Glenon Mateus Barbosa Araújo

Análise de Sistemas de Detecção de Intrusão

Brasil

2017

Glenon Mateus Barbosa Araújo

Análise de Sistemas de Detecção de Intrusão

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

2017

fichacatalografica

Glenon Mateus Barbosa Araújo Análise de Sistemas de Detecção de Intrusão/ Glenon Mateus Barbosa Araújo. – Brasil, 2017- 39 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, 2017.
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

Figura 1 – Estatísticas de ataques reportadas ao CERT.br 17
Figura 2 – Ataque de Negação de Serviço Distribuído 21
Figura 3 – Exemplo de saída do Nmap 23
Figura 4 – Arquitetura do Metasploit 24
Figura 5 – Arquitetura do *framework* Pytbull 25
Figura 6 – Exemplo de arquitetura de NIDS passivo 29
Figura 7 – Exemplos de Arquitetura de NIDS ativo 29
Figura 8 – Componentes do Snort 30
Figura 9 – Infraestrutura do ambiente de teste 32
Figura 10 – Busca e união dos dados de diferentes fontes 33

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*
DoS	*Denial of Service*
CAIS	*Centro de Atendimento a Incidentes de Segurança*
DoS	*Denial of Services*
DDoS	*Distributed Denial of Services*
URL	*Uniform Resource Locator*

Sumário

1 INTRODUÇÃO . . . 13
1.1 Motivação . . . 13
1.2 Objetivos . . . 13
1.3 Metodologia . . . 13
1.4 Trabalhos Relacionados . . . 13

2 SEGURANÇA EM REDES DE COMPUTADORES . . . 15
2.1 Definições . . . 15
2.2 Cenário Geral . . . 16
2.3 Pontos de Vulnerabilidade . . . 16
2.4 Ataques Comuns à Redes de Computadores . . . 16
2.4.1 *Scanners* . . . 17
2.4.2 Exploit . . . 18
2.4.3 Força Bruta . . . 19
2.4.4 Desfiguração de páginas . . . 20
2.4.5 Negação de Serviços . . . 20
2.4.6 Malwares . . . 21
2.5 Ferramentas para Avaliação de Segurança . . . 22
2.5.1 Nmap . . . 22
2.5.2 Metasploit Framework . . . 23
2.5.3 Pytbull . . . 23
2.6 Conclusão . . . 25

3 SISTEMAS DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO . . . 27
3.1 Definições de IDS/IPS . . . 27
3.2 Tipos de IDS/IPS . . . 27
3.3 Principais Ferramentas de IDS . . . 29
3.3.1 Snort . . . 29
3.3.2 Suricata . . . 30
3.4 Conclusão . . . 30

4 DETECÇÃO DE INTRUSÃO EM UM CENÁRIO REAL . . . 31
4.1 Metodologia dos Testes . . . 31
4.1.1 Cenário de Testes . . . 31
4.1.2 Infraestrutura Definida para Testes . . . 31
4.2 Resultados . . . 34

4.3 Conclusão 34
4.4 Métricas de Comparação 34

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 35

REFERÊNCIAS 37

1 Introdução

1.1 Motivação

1.2 Objetivos

1.3 Metodologia

1.4 Trabalhos Relacionados

2 Segurança em Redes de Computadores

Esse

2.1 Definições

Para entendemos melhor o que é segurança da informação, precisamos conceituar alguns termos que serão detalhados 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.

Segurança da Informação 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 a norma ISO/IEC 27002 (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 (identificar se uma informação foi alterada) e a disponibilidade (tentando acessar um serviço e verificando se o mesmo está respondendo adequadamente). No entanto, só é possível identificar se houve quebra da confiabilidade através de auditorias, analisando os registros de acesso (se houver), tornando a identificação complicada 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.

2.2 Cenário Geral

2.3 Pontos de Vulnerabilidade

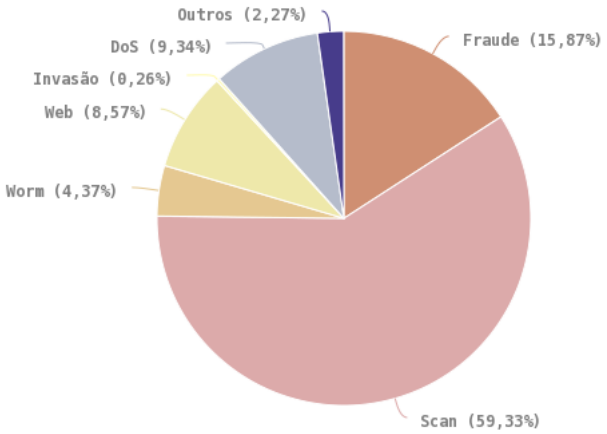
2.4 Ataques Comuns à Redes de Computadores

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 1).

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.

Figura 1 – Estatísticas de ataques reportadas ao CERT.br



Fonte: (INTERNET, 2017b)

O CAIS é responsável por zela pela segurança da rede Ipê (infraestrutura de rede dedicada à comunidade brasileira de 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 (CAIS, 2017).

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* 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 comumente exploram *bugs* ou vulnerabilidades em programas para ter acesso ao sistema alvo. Infelizmente, existem milhares de *bugs*, por exemplo, em 2013, foram

reportados 103,000 *bugs* no sistema operacional Ubuntu. Outro projetos de códigos fechados, possuem estatísticas similares (AVGERINOS et al., 2014).

Diante das vulnerabilidades encontradas utilizando os *scanners*, próximo passo 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).

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,).

Nos serviços *web*, como por exemplo, apache2 existe um usuário especial, comumente chamado de www-data ou algo semelhante. O servidor *web*, 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).

Em 2010, o site Zone-h registrou mais de 1,4 milhões de páginas desfiguradas, muitas delas associadas a ataques *Cross-site scripting* (XSS). Os *defacements* são considerados ataques passivos pois é gerado apenas uma mensagem na tela (NUNAN, 2012).

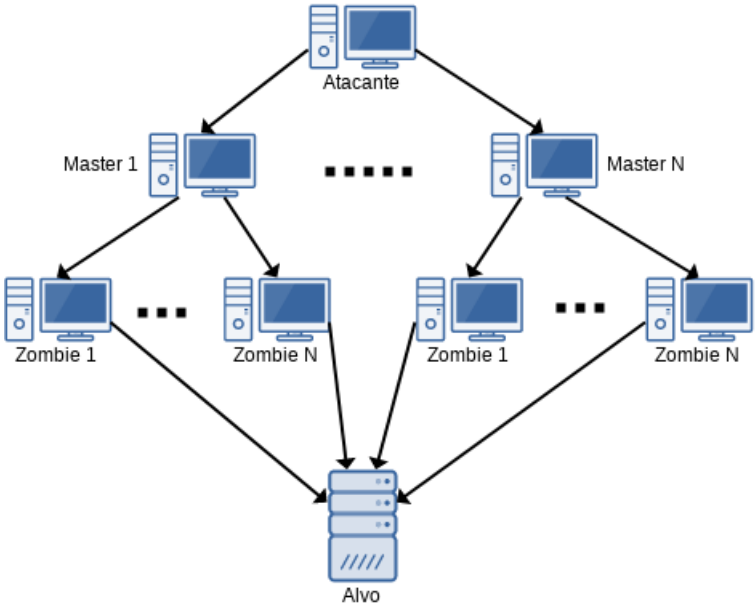
2.4.5 Negação de Serviços

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).

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 um forma mais sofisticada de ataque de Negação de Serviço chamada Negação de Serviço Distribuído (*Distributed Denial of Services* - DDoS), enquanto o DoS básico as requisições partem de apenas uma fonte (Figura ??), 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).

Figura 2 – 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. Os *malwares* mais conhecidos são os vírus, worms e trojans.

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, 2017a).

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, 2017a).

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 3). As portas podem assumir quatro estados, temos: aberto (open), significa que existe alguma aplicação escutando conexões; filtrado (*filtered*), há um obstáculo na rede, podendo ser algum *firewall*, que impossibilita que o Nmap determine se a porta está aberta ou fechada; fechado (*closed*), não possui aplicação escutando na porta; não-filtrado (*unfilterd*), a porta responde requisição porém o Nmap não consegue determinar se estão fechadas ou abertas (NMAP, 2017)

Figura 3 – 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
```

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).

O Metasploit possui uma biblioteca divida em três partes: **Rex**: É a biblioteca fundamental, a maioria das tarefas executadas pelo *framework* usaram essa biblioteca; **MSF Core**: É o *framework* em si, possui, por exemplo, gerenciador de módulos e a base de dados; **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 arquitetura é mostrada com mais detalhes na Figura 4.

Os módulos são divididos da seguinte maneira: Payload: são código executados no alvo remotamente; Exploit: explora *bugs* ou vulnerabilidade existente em aplicações do alvo; Módulos Auxiliares: usado para escanear as vulnerabilidades e executar várias tarefas; Encoder: codifica o *payload* para evitar qualquer tipo de detecção pelo anti vírus.

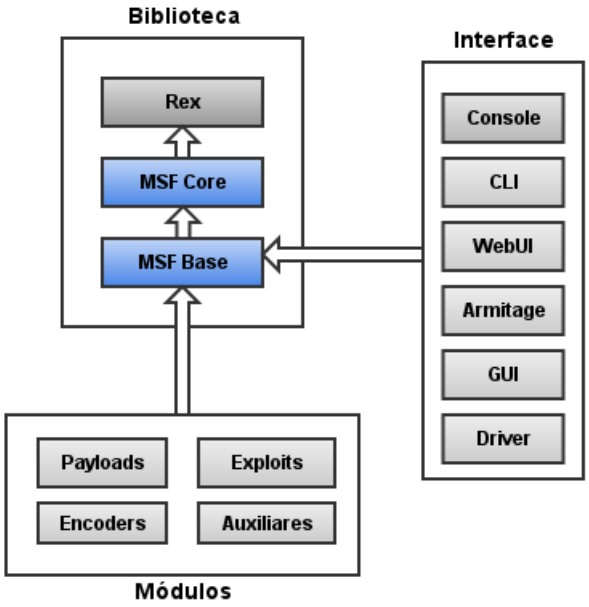
2.5.3 Pytbull

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 (**??**). 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;

Figura 4 – Arquitetura do Metasploit



Fonte: Autoria própria

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;

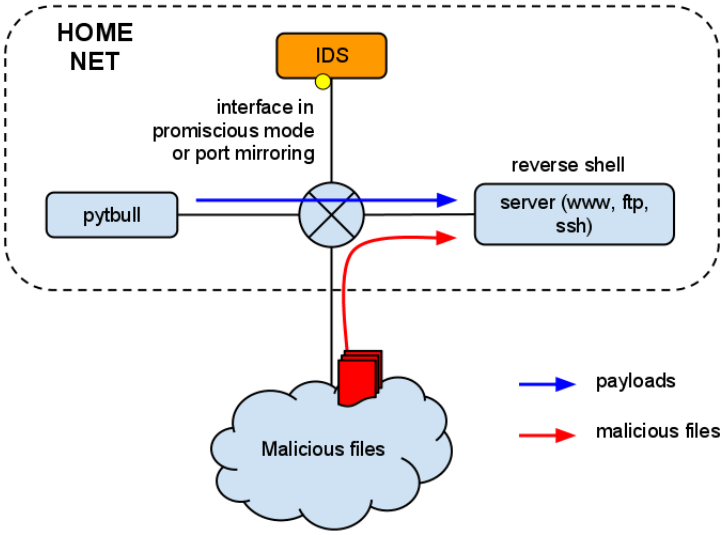
j) **shellCodes**: envia *shellcodes* para o servidor na porta 21/ftp testando a capacidade de detectar e/ou bloquear o mesmo;

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

Existem basicamente 5 tipos de testes: socket, abre um *socket* em uma porta e envia o *payload* para o alvo remoto na porta especificada; command, envia um comando para alvo remoto com a função python subprocess.call(); scapy, envia cargas úteis especificas baseadas na sintaxe de Scapy; client side attacks, usa um *shell* reverso no alvo remoto e envia comandos

para serem processados no servidor; pcap replay, permite reproduzir tráfego com base em arquivos de pcap.

Figura 5 – Arquitetura do *framework* Pytbull



Fonte: (??)

2.6 Conclusão

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 alguns possíveis incidentes, minimizando os impactos causados por sistemas comprometidos (MUKHOPADHYAY; CHAKRABORTY; CHAKRABARTI, 2011).

Basicamente os sistemas de detecção de intrusão são compostos por quatro componentes, temos: Sensor ou Agente, responsável pelo monitoramento e analise do trafego capturado; Banco de Dados, usado como repositório das informações de eventos detectados pelos sensor ou agente que serão processados; Gestor, é o dispositivo central que recebe, analisa e gerencia as informações de eventos vindos do sensor ou agente; Console, fornece uma interface para administração e monitoramento das atividades do IDS.

3.2 Tipos de IDS/IPS

Os IDSs 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).

Em um HIDS o sensor é instalado no *host* que será monitorado, analisando as informações contidas na própria máquina. Esse tipo de IDS tem o objetivo de analisar aspectos internos ao *host* como processos em execução, modificações na configuração do sistema e

serviços, monitorar o tráfego de rede somente do *host* e acesso a arquivos não autorizados, entre outros.

Os HIDS possuem algumas vantagens como evitar que alguns códigos sejam executados; bloqueia o tráfego de entrada e saída contendo ataques e uso não autorizado de protocolos e programas; evita que arquivos possam ser acessados, modificados e deletados impedindo o instalação de *malware* e outros ataques envolvendo acesso inapropriado a arquivos. Por outro lado, um HIDS possui algumas limitações como, por exemplo, difícil instalação e manutenção; interferir no desempenho do *host*; demora para identificar alguns eventos consequentemente a resposta a esses incidentes sofrerá um *delay* (SCARFONE; MELL, 2007).

Já em um NIDS, o sensor é instalado na rede e a interface de rede atua em um modo especial chamado "promíscuo", passando a ter a capacidade de capturar o tráfego da rede mesmo que os pacotes não sejam destinados ao próprio sensor.

Quanto a localização o NIDS pode ser classificado como passivo ou ativo. No modo passivo, 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. Enquanto no modo ativo, 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.

Os NIDS possuem algumas vantagens como serem independentes de plataforma; não interfere no desempenho do *host*; fácil implantação e transparente para o atacante. Por outro lado, possuem desvantagens como: pode adicionar retardados nos pacotes quando instalado no modo ativo, isso ocorre principalmente se houver um subdimensionamento do *hardware*; dificuldade de tratar dados de redes de alta velocidade; quando em modo passivo, trata apenas o segmento da rede que o IDS esta instalado e dificuldade de tratar dados criptografados. Esse tipo de IDS é mais utilizado devido a grande heterogeneidade de dispositivos e sistemas operacionais disponíveis na rede, tornando a administração mais simples se comparados com o HIDS.

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. Tem como vantagem, usar poucos recursos do servidor e rápido processamento. Porém, as desvantagens são: exigi uma atualização constante da base de assinaturas; alto conhecimento para geração da base e possuir um alto índice de falsos positivos e negativos.

Já os 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 desvio de comportamento. Porém existe a desvantagem

de gerar um grande número de falsos alertas em decorrência a modificações na rede ou *host* nem sempre representar um tráfego malicioso.

Figura 6 – Exemplo de arquitetura de NIDS passivo

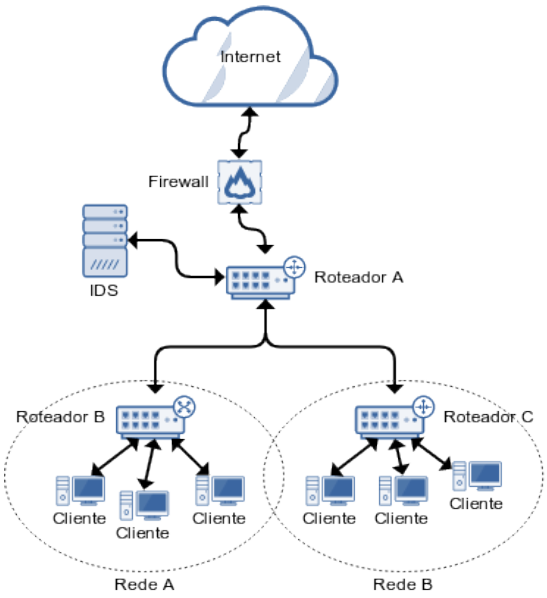
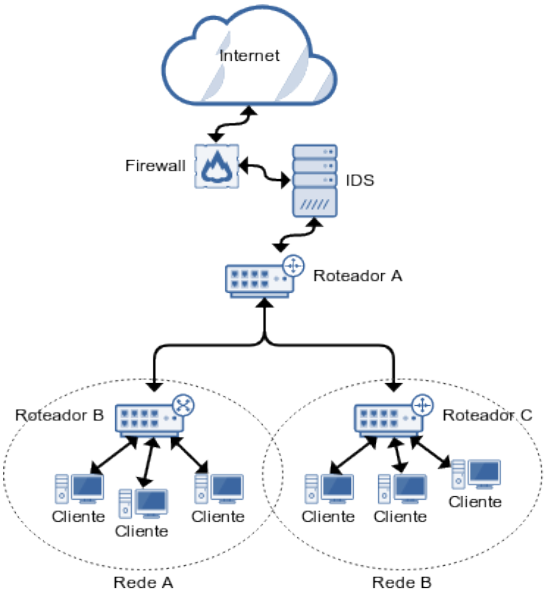


Figura 7 – Exemplos de Arquitetura de NIDS ativo



3.3 Principais Ferramentas de IDS

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) (**??**).

No modo *Sniffer*, o Snort captura os pacotes e exibi as informações no console. No modo *Packet Logger*, além de capturar o tráfego, ele registrar essas informações em disco (arquivos de logs). E no modo NIDS, é o modo mais complexo, permite analise do pacotes de rede em tempo real.

Existe quatro componentes no Snort: O *Sniffer*, o Pré-processador, o Motor de Detecção e Módulo de Saída. Os componentes são organizados de acordo com a figura 8 (**??**).

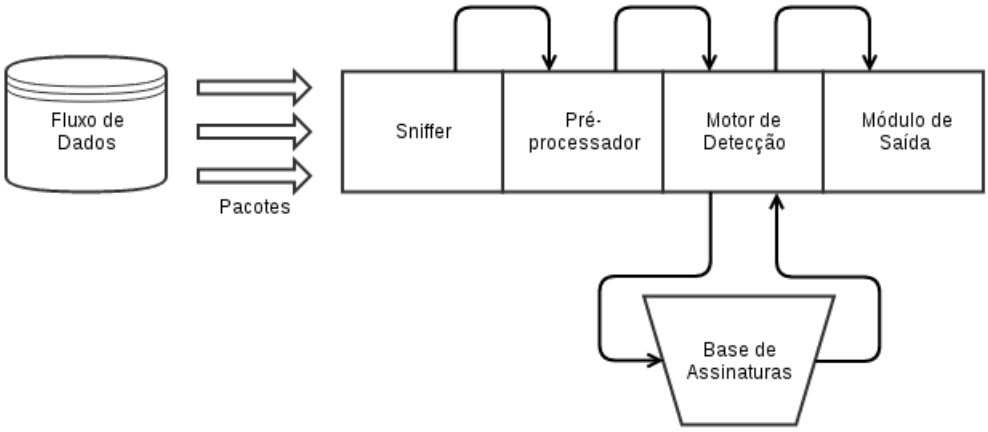


Figura 8 – Componentes do Snort

3.3.2 Suricata

3.4 Conclusão

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 4.1.2 será descrito os testes realizados com suas respectivas justificativas. Na seção 4.2 será apresentado os resultados esperados e obtidos, problemas encontrados e a comparação das ferramentas e por último, na seção 4.3, 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. Figura ??

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 que tinha com a plataforma e por existir uma interface para gerencia chamada XenCenter que roda no Windows. Uma alternativa *opensource* desse software é o OpenXenManager (LINTOTT, 2017).

No primeiro momento, foi instalado uma máquina virtual com o sistema operacional Debian 7.11 *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 9) 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 três 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. A infraestrutura final pode ser visualizada na Figura 9.

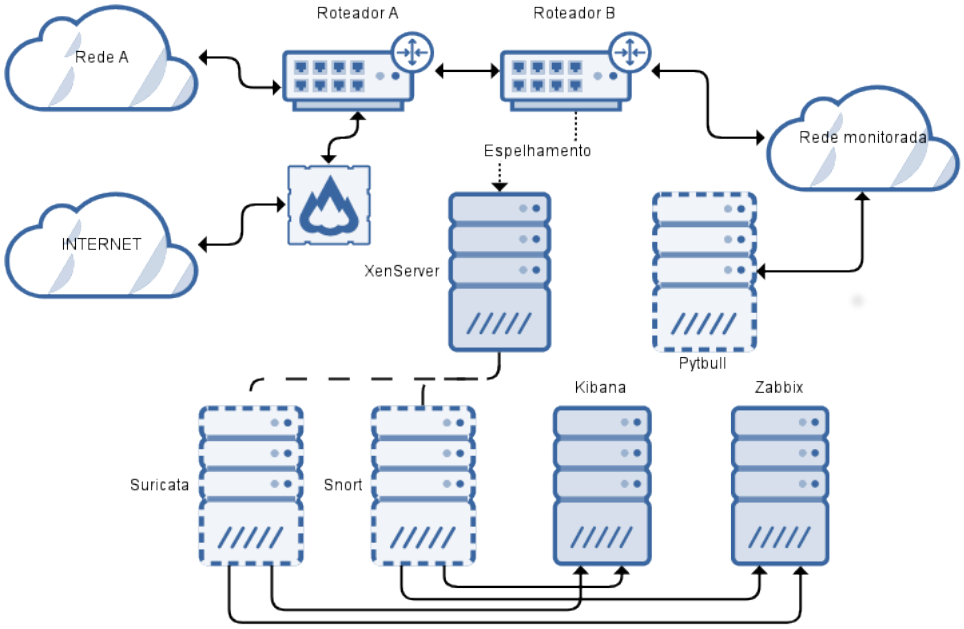


Figura 9 – Infraestrutura do ambiente de teste

Para coleta das informações de uso de recurso de hardware como memória, processamento e I/O das máquinas com os IDSs foi usado o *daemon* Collectd (COLLECTD, 2017). Outra opção para esse fim é a utilização de um servidor de monitoramento com o Zabbix (ZABBIX, 2017). A ideia de usar duas ferramentas para 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, compatível com

as ferramentas estudas. 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, o 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. O 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. Por ultimo, o Logstach (ELASTIC, 2017c), um motor de coleta de dados em tempo real, unificando os dados de diferentes fontes dinamicamente, normalizando-os nos destinos escolhidos (Figura 10). Dessa forma centralizou-se os *logs*, facilitando a visualização das ocorrências dos IDSs.

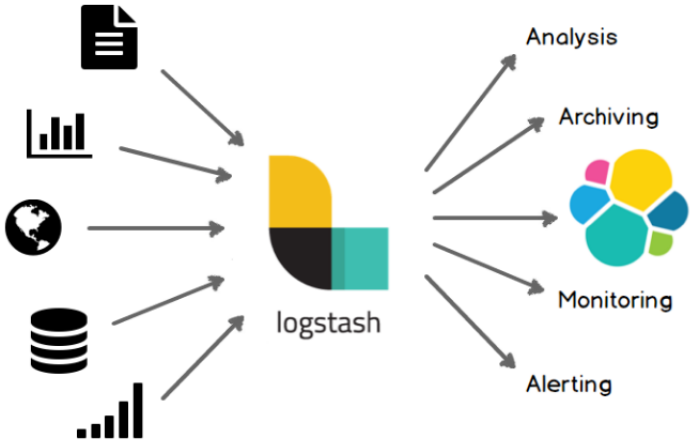


Figura 10 – Busca e união dos dados de diferentes fontes

sectionTestes 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 por engenharia social, analisando as pessoas que trabalharam no 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 um estudo e mapeamento da rede, a procura de um *host* vulnerável. A ferramenta escolhida para essa finalidade é o Nmap 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 possíveis em um *host*.

nmap -F 200.239.82.0/24

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*, 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 200.239.82.0/24

De posse de um alvo em potencial, próximo passo seria rodar um scan 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 2.5.2 nativo do sistema operacional Kali Linux.

4.2 Resultados

4.3 Conclusão

4.4 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 23.

AVGERINOS, T. et al. Automatic exploit generation. Communications of the ACM, 2014. Citado na página 19.

BASSO, T. Uma abordagem para avaliação da eficácia de scanners de vulnerabilidades em aplicação web. Faculdade de Engenharia Elétrica e de Computação - UNICAMP, 2010. Citado na página 18.

CAIS. *Segurança*. 2017. Disponível em: <https://www.rnp.br/servicos/seguranca>. Acesso em: 12 jul. 2017. Citado na página 17.

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 15.

COLLECTD. *Collectd – The system statistics collection daemon*. 2017. Disponível em: <https://collectd.org/>. Acesso em: 12 jul. 2017. Citado na página 32.

DEBIAN. *Afinal de contas, o que é o Debian?* 2017. Disponível em: <https://www.debian.org/intro/about>. Acesso em: 12 jul. 2017. Citado na página 31.

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 33.

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 33.

ELASTIC. *Logstash Reference*. 2017. Disponível em: <https://www.elastic.co/guide/en/logstash/current/introduction.html>. Acesso em: 12 jul. 2017. Citado na página 33.

INTERNET, C. G. da. *Ataques na Internet*. Citado na página 20.

INTERNET, C. G. da. *Códigos maliciosos* $Malware$. 2017. Disponível em: <https://cartilha.cert.br/malware/>. Acesso em: 12 jul. 2017. Citado 2 vezes nas páginas 21 e 22.

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 17.

ISH, J. *py-idstools*. 2017. Disponível em: <https://github.com/jasonish/py-idstools>. Acesso em: 12 jul. 2017. Citado na página 32.

JAIN, J.; PAL, P. R. Detecting worms based on data mining classification technique. IJESC, 2017. Citado na página 21.

KALI. 2017. Disponível em: <http://docs.kali.org/introduction/what-is-kali-linux>. Acesso em: 12 jul. 2017. Citado na página 32.

LINTOTT, D. *OpenXenManager introduction*. 2017. Disponível em: <https://github.com/OpenXenManager/openxenmanager>. Acesso em: 12 jul. 2017. Citado na página 31.

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 31.

MCCLURE, S.; SCAMBRAY, J.; KURTZ, G. *Hackers Expostos Segredos e Soluções para a Segurança de Redes*. 7. ed. [S.l.]: Bookman Editora LTDA, 2014. Citado na página 18.

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 27.

NAGAHAMA, F. Y. Ipsflow: Um framework para sistema de prevenção de intrusão baseado em redes definidas por software. 2013. Citado na página 27.

NMAP. *Nmap Manual*. 2017. Disponível em: <https://nmap.org/>. Acesso em: 12 jul. 2017. Citado 2 vezes nas páginas 22 e 34.

NUNAN, A. E. Detecção de cross-site scripting em páginas web. Instituto de Computação - UFAM, 2012. Citado na página 20.

NUNES, C. H. F. Exploit e ferramentas para sua utilização. FATEC OURINHOS, 2011. Citado na página 19.

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 16.

SCARFONE, K.; MELL, P. Guide to intrusion detection and prevention systems $idps$. National Institute Of Standards and Technology, 2007. Citado na página 28.

SCHARDONG, F.; ÁVILA, R. Interface de apoio para ataques de força bruta com o gpu md5 crack. ERAD, 2012. Citado na página 19.

SILVA, D. R. P. da; STEIN, L. M. Segurança da informação: uma reflexão sobre componentes humanos. *Ciência e Cognição*, 2007. Citado na página 19.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. [S.l.]: Prentice Hall, 2011. Citado na página 20.

STALLINGS, W.; BROWN, L. *Segurança de Computadores: Princípios e Práticas*. 2. ed. [S.l.]: Elsevier Editora LTDA, 2014. Citado na página 20.

TÉNICAS, A. B. de N. Nbr iso/iec 27002:2013. *ABNT*, 2013. Citado na página 15.

TILBORG, H. C. A. van; JAJODIA, S. *Encyclopedia of Cryptography and Security*. [S.l.]: Springer Science+Bysubesse Media, 2011. Citado na página 21.

ULBRICH, H. C.; VALLE, J. D. *Universidade Hacker*. 5. ed. [S.l.]: Digerati Books, 2007. Citado 3 vezes nas páginas 17, 18 e 19.

XENSERVER. *About Xenserver*. 2017. Disponível em: <https://xenserver.org/about-xenserver-open-source.html>. Citado na página 31.

ZABBIX. *What is Zabbix*. 2017. Disponível em: <https://www.zabbix.com/product>. Acesso em: 12 jul. 2017. Citado na página 32.

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 20.