

A. *Fase 1: Arquitetura Proposta*

Nesta fase, a arquitetura foi desenvolvida com o *Hypervisor XenServer*, uma solução de virtualização de código aberto que possibilita gerenciar infraestruturas virtuais [12] [13]. Toda a estrutura do projeto foi feita em apenas um servidor físico. Recomenda-se a utilização de *hardware* dedicado para que as máquinas virtuais possam ser executadas corretamente. A Tabela III apresenta as características do servidor e plataforma de virtualização utilizada.

TABELA III. CARACTERÍSTICAS DO SERVIDOR

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

O servidor foi configurado com RAID 10, para garantir desempenho e redundância dos dados. Portanto, foram utilizados 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.

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*). A Figura 2 mostra os *honeypots* virtuais que foram criados pelo *XenCenter*. Vale ressaltar que todos os sistemas foram testados e funcionaram com sucesso dentro do *XenServer*. As máquinas virtuais foram configuradas de acordo com a Tabela IV.

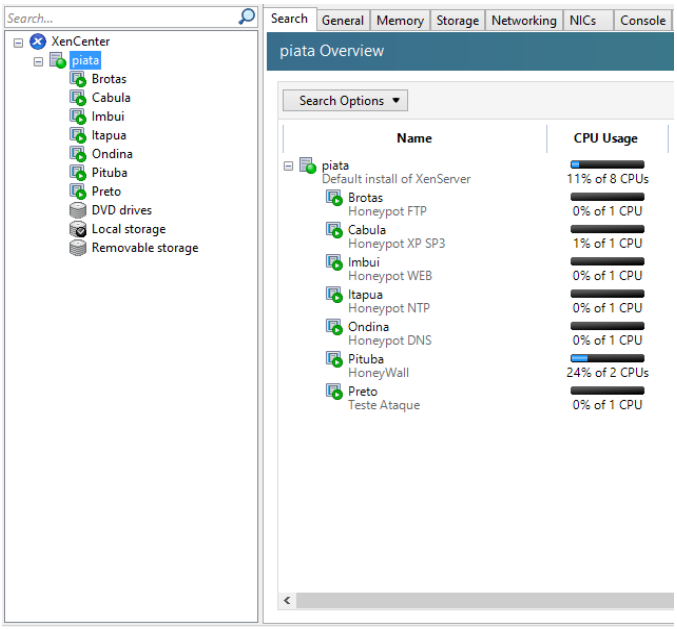

Fig. 2. Gerência do *XenServer* com *XenCenter*.

Conforme a Figura 1, 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, integrada por um conjunto de *honeypots* para serem comprometidos; e a rede de gerência 172.30.10.0/24, para gerência do *honeywall e XenServer*.

O *honeywall* [14], [15] foi configurado com três interfaces virtuais. A primeira interface virtual *eth0* comunica-se com o *firewall* (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 e coleta de dados do *honeywall*. Vale ressaltar que o funcionamento deste dispositivo na camada 2 apresenta duas grandes vantagens: a primeira é que não há *hops* de roteamento nem decremento do TTL (*Time To Live)* no cabeçalho IP; a segunda vantagem é a dificuldade por parte dos intrusos em detectar o ambiente.

TABELA IV. 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, ntpdate 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 *ntp-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, ntpdate 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, ntpdate 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

Todos os *honeypots* foram implementados na rede *honeynet*, que foi configurada como *host-only* (rede virtual privada) para fazer a comunicação entre os *honeypots e* a interface virtual *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 usada para coletar e analisar remotamente os dados. Esta rede deverá ser utilizada ainda para administrar o *honeywall* (IP 172.30.10.30) e o *XenServer* (IP 172.30.10.10*).* A gerência vabe a um *host* dedicado exclusivo para esta finalidade. Um cabo *crossover* é 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 processo de *hardening* para manter os sistemas mais seguros.

B. *Fase 2: Controle de Dados*

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

- Qualquer individuo poderá realizar uma conexão da internet para a *honeynet*. 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* comprometidos para atacar outros sistemas em produção. Esta regra será replicada também no *firewall iptables* implementado no *honeywall*, para que haja uma redundância de controle de fluxo;
- A rede *honeypot* e a rede gerência não poderão se comunicar. Isso garante que os *honeypots* comprometidos não modifiquem ou destruam os dados coletados.

Ao mesmo tempo, um *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* em uma escala de tempo (mensal ou diária).

Nesta fase, foram implementadas duas camadas de segurança para diminuir o impacto de falhas durante o controle do tráfego de dados.

C. *Fase 3: Captura de Dados*

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), mais se espera ter sucesso no 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.

Nesta arquitetura, o software detector de intrusões *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 todo 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 obter registros que permitam mais tarde recriar com precisão ataques em um *honeypot* [16]. 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).

IV. 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* [17] (IP 172.30.20.100) na rede externa para simular alguns ataques.

O primeiro teste foi realizado quanto ao requisito 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*, foi possível capturar os pacotes ICMP (Figura 3).

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

04/27-15:56:18.831361 26:0:78:DC:4E:F8 -> 2E:C:53:6B:2B:85 type:0x800 len:0x62
172.30.20.100 -> 172.30.20.13 ICMP TTL:64 TOS:0x0 ID:29641 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:6594   Seq:2  ECHO
25 5C 3E 55 00 00 00 00 1B 58 0B 00 00 00 00 00  %\>U.....X......
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

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

04/27-15:56:18.831979 2E:C:53:6B:2B:85 -> 26:0:78:DC:4E:F8 type:0x800 len:0x62
172.30.20.13 -> 172.30.20.100 ICMP TTL:64 TOS:0x0 ID:55708 IpLen:20 DgmLen:84
Type:0  Code:0  ID:6594  Seq:2  ECHO REPLY
25 5C 3E 55 00 00 00 00 1B 58 0B 00 00 00 00 00  %\>U.....X......
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

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

Fig. 3. Captura do pacote ICMP.

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:

- 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.

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.

```
May 11 09:26:59 pituba kernel: OUTBOUND ICMP: IN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36 DST=172
.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=266 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=7424
May 11 09:27:00 pituba kernel: OUTBOUND ICMP: IN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36 DST=172
.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=268 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=7680
May 11 09:27:01 pituba kernel: OUTBOUND ICMP: IN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36 DST=172
.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=270 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=7936
May 11 09:27:02 pituba kernel: OUTBOUND ICMP: IN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36 DST=172
.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=272 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=8192
May 11 09:27:03 pituba kernel: OUTBOUND ICMP: IN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36 DST=172
.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=274 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=8448
May 11 09:27:04 pituba kernel: Drop icmp > 30 attemptsIN=br0 OUT=br0 PHYSIN=eth1 PHYSOUT=eth0 SRC=172.30.20.36
 DST=172.30.20.100 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=276 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=8704
[root@pituba log]#
```

Fig. 4. Limites de conexões de saída do protocolo ICMP.

Quando o limite de conexão de saída ICMP (*HwICMPRATE=30*) for atingido, o *script rc.firewall* executará uma entrada “DROP ICMP” e bloqueará durante uma hora estas conexões de saída (Figura 4). Todas as conexões serão registradas pelo *iptables* no *honeywall* em */var/log/iptables*.

O terceiro teste (referente ao requisito de 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 5).

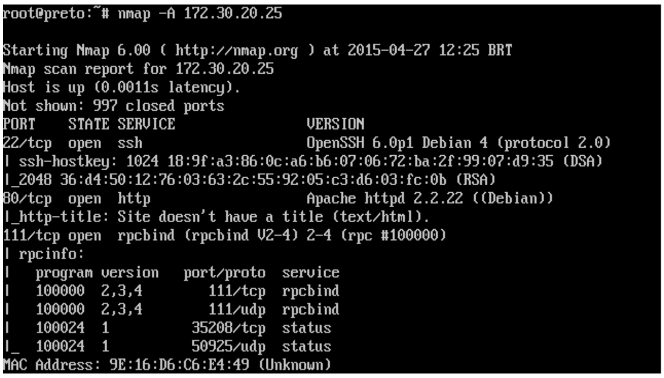

Fig. 5. Ataque *portscan* com *nmap* no *Honeypot* WEB.

Este ataque mostrou (vide Figura 5) 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.

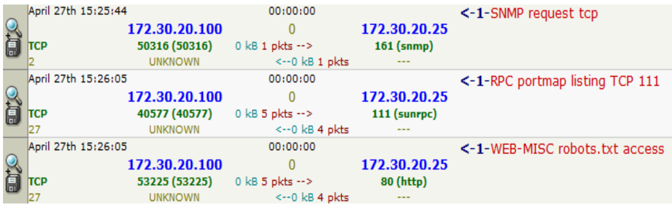

Fig. 6. Captura do ataque *portscan* feito pelo *nmap* no *Honeypot* WEB.

Conforme Figuras 5 e 6, o *snort* conseguiu detectar três ataques (em vermelho na Figura 6): *portscan* executado pelo *nmap* como uma tentativa de obter informações do *Honeypot* WEB através do protocolo SNMP direcionado para a porta 161/TCP. O ataque WEB-MISC *robots.txt Access* [18] detectado pelo *snort* informa que houve uma tentativa de coleta de informações a uma aplicação *web* potencialmente vulnerável.

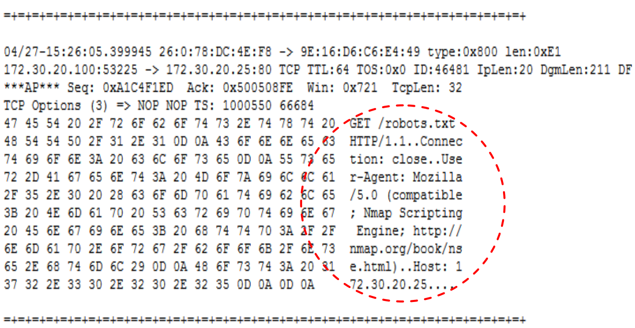

Fig. 7. Captura do ataque *portscan* em formato hexadecimal e ASCII.

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 7 informa que foi executado um *portscan* através do *nmap*.

V. Simulações e Resultados Obitidos

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.

A. *O Ataque*

Neste estudo de caso, foram realizados 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).

Primeiramente foi executado um *portscan* com o *nmap* pelo intruso (Figura 8). Após a varredura, verificou-se que vários serviços estavam com estado OPEN, inclusive o FTP e SSH.

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 9 apresenta os usuários (luiza e marcia) e as senhas (senhaluiza e senhamarcia) que foram encontrados pelo *xHydra* durante o ataque.

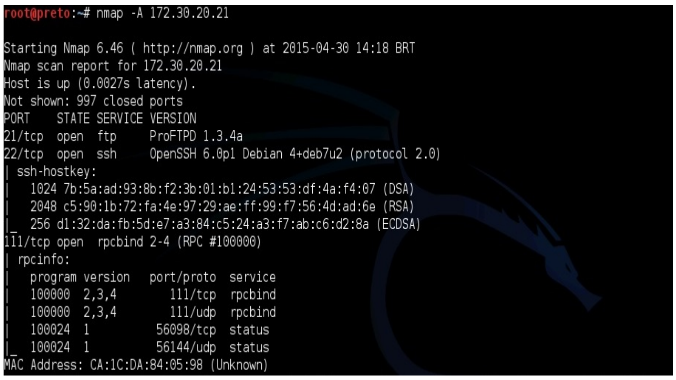

Fig. 8. Ataque *portscan* com *nmap* no *Honeypot* FTP.

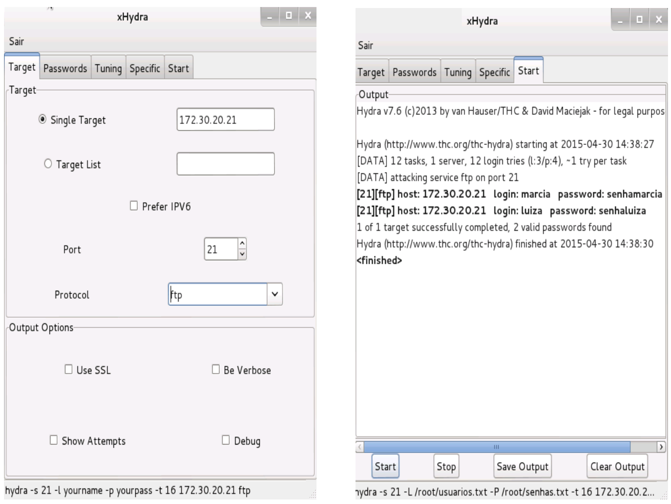

Fig. 9. Ataque de força bruta com *xHydra*.

Foram utilizadas 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 10. Após o ataque, a ferramenta retorna os usuários (luiza e marcia) e as senhas (senhaluiza e senhamarcia) que foram encontrados.

Em uma última etapa (Figura 11), 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.*

```
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_6.0pl -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:~#
```

Fig. 10. Ataque de força bruta com *medusa*.

```
root@preto:~# ftp 172.30.20.21
Connected to 172.30.20.21.
220 ProFTPD 1.3.4a Server (FTP RAEOO) [::ffff:172.30.20.21]
Name (172.30.20.21:root): luiza
331 Password required for luiza
Password:
230-Welcome, archive user luiza@172.30.20.100 !
230-
230-The local time is: Thu Apr 30 17:44:34 2015
230-
230-This is an experimental FTP server.  If you have any unusual problems,
230-please report them via e-mail to <root@brotas.raeoo.com.br>.
230-
230 User luiza logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
drwxr-xr-x   2 root     root         4096 Apr 27 14:28 Arquivos
drwxr-xr-x   2 root     root         4096 Apr 30 16:15 Documentos
drwxr-xr-x   2 root     root         4096 Apr 27 14:28 Palestras
-rw-r--r--   1 root     root          170 Sep  4  2014 welcome.msg
226 Transfer complete
ftp> cd Documentos
250 CWD command successful
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 root     root            76 Apr 30 16:15 seminario.txt
226 Transfer complete
ftp> get seminario.txt
local: seminario.txt remote: seminario.txt
200 PORT command successful
150 Opening BINARY mode data connection for seminario.txt (76 bytes)
226 Transfer complete
76 bytes received in 0.01 secs (14.7 kB/s)
ftp> delete se
seminario.txt  senhas.txt
ftp> delete seminario.txt
550 seminario.txt: Permission denied
ftp> exit
221 Goodbye.
root@preto:~#
```

Fig. 11. Conexão FTP realizada pelo intruso.

B. 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.

No mesmo dia, o *snort* alertou que um dos *honeypots* havia sido comprometido. Neste caso, o ataque foi detectado e registrado conforme Figura 12. 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 13):

- 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*).

Para obter informações detalhadas sobre os pacotes enviados, analisamos os dados que foram detectados no *payload* do pacote. Conforme a Figura 14, confirmamos que o intruso realizou um ataque de força bruta através da ferramenta *Medusa* no serviço SSH.

Ainda no mesmo dia, às 14h 35min recebemos outro alerta referente a uma conexão FTP no mesmo *honeypot* (Figura 15). 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).

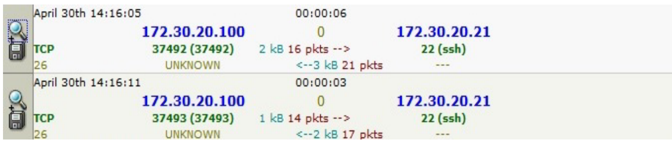

Fig. 12. Captura do ataque de força bruta com *medusa*.

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

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
```

Fig. 13. 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
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..

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

04/30-14:16:05.164615 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..

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

Fig. 14. Captura do ataque de força bruta em hexadecimal e ASCII.

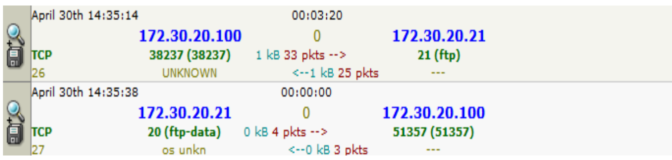

Fig. 15. Captura do ataque no *Honeypot* FTP.

Analisando novamente o *payload* do pacote, verificamos que a conexão FTP foi feita em texto simples, ou seja, 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 16).

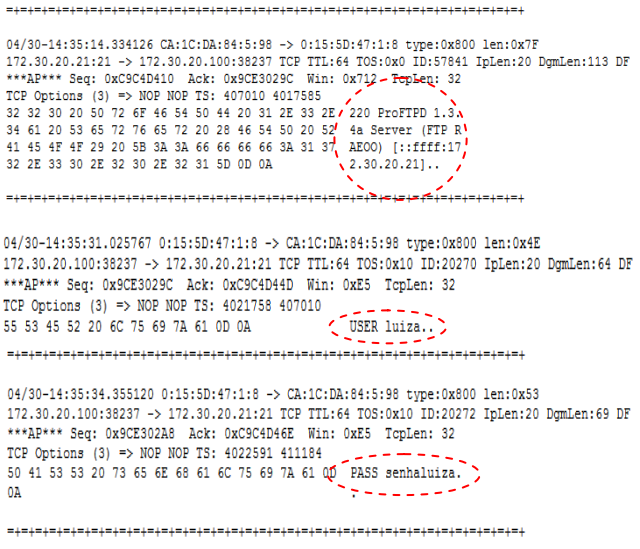

Fig. 16. Captura do ataque em *hexadecimal* e ASCII no *Honeypot* FTP.

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.

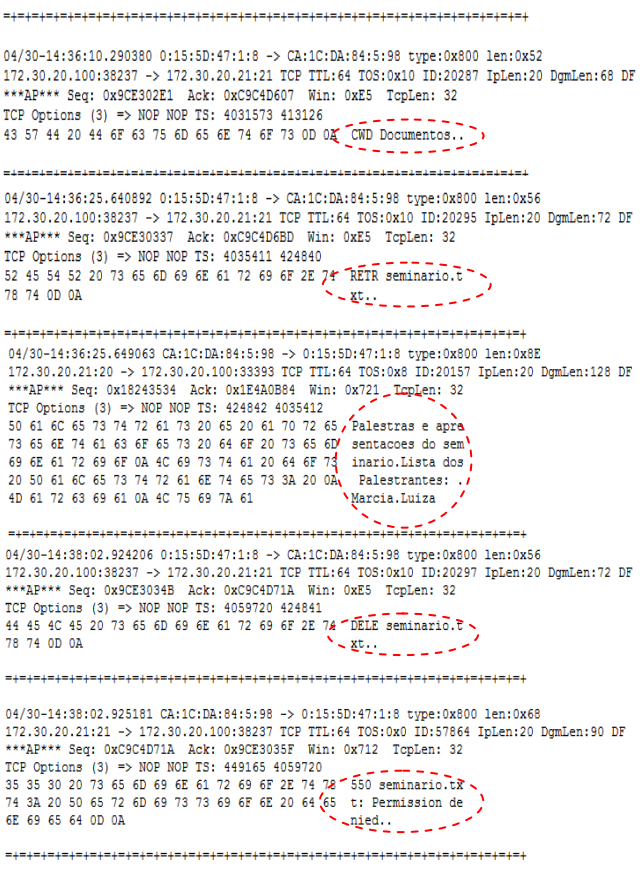

Fig. 17. Exploração de ataque no *Honeypot* FTP.

Neste mesmo cenário, continuando a análise do ataque no *Honeypot* FTP, a Figura 17 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)*.

VI. Conclusões

Neste trabalho, um ambiente *honeynet* virtual de autocontenção foi desenvolvido como solução de pesquisa para analisar vulnerabilidades e acompanhar novas formas de atividades de intrusos em redes. O desenvolvimento do ambiente proposto foi dividido em três fases, buscando uma otimização dos processos apresentados. Na primeira fase mostramos a arquitetura proposta, definindo o *hypervisor* utilizado e descrevendo como o *honeywall* e os *honeypots* foram configurados. Na segunda fase, um *firewall* e um *script* implementados 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.

Tais procedimentos indicam a validade de utilizar o ambiente em aplicações governamentais, visto que são providas todas as funcionalidades ao alcance do gestor para a captura de detalhes de ataques, seja visando a atualização de medidas de proteção, seja para efeito de demonstração forense.

Como trabalhos futuros, cabe testar outras formas de ataques; executar o 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.*

Agradecimentos

Os autores agradecem às Agências brasileiras de pesquisa e inovação CAPES (Projeto FORTE, Edital CAPES Ciências Forenses 25/2014), FINEP (Convênio RENASIC/PROTO 01.12.0555.00), pelo suporte a este trabalho..

Referências

[1] 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>. Acessado em: 15/04/2015.

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

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

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

[5] 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.

[6] COHEN F. *A Note on the Role of Deception in Information Protection - 1998*. Disponível em: <http://all.net/journal/deception/deception.html/>. Acessado em 17/04/2015.

[7] THE HONEYNET PROJECT. Disponível em: < https://www.honeynet.org//>. Acessado em: 01/04/2015.

[8] SOUSA JR, R. T.; SILVA, T. A.; ALBUQUERQUE, R. O. *Ambiente baseado em agentes de software para o auxílio na detecção e estudo de ataques em redes de computadores*. Proceedings of the 1st International Conference on Cyber Crime Investigation ICCyber'2004. Brasília, 2004. p. 156-161.

[9] SILVA, T. A.; ALBUQUERQUE, R. O.; BUIATI, F. M.; PUTTINI, R. S.; SOUSA JR, R. T. *A Community of Agents for Trapping Attacks Against Network Services and Redirecting Traffic Attacks to a Honeynet*. Proceedings of the First International Conference on Internet Technologies and Applications (ITA 05), Wrexham (UK), 2005. p. 135-143.

[10] SPITZNER, L. *Honeypots – Tracking Hackers*. *Indianápolis, IN: AddisonWesley*, 2002.

[11] THE HONEYNET PROJECT. *Know Your Enemy: Defining Virtual Honeynets*. Janeiro de 2003. Disponível em: <http://old.honeynet.org/papers/virtual/>. Acessado em: 01/04/2015.

[12] CITRIX. *Optimized server virtualization for all your workloads*. Disponível em < http://www.citrix.com/products/xenserver/overview.html>. Acessado em 11/04/2015.

[13] XENSERVER. *XenServer Open Source Virtualization Plataform*. Disponível em < http://xenserver.org/>. Acessado em 10/04/2015.

[14] THE HONEYNET PROJECT. *Know Your Enemy: Honeywall CDROM Roo*. Agosto de 2005. Disponível em:<http://old.honeynet.org/papers/cdrom/roo/>. Acessado em: 02/04/2015.

[15] THE HONEYNET PROJECT. *Roo CDROM User's Manual*. Maio de 2007. Disponível em: <http://old.honeynet.org/tools/cdrom/roo/manual/6-maintain.html/>. Acessado em: 02/04/2015.

[16] THE HONEYNET PROJECT. *Know Your Enemy: Sebek*. Novembro de 2003. Disponível em: <http://www.honeynet.org/papers/sebek/>. Acessado em: 05/04/2015.

[17] KALI LINUX. *Our Most Advanced Penetration Testing Distribution, Ever*. Disponível em < https://www.kali.org/>. Acessado em 20/04/2015

[18] SNORT. *Snort FAQ*. Disponível em <https://www.snort.org/faq/readme-thresholding>. Acessado em 22/04/2015.