Glenon Mateus Barbosa Araújo

Análise de Sistemas de Detecção de Intrusão *Opensource* Snort e Suricata em uma Rede Acadêmica

Brasil

2018

Glenon Mateus Barbosa Araújo

Análise de Sistemas de Detecção de Intrusão *Opensource* Snort e Suricata em uma Rede Acadêmica

Trabalho de Conclusão de Curso submetida a graduação em Ciência da Computação da UFPA

Universidade Federal do Pará – UFPA

Faculdade de Computação

Bacharelado em Ciência da Computação

Orientador: Dr. Roberto Samarone dos Santos Araújo

Brasil

2018

fichacatalografica

Glenon Mateus Barbosa Araújo Análise de Sistemas de Detecção de Intrusão *Opensource* Snort e Suricata em uma Rede Acadêmica/ Glenon Mateus Barbosa Araújo. – Brasil, 2018- 56 p. : il. (algumas color.) ; 30 cm.
Orientador: Dr. Roberto Samarone dos Santos Araújo
Trabalho de Conclusão de Curso – Universidade Federal do Pará – UFPA
Faculdade de Computação
Bacharelado em Ciência da Computação, 2018.
1. Suricata. 2. Snort. 3. IDPS. I. Orientador. II. Universidade Federal do Pará. III. Faculdade de Computação. IV. Análise de IDPSs

Resumo

Palavras-chave: Segurança da Informação, Suricata, Snort, Sistema de Detecção de Intrusão, Sistema de Prevenção de Intrusão, IDS, IPS.

Abstract

Keywords: Security Information, Suricata, Snort, Intrusion Detection System, Intrusion Prevention System, IDS, IPS.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IDS	*Intrusion Detection System*
IPS	*Intrusion Prevention System*
SDI	*Sistema de Detecção de Intrusão*
SPI	*Sistema de Prevenção de Intrusão*
IDPS	*Intrusion Detection and Prevention System*
HIDS	*Host Based Intrusion Detection Systems*
NIDS	*Network Based Intrusion Detection Systems*
MB	*Megabytes*
GB	*Gigabytes*
SO	*Sistema Operacional*
JSON	*JavaScript Object Notation*
CAIS	*Centro de Atendimento a Incidentes de Segurança*
DoS	*Denial of Services*
DDoS	*Distributed Denial of Services*
URL	*Uniform Resource Locator*
LAN	*Local Area Network*
SPF	*Stateful Packet Filter*
SQL	*Structured Query Language*
XSS	*Cross-site scripting*
OISF	*Open Information Security Foundation*
ET	*Emerging Threats*

Sumário

1 Introdução

Este trabalho apresenta um comparativo entre sistemas de detecção e prevenção de intrusão (Intrusion Detection and Prevention System - IDPS) de código aberto mais populares na comunidade de segurança da informação. Essas ferramentas permitem monitorar sistemas computacionais, alertando os administradores sobre possíveis ameaças.

Neste capítulo, apresentam-se, na seção 1.1, as motivações deste trabalho, evidenciando a importância do IDPS em um ambiente corporativo real. Em seguida, no seção 1.2, os objetivos do trabalho, na seção 1.3, uma descrição das metodologias utilizadas, na seção 1.4 os trabalhos relacionados, e por fim, na seção 1.5, a organização do trabalho.

1.1 Motivação

A Internet é um conjunto de redes físicas heterogênea (uma variedade de dispositivos conectados, *smartphones*, *desktops*, *notebooks*, servidores, *switches*, roteadores, entre outros) funcionando como uma rede lógica única de alcance mundial. O grande e contínuo crescimento da Internet gerou um aumento da sua complexidade, que a expõe a diversas vulnerabilidades.

A todo momento, novos ataques ou mesmo variações de ataques já existentes surgem e são lançados a várias redes indiscriminadamente em busca de vulnerabilidades. Em 2015, foram reportados ao Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil (CERT.br) cerca de 722.205 incidentes, em 2016, esse número diminuiu, chegando a 647.112 (INTERNET, 2017c). Apesar da diminuição, esse número é considerado grande, presumindo-se que há muitos incidentes que não são reportados e/ou identificados.

As empresas, de qualquer segmento e tamanho, devem ter o trabalho de manter os ativos seguros e isso vai além da utilização de anti-vírus nos computadores. Ter uma política de atualização de *software* em estações de trabalho e servidores, aliados com boas práticas nas configurações de serviços, dificultam a exploração de vulnerabilidades.

A utilização de *firewall*, não pode ser encarado com uma solução definitiva, passando uma falsa sensação de segurança, pois muitas portas legítimas podem estar vulneráveis, como acontece, por exemplo, com a porta 80 que hospeda *websites* vulneráveis (MARTINELO; BELLEZI, 2014).

Diante desse cenário, torna-se cada vez mais importante para o administrador de rede e/ou segurança da informação o uso de ferramentas de IDPS, permitindo identificar e tratar de forma automatizada os incidentes de segurança.

Para implementa tais ferramentas em uma rede, deve-se levar em consideração a

flexibilidade e a administração simplificada para não resultar que a empresa/instituição fique na dependência de um único fabricante ou fornecedor da solução. Além disso, a ferramenta deve ter um bom desempenho e eficiência para não passar a falsa sensação de proteção ou degradar o desempenho da rede.

1.2 Objetivos

Diante do apresentado, este trabalho tem como objetivo geral, analisar e apresentar um comparativo entre as soluções de código aberto de IDPS mais conhecidas: Snort e Suricata. Como desdobramento de tal objetivo, os seguintes objetivos secundários foram definidos:

a) Apresentar conceitos sobre segurança da informação e sua importância em um ambiente corporativo;

b) Descrever problemas relacionados a ataques envolvendo redes de computadores;

c) Descrever as ferramentas, compreendendo requisitos, características, modos de atuação e funcionalidades;

d) Descrever o ambiente experimental;

e) Realizar experimentos e coletar dados para validar o funcionamento e a eficiência das ferramentas;

1.3 Metodologia

Esse trabalho se configura numa pesquisa qualitativa. Utilizou-se uma máquina com *hardware* robusto, configurada com 130G de memória RAM, com um processador possuindo 40 núcleos e duas interfaces de rede, uma para gerência e outra configurada em modo '*promisc*' com espelhamento do roteador. Nela instalou-se o SO XenServer versão 7, da Citrix, SO voltado para virtualização com um bom desempenho num ambiente de produção real.

No *host*, instalou-se duas máquinas virtuais, uma para cada ferramenta de IDPS em teste, Snort e Suricata, nas suas versões mais recentes, 2.9.8.3 e 4.0.0, respectivamente. A mesma quantidade de recurso de *hardware* foi alocada para as máquinas, criando assim um ambiente igual para ambas as ferramentas. Além disso, após configuradas, usou-se a mesma base de assinaturas aberta da *Emerging Threats* (ET), que possui frequentes atualizações.

As métricas para comparação avaliadas são a quantidade de recurso usado pela ferramenta para análise do tráfego em um período de tempo determinado, quantidade de falsos positivos e negativos e a taxa de detecção. Para avaliar os recursos de *hardware*, usou-se dois recursos. Primeiro, foi instalado nas máquinas virtuais um *daemon* collectd, a segunda forma, foi instalar um servidor de monitoramento Zabbix.

Para facilitar a avaliação dos alertas gerados pelas ferramentas e determinar os falsos positivos e negativos, optou-se por centralizar o *logs* em um servidor. Para tal, usou-se uma infraestrutura que reúne três serviços, descrito na subseção 4.1.2.

1.4 Trabalhos Relacionados

O uso de ferramentas IDS *opensource* Snort e/ou Suricata, importantes na área de segurança, já foi abordado e apresentou alguns resultados satisfatórios em pesquisas anteriores, por exemplo, nas pesquisas de Nagahama *et al.* (2013), Martín *et al.* (2014) e Cléber *et al* (2014).

No trabalho do Nagahama *et al.* (2013), é usado redes definidas por *softwares*, que desacopla os planos de controle e de dados, permitindo adaptar o funcionamento da rede de acordo com a necessidade de cada um. O *software* utilizado na pesquisa foi o OpenFLOW. Tal uso, tem como objetivo mitigar a falta de integração do IDS com os equipamentos de rede como switches e roteadores, o que limita a atuação destas ferramentas.

Já Martín *et al.* (2014), utiliza-se do BroFlow que possui uma série de vantagem, como, detecção de intrusão através de algoritmos simples, modular e flexível, reação imediata a um ataque descantando pacotes dos atacantes os mais próximo da origem. Dentre os resultados obtidos, destacam-se que a ferramenta conseguiu garantir o encaminhamento de pacotes legítimos na rede na taxa máxima do enlace e reduziu, em até dez vezes, o atraso na rede provocado pelo ataque.

No trabalho de Cléber *et al* (2014), foi feito uma comparação de desempenho das ferramentas de IDS (Snort e Suricata) porém usou-se dados sintéticos fornecidos pela *Defense Advanced Research Projects Agency* (DARPA). Ao final, listou-se as vantagens e desvantagens existentes de cada ferramenta. No trabalho proposto, no entanto, serão usados dados mais próximo de um ambiente de produção.

1.5 Organização do Trabalho

Além deste capítulo introdutório, esse trabalho está divido da seguinte forma:

No Capítulo 2, são definidos conceitos sobre redes de computadores e segurança da informação, também são descritos os ataques comuns e ferramentas utilizadas para validar e avaliar as soluções de IDPS.

Em seguida, no Capítulo 3, a definição IDPS, os tipos existentes, as funcionalidades e descrição das ferramentas avaliadas: Snort e Suricata.

O Capítulo 4 detalhará o cenário real e a infraestrutura utilizada para os realização dos testes das ferramentas, os testes realizados e o resultados obtidos.

Por fim, no Capítulo 5, as considerações finais e trabalhos futuros.

2 Segurança em Redes de Computadores

Esse capítulo apresentará conceitos e definições sobre segurança da informação e rede de computadores, mostrando seus principais componentes e os ataques mais utilizados contra essas redes. Ao final, foram apresentadas as ferramentas usadas para simular ataques com o intuito de avaliar o comportamento dos IDPS (Snort e Suricata).

Este capítulo esta organizado da seguinte forma: A próxima seção apresenta as definições sobre segurança da informação. Na seção 2.2 será apresentado uma topologia de rede comum, que existe na organizações. Na seção 2.3 será abordado os pontos vulneráveis em uma rede. Na seção 2.4 será apresentado o principais ataques a rede de computadores. Por fim, na seção 2.5 será apresentada as ferramentas usadas para gerar os ataques.

2.1 Definições

Quando se fala e segurança de sistemas computacionais, logo vem a mente da maioria dos usuários da rede, roubo de número de cartões de crédito, *hackers* danificando páginas (subseção 2.4.4) e aplicações *Web* e ataques de negação de serviço (subseção 2.4.5). Também tens a imagem dos *malwares*, como vírus, cavalos de tróia e *worms* (subseção 2.4.6). Esses possuem maior visibilidade pois representam uma parte significativa das ameaças existentes na Internet.

Porém existem outros problemas que apresentam riscos que normalmente não são levados em consideração, como administradores desonestos, funcionários descontentes e usuários que utilizam dados sigilosos de forma equivocada.

Para um melhor entendimento sobre segurança da informação, precisa-se entender alguns elementos listados abaixo (COELHO; ARAUJO; BEZERRA, 2014):

a) **Incidente de segurança**: qualquer evento oposto a segurança; por exemplo, ataques de negação de serviços (*Denial of Service* - DoS), roubo de informações, vazamento e obtenção de acesso não autorizado a informações;

b) **Ativo**: qualquer coisa que tenha valor para a organização e para seus negócios. Alguns exemplo: banco de dados, softwares, equipamentos (computadores e notebooks), servidores, elementos de redes (roteadores, switches, entre outros), pessoas, processos e serviços;

c) **Ameaça**: qualquer evento que explore vulnerabilidades. Causa potencial de um incidente indesejado, que pode resultar em dano para um sistema ou organização;

d) **Vulnerabilidade**: qualquer fraqueza que possa ser explorada e comprometer a

segurança de sistemas ou informações. Fragilidade de um ativo ou grupo de ativos que pode ser explorada por uma ou mais ameaças. Vulnerabilidades são falhas que permitem o surgimento de deficiências na segurança geral do computador ou da rede. Configurações incorretas no computador ou na segurança também permitem a criação de vulnerabilidades. A partir dessa falha, as ameaças exploram as vulnerabilidades, que, quando concretizadas, resultam em danos para o computador, para a organização ou para os dados pessoais;

e) **Risco**: probabilidade de uma ameaça se concretizar;

f) **Ataque**: qualquer ação que comprometa a segurança de uma organização;

g) **Impacto**: consequência de um evento.

Diante desses elementos, podemos definir segurança da informação como sendo a proteção das informações, sistemas, recursos e demais ativos contra desastres, erros (intencionais ou não) e manipulação não autorizada, objetivando a redução da probabilidade e do impacto de incidentes de segurança.

Segundo (TÉNICAS, 2013), segurança da informação é a preservação da confidencialidade, da integridade e da disponibilidade da informação, adicionalmente, outras propriedades, tais como autenticidade, responsabilidade, não repúdio e confiabilidade, podem também estar envolvidas.

Dentre vários conhecimentos que um profissional de segurança deve possuir, o conceito mais básico e considerado o pilar de toda a área de segurança corresponde à sigla CID (Confidencialidade, Integridade e Disponibilidade), de modo que um incidente de segurança é caracterizado quando uma dessas áreas é afetada (PEIXINHO; FONSECA; LIMA, 2013). Abaixo será detalhado cada item.

a) **Confidencialidade**: termo ligado à privacidade de um ativo ou recurso, que deve ser acessível somente por pessoas ou grupos autorizados;

b) **Integridade**: possui duas definições, a primeira está relacionada com o fato da informação ter valor correto, a segunda, está ligada à inviolabilidade da informação;

c) **Disponibilidade**: está relacionada ao acesso à informação, que deve está disponível quando necessária.

Dois dos termos citados são fáceis de ser monitorados pois é perceptível para o usuário: a integridade (capacidade de identificar se uma informação foi alterada) e a disponibilidade (através da tentativa de acesso a um serviço e verificando se o mesmo está respondendo adequadamente). No entanto, só é possível identificar se houve quebra da confidencialidade com auditorias, analisando os registros de acesso (se houver), isso torna a identificação custosa e em muitos casos impossível (PEIXINHO; FONSECA; LIMA, 2013).

Além dos conceito listados, a literatura moderna considera mais alguns conceitos auxiliares, temos:

a) **Autenticidade**: garantia que uma informação, produto ou documento foi elaborado ou distribuído pelo autor a quem se atribui;

b) **Legalidade**: garantia de que ações sejam realizadas em conformidade com os preceitos legais vigentes e que seus produtos tenham validade jurídica;

c) **Não repúdio**: conceito bastante utilizado em certificação digital, onde o emissor de uma mensagem não pode negar que a enviou;

d) **Privacidade**: habilidade de uma pessoa controlar a exposição e a disponibilidade de informações acerca de si.

Os ataques são classificados em passivo e ativo. Em um ataque passivo não há interação direta (modificações de arquivos ou afetando os recursos) com o sistema alvo, o atacante apenas monitora com o objetivo de obter informações. Por outro lado, os ataques ativos há modificações de dados que afetam as operações do sistema. Os ataques podem ser divididos em categorias apresentadas na tabela.

Tabela 1 – Classificação dos ataques passivos e ativos

Ataque	**Categoria**	**Descrição**
Passivo	Liberação de conteúdo da mensagem	Ocorre quando uma informação é captada e seu conteúdo é lido pelo atacante
	Análise de tráfego	Ocorre quando o tráfego da troca de uma informação (criptografada ou não) é analisado para identificar padrões nas mensagens
Ativo	Disfarce	Ocorre quando uma entidade finge ser outra entidade
	Repetição	Ocorre quando os dados são capturados passivamente e, subsequentemente, retransmitidos para produzir um efeito não autorizado
	Modificação da mensagem	Ocorre quando alguma parte da mensagem original é alterada para produzir um efeito não autorizado
	Negação de serviço	Ocorre quando há um impedimento ou inibição do uso ou gerenciamento normal das instalações de comunicação

Fonte: Autoria própria

Além disso, podemos dividir os ataques em quatro categorias, que são (CLARO, 2015):

a) **Interrupção**: Esse ataque tem como objetivo interromper ou destruir o serviço, afetando a disponibilidade da informação, como ocorre, por exemplo, nos ataques de negação de serviço (DoS) e ataques de negação de serviço distribuído (DDoS);

b) **Interceptação**: Esse ataque visa capturar informações que estão em transito sem a percepção do vítima, comprometendo sua privacidade. Seu objetivo principal é gerar cópias de informações, arquivos e programas de forma não autorizada. Um exemplo desse tipo de ataque é o *Man-in-the-Middle*.

c) **Modificação**: Esse ataque ocorre quando as informações transmitidas são alteradas, após serem captadas, afetando sua integridade. Como exemplo desse ataque temos o *Replay Attack*.

d) **Falsificação**: Esse ataque tem como finalidade se passar por um usuário do sistema para obter informações e transmiti-las na rede, comprometendo a autenticidade da informação. Como exemplo desse ataque temos o IP *Spoofing*.

2.2 Cenário Geral

Nessa seção será explicado alguns conceito básico sobre redes de computadores e descrever uma topologia de rede genérica conectada a internet.

Uma rede de computadores é um conjunto de dispositivos interconectados para compartilhar recursos como *hardware*, *software*, interação e interatividade, onde existem máquinas que desempenham os papéis de clientes, servidores e/ou parceiros dependendo do serviços disponíveis na rede (JUSTO; TAMARIZ, 2012).

As características de uma rede são: dois ou mais computadores interligados; meio físico de comunicação (com fio, sem fio, metálico, fibra, etc); vários tipos de equipamentos (estações de usuários, servidores, concentradores, etc); software para comunicação entre os equipamentos (protocolos); aplicativos para transferência de informação (ELIAS; LOBATO, 2013).

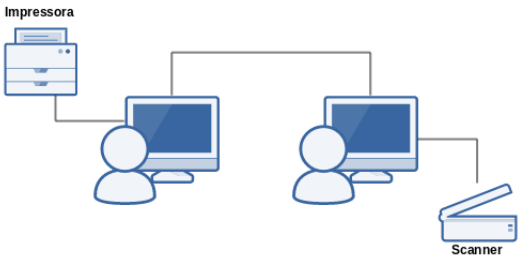
As redes podem ser classificadas em duas categorias (ELIAS; LOBATO, 2013):

a) **Redes par-a-par ou peer-to-peer**: Aqui não existem servidores dedicados ou hierarquia entre os computadores, todos são iguais, onde cada computador funciona como cliente e/ou servidor, cabendo o usuário determinar o que será compartilhado (Figura 1);

b) **Redes cliente-servidor**: Aqui há servidores dedicados que oferecem serviços à rede (servidores de arquivos e impressão, correio, fax, comunicação, aplicações, etc), geralmente são otimizadas para processar rapidamente as requisições dos clientes da rede e para garantir a segurança dos arquivos e pastas (Figura 2).

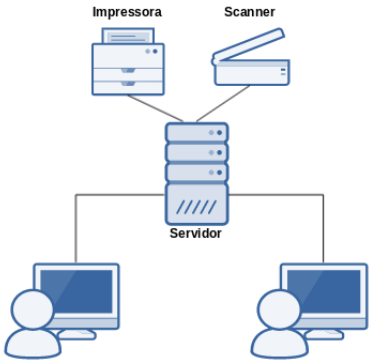
Uma Internet é uma rede de computadores que interconecta centenas de dispositivos de computadores ao redor do mundo (KUROSE; ROSS, 2013). A União Internacional de Telecomunicações (UTI) estima que haja cerca de 3.578 milhões (Figura 3) de usuários usando diferentes tipos de dispositivos, como, celulares, automóveis, *webcams*, TVs, *laptops*, consoles para jogos, entre outros (TELECOMUNICAÇÕES, 2017).

Figura 1 – Rede par-a-par



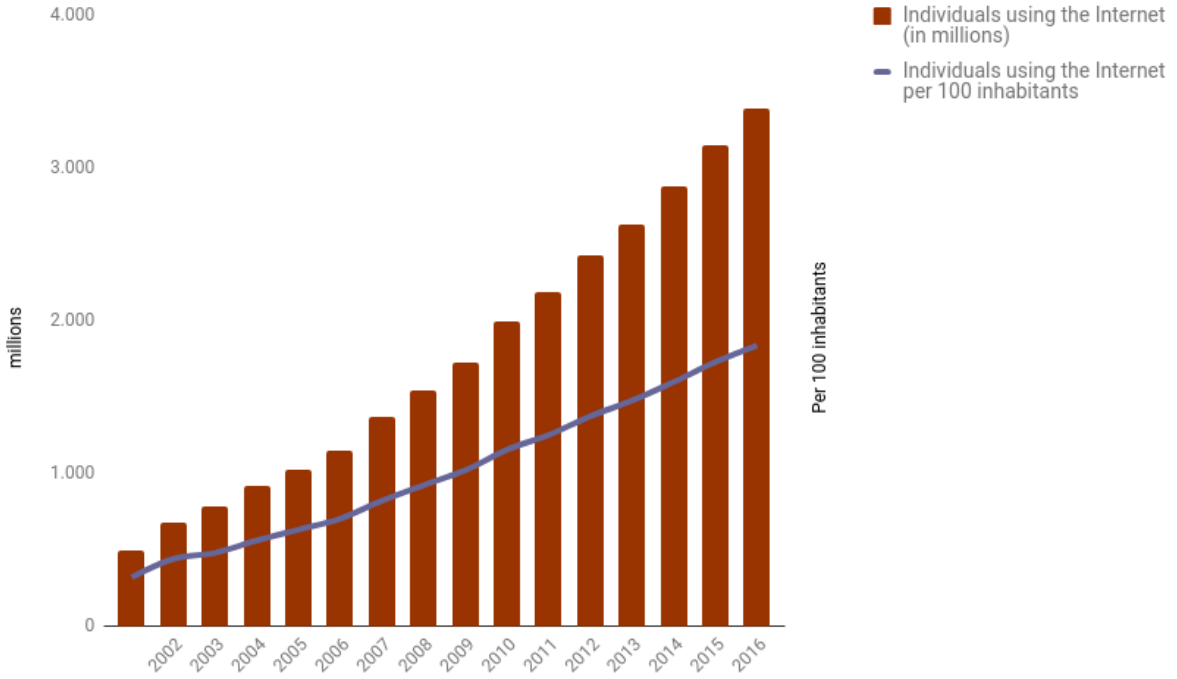
Fonte: Autoria própria

Figura 2 – Rede cliente-servidor



Fonte: Autoria própria

Figura 3 – Quantidade de usuário conectados na Internet



Fonte: (TELECOMUNICAÇÕES, 2017)

Os equipamentos que são comumente usados nas redes de computadores são:

a) **Concentradores (*hubs*)**: São pontos de conexões para dispositivos em uma rede, contendo várias portas usados para conectar os segmentos da LAN. Quando um pacote chega em uma porta, ele é replicado para as demais portas, assim, todos os clientes conectados ao *hub* podem ver todos os pacotes. Esse tipo de equipamento não é mais recomendado.

b) **Switches**: São equipamentos que se diferem dos *hubs* por serem capazes de ler o MAC de origem e destino. Além disso, realizam comutações (os pacotes são

individualmente encaminhadas entre os dispositivos conectados) de quadros na camada de enlace;

c) **Roteadores**: São dispositivos de rede mais tradicionais, como de backbone das intranets e da internet. Suas principais funções são seleção dos melhores caminhos de saída para os pacotes de entrada e roteamento destes pacotes para a interface de saída apropriada.

Um administrador, com o minimo de consciência sobre segurança, coloca em sua rede, um *firewall* de borda. Um *firewall* sempre é colocada na divisa entre duas ou mais redes, pode ser entre redes privadas ou entre uma rede privada e a Internet. Uma empresa pode ter muitas LANs conectadas de forma arbitrárias, mas todo o tráfego de saída ou de entrada da empresas para a internet deve ser feito através do *firewall*, permitindo assim, que alguns pacotes passem e bloqueando outros (TANENBAUM; WETHERALL, 2011).

Há três tipos básicos de *firewall*, os mais tradicionais são os filtros de pacotes e os *proxies*. O terceiro tipo é uma evolução do filtro de pacotes tradicional chamado de filtro de estados de pacotes ou *stateful packet filter* (SPF) (ULBRICH; VALLE, 2007).

Um *firewalls* de filtros de pacotes são baseados em tabelas configuradas pelo administrador da rede. Essas tabelas listam as origens e os destinos aceitáveis e/ou bloqueados e as regras padrões que orientam o que deve ser feito com os pacotes recebidos de outras máquinas ou destinados a elas, ou seja, o *firewall* tem como função controlar o tráfego entre as redes (TANENBAUM; WETHERALL, 2011).

Há vários *softwares* que implementam filtro de pacotes. Alguns são instalados em *hardwares* como roteadores outros são programas que rodam em computadores comuns (ULBRICH; VALLE, 2007). Um utilitário bastante conhecido e utilizado para essa finalidade é o *iptables*. A Tabela 2 apresenta um exemplo de uma tabela de regras.

Tabela 2 – Tabela de regras aplicadas no *firewall*

IP Origem	**IP Destino**	**Porta Origem**	**Porta Destino**	**Protocolo**	**Flag TCP**	**Ação**
Rede Externa	Servidor Web	Todas	80,443	TCP	Todos	Permitir
Rede Externa	Servidor Web	Todas	21,3000:3070	TCP	Todos	Permitir
Todas	Todas	Todas	Todas	Todos	Todos	Negar

Fonte: Autoria própria

No exemplo, são permitidas conexões na Intranet no Servidor Web pelas portas 80 e 443 (padrão nos protocolos HTTP e HTTPS) para todos os *Flags* TCP (ACK, ACK/SYN, SYN e FIN). Além disso, podemos definir um range de portas, como na linha 2, que são abertas as portas 21 e todas as portas entre 3000 e 3070, utilizadas por padrão pelo protocolo FTP.

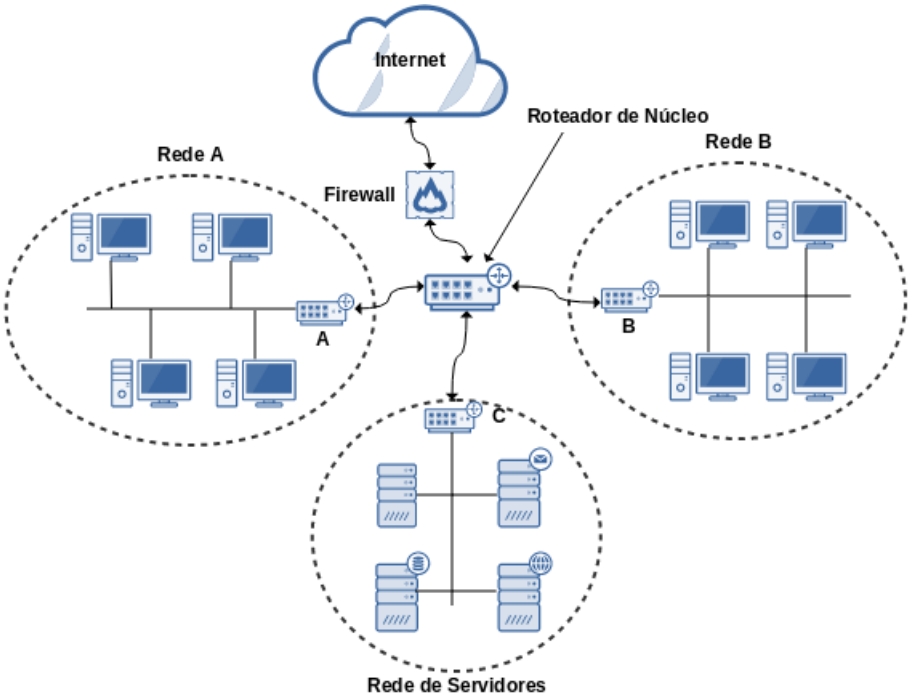
Um *proxie* trabalha na camada de aplicação interagindo com o programa e seus protocolos, independente de como esse protocolo será encapsulado na pilha TCP/IP. Por exemplo, um *proxy* para Web trabalha apenas com o protocolo HTTP, bloqueando os demais.

Além disso, pode-se configurar-lo para controlar quem pode ou não acessar serviços externos (ULBRICH; VALLE, 2007).

No *firewall* de filtros de pacotes por estado (SPF) uma nova tecnologia de análise de pacotes foi agregada, permitindo que eles lembrem-se de pacotes anteriores antes de permitir outro mais recente entrar. Isso é implementado na forma de uma tabela de conexões ativas. Quando uma conexão é iniciada, todos os dados do pacote são guardados nela. Se um novo pacote chegar em direção à mesma máquina, o SPF consulta a tabela. O novo pacote e aceito caso seja dada a continuação da conexão ou rejeitado, se não for (ULBRICH; VALLE, 2007).

Na Figura 4 apresenta uma típica rede composta por um roteador de núcleo que interliga roteadores (A, B e C) de outras redes da Intranet, que por sua vez interliga os clientes e/ou servidores. Todo trafego de saída e entrada da Intranet para a Internet passa pelo roteador de núcleo, além disso, os pacotes são tratados por um *firewall* de borda, que determina o que entra e o que sai da rede local.

Figura 4 – Topologia geral de uma rede de computadores



Fonte: Autoria própria

2.3 Pontos de Vulnerabilidade

Nessa seção será abordado os pontos fracos que uma pessoa má intencionada pode explorar para ter um ataque bem sucedido a um rede de computador.

Apesar da preocupação dos administradores em proteger suas redes de ataques, devido a sua heterogeneidade, sempre haverá uma breja a ser explorada. A literatura considera o ser humano como elo mais fraco, é bastante comum o usuário cadastrar senhas fracas, por conveniência, e fácil memorização (subseção 2.4.3).

Uma técnica bastante comum usadas por golpistas que visa obter informações financeiras ou informações pessoais da vítima, é o *phishing*. Os ataques de *phshing* mais conhecidos são os quais um atacante induz usuários a acessar um site clonado de uma instituição financeira, de modo a coletar suas credenciais de acesso (CERON, 2015). Muitos usuário, por falta de conhecimento ou até por ingenuidade, acabam informando seus dados em sites falsos, sem ao menos verificar a veracidade do mesmo.

Outros serviços visados por esse tipo de ataque são (CERON, 2015):

a) Credenciais de serviços: e-mails, redes sociais ou armazenamento;

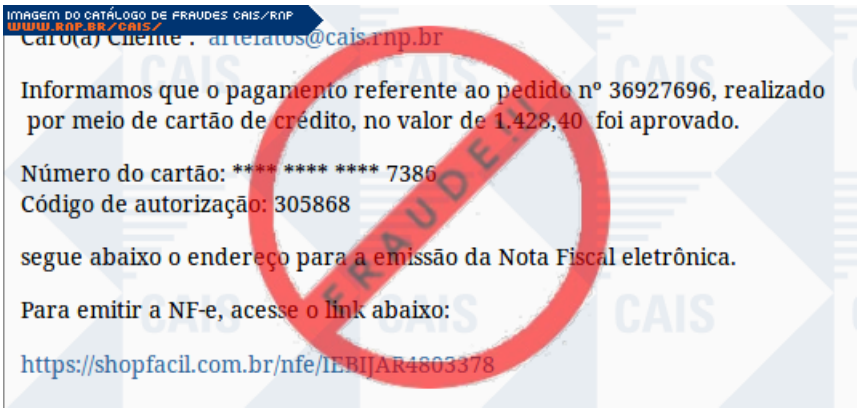
b) Webmail corporativo;

c) Programa de milhagem (companhias aéreas ou redes de supermercados);

d) Comércio eletrônico.

Uma forma de difundir *phishing* é através de *spam*. Um *spam* é uma mensagem, na maioria das vezes de conteúdo falso, enviado para diversos e-mails, nessas mensagens podem conter uma URL de um site falso ou um site com códigos maliciosos subseção 2.4.6 que infectam a vítima. Um *spammer*, como é chamado quem envia *spam*, também usa técnica de *phishing* para obter dados de acesso de e-mails pessoais. Dessa forma, o *spammer* pode acessar a conta de e-mail da vítima e enviar *spam* para todos os seus contatos, sem que a vítima perceba.

Figura 5 – Fraude identificada pelo CAIS



Fonte: (SEGURANÇA, 2017)

O CAIS mantém um catálogo de fraudes identificadas sobre os principais golpes que estão em circulação. Na Figura 5, temos uma fraude recebida por e-mail (*spam*) contendo

um *link* para download de um arquivo malicioso criado para roubar informações da vítima e instalar outros arquivos (SEGURANÇA, 2017).

Outro problema comum é o desenvolvimento de aplicações *web* sem nenhuma preocupação com segurança, podendo comprometer, não somente o serviço, mas também, em casos mais extremos, o servidor inteiro. Para tal, atacante pode usar vários artifícios, os mais conhecidos são *sql injection* e *cross-site script*.

A inserção de *Structured Query Language* (SQL) via formulário na aplicação *web* resulta num ataque de *sql injection*. O atacante injeta um código dentro dos campos de entrada, como usuário e senha, de uma aplicação onde a declaração condicional sempre será verdadeiro quando executado. Em casos bem sucedidos, o atacante pode alterar o banco de dados, acessar informações sensíveis ou ter acesso ao sistema (S; S; M, 2014).

No exemplo abaixo, a declaração condicional 'OR 1=1' torna toda a clausura WHERE verdadeiro pois a expressão 1=1 é uma tautologia. A consulta retorna todos os dados da tabela *user_info*. Perceba que os dois hífens fornecidos no final da entrada comenta o resto da linha.

```
SELECT * FROM user_info WHERE logID="" OR 1=1 -- AND pass1=""
```

O *Cross-site scripting* (XSS) é uma forma de ataque que permite utilizar um aplicação vulnerável para transporta códigos maliciosos até o navegador de outros usuário. O navegador da vítima entende que o código recebido é legítimo e, por isso, informações sensíveis, como o identificador de sessão do usuário, por exemplo, podem ser acessadas programaticamente (UTO, 2013).

Com o XSS pode-se roubar histórico de navegação, fazer uma varredura de redes privadas, descobrir consultas realizadas em mecanismos de busca, escravizar o navegador *web* e proliferar *worms* (subseção 2.4.6) baseados em XSS (UTO, 2013).

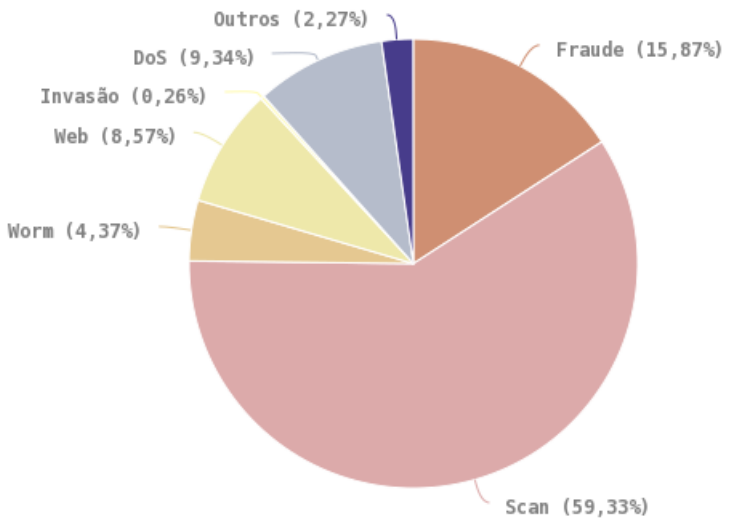
2.4 Ataques Comuns à Redes de Computadores

Nessa seção será descritos os ataques mais comuns à redes e serviços de organizações privadas e públicas, financeiras ou acadêmicas. Para licitar os ataques dessa seção, levou-se em consideração as estatísticas divulgada pelo CERT.br (Figura 6).

O CERT.br é o grupo de resposta a incidentes de segurança para a internet brasileira, mantido Comitê Gestor da Internet no Brasil. Atua na notificação e tratamento de incidentes de segurança dando apoio no processo de resposta. Além disso, faz um trabalho de conscientização e treinamento sobre problemas de segurança no Brasil.

Paralelamente ao CERT.br temos o Centro de Atendimento a Incidentes de Segurança (CAIS), mantido pela Rede Nacional de Ensino e Pesquisa (RNP). O CAIS é responsável por zela pela segurança da rede Ipê (infraestrutura de rede dedicada à comunidade brasileira de

Figura 6 – Estatísticas de ataques reportadas ao CERT.br



Fonte: (INTERNET, 2017d)

ensino superior), detectando, resolvendo e prevenindo incidentes de segurança. Além disso, tem o papel de orientar (através de publicações de cartilhas) e disseminar boas práticas de segurança da informação, educando e conscientizando usuários de todos os níveis sobre os principais riscos em segurança da informação (SEGURANÇA, 2017).

Desde 2008, todas a fraudes identificadas pelo CAIS estão sendo ordenadas e disponibilizadas para consulta (Figura 7). Adicionalmente, são enviados alertas através de uma lista quando uma fraude mostra-se particularmente perigosa aos usuários e computadores.

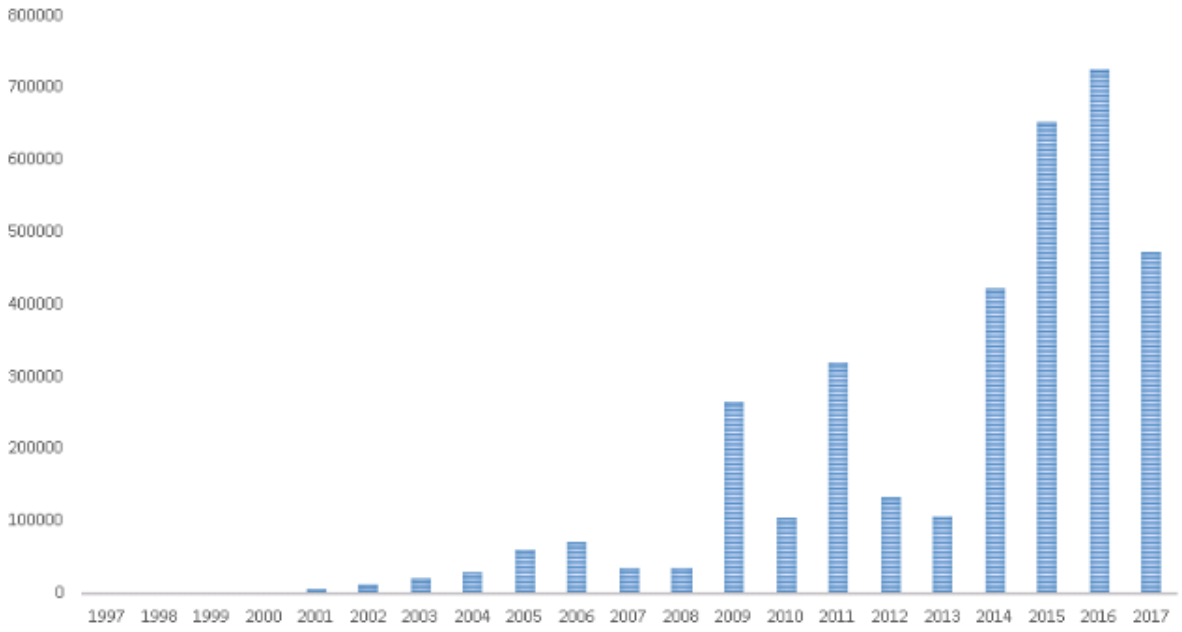
2.4.1 *Scanners*

Conforme tratado na seção 2.1, uma vulnerabilidade é a fraqueza em sistemas de informação, procedimentos de segurança do sistema e controles internos, ou aplicação que pode ser explorada tendo como origem uma ameaça.

Scanners são programas usados para varrer uma rede à procura de computadores (tanto pessoais como servidores) com alguma vulnerabilidade. Podemos dividir os *scanners* em dois tipos (ULBRICH; VALLE, 2007):

a) ***scanner* de portas TCP/IP abertas (ou *portscanner*)**: cada serviço de rede que estiver disponível em uma determinada máquina é uma porta de entrada em potencial. Existem um total de 128 mil porta, sendo 65536 portas para o protocolo TCP e 65536 portas para o protocolo UDP. O *portscanner* verifica quais portas TCP/IP estão abertas com o objetivo de determinar quais serviços de rede TCP/IP disponíveis. Quase todas as técnicas de *portscanning* valem-se de sinais (ou *flags*), TCP, UDP ou ICMP, e a partir da análise desses sinais, os *scanners*

Figura 7 – Estatísticas de incidentes reportados ao CAIS



Fonte: (SEGURANÇA, 2017)

retiram informações sobre o sistema; (ULBRICH; VALLE, 2007)

b) ***scanner* de vulnerabilidades conhecidas**: Um vez determinados os serviços que uma máquina disponibiliza na rede entra em cena o *scanner* de vulnerabilidade. A ideia é checar, através de uma lista de falhas conhecidas, se o sistema está ou não executando um serviço com problemas (ULBRICH; VALLE, 2007).

Normalmente, essas ferramentas funcionam em três estágios (BASSO, 2010):

a) **Configuração**: aqui será definido o endereço IP do alvo ou a URL (Uniform Resource Locator) da aplicação Web e demais parâmetros, como, por exemplo, utilização de *proxy*.

b) **Rastreamento**: esse estágio, em *scanners* de vulnerabilidade de aplicações web, o *scanner* chama a primeira página web e então examina seu código procurando *links*. Cada *link* encontrado é registrado e este procedimento é repetido várias vezes até que *links* e páginas não sejam mais encontrados.

c) **Exploração**: vários testes são executados e as requisições e respostas são armazenadas e analisadas. Ao final, os resultados são exibidos ao usuário e podem ser salvos para uma análise posterior.

Um bom *scanner* de vulnerabilidade verifica itens como (ULBRICH; VALLE, 2007):

a) **Erros comuns de configuração**: portas não utilizadas por nenhum serviço abertas;

b) **Configurações e senhas-padrões**: instalação de softwares deixando-os com as configurações de fabrica (com usuário e senha-padrão), por exemplo, usuário: admin, senha: admin. Outro problema é deixar serviços desnecessários ativados;

c) **Combinação óbvias de usuário e senha**: Usuário comuns tendem a colocar senhas fáceis de lembrar;

d) **Vulnerabilidades divulgadas**: Sempre que uma falha de segurança é divulgada há uma corrida dos desenvolvedores para saná-las. Em paralelo, existem *hackers* que querem chegar aos sistemas vulneráveis antes de serem consertados.

Os *scanners* de vulnerabilidades automatizados contêm, e atualizam regularmente, enormes bancos de dados de assinaturas de vulnerabilidades conhecidas para basicamente tudo o que está recebendo de informações em uma porta de rede, inclusive sistemas operacionais, serviços e aplicativos web (MCCLURE; SCAMBRAY; KURTZ, 2014).

2.4.2 *Exploit*

Os atacantes exploram *bugs* ou vulnerabilidades em programas para ter acesso ao sistema alvo. Infelizmente, existem milhares de *bugs*, em 2013, por exemplo, foram reportados 103,000 *bugs* no sistema operacional Ubuntu. Outro projetos de códigos fechados, possuem estatísticas similares (AVGERINOS et al., 2014).

Diante das vulnerabilidades obtidas por um *scanners* (subseção 2.4.1), o passo seguinte seria usar um *exploit* adequado. Os *exploit* são pequenos utilitários usados para explorar vulnerabilidades específicas, podendo ser utilizados de forma *"stand alone"*, ou seja, diretamente, ou podem ser incorporados à *malwares* (NUNES, 2011).

Para alguns *exploits* funcionar, é necessário ter acesso ao *shell* da máquina-alvo. Tal artificio pode ser conseguido através da execução de um cavalo de tróia (subseção 2.4.6) pela vítima em seu sistema. O *trojan* abre uma porta de comunicação e permite que o invasor tenha total controle sobre a máquina, dessa forma é possível executar *exploits* para quebrar outros níveis de segurança (ULBRICH; VALLE, 2007).

2.4.3 Força Bruta

Na segurança da informação, a autenticação é umas das áreas-chaves onde há a distinção de usuários autorizados de outros não-autorizados, tendo como principal vantagem ser de fácil implementação, não requerendo equipamentos, como leitores biométricos (SILVA; STEIN, 2007).

Na literatura sobre segurança da informação, o fator humano é considerado o elo mais fraco. Muitos usuários, por conveniência, criam senhas de acesso fáceis e, em muitos casos,

única para acessar diversos sistemas. Nesse ponto que *hackers* iram atuar para ter acesso não-autorizado ao sistema.

Existem três métodos mais usados por programas de quebra de senha: ataques de dicionário (ou lista de palavras), ataques híbridos e ataques de força-bruta. Nos ataques por dicionários, utilizam-se listas de palavras comuns: nomes próprios, marcas conhecidas, gírias, nomes de canções, entre outros, tais elementos conseguidos por engenharia social (ULBRICH; VALLE, 2007).

Um ataque de força bruta consiste em gerar todas as permutações e combinações possíveis de senha, criptografar cada uma e comparar a senha gerada com a senha criptografada original até encontrar uma que seja igual (SCHARDONG; ÁVILA, 2012).

Esse tipo de ataque é facilmente detectável pois, além de gerar uma alta carga no servidor, gera uma grande quantidade de registros de logs. No entanto, caso a pessoa má intencionada, de alguma outra forma, tenha acesso ao arquivo de *hash* ou a tabela de usuário de um banco de dados, com as senha criptografadas do sistema, ela pode usar o ataque de força bruta no arquivo em qualquer máquina, assim, impossibilitando a detecção do ataque.

Muitos sistemas já possuem formas de contornar esse tipo de ataque, por exemplo, bloqueio de usuário ao errar a palavra-chave por uma certa quantidade de vezes. Outra forma, é colocar um tempo de expiração da senha, por exemplo, a senha deve ser trocada a cada trinta dias por uma diferente e nunca usada anteriormente, dessa maneira, inviabilizando a quebra de senha por força bruta.

2.4.4 Desfiguração de páginas

A desfiguração de páginas, *defacement* ou pichação ocorre quando o conteúdo da página *web* de um site é alterado. O atacante (*defacer*) consegue fazer alterações em páginas explorando vulnerabilidade nas aplicações *web* que permite injeção de *script* malicioso ou através de furto de senha de acesso à interface *web* usadas para administração remota (INTERNET, 2017a).

Nos serviços *web*, como por exemplo, apache2 existe um usuário especial, comumente chamado de *www-data* ou algo semelhante. O usuário *www-data*, na maioria das vezes, precisa apenas de permissões de leitura nos arquivos porém muitos gerentes de sistemas cujo a conscientização sobre segurança é insuficiente, designa permissões errôneas (escrita ou alteração), e caso haja um comprometimento, através, por exemplo, de injeção de código remoto PHP, do servidor, o atacante poderá alterar a maioria dos arquivos. A ocorrência amplamente disseminada de ataques de desfiguração de páginas Web é uma consequência direta dessa prática (STALLINGS; BROWN, 2014).

Esse tipo de ataque pode trazer sérias consequências à instituição, entre elas (CERON, 2015):

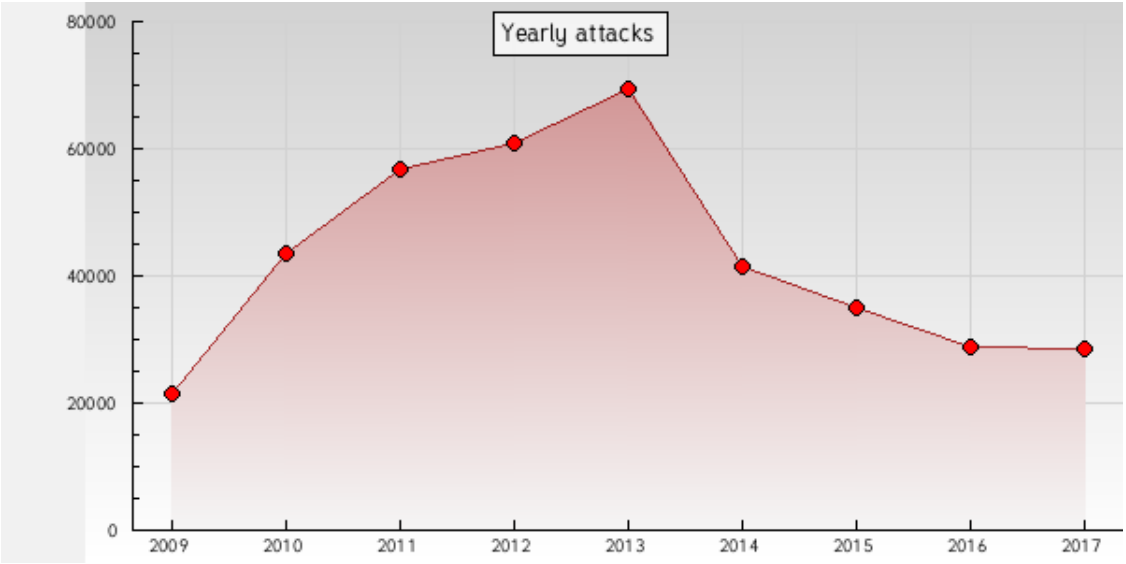
a) **Constrangimento**: A instituição pode ter a imagem de confiabilidade afetada, em certos casos, refletir o descaso com que as informações críticas são tratadas;

b) **Disseminação de inverdades**: Algumas alterações no *website*, por exemplo, alterações de preços de produtos, podem resultar em consequências negativas;

c) **Prejuízo de serviços**: Pode indisponibilizar serviços prestados pela instituição, por exemplo, em *e-commence*.

Existem ferramentas que automatizam esse tipo de ataque, elas identificação aplicações web populares vulneráveis, de modo explorar falhas de segurança e alterar o conteúdo da página (CERON, 2015).

Figura 8 – Estatísticas de *defacement*



Fonte: (ZONE-H, 2017)

O site Zone-H mantém um arquivo de páginas alteradas. Os próprios *hackers* submetem os *websites* comprometidos no intuito de ter seus minutos fama. Nas submissões, os sites são espelhados para o Zone-H, então os moderadores verificam a veracidade do *defacement*. Em 2013, foram identificados cerca de 70000 páginas comprometidas, desde então houve uma redução nesse número (Figura 8). Esse tipo de ataque é considerado passivo pois é gerado somente uma mensagem na tela (NUNAN, 2012).

2.4.5 Negação de Serviços

Um ataque de negação de serviço (*Denial of Service* - DoS) tem como principal objetivo deixar um serviço (servidor *web*, banco de dados) ou recurso (memória, processador) indisponível, impossibilitando que usuário legítimos tenham acesso a esses recursos. Para tal, o atacante gera diversas requisições inúteis para o servidor, consumindo seus recursos até que o serviço não esteja mais disponível ou degradando a qualidade do serviço (STALLINGS, 2011).

Pode-se dividir os ataques de Dos em três categorias (KUROSE; ROSS, 2013):

a) **Ataque de vulnerabilidade**: Envolve o envio de um série de mensagens a uma aplicação ou sistema operacional vulnerável, como consequência o serviço pode parar ou, no pior caso, o hospedeiro pode pifar;

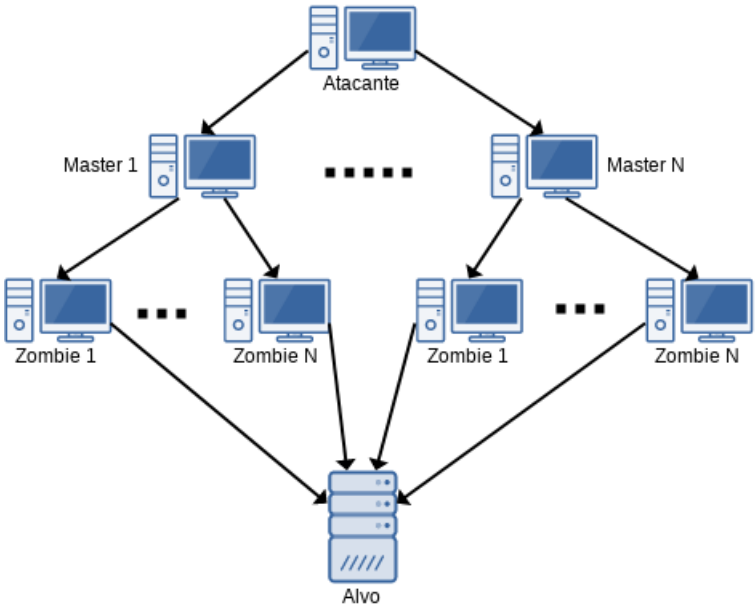
b) **Inundação na largura de banda**: O atacante envia um grande quantidade de pacotes ao hospedeiro, fazendo com que o enlace de acesso do alvo fique indisponível, impedindo os pacotes legítimos de alcançarem o servidor;

c) **Inundação na conexão**: O atacante estabelece um grande número de conexões no hospedeiro-alvo fazendo-o deixar de aceitar conexões legítimas.

Esse tipo de ataque pode gerar grandes prejuízos financeiros para as empresas, principalmente *e-commence*, pois enquanto o sistema está fora ou com uma resposta lenta, as transações financeiras são prejudicadas. Com isso, cria-se também, uma insatisfação pelo usuário do serviço prestado pela empresa.

Existe uma forma mais sofisticada de ataque de DoS chamada Negação de Serviço Distribuído (*Distributed Denial of Services* - DDoS), enquanto o DoS básico as requisições partem de apenas uma fonte, no entanto, no DDoS o atacante tem acesso a um grande número de computadores (*zombies*) explorando suas vulnerabilidades criando o que chamamos de *botnet* (Figura 2.4.5). Com isso, basta o atacante indicar as coordenadas de um ou mais alvos para o ataque (ZARGAR; JOSHI; TIPPER, 2013). O DDoS são mais difíceis de detectar e de prevenir do que um ataque DoS de um único hospedeiro.

Figura 9 – Ataque de Negação de Serviço Distribuído



Fonte: Autoria própria

2.4.6 *Malwares*

Os *malwares*, também conhecidos como *softwares* maliciosos, são um grande problema para sistemas de informação, sua existência ou execução tem consequências negativas ou involuntárias. Nessa seção será apresentado os *malwares* mais popularmente conhecidos que são os vírus, *worms*, *trojans* e, devido sua repercussão, os *ransomwares*.

É importante entender o funcionamento e o comportamento desses códigos maliciosos para, a partir daí, buscar soluções contra esse ataque. Existe dois tipos de análise: análise estática, requer uma verifica linha a linha do código malicioso, geralmente o código não está disponível e até mesmo se estiver, o autor do *malware* muitas vezes ofusca o código, tornando esse tipo de análise difícil. Por outro lado, existe a análise dinâmica, o analista monitora a execução e o comportamento do *malware*, esse tipo de análise é imune a ofuscação de código (TILBORG; JAJODIA, 2011).

O Vírus é um programa que se propaga inserindo cópias de si mesmo e se tornando parte de outros programas e arquivos. Para dar continuidade ao processo de infecção, o vírus depende da execução do programa ou arquivo hospedeiro. O principal meio de propagação desse tipo de *software* malicioso são as mídias removíveis, como, por exemplo, *pen-drives* (INTERNET, 2017b).

O *Worm* é um *malware* que se propaga através de e-mails, sites ou *software* baseados em rede, explorando as vulnerabilidades das aplicações. Uma das principais características desse tipo de *software* é a propagação automática, ou seja, sem a intervenção do usuário (JAIN; PAL, 2017).

O *Trojan* ou Cavalo de Troia são programas que precisam ser explicitamente executados para serem instalados no computador. Esse *malware* se disfarça de um programa benigno, por exemplo, cartões virtuais animados, álbuns de fotos, jogos e protetores de tela que ao serem executados o *trojan* é instalado sem o consentimento do usuário. No entanto, o atacante, após invadir um computador, pode instalar o *trojan* alterando as funções já existentes de programas para executarem ações maliciosas (INTERNET, 2017b).

Por fim, temos os *ransomwares*. O *ransomware* é um *malware* que criptografa os dados de um computador ou uma rede. A pessoa ou a organização responsável pelo ataque pede um resgate, geralmente pago em cripto moedas, como por exemplo, bitcoin, para manter sua anonimidade, fornecendo uma chave para descriptografar os arquivos mediante o pagamento (ENIS, 2017).

A melhor medida contra esse tipo de *malware*, uma vez que, não há garantias que o atacante irá fornecer a chave depois do pagamento, além de manter o sistema sempre atualizado, é ter uma politica de *backup* regular. O armazenamento de arquivos importantes em outros tipos de mídias não conectadas regularmente ao sistema (removíveis) ou *backup* baseados em nuvens (ENIS, 2017).

2.5 Ferramentas para Avaliação de Segurança

Nessa seção será descrito as ferramentas auxiliares utilizadas para geração de ataques abordados na seção 2.4 com objetivo de testar e validar as configurações das ferramentas de IDPS estudas.

2.5.1 Nmap

O Nmap é uma ferramente de código aberto utilizada para auditoria de segurança e descoberta de rede. A ferramenta é capaz de determinar quais *hosts* estão disponíveis na rede, quais serviços cada *host* está oferecendo, incluindo nome e versão da aplicação, o sistema operacional usado, dentre outras características.

Muitos administradores de sistemas utilizam o Nmap para tarefas rotineiras como, criação de inventário de rede, gerenciamento de serviços, visto que é de suma importância manter os mesmos atualizados e monitoramento de *host*.

Diversos parâmetros podem ser utilizados com o Nmap, possibilitando realizar varreduras das mais variadas maneiras, dependendo do tipo desejado. A lista completa de opções podem ser consultadas na documentação oficial que vem junto da ferramenta ou no site do projeto (NMAP, 2017).

Na execução do Nmap, o que não for opção ou argumento da opção é considerado especificação do *host* alvo. O alvo pode ser um ou vários, usando uma notação de intervalo por hífen ou uma lista separada por vírgula. Os *hosts* alvos também podem ser definidos em arquivos.

O resultado do Nmap é uma tabela de portas e seus estados (Figura 10). As portas podem assumir quatro estados, temos (NMAP, 2017):

a) ***open***: significa que existe alguma aplicação escutando conexões;

b) ***filtered***: há um obstáculo na rede, podendo ser algum *firewall*, que impossibilita que o Nmap determine se a porta está aberta ou fechada;

c) ***closed***: não possui aplicação escutando na porta;

d) ***unfilterd***: a porta responde requisição porém o Nmap não consegue determinar se estão fechadas ou abertas.

2.5.2 Metasploit Framework

O Metasploit é um *framework* de código aberto cujo principio básico é desenvolver e executar *exploit* contra alvos remotos e fornecer uma lista de vulnerabilidades existentes no alvo. É uma ferramenta que combina diversos *exploits* e *payloads* dentro de um local, ideal para levantamento de segurança de serviços e testes de penetração (ARYA et al., 2016).

Figura 10 – Exemplo de saída do Nmap

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-12 10:30 -03
Nmap scan report for portal.ufpa.br (200.239.64.160)
Host is up (0.00041s latency).
Other addresses for portal.ufpa.br (not scanned): 2801:80:240:8000::5e31:160
rDNS record for 200.239.64.160: marahu.ufpa.br
Not shown: 94 filtered ports
PORT     STATE  SERVICE
21/tcp   open   ftp
22/tcp   open   ssh
80/tcp   open   http
81/tcp   open   hosts2-ns
443/tcp  open   https
3000/tcp closed ppp

Nmap done: 1 IP address (1 host up) scanned in 1.65 seconds
```

Autoria própria

O Metasploit possui uma biblioteca divida em três partes:

a) **Rex**: É a biblioteca fundamental, a maioria das tarefas executadas pelo *framework* usarão essa biblioteca;

b) **MSF Core**: É o *framework* em si, possui, por exemplo, gerenciador de módulos e a base de dados;

c) **MSF Base**: Guarda os módulos, sejam eles, *exploit*, *encoders* (ferramentas usadas para desenvolver o *payloads*) e os *payloads*. Além disso, são guardadas informações de configuração e sessões criadas pelos *exploits*.

A Interface permite que o usuário interaja com o *framework*. Nele há o *msfconsole* uma interface de linha de comando interativa, o *msfcli* interface de linha de comando não-interativa, e o *msfweb* interface baseada em *web* (MAYNOR et al., 2007). Por fim, temos o Armitage, que é uma interface gráfica baseada em Java desenvolvido por Raphael Mudge.

A arquitetura é mostrada com mais detalhes na Figura 11.

Os módulos são divididos da seguinte maneira:

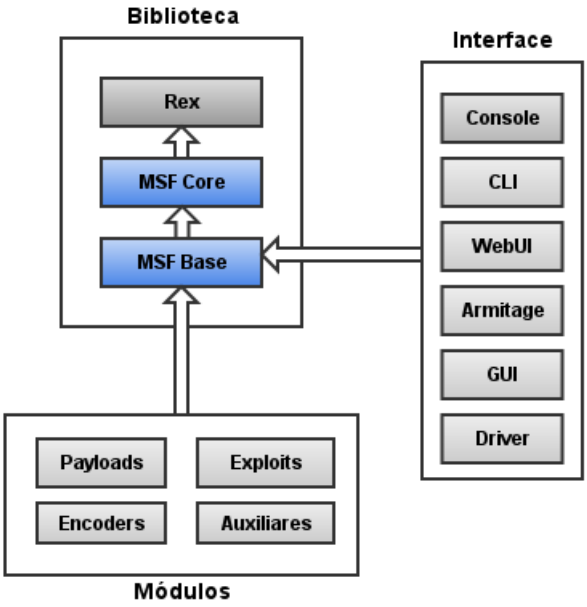
a) **Payload**: são código executados no alvo remotamente;

b) **Exploit**: explora *bugs* ou vulnerabilidade existente em aplicações do alvo;

c) **Módulos Auxiliares**: usado para escanear as vulnerabilidades e executar várias tarefas;

d) **Encoder**: codifica o *payload* para evitar qualquer tipo de detecção por antivírus.

Figura 11 – Arquitetura do Metasploit



Fonte: Autoria própria

2.5.3 Pytbull

O Pytbull é um *framework* para teste de IDPS, capaz de determinar a capacidade de detecção e bloqueio do mesmo, além de fazer uma comparação entre diversas soluções e verifica as configurações (DAMAYE, 2016). O *framework* Pytbull possui cerca de 300 testes agrupados em 11 módulos, temos:

a) **badTraffic**: pacotes não compatíveis com a RFC são enviados para o servidor para testar como os pacotes são processados;

b) **bruteForce**: testa a capacidade do IDPS de rastrear ataques de força bruta;

c) **clientSideAttacks**: usa um *shell* reverso para fornecer ao servidor instruções para baixar arquivos maliciosos;

d) **denialOfService**: testa a capacidade do IDPS de proteger contra tentativas de DoS;

e) **evasionTechniques**: testa a capacidade do IDPS de detectar técnicas de evasão;

f) **fragmentedPackets**: várias cargas úteis fragmentadas são enviadas ao servidor para testar sua capacidade de recomposição e detectar os ataques;

g) **ipReputation**: testa a capacidade do servidor detectar tráfego de servidores com reputação baixa;

h) **normalUsage**: cargas úteis que correspondem a uso normal;

i) **pcapReplay**: permite reproduzir arquivos pcap;

j) **shellCodes**: envia *shellcodes* para o servidor na porta 21/ftp testando a capacidade de detectar e/ou bloquear o mesmo;

k) **testRules**, testa a base de assinaturas configuradas no servidor IDPS.

Existem basicamente 5 tipos de testes (DAMAYE, 2016):

a) ***socket***: Abre um *socket* em uma porta e envia o *payload* para o alvo remoto na porta especificada;

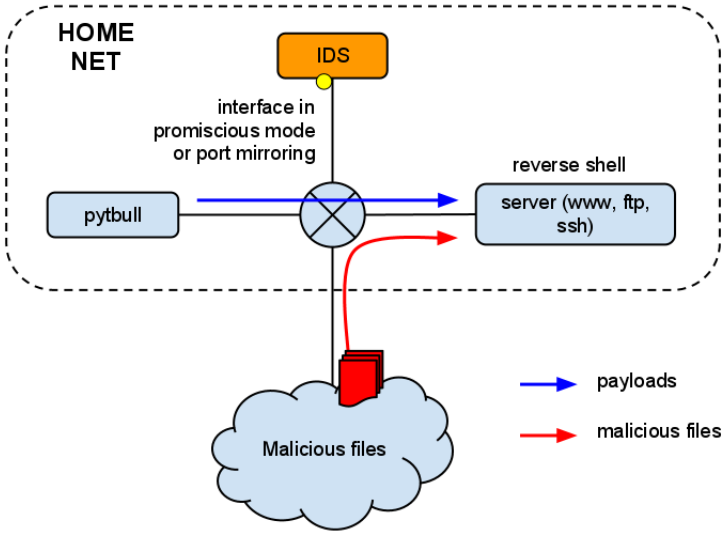
b) ***command***: Envia um comando para alvo remoto com a função python subprocess.call();

c) ***scapy***: Envia cargas úteis especificas baseadas na sintaxe de Scapy;

d) ***client side attacks***: Usa um *shell* reverso no alvo remoto e envia comandos para serem processados no servidor;

e) ***pcap replay***: Permite reproduzir tráfego com base em arquivos de pcap.

Figura 12 – Arquitetura do *framework* Pytbull



Fonte: (DAMAYE, 2016)

2.6 Conclusão

Este capítulo apresentou definições sobre segurança da informação, mostrando os elementos envolvidos. Definindo os tipos de ataques existentes e suas categorias e os possíveis impactos. Mostrou-se um cenário geral de uma rede de computador e seus componentes assim como suas definições e os ataques comuns envolvendo essas redes. Ao final do capítulo, mostrou-se as ferramentas auxiliares usadas para simular ataques com objetivo de analisar o comportamento dos IDPS.

3 Sistemas de Detecção e Prevenção de Intrusão

Os sistemas de detecção e prevenção de intrusão (*Intrusion Detection and Prevention System* - IDS/IPS) são ferramentas de importância reconhecida pela comunidade da segurança da informação. Nesse capítulo, vamos apresentar os principais conceitos relacionados a IDS e IPS, uma breve descrição do funcionamento e classificação, para melhor entendimento das ferramentas que iremos apresentar e avaliar em um ambiente de real.

3.1 Definições de IDS/IPS

Intrusion Detection Systems (IDS) ou Sistemas de Detecção de Intrusão (SDI) são ferramentas utilizadas para monitoramento de eventos que ocorrem em redes e sistemas computacionais, analisando sinais de possíveis ataques que podem levar a uma violação das politicas de segurança da organização, alertando os administradores do sistema que estes eventos estão ocorrendo.

O *Intrusion Detection Systems* (IPS) ou Sistema de Prevenção de Intrusão (SPI) possui todas as funcionalidades do IDS com uma diferença, ele é capaz de deter os incidentes, minimizando os impactos causados por sistemas comprometidos (MUKHOPADHYAY; CHAKRABORTY; CHAKRABARTI, 2011).

Os IDS's são compostos basicamente por quatro componentes, temos:

a) **Sensor ou Agente**: responsável pelo monitoramento e analise do trafego capturado;

b) **Base de Dados**: usado como repositório das informações de eventos detectados pelo sensor e que posteriormente serão processados;

c) **Gestor**: é o dispositivo central que recebe, analisa e gerencia as informações de eventos vindo do sensor;

d) **Console**: é uma interface para administração e monitoramento das atividades.

3.2 Tipos de Sistemas de Detecção e Prevenção de Intrusão

Os IDPS's são classificados de acordo com o local onde o sensor é instalado, *Host Based Intrusion Detection Systems* (HIDS) e *Network Based Intrusion Detection Systems* (NIDS), e a técnica utilizada para o monitoramento, baseado em assinaturas e anomalias (NAGAHAMA, 2013).

3.2.1 Sistemas de Detecção de Intrusão Baseados em Host (HIDS)

Em um HIDS o sensor é instalado no *host*, monitorando as informações contidas na própria máquina. Esse tipo de IDS não observa o tráfego que passa pela rede (somente o trafego que passa pela placa de rede do *host*), seu uso volta-se a verificação de informações relativas aos eventos e registros de logs e sistemas de arquivos (permissão, alteração, acesso a arquivos não autorizados) (NAGAHAMA, 2013).

As vantagens do HIDS são:

a) Evita a execução de códigos maliciosos;

b) Bloqueia tráfego de entrada e saída contendo ataques e uso não autorizado de protocolos e programas;

c) Evita que arquivos possam ser acessados, modificados e deletados impedindo a instalação de *malwares* e ataques envolvendo acesso inapropriado a arquivos;

Por outro lado, o HID possui alguns desvantagens como (SCARFONE; MELL, 2007):

a) Difícil instalação e manutenção;

b) Interfere no desempenho do *hosts*;

c) Demora para identificar eventos consequentemente a resposta ao incidente terá um atraso.

3.2.2 Sistemas de Detecção de Intrusão Baseados em Rede (NIDS)

No NIDS, o sensor é instalado na rede e a interface de rede atua em um modo especial chamado "promíscuo", tendo a capacidade de capturar o tráfego mesmo que os pacotes não sejam destinados ao sensor. Dessa forma, o NIDS monitora e analisa todo o trafego no segmento da rede, detectando atividades maliciosas, como ataques baseados em serviço, *portscans*, entre outros, além de detectar se algum usuário legítimo está fazendo mau uso da rede (NAGAHAMA, 2013).

Quanto a localização o NIDS pode ser classificado como passivo ou ativo. No modo passivo (Figura 14), o IDS monitora copias dos pacotes da rede que passam pelo *switch* ou *hub* onde está conectado, ficando limitado somente a gerar notificações quando encontrado algum tráfego malicioso.

No entanto, no modo ativo (Figura 13), o IDS é instalado da forma que o tráfego da rede passe através do sensor parecendo com o fluxo de dados associado com um *firewall*. Dessa forma, ele é capaz de parar ataques bloqueando o fluxo malicioso.

É necessário uma analise minuciosa na instalação de um IDS ativo pois um mal dimensionamento de *hardware* pode degradar a rede, adicionando atrasos excessivos aos pacotes.

As principais vantagens do um NIDS são:

a) São independentes de plataformas;

b) Não interfere no desempenho do *host*;

c) Fácil implantação e transparente para o atacante.

Dentre as desvantagens, temos:

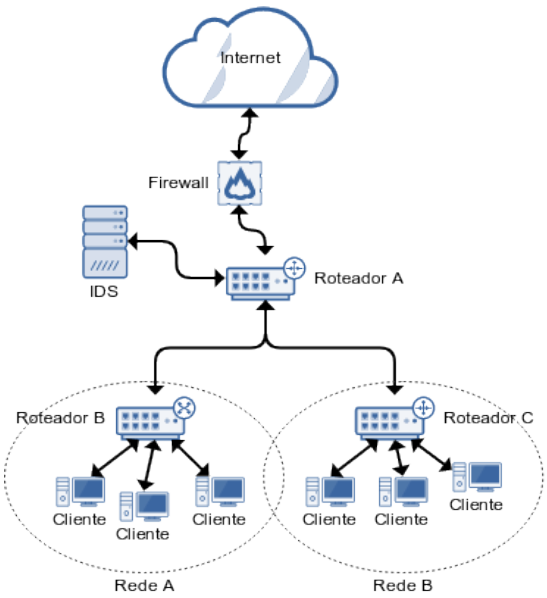
a) Pode adicionar retardados aos pacotes quando instalado no modo ativo;

b) Dificuldade de tratar dados de redes de alta velocidade;

c) Trata apenas segmentos de rede;

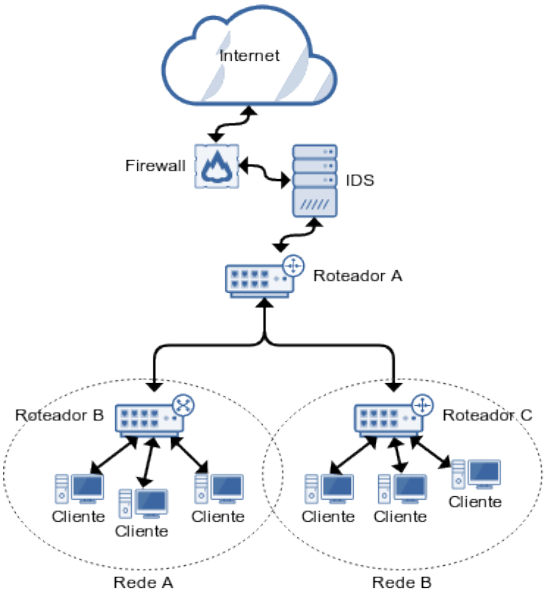
d) Dificuldade de tratar dados criptografados.

Figura 13 – Exemplo de arquitetura de NIDS passivo



Fonte: Autoria própria

Figura 14 – Exemplo de Arquitetura de NIDS ativo



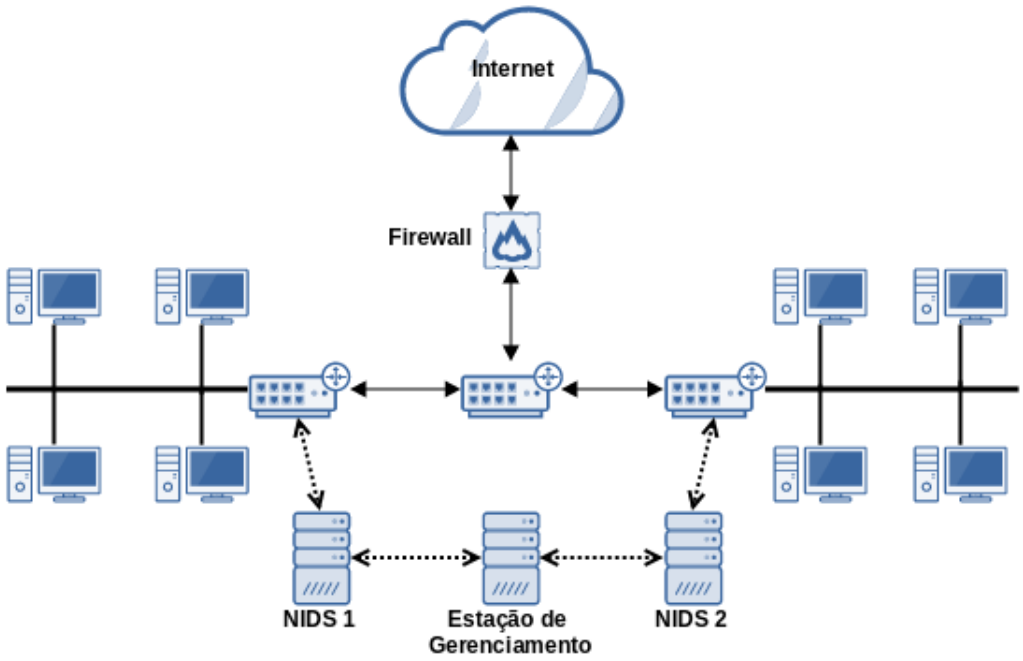
Fonte: Autoria própria

3.2.3 Sistema de Detecção de Intrusão Distribuídos

A função de um Sistema de Detecção de Intrusão Distribuído (SDID) é de gerencia. Os sensores (pode ser NIDS, HIDS ou a combinação de ambos), localizados remotamente, reportam os alertas para um centralizador. Os *logs* de ataques são, periodicamente, enviados para a estação de gerenciamento, armazenando em uma base única e centralizada, além disso, novas assinaturas de ataques podem ser enviadas para os sensores (BAKER; CASWELL; BEALE, 2007).

Na Figura 15 mostra um SDID composto por dois sensores e um estação de gerenciamento centralizado. O sensor NIDS 1 e NIDS 2 estão operando em modo *promiscuos* e está

Figura 15 – Sistema de Detecção de Intrusão Distribuído



Fonte: Autoria própria

protegendo segmentos de rede. É recomendando que a conexão entre os sensores e o centralizado seja feita por uma rede privada, em caso de utilização de rede públicas, recomenda-se adicionar uma camada de segurança, como criptografia, ou VPN.

3.2.4 Formas de Detecção

Quanto a técnica de monitoramento utilizado, o IDS pode ser baseados em assinaturas ou anomalias. IDSs baseados em assinaturas compara os pacotes com uma base de assinaturas de ataques previamente conhecidos e reportados por especialistas, cada assinatura identifica um ataque (NAGAHAMA, 2013).

As vantagens de um IDS baseados em assinaturas são:

a) Usa pouco recurso de hardware do servidor;

b) Possui, de certa forma, um rápido processamento.

Dentre as desvantagens temos:

a) Exige uma atualização constante da base de assinaturas;

b) Para a geração de uma base própria, a equipe precisa de um alto conhecimento técnico;

c) Possui altos índices de falsos positivos e negativos.

Os IDS baseados em anomalias, procuram determinar um comportamento normal na fase de aprendizagem do sistema computacional ou rede e sempre que existir um desvio desse padrão alertas são gerados.

Possui a vantagem de detectar novos ataques sem necessariamente conhecer a fundo a intrusão através dos desvios de comportamento. Porém, tem como desvantagem a geração de um grande número de falsos alertas em decorrência a modificações na rede ou *host* nem sempre representar um tráfego malicioso.

3.3 Principais Ferramentas de IDS

Nesse capitulo, será apresentado as ferramentas de IDPS analisadas. A escolha dessas ferramentas deu-se devido ser de código aberto e de livre uso, e também, pela sua popularidade diante da comunidade de segurança da informação.

3.3.1 Snort

O Snort é um sistema de detecção e prevenção de intrusão de código fonte aberto escrita na linguagem de programação C bem conhecido pela comunidade da segurança da informação. Seu primeiro *release* foi lançado em 1998 e desde então passa por constantes revisões e aperfeiçoamentos, com o passar dos anos se tornou o IDS mais utilizado no mundo. Ele combina análise baseada em assinaturas e anomalias, podendo operar em três modos: *sniffer*, *packet logger* e de sistema de detecção de intrusão (NIDS) (ROESCH; GREEN, 2017).

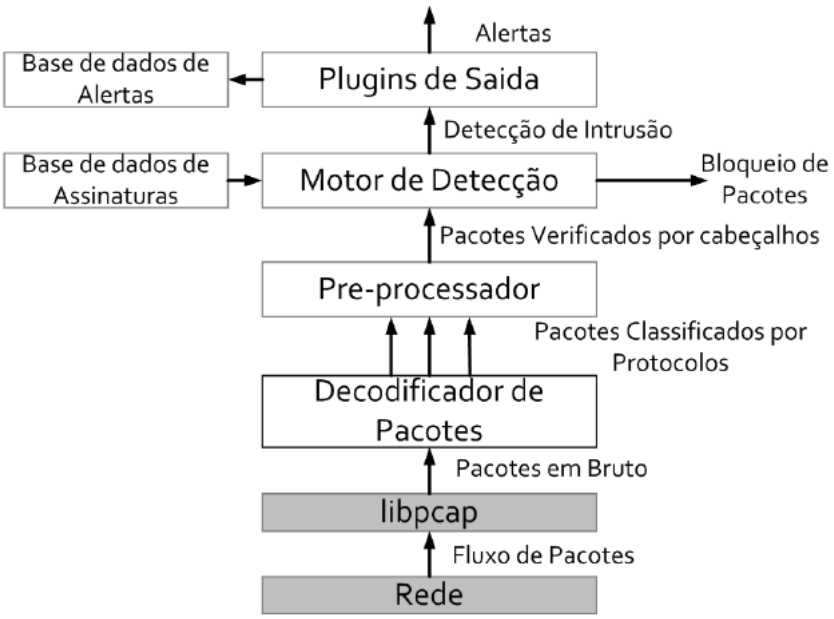
No modo *Sniffer*, o Snort captura os pacotes e exibi as informações no console de forma continua. No modo *Packet Logger*, além de capturar o tráfego, o Snort escreve essas informações em arquivos (chamados de logs) que são armazenados no disco. Por fim, o *Network Intrusion Detection System* - NIDS, sendo o modo mais complexo e completo, permitindo capturar e analisar os pacotes de rede em tempo real (ROESCH; GREEN, 2017).

Existe quatro componentes no Snort: O *sniffer*, o pré-processador, o motor de detecção e módulos de saída. A Figura 16 mostra a arquitetura e disposição dos componentes (BAKER; CASWELL; BEALE, 2007).

O pré-processador, o motor de detecção e os componentes de alerta do Snort são todos *plugins*. Os *Plugins* são programas escritos em conformidade com a API de *plugins* do Snort. Esses programas são usados no core do Snort, mas eles são separados para que as modificações feitas no *core* sejam mais confiáveis e mais fáceis de realizar (BAKER; CASWELL; BEALE, 2007).

O *sniffer* é um dispositivo (*software* ou *hardware*) usado para ver o trafego passante em algum segmento de rede. No caso da Internet, consiste geralmente de trafico IP (composto por

Figura 16 – Arquitetura do Snort



Fonte: (LOPEZ, 2014)

diferentes protocolos de alto nível como, TCP, UDP, ICMP, protocolos de roteamento e IPSec). Os pacotes são analisado, interpretados e exibidos de uma forma legível para os humanos.

Um *sniffer* tem os seguintes usos:

a) Analisador de rede e resolução de problemas;

b) Analisador de performance e avaliação comparativa;

c) Capturar senhas em texto plano e outros dados sensíveis.

Assim como qualquer outra ferramenta de rede, os *sniffers* podem ser usados tanto para o bem quanto para o mal. Então, criptografar o trafego de rede previne que pessoas sejam capazes de lerem os pacotes capturados (BAKER; CASWELL; BEALE, 2007).

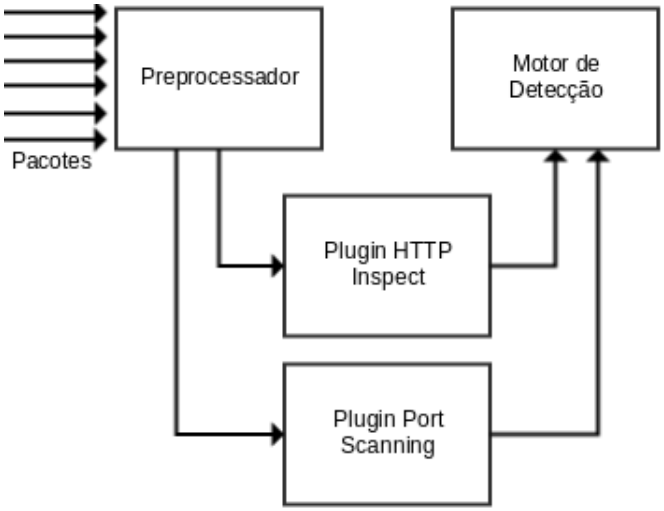
O pré-processador pega o pacote bruto e faz uma checagem utilizando um determinado *plugin*. Esses *plugins* verificam se o pacote tem um tipo particular de comportamento, uma vez determinado, o pacote é enviado para o motor de detecção caso contrário é descartado.

Na Figura 17, pode-se ver como o pré-processador utiliza *plugins* para chegar pacotes. O Snort suporta muitos tipos de pré-processadores, cobrindo vários protocolos comumente usados como, IP *fragmentation handling*, *port scanning* e controle de fluxo.

O uso de *plugins* é uma característica muito útil para o IDS, pois os *plugins* podem ser ativados e desativadas a medida do necessário, otimizando a utilização dos recursos computacionais e geração de alertas (BAKER; CASWELL; BEALE, 2007).

Os pacotes, após passarem por todos os pré-processador, são entregues para o motor

Figura 17 – Uso de *plugins* no pré-processador



Fonte: Autoria própria

de detecção. O motor de detecção pega esses dados e faz uma checagem utilizando uma base de regras pré-configurado pelo administrador. Se a regra for compatível com os dados do pacote, eles são enviado para o processador de alertas, caso contrário, são descartados (BAKER; CASWELL; BEALE, 2007).

Na Figura 18, temos os pacotes saindo dos pré-processadores e chegando no motor de detecção. No motor de detecção há uma base de regras configurada, os dados dos pacotes são comparados com as assinaturas da base, se coincidirem, uma ação é tomada, caso contrário, o pacote é descartado.

A base de regras é um conjunto de assinaturas de ataques conhecidos e catalogados. As regras são escritas em formato texto em uma única linha e constituídas por duas partes:

a) **Cabeçalho**: São definidos que ações serão tomadas, tipo de pacote (TCP, UDP, ICMP, etc), o IP de origem e destino e suas respectivas portas;

b) **Opções**: É o conteúdo do pacote que faz ele ser compatível com a regra.

Dentre as ações que podem ser tomadas temos:

a) ***Activation***: Alerta e chama regra do tipo *dynamic*;

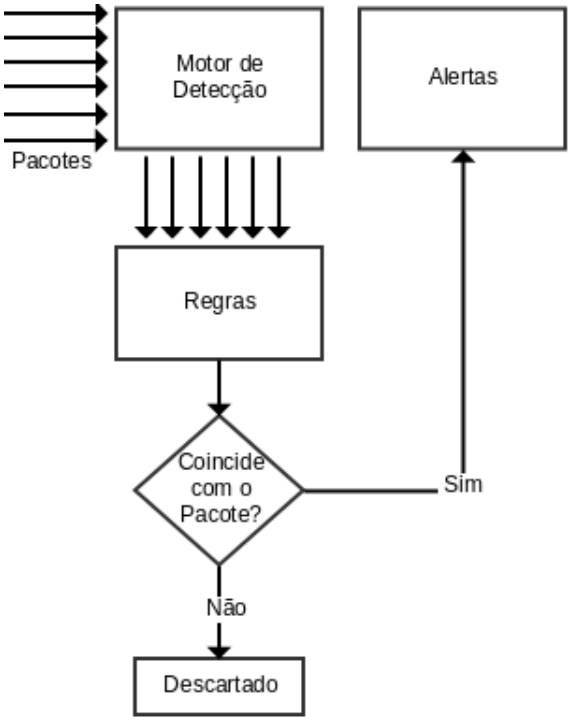
b) ***Dynamic***: permanece inativa até ser ativado por uma regra *activate*, registrando o tráfego;

c) ***Alert***: Gera um alerta usando um método selecionado e então registra os pacotes e dados;

d) ***Pass***: Ignora os pacotes;

e) ***Drop***: Descarta o pacote (quando configurado para atuar de forma ativa (IPS));

Figura 18 – Motor de Detecção do Snort



Fonte: Autoria própria

f) ***Log***: Registra e não alerta.

Abaixo temos um exemplo de regra.

```
alert icmp any any -> any any (msg:"Ping suspeito";
sid:1; resp:icmp_all;)
```

Com a regra acima, o Snort gerará um alerta de qualquer pacotes ICMP que estiver passando de qualquer máquina e porta origem (**any any**) para qualquer máquina e porta destino (**any any**) e enviará pacotes ICMP para a máquina de origem com as mensagens *host unreachable; network unreachable*.

Se um dado for compatível com uma regra um alerta é gerado. Os alertas podem ser enviados para arquivos de *logs*, através da rede, através de *sockets* UNIX ou Windows Popup (SMB). Os alertas também podem ser armazenados em banco de dados SQL como MySQL e Postgres. Existem vários *plugins* para Perl, PHP e servidores Web para exibir os *logs* através de um interface Web (BAKER; CASWELL; BEALE, 2007).

3.3.2 Suricata

Suricata é um NIDS de código aberto, seu primeiro *release* oficial foi lançando em 2010 e foi desenvolvido e atualmente é mantido pela *Open Information Security Foundation*

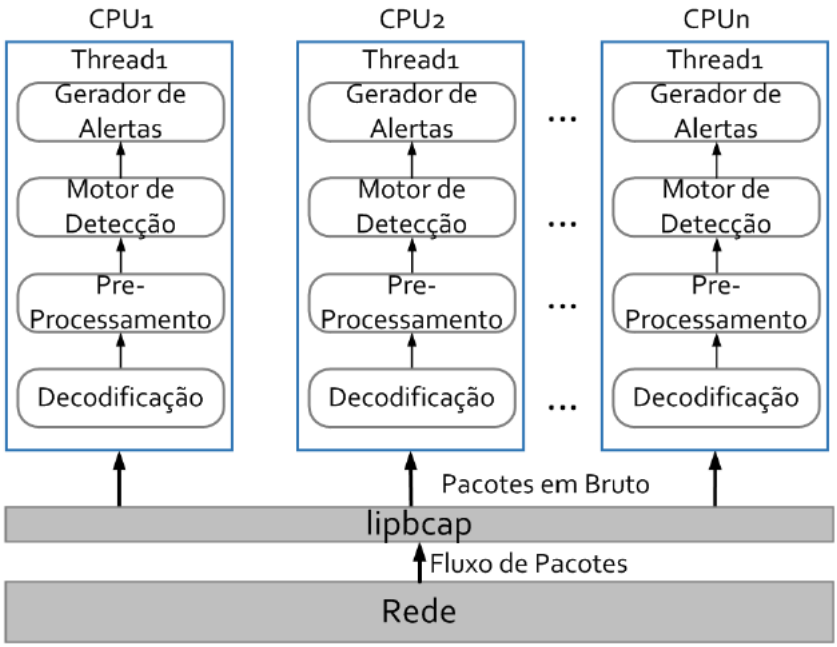
(OISF). A OISF é uma fundação sem fins lucrativos formada por um grupo multinacional de desenvolvedores (OISF, 2017a).

Uma característica marcante desse IDS é a utilização de uma tecnologia de processamento *multithread* para ter benefício dos múltiplos núcleos de um computador. Além de utilizar *hardware* de aceleração para ter um melhor desempenho (LOPEZ, 2014).

O Suricata utilizada detecção baseada em assinatura e em anomalias. As assinaturas desenhadas para o Snort funcionam no Suricata, podendo ser otimizadas para o uso em seu motor de detecção. As anomalias dos protocolos são fornecidas pelos pré-processadores, e quando implementado no modo ativo, atua na modalidade de prevenção (LOPEZ, 2014).

Embora o código do Suricata ser original, os desenvolvedores não hesitam afirmar que a arquitetura foi inspirada no Snort. Na Figura 19 representa a mesma arquitetura do Snort porém com o mecanismo de *multithread* (LOPEZ, 2014). Dessa forma não há necessidade de descrever os componentes pois já foram citados na subseção 3.3.1.

Figura 19 – Arquitetura *Multithread* do Suricata



Fonte: (LOPEZ, 2014)

As assinaturas são de grande importância tanto no Snort quanto no Suricata. Muitas pessoas, por conveniência, utilizam conjunto de regras prontas. As mais usadas são da Emerging Threats (ET) e Talos (anteriormente chamado de VRT) (OISF, 2017b).

Talos é um grupo de especialistas em segurança de rede trabalhando o tempo todo para descobrir, avaliar e responder de forma proativa as últimas tendências em atividades de *hacking*, tentativa de intrusão, *malware* e vulnerabilidades (TALOS, 2017).

A base de assinaturas ET é mantida pela Proofpoint. A Proofpoint é uma empresa especialista em segurança da informação, no site oficial há vários produtos que visam proteger pessoas e dados, detectando e bloqueando ataques e respondendo a essas ameaças (PROOFPOINT, 2017).

O fato de existir equipes dedicadas para o desenvolvimento de uma base de regras, torna o uso dessas bases confiáveis, menos custoso, tem termos de tempo, recursos financeiros e de pessoal e fácil implementação.

3.4 Conclusão

Este capítulo apresentou definições sobre IDPS e seus componentes, os tipos existentes e a forma de detecção utilizada. Também foi descrito as ferramentas avaliadas nesse trabalho (Snort e Suricata), destacando suas diferenças e arquiteturas.

4 Detecção de Intrusão em um Cenário Real

Este capítulo esta organizado da seguinte forma: A próxima seção apresenta o cenário de testes, descrevendo características gerais da rede selecionada para os teste. Na seção 4.1.2 será abordado a infraestrutura usada para os testes, ferramentas utilizadas e as configurações feitas. Na seção 4.2 será descrito os testes realizados com suas respectivas justificativas. Na seção 4.3 será apresentado os resultados esperados e obtidos, problemas encontrados e a comparação das ferramentas e por último, na seção 4.4, uma breve conclusão.

4.1 Metodologia dos Testes

4.1.1 Cenário de Testes

A rede selecionada para ser monitorada tem os valores especificados na tabela. Podemos verificar que em um determinado período do dia o pico de trafego chega a 107,25 Mbps, valores considerados ideais para o experimento, inclusive para tentar validar os recursos alocados.

Em um primeiro momento, selecionou-se uma rede

4.1.2 Infraestrutura Definida para Testes

No ambiente de teste foi usado uma máquina Dell com 134 Megabytes (MB) de memória RAM e 40 núcleos. Usou-se XenServer (XENSERVER, 2017) versão 7, sistema operacional (SO) *opensource* da Citrix voltado para virtualização. Foram testados outros SOs porém somente o XenServer possuía, na época da instalação do ambiente, *firmware* da placa de rede do *host* compatível e que funcionava com instabilidade.

Outro fator que pesou na escolha do SO foi a experiência com a plataforma e por existir uma interface para gerencia chamada XenCenter que roda no Windows. Uma alternativa *opensource* desse software é o OpenXenManager que funciona nos sistemas Unix (LINTOTT, 2017).

No primeiro momento, foi instalado uma máquina virtual com o sistema operacional Debian 9.3 *codename* Wheezy (DEBIAN, 2017), uma distribuição linux com uma proposta de ser totalmente livre, usada como base para instalação de outras máquinas utilizando o recurso de *snapshot*, uma cópia de uma máquina virtual rodando em um certo momento, do XenServer. O uso desse recurso foi necessário para criar um ambiente igual para os IDSs.

Foi alocado 8 MB memória RAM, 4 processadores e 100 Gigabytes(GB) de espaço em disco para o *snapshot*. Esses valores foram definidos com base em um estudo (LOCOCO, 2011) que considerava vários fatores, como largura da rede, localização do IDS e versão, tipo

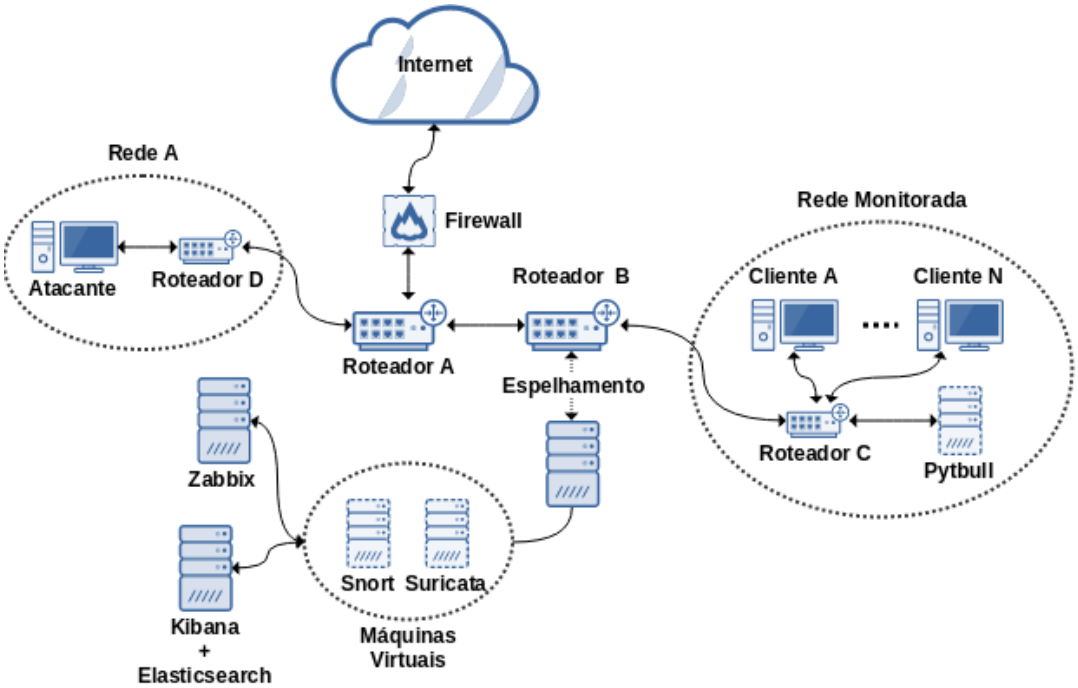
do capturador de tráfego e tamanho da base de assinaturas para dimensionar os recursos de memória e processamento, aplicado especificamente ao Snort. A mesma regra foi aplicada ao Suricata.

Para o *host* conseguir pegar o pacotes destinados a rede escolhida para ser monitorada foi necessário uma configuração de espelhamento no roteador B (Figura 20) que consiste na copia dos pacotes que saem pela porta dessa rede no roteador para a porta conectada no *host* que possui uma largura de banda de 10 Gigabits. A interface de rede do *host* precisou ser configurada no modo *promisc*.

Posteriormente criou-se quatro máquinas virtuais, duas usadas para instalação dos IDSs (Suricata e Snort) e a terceira para instalação das ferramentas usadas para simular ataques a rede. Optou-se pela instalação do sistema Kali Linux (KALI, 2017) para geração de ataques pois nele existe várias ferramentas nativas para testes de penetração e auditoria de segurança (Metasploit subseção 2.5.2 e NMAP subseção 2.5.1).

Na quarta máquina foi instalado o *framework* Pytbull (subseção 2.5.3), ela servirá também como alvo das simulações dos ataques. A infraestrutura final pode ser visualizada na Figura 20.

Figura 20 – Infraestrutura do ambiente de teste



Fonte: Autoria própria

Para coleta das informações de uso de recurso de hardware como memória, processamento e I/O das máquinas com os IDSs foi utilizado o *daemon* Collectd (COLLECTD, 2017). Outra opção para esse finalidade é a utilização de um servidor de monitoramento como o Zabbix

(ZABBIX, 2017). A ideia de ter duas ferramentas para essa analise é fazer um comparativo e validar as informações coletadas.

O formato usado para facilitar a análise do *logs* foi JavaScript Object Notation (JSON), um formato simples, leve e de fácil leitura. O Motor de Saída do Suricata já tem suporte a esse tipo de formato o que não acontece no Snort. Para tal, usou-se o IDSTools (ISH, 2017), uma coleção de bibliotecas na linguagem python que trabalha para auxiliar o IDS Snort. Dentre os utilitários presentes nessa coleção, temos o idstools-u2json, que converte, de forma continua, arquivo no formato unified2, uma das saídas disponível no Snort, para o formato JSON.

Para analisar os *logs*, usou-se uma infraestrutura que combina três ferramentas:

a) **Kibana** (ELASTIC, 2017b): Uma plataforma de análise e visualização desenhada para trabalhar com os índices do Elasticsearch (ELASTIC, 2017a), a grosso modo, podemos dizer que ela é uma interface gráfica para o Elasticsearch.

b) **Elasticsearch**: Um motor de busca e análise altamente escalável, capaz de armazenar, buscar e analisar uma grande quantidade de dados em tempo próximo ao real.

c) **Logstach** (ELASTIC, 2017c): Um motor de coleta de dados em tempo real, unificando os dados de diferentes fontes dinamicamente, normalizando-os nos destinos escolhidos.

Dessa forma centralizou-se os *logs*, facilitando a visualização e analise das ocorrências dos IDSs.

4.2 Testes Realizados

Os testes realizados são simulações de passos que uma pessoa má-intencionada iria tomar para alguma tentativa de invasão, entende-se por invasão, qualquer tipo de violação e alteração não autorizada de um serviço ou *host*.

O passo inicial seria um estudo do alvo utilizando várias técnicas mas principalmente a engenharia social, analisando as pessoas que trabalharam na organização, enviando spam e *phishing* na tentativa de capturar dados como senhas de acesso.

Posteriormente, verificando os serviços que o alvo oferece e observando (*sniffando*) a rede, a procura de alguma senha desprotegida (não criptografada). Esse passo inicial não será aplicados nos testes pois seria impossível o IDS detectar.

O passo seguinte seria uma garimpagem de informações e mapeamento da rede, a procura de um *host* vulnerável. A ferramenta escolhida para essa finalidade é o NMAP subseção 2.5.1.

No primeiro teste de Scan, usou-se o parâmetro '-F', habilitando a modo *fast* do NMAP. Nesse modo, são verificadas apenas a portas especificadas no arquivo nmap-services, na instalação padrão esse arquivo vem com 27372 portas descritas. Isso é muito mais rápido que verificar todas as 65535 portas tcp e 65535 portas udp, possíveis em um *host*.

```
nmap -F ALVO
```

O segundo teste, usou-se o parâmetro '-sV' do NMAP. Essa opção habilita a descoberta de versões, tentando determinar os protocolos de serviços, o nome da aplicação, o número da versão, o nome do *host* (utilizando o DNS reverso), tipo de dispositivo, sistema operacional usado, entre outras informações. Essas informações são de grande valor pois, a partir delas, pode-se explorar vulnerabilidades conhecidas de uma determinada versão de um serviço (NMAP, 2017).

```
nmap -sV ALVO
```

De posse de um alvo em potencial, próximo passo seria rodar um *scanner* de vulnerabilidade, em busca de brejas já conhecidas, e que, geralmente por descuido do administrador, não foi fechada. Para esses testes usou-se o *framework* Metasploit (subseção 2.5.2) nativo do sistema operacional Kali Linux.

4.3 Resultados

4.4 Conclusão

4.5 Métricas de Comparação

5 Considerações Finais e Trabalhos Futuros

Referências

ARYA, Y. et al. A study of metasploit tool. *International Journal Of Engineering Sciences & Research Technology*, 2016. Citado na página 33.

AVGERINOS, T. et al. Automatic exploit generation. Communications of the ACM, 2014. Citado na página 28.

BAKER, A. R.; CASWELL, B.; BEALE, J. *Snort IDS and IPS Toolkit*. 1. ed. [S.l.]: Syngress, 2007. Citado 5 vezes nas páginas 39, 41, 42, 43 e 44.

BASSO, T. Uma abordagem para avaliaçãgo da eficacia de scanners de vulnerabilidades em aplicaçgão web. Faculdade de Engenharia Eletrica e de Computaçãgo - UNICAMP, 2010. Citado na página 27.

CERON, J. *Tratamento de Incidentes de Segurança*. 2. ed. [S.l.]: Rede Nacional de Ensino e Pesquisa - RNP, 2015. 46 p. Citado 3 vezes nas páginas 24, 29 e 30.

CLARO, J. R. Sistemas ids e ips: Estudo e aplicação de ferramenta *OPEN SOURCE* em ambiente linux. Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense, 2015. Citado na página 19.

COELHO, F. E. S.; ARAUJO, L. G. S. de; BEZERRA, E. K. Gestão da segurança da informação - nbr 27001 e nbr 27002. *Escola Superior de Redes - RNP*, 2014. Citado na página 17.

COLLECTD. *Collectd – The system statistics collection daemon*. 2017. Disponível em: <https://collectd.org/>. Acesso em: 12 jul. 2017. Citado na página 48.

DAMAYE, S. *Oficial Documetation*. 2016. Disponível em: <http://pytbull.sourceforge.net/index.php?page=documentation>. Acesso em: 02 ago. 2017. Citado 2 vezes nas páginas 35 e 36.

DEBIAN. *Afinal de contas, o que e o Debian?* 2017. Disponível em: <https://www.debian.org/intro/about>. Acesso em: 12 jul. 2017. Citado na página 47.

ELASTIC. *Elasticsearch Reference*. 2017. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html>. Acesso em: 12 jul. 2017. Citado na página 49.

ELASTIC. *Kibana User Guide*. 2017. Disponível em: <https://www.elastic.co/guide/en/kibana/current/introduction.html>. Acesso em: 12 jul. 2017. Citado na página 49.

ELASTIC. *Logstash Reference*. 2017. Disponível em: <https://www.elastic.co/guide/en/logstash/current/introduction.html>. Acesso em: 12 jul. 2017. Citado na página 49.

ELIAS, G.; LOBATO, L. C. *Arquitetura e Protocolos de Rede TCP-IP*. 2. ed. [S.l.]: Rede Nacional de Ensino e Pesquisa - RNP, 2013. Citado na página 20.

ENIS, M. Ransomware hits govt., libraries. Library Journal, Maio. 2017. Acesso em: 19 out. 2017. Citado na página 32.

INTERNET, C. G. da. *Ataques na Internet*. 2017. Disponível em: <https://cartilha.cert.br/ataques/>. Acesso em: 21 jul. 2017. Citado na página 29.

INTERNET, C. G. da. *Codigos maliciosos* $Malware$. 2017. Disponível em: <https://cartilha.cert.br/malware/>. Acesso em: 12 jul. 2017. Citado na página 32.

INTERNET, C. G. da. *Estatisticas dos Incidentes Reportados ao CERT.br*. 2017. Disponível em: <https://www.cert.br/stats/incidentes/>. Acesso em: 30 nov. 2017. Citado na página 13.

INTERNET, C. G. da. *Incidentes Reportados ao CERT.br - Tipos de Ataques*. 2017. Disponível em: <https://www.cert.br/stats/incidentes/2016-jan-dec/tipos-ataque.html>. Acesso em: 02 ago. 2017. Citado na página 26.

ISH, J. *py-idstools*. 2017. Disponível em: <https://github.com/jasonish/py-idstools>. Acesso em: 12 jul. 2017. Citado na página 49.

JAIN, J.; PAL, P. R. Detecting worms based on data mining classification technique. IJESC, 2017. Citado na página 32.

JUSTO, J. E. da S.; TAMARIZ, A. del R. Modelo de agente racional para auxiliar na gestãgo de serviços em redes de computadores. Universidade Federal do Norte Fluminense, 2012. Citado na página 20.

KALI. *What is Kali Linux ?* 2017. Disponível em: <http://docs.kali.org/introduction/what-is-kali-linux>. Acesso em: 12 jul. 2017. Citado na página 48.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma abordagem top-down*. 7. ed. [S.l.]: Pearson Education do Brasil Ltda, 2013. Citado 2 vezes nas páginas 20 e 31.

LINTOTT, D. *OpenXenManager introduction*. 2017. Disponível em: <https://github.com/OpenXenManager/openxenmanager>. Acesso em: 12 jul. 2017. Citado na página 47.

LOCOCO, M. *Capacity Planning for Snort IDS*. 2011. Disponível em: <http://mikelococo.com/2011/08/snort-capacity-planning/>. Acesso em: 12 jul. 2017. Citado na página 47.

LOPEZ, M. E. A. Um arquitetura de detecção e prevenção de intrusão para redes definidas por software. Programa de Engenharia Eletrica - COPPE - UFRJ, 2014. Citado 2 vezes nas páginas 42 e 45.

MARTINELO, C. A. G.; BELLEZI, M. A. Analise de vulnerabilidades com openvas e nessus. Universidade Federal de São Carlos - UFSCar, 2014. Citado na página 13.

MAYNOR, D. et al. *Metasploit Toolkit for penetration testing, exploit development, and vulnerability research*. 1. ed. [S.l.]: Syngress, 2007. Citado na página 34.

MCCLURE, S.; SCAMBRAY, J.; KURTZ, G. *Hackers Expostos Segredos e Soluções para a Segurançga de Redes*. 7. ed. [S.l.]: Bookman Editora LTDA, 2014. Citado na página 28.

MUKHOPADHYAY, I.; CHAKRABORTY, M.; CHAKRABARTI, S. A comparative study of related technologies of intrusion detection & prevention systems. JOURNAL OF INFORMATION SECURITY, 2011. Citado na página 37.

NAGAHAMA, F. Y. Ipsflow: Um framework para sistema de prevenção de intrusão baseado em redes definidas por software. 2013. Citado 3 vezes nas páginas 37, 38 e 40.

NMAP. *Nmap Manual*. 2017. Disponível em: <https://nmap.org/>. Acesso em: 12 jul. 2017. Citado 2 vezes nas páginas 33 e 50.

NUNAN, A. E. Detecçãgo de cross-site scripting em paginas web. Instituto de Computaçãgo - UFAM, 2012. Citado na página 30.

NUNES, C. H. F. Exploit e ferramentas para sua utilizaçãgo. FATEC OURINHOS, 2011. Citado na página 28.

OISF. *About*. 2017. Disponível em: <https://suricata-ids.org/about/>. Acesso em: 20 dez. 2017. Citado na página 45.

OISF. *Rule Format*. 2017. Disponível em: <http://suricata.readthedocs.io/en/latest/rules/intro.html>. Acesso em: 22 dez. 2017. Citado na página 45.

PEIXINHO, I. de C.; FONSECA, F. M. da; LIMA, F. M. M. Segurança de redes e sistemas. *Escola Superior de Redes - RNP*, 2013. Citado na página 18.

PROOFPOINT. *About Proofpoint*. 2017. Disponível em: <https://www.proofpoint.com/us/company/about>. Acesso em: 22 dez. 2017. Citado na página 46.

ROESCH, M.; GREEN, C. *Snort Users Manual*. 2017. Disponível em: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node2.html>. Acesso em: 29 set. 2017. Citado na página 41.

S, S.; S, S.; M, R. Review on sql injection attacks: Dection techniques and protection mechanisms. International Journal of Computer Science and Information Technologies, 2014. Acesso em: 26 out. 2017. Citado na página 25.

SCARFONE, K.; MELL, P. Guide to intrusion detection and prevention systems $idps$. National Institute Of Standards and Technology, 2007. Citado na página 38.

SCHARDONG, F.; ÁVILA, R. Interface de apoio para ataques de força bruta com o gpu md5 crack. ERAD, 2012. Citado na página 29.

SEGURANÇA, C. de Atendimento a Incidentes de. *Segurança*. 2017. Disponível em: <https://www.rnp.br/servicos/seguranca>. Acesso em: 12 jul. 2017. Citado 4 vezes nas páginas 24, 25, 26 e 27.

SILVA, D. R. P. da; STEIN, L. M. Segurança da informaçgãgo: uma reflexão sobre componentes humanos. *Ciência e Cogniçãgo*, 2007. Citado na página 28.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. [S.l.]: Prentice Hall, 2011. Citado na página 30.

STALLINGS, W.; BROWN, L. *Segurança de Computadores: Principios e Praticas*. 2. ed. [S.l.]: Elsevier Editora LTDA, 2014. Citado na página 29.

TALOS. *Talos*. 2017. Disponível em: <https://www.snort.org/talos>. Acesso em: 22 dez. 2017. Citado na página 45.

TANENBAUM, A.; WETHERALL, D. *Redes de Computadores*. 5. ed. [S.l.]: Editora Pearson, 2011. Citado na página 22.

TELECOMUNICAÇÕES, U. I. de. *Statistics*. 2017. Disponível em: <https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>. Acesso em: 29 set. 2017. Citado 2 vezes nas páginas 20 e 21.

TÉNICAS, A. B. de N. Nbr iso/iec 27002:2013. *ABNT*, 2013. Citado na página 18.

TILBORG, H. C. A. van; JAJODIA, S. *Encyclopedia of Cryptography and Security*. [S.l.]: Springer Science+Bysubesse Media, 2011. Citado na página 32.

ULBRICH, H. C.; VALLE, J. D. *Universidade Hacker*. 5. ed. [S.l.]: Digerati Books, 2007. Citado 6 vezes nas páginas 22, 23, 26, 27, 28 e 29.

UTO, N. *Teste de Invasão de Aplicações Web*. 1. ed. [S.l.]: Rede Nacional de Ensino e Pesquisa - RNP, 2013. 179 p. Citado na página 25.

XENSERVER. *About Xenserver*. 2017. Disponível em: <https://xenserver.org/about-xenserver-open-source.html>. Acesso em: 12 jul. 2017. Citado na página 47.

ZABBIX. *What is Zabbix*. 2017. Disponível em: <https://www.zabbix.com/product>. Acesso em: 12 jul. 2017. Citado na página 49.

ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. IEEE, 2013. Citado na página 31.

ZONE-H, E. *Estatisticas*. 2017. Disponível em: <http://www.zone-h.com.br/stats>. Acesso em: 22 nov. 2017. Citado na página 30.