SEGURANÇA EM *VIRTUAL PRIVATE SERVERS* LINUX: UMA ABORDAGEM ANALÍTICA E COMPARATIVA DE FERRAMENTAS *OPEN SOURCE*

Thiago José Lucas[1]; André Domingues[2]

Resumo. O cenário atual de *cloud computing* frente às ameaças existentes na rede mundial de computadores motiva os administradores de sistemas a se preocupar com a segurança de seus servidores. Num cenário onde *Virtual Private Servers* são uma importante alternativa na redução de custos com *datacenters* e infraestruturas de rede local, tem-se a necessidade de se agregar segurança aos sistemas com o uso de *software*s para tal. O presente artigo introduz à importância da segurança da informação nesses ambientes, além de realizar um estudo de caso com a implementação e análise de alguns *software*s open source de segurança para sistemas operacionais Linux.
Palavras-chave: *Cloud computing; Virtual Private Servers;* segurança de servidores.

Abstract. The current scenario of cloud computing in the face of threats existing in the World Wide Web encourages system administrators to worry about the security of your servers. In a scenario where Virtual Private Servers are an important alternative for cost reduction and data center infrastructure network, there is the necessity of adding security to the system by using *software* to do so. This paper introduces the importance of information security in these environments, in addition to conducting a case study with the implementation and analysis of some open source *software* security for Linux operating systems.
Key-Words: Cloud computing**;** Virtual Private Servers*;* security servers.

1 Introdução

A necessidade de redução de custos vem mudando o cenário das infraestruturas de rede nos parques tecnológicos. A adoção cada vez mais comum de serviços em *cloud computing* traz benefícios consideráveis. Os *Virtual Private Servers* são servidores remotos e virtualizados, operando na nuvem e sem a necessidade de adoção e manutenção de hardwares, *datacenters* e *links* de acesso para os servidores locais.

O ambiente onde os *Virtual Private Servers* operam - principalmente pelo fato de sua disponibilidade na Internet - é crítico no tocante à segurança. Essa necessidade faz com que a implementação de *softwares* de segurança seja imprescindível à manutenção do sistema operacional.

[1] Professor da Faculdade de Tecnologia de Ourinhos-FATEC; e-mail: thiagojlucas@gmail.com.
[2] Professor da Universidade Tecnológica Federal do Paraná (UTFPR); e-mail: anddomingues@utfpr.edu.br.

O objetivo deste trabalho é disponibilizar um cenário onde se possam analisar os principais recursos de algumas ferramentas *open source* para prover segurança em *Virtual Private Servers* Linux. A seção 2 fornecerá ao leitor informações sobre a importância da segurança de informação em ambientes de *cloud computing*. A seção 3 estabelecerá uma comparação entre algumas ferramentas que são categorizadas de acordo com o seu objetivo. Já a seção ocupar-se-á se dá o estudo de caso da implementação de uma ferramenta de cada categoria.

2 Principais Requisitos de Segurança

A segurança de sistemas é hoje um requisito essencial, uma vez que a informação é um dos ativos mais importantes de uma organização ou um dos bens de maior valor para um indivíduo. Do ponto de vista de Goodrich e Tamassia (2013), a segurança da informação pode ser definida nos termos do acrônimo C.I.D. (confidencialidade, integridade e disponibilidade), que podem ser caracterizados como seguem:

- Confidencialidade - propriedade que evita acesso não autorizado à informação;
- Integridade - propriedade de garantir que a informação não seja alvo de alteração não autorizada;
- Disponibilidade - propriedade que garante que a informação esteja acessível e modificável para usuários autorizados.

O fato de *Virtual Private Servers* possuírem, na maioria dos casos, um cenário inseguro - dada a sua disponibilidade na internet - faz com haja preocupação com a segurança destes tipos de sistemas. Walker (2013) elenca as principais ameaças existentes em um cenário compartilhável, como o de *Cloud Computing* que segue os mesmos princípios do cenários dos *Virtual Private Servers*. Entre eles estão as ameaças de "violação de dados", "perda de dados", "uso de aplicações inseguras" e "negação de serviço".

2.1 Violação de Dados

A violação de dados é um incidente que ameaça a integridade da informação. Para se compreender a importância dessa preocupação e, principalmente, para justificar a presente pesquisa, nos remetemos à Madureira (2013) que, por sua vez, afirma que, em 2012, o custo total médio de um incidente de violação de dados no Brasil foi de R$ 2,64

milhões. O autor ainda complementa afirmando que há registros de casos onde o prejuízo chega a quase R$ 10 milhões com tal incidente.

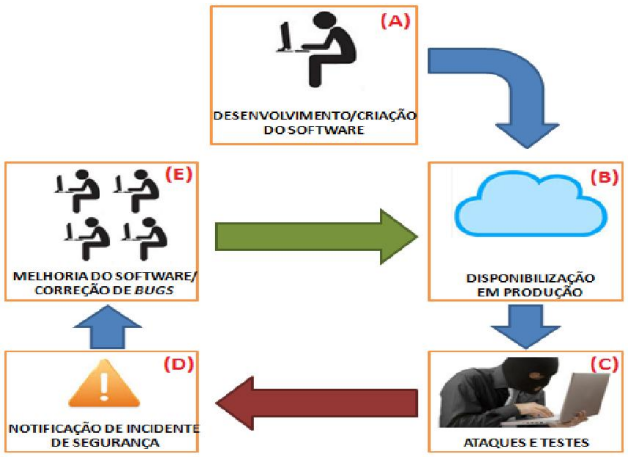
2.2 Perda de Dados

A perda de dados é um incidente que ameaça a disponibilidade da informação. Nesse sentido, o uso de *Virtual Private Servers* em *Cloud Computing*, segundo estudo da Symantec vem crescendo em decorrência de uma alternativa para pequenas e médias empresas contra a perda de dados. A pesquisa realizada contou com a colaboração de 2053 organizações em 30 países revela que 34% (um total aproximado de 698 organizações) adotam o uso de nuvens públicas.

2.3 Uso de Aplicações Inseguras

Quando falamos em *Virtual Private Servers Linux* estamos falando de um servidor virtual, executado em um ambiente de *Cloud Computing* com um Sistema Operacional que opera sobre o *Kernel Linux*. Isso nos remete ao princípio do *open source*, uma vez que um Sistema Operacional Linux tem como principal vantagem os *softwares* agregados à ele e disponibilizados como código aberto - em geral esta disponibilização está discriminada em sua GPL[3].

Figura 1 - Ciclo de atualização de *software* motivada por incidente de segurança da informação.



Fonte: Elaborada pelo autor.

[3] *General Public License:* Licensa de programas da *Free Software Foundation*.

Softwares como Apache[4] e *My*SQL[5], devido ao uso massivo na internet, são constantes alvos de ataques. Grupos de segurança de redes ou mesmo *crackers* dedicam seu tempo e conhecimento à descobrir vulnerabilidades dos *software*s *open source* em sua versão estável. Isso cria um ciclo infindável que pode ser visualizado pela figura 1.

A figura 1 pode ser interpretada da seguinte maneira, conforme a sua legenda alfabética:

a) Desenvolvimento/Criação do *Software*: Diz respeito ao momento em que o *software* está em desenvolvimento. Esse primeiro item diz respeito ao processo que vai desde o projeto de acordo com as necessidades até a finalização de um pacote estável.

b) Disponibilização em Produção: Diz respeito à disponibilização do *software* em versão estável (ou não) à comunidade[6]. Pode ser resumido no fato da disponibilização do *software* para *download* na página do desenvolvedor.

c) Ataques e Testes: Um *software* - livre ou não - sempre que é disponibilizado, na medida em que sua aplicação torna-se acessível à comunidade, transforma-se em alvo de ataques de *hackers* ou *crackers*. Os primeiros têm por objetivo descobrir vulnerabilidades e potenciais ameaças que o uso de tal *software* traz consigo. Já os segundos, além do mesmo objetivo dos *hackers*, querem explorar tais vulnerabilidades de forma à prejudicar a comunidade. Fato é que ambos contribuem - direta ou indiretamente - para o aprimoramento do *software* em questão.

d) Notificação de Incidente de Segurança: Tão *log*o uma vulnerabilidade é descoberta, a comunidade é comunicada por meio de listas de segurança de *softwares*. Um exemplo de lista é a "*Security Tracker*"[7], uma ferramenta que permite ao usuário receber notificações de incidentes de segurança relacionados a determinados *softwares*.

e) Melhoria do *Software*/Correção de *Bugs*[8]: Última etapa do ciclo de atualização do *software*. A comunidade ou os desenvolvedores oficiais trabalham à fim de solucionar o *bug* reportado e devolver em produção do *software* atualizado, para que a etapa "a" seja executada novamente, fechando aqui um "ciclo infindável".

Por mais controles que uma aplicação possua, ela nunca poderá ser considerada 100% segura. Isso é o que Santos (2008) afirma, baseando-se no princípio de que na

[4] Servidor Web licenciado sob GPL *open source.*
[5] Sistema Gerenciador de Banco de dados licenciado sob GPL *open source.*
[6] Termo usado para nomear os usuários e contribuintes do software livre.
[7] Disponível em http://securitytracker.com/
[8] Termo informal que faz referência à problemas encontrados nos *softwares* em produção.

medida em que a eficácia dos controles aumenta, os *crackers* aprimoram suas técnicas de invasão e exploração das vulnerabilidades.

2.4 Negação de Serviço

Ataques de negação de serviço são frequentes em redes de computadores conectadas à internet. Estes ataques são divididos em dois tipos, de acordo com a quantidade de máquinas atacantes: DoS ou DDoS:

- DoS – *Denial of Service*: Ataque de negação de serviço onde apenas um *host* envia um grande número de pacotes ao destino.
- DDoS – *Distributed Denial of Service*: Ataque de negação de *service* onde dois ou mais *hosts* enviam grande quantidade de pacotes ao destino.

Nos dois casos o objetivo é o mesmo: causar um cenário onde o servidor ocupa todo o seu processamento para responder às requisições dos atacantes, tornando demorada ou impossível a resposta de requisições de clientes reais.

Para se ter ideia da importância de se prevenir dessa ameaça, no segundo semestre de 2011, o ataque de DDoS mais poderoso identificado pela Kaspersky[9] foi 20% mais forte se comparado ao semestre anterior ao levantamento, com o foco principal dos ataques tendo como alvo páginas de comércio eletrônico por meio de uma técnica denominada HTTP *flood* (alto número de envio de requisições HTTP, causando a sobrecarga do *webserver*).

3 Comparativo de Ferramentas *Open Source*

Ferramentas *open source* que agregam segurança em *Virtual Private Server* Linux são uma importante alternativa para usuários ou empresas que não desejam (ou não possam) investir consideravelmente na compra de *softwares* proprietários ou aquisição de licenças. Em contrapartida, é necessário um preparo do administrador do sistema para a manutenção dessas ferramentas em produção. As ferramentas serão estudadas no decorrer deste trabalho, com objetivo de proporcionar ao leitor um cenário analítico entre elas, com suas vantagens e desvantagens. Elas se dividem nas seguintes categorias:

[9]Empresa russa de desenvolvimento de *softwares* para segurança na internet.

- IDS/IPS (*Intrusion Detection System/Intrusion Prevention System*);
- NSM (*Network Security Monitoring*);
- LAS (*Log Analysis System*);
- Antivirus.

O quadro 1 mostra de forma dividida os *softwares* que serão analisados, de acordo com a sua categoria:

Quadro 1 - Categorização dos *softwares* de acordo com sua natureza.

IDS/IPS	NSM	LAS	Antivirus
- Fail2ban - Snort - Suricata	- IPPL - SentryTools	- CipherDyne PSAD	- ClamAV

Fonte: Elaborada pelos autores.

3.1 *Intrusion Detection System* e *Intrusion Prevention System*

Sistemas de detecção e/ou prevenção de intrusão são *softwares* que possuem uma base de dados de comportamentos suspeitos e analisam o tráfego na rede ou o comportamento de determinados *softwares* secundários à fim de identificar se trata-se de uma tentativa de intrusão. Existem IDS e IPS baseados em *hardware* (HIDS e HIPS) e IDS e IPS baseados em *host* (HIDS e HIPS). Os IDS e IPS analisados neste tópico são baseados em *host.*

3.1.1 Fail2ban

De acordo com a página oficial do projeto Fail2ban (2013), o *software* trabalha analisando arquivos de *log* de conexões e "banindo" endereços IP com comportamentos maliciosos.

O fail2ban pode ser definido como uma ferramenta que protege uma variedade de serviços contra tentativas de acessos indesejados: "[...] *fail2ban is a tool that serves to protect a variety of services against unwanted visitors* [...]" (HOBSON, 2013, p. 161).

Um exemplo é a tentativa de *login* em determinado serviço (vsFTPd[10], por exemplo) com usuário(s) e senha(s) inválido(s). Para cada tentativa de *login* frustrada, o serviço (nesse caso o servidor FTP) gera uma linha de texto nova no *log* (vsftpd.*log*), sendo um alerta para o Fail2ban, que monitorara esse *log*.

[10] vsFTPd – *Very Secure File Transfer Protocol Daemon* – Um servidor FTP *open source*.

Detalhes da implementação e funcionamento deste *software* serão abordados no tópico 4.

3.1.2 Snort

A página oficial do projeto Snort (2013) o define como o sistema de detecção e prevenção de intrusões mais utilizado no mundo, com cerca de 400 mil usuários registrados.

Em uma definição ainda mais detalhada e técnica desta ferramenta, Cox e Gerg (2007) o caracterizam como o provável melhor *open source* IDS disponível. Os autores complementam a caracterização da ferramenta afirmando que ele foi inicialmente desenvolvido para operar via linha de comando (sem *interface* gráfica) mas que, devido à sua popularidade e desempenho, várias aplicações desenvolvidas por terceiros passaram a integrá-lo (inclusive algumas *interfaces* gráficas), o que o torna o melhor *open source* IDS disponível para *download.*

Caswell; Beale e Baker (2007) apresentam uma definição alternativa do Snort, afirmando ser um *open source* NIDS[11] capaz de realizar análise em tempo real de tráfego e registrar *logs* de pacotes em redes IP[12] para posterior análise.

O *software* Snort por ser configurado para executar em três diferentes modos, de acordo com a página oficial do projeto (2013):

1. *Sniffer mode* – Nesse modo o Snort apenas exibe na saída padrão – monitor de vídeo, por exemplo – os pacotes que estão sendo trafegados pela rede.
2. *Packet logger mode* – Similar ao *sniffer mode*, porém armazena os dados coletados em um arquivo no disco rígido.
3. *Network IDS* – É o modo mais que apresenta mais flexibilidade de configuração. Aqui ele opera como detector e analisador do tráfego, de forma à identificar tentativas de acessos não-autorizados.

O Snort se mostra uma ferramenta mais robusta e com muitos recursos adicionais em comparação com o Fail2ban. Enquanto o segundo apenas trabalha buscando padrões de *strings* em arquivos de *log*, o Snort possui ferramentas que possibilitam uma análise mais inteligente do tráfego, tendo em vista seus três modos de operação.

[11] *Network Intrusion Detection System* – Sistema de Detecção de Intrusão que opera em rede.
[12] Protocolo da camada de rede utilizado na Internet.

3.1.3 Suricata

Suricata é um IDS, IPS e NSM *open source* que se destaca pelo seu alto desempenho. Ele foi desenvolvido e é mantido pela OISF[13]. Existem três motivos para ao menos se testar o Suricata, são eles:

1. É um sistema *multi threaded*, ou seja, suporta processamentos simultâneos no mesmo sistema operacional, porém por processadores diferentes. Possui a flexibilidade de se configurar qual é o máximo de carga que cada instância de execução do Suricata vai poder exercer em cada núcleo de processamento.
2. Tem suporte à identificação de protocolo, de forma nativa. Esse recurso possibilita ao administrador de rede a criação de regras por protocolos e não por portas, o que de certa forma é o mais inteligente.
3. Nativamente o Suricata é capaz de identificar centenas de tipos de arquivos que estejam trafegando pela rede, pois possui uma base de dados que possibilita tal ação. Essa *feature* pode ser útil na necessidade de se identificar determinada transferência na rede local, como por exemplo a disseminação de um vírus.

O Suricata é uma excelente alternativa ao Fail2ban, mas também não se comparam os seus recursos com a forma de operação do Snort, que é mais completo por possuir modos de operação diferentes em comparação dos os IDS/IPS Fail2ban e Suricata.

3.2 *Network Security Monitoring*

Softwares de *Network Security Monitoring* têm o objetivo de analisar o comportamento da rede e – em caso de algum comportamento anormal – notificar o administrador ou realizar ações previamente configuradas.

3.2.1 IPPL

O IPPL (*Internet Protocol Packet Logger*) é um NSM, de acordo com a categorização exposta na Tabela 1. Hass e Bernard (2000), desenvolvedores do *software* definem o IPPL como um *daemon* [14] que registra, em arquivos de *log,* as informações

[13] *Open Information Security Foundation* – Fundação de desenvolvimento de softwares.
[14] *Software* independente que é executado em segundo plano.

dos pacotes IP destinados a determinado *host* na rede. Ele é executado em segundo plano e exibe informações sobre os pacotes IP; possui um filtro onde se especifica o que monitorar e o que não monitorar. É um *software open source* licenciado sob GPL.

Em seus aspectos técnicos, os autores complementam detalhando o funcionamento interno do IPPL da seguinte maneira:

- *Log* de Pacotes – O IPPL, assim como o Suricata, é *multi threaded.* Para cada evento novo (cada novo protocolo registrado), a *thread* principal cria um *socket*[15] e então as *subthreads* decidem, de acordo com os filtros, se ele será analisado ou não.
- Mecanismo de Filtragem – Para cada pacote recebido, uma lista de regras é lida e o seu conteúdo é comparado com o de cada item da lista. Caso o item da lista case com o tipo de pacote, ele é registrado.
- Cache DNS[16] – Para tornar mais eficiente a resolução reversa de DNS, o IPPL cria uma base de dados de nomes, que é renovada de tempos em tempos. Caso esse recurso não fosse implementado pela equipe de desenvolvimento, cada pacote recebido necessitaria de uma consulta à um servidor DNS, causando a sua sobrecarga e também uma sobrecarga desnecessária na rede local.

3.2.2 SentryTools

A ferramenta, segundo Open Source *Software* (2013), provê segurança à nível de *host* implementando serviços para plataforma Linux, dentre outras. As ferramentas contidas no SentryTools são: PortSentry, *Log*check/*Log*Sentry e o *Host*Sentry. Na sequência segue a caracterização dos três *software*s acima citados: (i) Fonseca (2011) define o PortSentry como um sistema que faz uma espécie de "simulação", abrindo portas no servidor – simulando serviços – e baseado no que receber de conexão nessas portas ele pode tomar alguma atitude, como por exemplo solicitar ao *firewall* que bloqueie o endereço IP de origem; (ii) antigamente chamado de *Log*check, Negus (2013) afirma ser uma ferramenta muito útil para que se possa gerenciar os arquivos de *log* do sistema. O *software* trabalha de forma a analisar os arquivos de *log* gerados pelo *syslog*[17] e destacando as mensagens que possam caracterizar algum problema de segurança para o *host* ou o sistema operacional; e (iii) Rowland (2003) caracteriza o

[15] Um ponto de comunicação entre processos em redes de computadores ou *localhost*.
[16] *Domain Name System,* Sistema de Nome de Domínios.
[17] Padrão criado pela IETF para a transmissão de mensagens de log em redes IP.

*Host*sentry como um sistema de detecção de intrusão baseado em *host*. Pode ser definido como um *Login Anomaly Detection*, pois monitora tentativas de *login* no *host* em busca de anomalias ou comportamentos suspeitos. Para isso, conta com uma base de dados que o auxilia na comparação em busca de ameaças.

Na comparação com o IPPL, o Sentrytools tem a vantagem de ser modular. As três ferramentas descritas acima (PortSentry, *Log*Sentry e *Host*Sentry) trabalham independentes umas das outras, além de ter objetivos distintos na garantia de segurança de *Virtual Private Servers* Linux. O IPPL, como já citado, é apenas um *software* que registra as atividades de rede do *host*, tornando-o menos robusto, porém mais leve em comparação com o SentryTools.

3.3 *Log Analisys System*

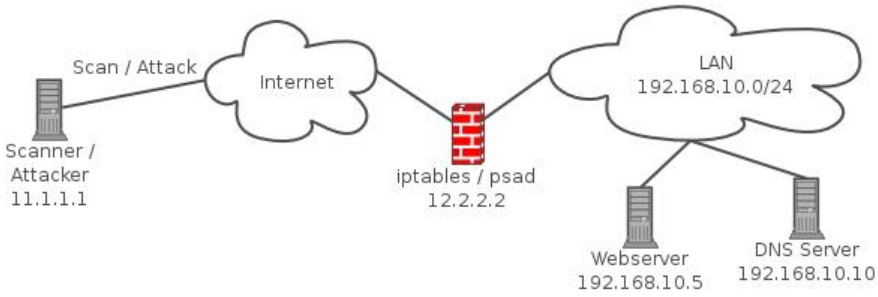
Softwares de *Log Analisys System* têm como objetivo analisar arquivos de *log* gerados por outras aplicações em busca de padrões. Estes padrões podem indicar alguma tentativa de fraude na rede.

3.3.1 CipherDyne PSAD

O CipherDyne PSAD pode detectar vários tipos de tráfego suspeito na rede. RACH (2007).

A página oficial do projeto[18] o define como uma coleção de *daemon s* que, ao serem executados em *hosts* Linux, analisam os *logs* criados pelo *netfilter/iptables*[19] em busca de comportamentos suspeitos na rede.

Figura 2 - Estrutura típica de implementação do CipherDyne PSAD em uma rede LAN com acesso à Internet.



Fonte: Disponível em http://cipherdyne.org/psad/

[18] Disponível em http://cipherdyne.org/psad/.
[19] Módulo do sistema operacional linux que fornece funções de *firewall,* NAT e *log* de dados.

A figura 2 ilustra um exemplo de topologia de rede com acesso à internet onde o *software* pode ser usado. Nesse caso, o atacante – denominado por "*Scanner/Attacker*" – sempre que necessitar acessar os dois ativos da rede local LAN – denominados "*Webserver*" e "*DNS Server*" – terá o tráfego analisado pelo *software*, denominado na figura 2 por "*iptables/psad*".

3.4 Antivírus

Softwares de Antivírus podem trabalhar de duas formas. A primeira é analisando o tráfego da rede em busca de padrões suspeitos de códigos maliciosos, porém no caso de conexões criptografadas acaba não sendo eficaz. Já a segunda forma é analisando arquivos que estão armazenados em algum dispositivo como discos rígidos e *pen drives*.

3.4.1ClamAV

A página oficial do projeto[20] na Internet define o ClamAV como um antivírus *open source* desenvolvido com o objetivo de detectar *Trojans*[21], Virus[22], *Malwares*[23], dentre outras ameaças. Ele fornece *daemon s* de alto desempenho (por ser *multi threaded*) além de ferramentas poderosas de linha de comando. O ClamAV conta com uma biblioteca de padrões de vírus que é usada como base nas comparações dos arquivos para que o *software* seja capaz de identificar se determinado arquivo é suspeito ou não.

Turnbull (2005) afirma que uma vantagem significativa na escolha do ClamAM, além do fato de ser uma ferramenta *open source*, é a de possuir as definições (biblioteca) de vírus frequentemente atualizadas.

No tópico 4 (Estudo de Caso) uma ferramenta de cada categoria será instalada e testada, ação que servirá de base para que se possa afirmar a eficácia da ferramenta instalada.

[20] Disponível em http://www.clamav.net/
[21] Também conhecido como Cavalo de Tróia, por parecer inofensivo. Trata-se de um código malicioso.
[22] Vírus – Código malicioso que se reproduz e é invisível ao usuário ou administrador do sistema.
[23] Categoria de código malicioso onde o *trojan* se encaixa.

4 Estudo de Caso

Neste tópico, uma ferramenta de IDS/IPS (Fail2ban), uma ferramenta de NSM (IPPL) e uma ferramenta de LAS (Cipherdyne PSAD) elencadas na Tabela 1 serão instaladas e testadas. O objetivo é analisar o processo de instalação, bem como algumas de suas funcionalidades básicas.

4.1 Instalação da Ferramenta Fail2ban e do servidor FTP vsFTPd

Para a instalação do IDS/IPS Fail2ban será utilizado a distribuição Debian[24] na sua versão 6 com Kernel[25] Linux 2.6.32. Neste processo será testada a eficácia do Fail2ban quando um cliente FTP tentar, por diversas vezes, se autenticar no servidor "vsFTPd" com credenciais inválidas, simulando uma tentativa de acesso não autorizado ou mesmo um ataque de negação de serviço.

Os *softwares* "Fail2ban" e "vsFTPd" foram instalados pelo gerenciador de pacotes padrão da distribuição:

Figura 3 - Linhas de comando.

```
root@localserver:~# apt-get update && apt-get -y install fail2ban
root@localserver:~# apt-get -y install vsftpd
```

Fonte: Elaborado pelos autores.

Após a instalação com sucesso dos dois *softwares* e suas dependências, o arquivo `"/etc/fail2ban/jail.conf"` foi editado e a diretiva `[vsftpd]` alterada para o seguinte padrão:

Figura 4 - Linhas de comando.

```
[vsftpd]
enabled  = true
port     = ftp,ftp-data,ftps,ftps-data
filter   = vsftpd
logpath  = /var/log/vsftpd.log
maxretry = 6
```

Fonte: Elaborado pelos autores.

As opções (ver figura 4) acima podem ser interpretadas da seguinte maneira:

- *enabled* – Pode ser *false* ou *true*. Essa opção habilita ou desabilita a função da diretiva [vsftpd].

[24] Distribuição/Sistema Operacional com Kernel Linux.
[25] Kernel – O núcleo do sistema operacional.

- *port* – Quais as portas que o Fail2ban deve bloquear assim que determinado comportamento suspeito for encontrado. As portas não são numéricas e as referências nominais estão em */etc/services*.
- *filter* – Qual o tipo de filtro. O *path* de filtros é o diretório */etc/fail2ban/filter.d/*
- *log*path – Qual o arquivo de *log* que o Fail2ban irá analisar em busca de comportamentos suspeitos. Padrões contidos no filtro especificado na opção *filter* serão buscados nesse arquivo.
- *maxretry* – Número de tentativas de *login* no servidor vsftpd sem sucesso que o Fail2ban deve interpretar como normal. Tentativas maiores que o valor especificado nessa opção serão interpretados como ataque.

Após a realização destes ajustes, basta iniciar o serviço Fail2ban com o execução do script a seguir: "`root@localserver:~# /etc/init.d/fail2ban start`".

O servidor FTP precisará apenas de dois ajustes na sua configuração contida em "`/etc/vsftpd.conf`":

- `anonymous_enable=NO;`
- `local_enable=YES`.

A opção `anonymous_enable=NO` somada à opção `local_enable=YES` indica que apenas usuários do sistema poderão fazer *login* no servidor FTP. Para iniciá-lo, basta executar o script conforme a seguir: `root@localserver:~# /etc/init.d/vsftpd start`.

Para verificar se o servidor FTP está em execução, executa-se o comando ftp e realiza-se o *login* com o usuário *anonymous* e senha nula. A informação esperada é 230 *Login successfu*, que pode ser vista na sétima linha abaixo:

Figura 5 - Linhas de comando.

```
root@localserver:~# ftp localhost
Connected to localhost.
220 (vsFTPd 2.3.2)
Name (localhost:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Fonte: Elaborada pelos autores.

Simularemos agora uma tentativa de *login* com usuário e senha desconhecidos: *user* e *password*. O objetivo deste teste é analisar a saída de texto que o servidor vsFTPd adicionará no *log* `/var/log/vsftpd.log`. A saída do *log* segue abaixo:

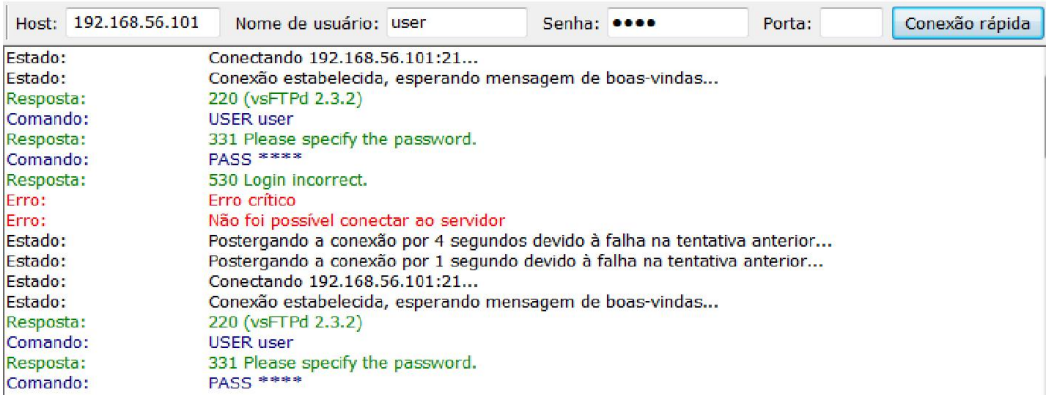
Figura 6 - Linhas de comando.

```
Fri Jul 12 12:20:13 2013 [pid 2] CONNECT: Client "127.0.0.1"
Fri Jul 12 12:20:21 2013 [pid 1] [user] FAIL LOGIN: Client "127.0.0.1"
```

Fonte: Elaborada pelos autores.

Com um cliente FTP, simularemos agora uma tentativa de *login* via rede com seis tentativas inválidas, conforme ilustra a figura 7:

Figura 7 - Tentativas de fraude ao servidor FTP vsFTPd executados pelo cliente FileZilla.



Fonte: Elaborada pelos autores.

Após a tentativa de fraude, o *log* do IDS/IPS Fail2ban mostra que o mesmo reconheceu a tentativa de fraude, conforme figura 8.

Figura 8 - Linhas de comando.

```
root@localserver:~# tail -1 /var/log/fail2ban.log
2013-07-12 12:31:38,050 fail2ban.actions: WARNING [vsftpd] Ban
192.168.56.1
```

Fonte: Elaborada pelos autores.

O trecho final `Ban 192.168.56.1` indica que o Fail2ban comunicou o *firewall* netfilter/iptables solicitando o bloqueio do endereço IP 192.168.56.101 às portas FTP do servidor, o que pode ser provado com a saída do comando abaixo:

Figura 9 - Linhas de comando.

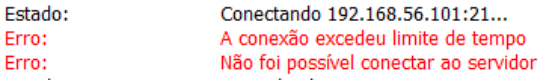
```
root@localserver:~# iptables-save | grep 192.168.56.1
-A fail2ban-vsftpd -s 192.168.56.1/32 -j DROP
```

Fonte: Elaborada pelos autores.

A saída do comando mostra que existe uma chamada `DROP` para a origem `192.168.56.101`.

Para provar que o *firewall* realmente bloqueou o acesso, novamente foi testada a conexão FTP ao servidor, teste ilustrado na figura 10:

Figura 10 - Tentativa de *login* no servidor FTP após bloqueio realizado pelo *Firewall*.



Fonte: Elaborada pelos autores.

Neste momento conclui-se – em relação à ferramenta Fail2ban – que se trata de um *software* extremamente simples de ser instalado e configurado, além de funcional. O IDS/IPS agiu conforme o esperado, assim que identificou uma tentativa de fraude no *login* ao servidor FTP, comunicou-se com o *firewall* e solicitou o bloqueio do endereço IP de origem.

4.2 Instalação da Ferramenta IPPL

A instalação da ferramenta IPPL será realizada no mesmo sistema operacional utilizado na instalação da ferramenta Fail2ban exposta no tópico 4.1.

Por se tratar apenas de um *daemon* que armazenará em arquivo de *log* as conexões desejadas para posterior auditoria (se necessário), sua instalação e configuração são bastante simples e podem ser exemplificada conforme segue abaixo:

A instalação do software foi realizada via gerenciador de pacotes padrão da distribuição, conforme a seguir: `root@localserver:~# apt-get install ippl.`

Após instalação, configuraremos o IPPL para gravar em *log* todas as conexões entrantes ao *host*. No arquivo `/etc/ippl.conf` basta adicionar a seguinte diretiva, conforme a seguir: `log-in all /var/log/ippl/all.log.`

Feito isso, basta iniciar o software com o comando "`ippl &`" para que o mesmo seja executado em segundo plano. Realizaram-se três tipos de conexões entrantes ao *host*, sendo elas: `SSH`, `FTP` e `ICMP ECHO REQUEST`. O arquivo de *log* `/var/log/ippl/all.log` ficou com o seguinte conteúdo:

Figura 11- Linhas de comando.

```
root@localserver:~# cat /var/log/ippl/all.log
Jul 12 13:17:28 ssh connection attempt from 127.0.0.1
Jul 12 13:17:32 ftp connection attempt from 192.168.56.1
Jul 12 13:17:48 ICMP message type echo request from 192.168.56.1
```

Fonte: Elaborada pelos autores.

Em relação à ferramenta IPPL, conclui-se que é um poderoso aliado do administrador do sistema em um momento de análise ou auditoria. Após a ocorrência de incidentes ou mesmo para se analisar o comportamento da rede, um *log* de ocorrências é muito útil. Por se tratar de um simples arquivo de texto, filtros podem ser usados para se buscar padrões ou expressões desejadas.

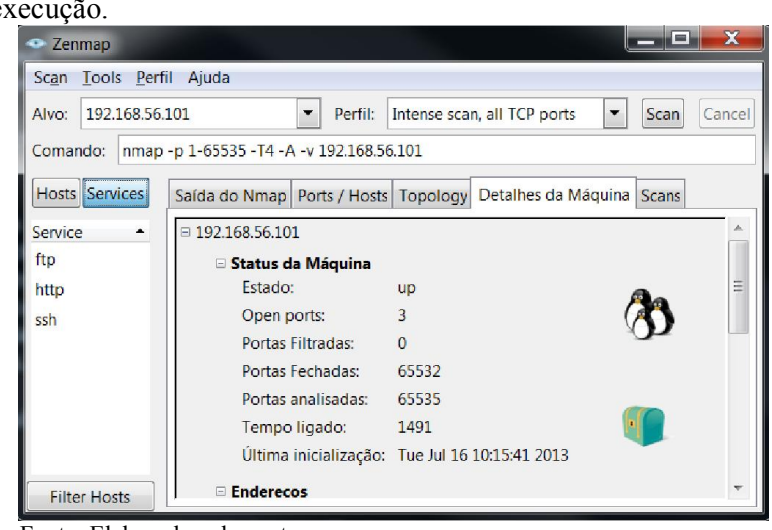
4.3 Instalação da Ferramenta Cipherdyne PSAD

A instalação da ferramenta PSAD será realizada no mesmo sistema operacional dos tópicos 4.2 e 4.1. Foi utilizado o gerenciador de pacotes padrão da distribuição para a `root@localserver:~# apt-get install psad`.

Como o PSAD trabalha analisando logs de firewall, nada foi alterado em seu arquivo de configuração e o mesmo foi iniciado com o comando abaixo: `root@localserver:~# /etc/init.d/psad start`.

A ferramenta *iptables* para o *firewall netfilter* foi utilizada para habilitar a gravação de *logs* de conexões entrantes na interface de rede `eth1` em `/var/log/messages`: `root@localserver:~# iptables -I INPUT -i eth1 -j LOG`.

Figura 12 - Execução de *Scanner* de rede no sistema operacional com o CipherDyne PSAD em execução.



Fonte: Elaborada pelos autores.

Após isso, a ferramenta "Zenmap NMAP GUI"[26] foi utilizada para que um *scanner* fosse executado no servidor, conforme ilustra a figura 12:

[26] *Scanner* de rede gráfico que executa NMAP em *background*

Tão logo o scanner foi executado, o CipherDyne PSAD já percebeu o comportamento na rede e criou em seu diretório de *logs* */var/log/psad* o diretório *192.168.56.1* que é justamente o endereço IP de origem do atacante (ou *scanner*). o conteúdo do diretório pode ser verificado com a saída do comando conforme figura 13.

Figura 13- Linhas de comando.

```
root@localserver:~# ls -l /var/log/psad/192.168.56.1/
192.168.56.101_email_alert
192.168.56.101_packet_ctr
192.168.56.101_start_time
danger_level
```

Fonte: Elaborada pelos autores.

Cada um destes arquivos tem uma função importante na análise do administrador do sistema operacional (sem contar a possibilidade de receber uma notificação por e-mail). Os arquivos são explicados a seguir:

- 192.168.56.101*_email_alert* – Trata-se do conteúdo do e-mail que é enviado ao administrador do sistema – caso configurado e especificado em */etc/psad/psad.conf*.
- 192.168.56.101*_packet_ctr* – Contado de pacotes. Cria-se no conteúdo desse arquivo uma divisão por protocolos (TCP ou UDP ou ICMP).
- 192.168.56.101*_start_time* – Data e horário do início da execução do *scanner*. Especificado em *timestamp* no conteúdo do arquivo.
- *danger_level* – Nível de perigo que o *scanner* apresenta de acordo com as informações que conseguiu coletar. Vai de 0 à 5 e pode ser personalizado em */etc/psad/psad.conf*.

Quanto à ferramenta PSAD, pode-se dizer que ela cumpre com o que promete. Sem necessidade de configuração inicial personalizada e apenas habilitando o *firewall netfilter* por meio do utilitário *iptables* o *software* foi capaz de identificar um *scanner* na rede local. O fato da ferramenta notificar o administrador via e-mail também é um diferencial.

5 Conclusão

O estudo de caso proposto e executado mostrou que – de forma simples, gratuita e intuitiva – é possível agregar segurança em *Virtual Private Servers* Linux de uma forma que o administrador tenha autonomia para tomar decisões em relação à segurança do sistema. Outros sistemas operacionais possuem *softwares* de IDS/IPS, NSM, LAS e Antivirus similares, contudo este estudo de caso fortalecido pela revisão bibliográfica

realizada mostrou a eficácia destes *softwares* de segurança em distribuições Linux, com o diferencial agregado do *software open source*.

No decorrer deste artigo os *softwares* foram comparados – à medida do possível – com os outros *software*s de sua mesma categoria. Em resumo, quanto às ferramentas de IDS/IPS, o Snort foi a que apresentou mais recursos e uma forma mais modularizada de trabalhar, embora o Fail2ban seja a ferramenta de implementação e utilização mais simples dentre as demais de sua categoria. As ferramentas de NSM, por sua vez, embora apenas a IPPL tenha sido implementada, têm a vantagem de auxiliar e resguardar o administrador em casos de pós-ataque (uma perícia, por exemplo). É aconselhado que se use em conjunto com ferramentas de IDS's e IPS's para que a segurança da rede não seja trocada apenas por um "monitoramento" ou "*logging* de eventos". Quanto à ferramenta de LAS, Cipherdyne PSAD, pode-se dizer que é bastante útil, pois trabalha de forma reativa, analítica. Deve ser utilizada em conjunto com a função de *log* do *netfilter*.

Conclui-se, portanto, que à medida em que a necessidade de segurança se torna maior, *softwares* e tecno*log*ias surgem com o objetivo de garantir a confidencialidade, a integridade e a disponibilidade dos dados servidos, fator provado por este artigo.

6 Referências

BARROS, J.; MORENO, J. **Mestres da espionagem digital**. Editora Universo dos Livros, 2008.

CASWELL, B.; BEALE, J.; BAKER, A. **Snort Intrusion Detection and Prevention Toolkit**, Editora Syngress, EUA. 2007.

COX, K.; GERG, C. **Managing Security with Snort & IDS Tools.** Editora O'Reilly, 2009.

FAIL2BAN. **Fail2ban Project Home-page.** Disponível em: <http://www.fail2ban.org/wiki/index.php/Main_Page>. Acesso em junho, 2013.

GOODRICH, M.; TAMASSIA, R. **Introdução À Segurança de Computadores**. Editora Bookman, 2013.

HASS, H; BERNARD; E. **IPPL Project Home-page.** 2013. Disponível em: <http://pltplp.net/ippl/>. Acesso em junho, 2013.

HOBSON, J. **CentOS 6 Linux Server Cookbook**. Editora Packt, 2013.

MADUREIRA, D. Violação de dados no Brasil custa R$ 2,6 milhões ao ano para empresas. **Jornal Valor Econômico**, 05 jun. 2013. Disponível em: <

http://www.valor.com.br/empresas/3150584/violacao-de-dados-no-brasil-custa-r-26-milhoes-ao-ano-para-empresas >. Acesso em: 17 dez. 2014.

NEGUS, C. **Red Hat Linux Bible - Fedora and Enterprise Edition**, Editora Wiley, 2004.

OPEN SOURCE *SOFTWARE*. **SentryTools Project Home-page**, 2013. Disponível em: <http://sentrytools.sourceforge.net/>. Acesso em junho, 2013.

RACH, M. **Linux *Firewalls***: Attack Detection and Response with Iptables, Psad, and Fwsnort. Editora No Starch, 2007.

ROWLAND, C. ***Host*sentry.** *HOST*SENTRY**.** 2003**.** Disponível em: <http://www.securityfocus.com/tools/275>. Acesso em Junho, 2013.

SANTOS, A. **Quem mexeu no meu sistema?** Editora Brasport, 2008.

SNORT. **Snort Project Home-page.** 2013**.** Disponível em: <http://www.snort.org/>. Acesso em: junho, 2013. Symantec (2013) "Pesquisa Global de PMEs 2013", http://www.symantec.com/pt/br/theme.jsp?themeid=smb-disaster-recovery, Junho.

TURNBULL, J. **Hardening Linux.** Editora Springer, 2005.

WALKER, K. **The Notorious Nine**: Cloud Computing Top Threats in 2013. Cloud Security Alliance. 2013, Disponível em: < https://cloudsecurityalliance.org/csa-news/ca-warns-providers-of-the-notorious-nine-cloud-computing-top-threats-in-2013/>. Acesso em: junho, 2013.