Glenon Mateus Barbosa Araújo

Análise de IDPSs

Brasil

2017

Glenon Mateus Barbosa Araújo

Análise de IDPSs

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 IDPSs/ Glenon Mateus Barbosa Araújo. – Brasil, 2017- 37 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

Errata

Elemento opcional da **??**, 4.2.1.2). Exemplo: FERRIGNO, C. R. A. **Tratamento de neoplasias ósseas apendiculares com reimplantação de enxerto ósseo autólogo autoclavado associado ao plasma rico em plaquetas**: estudo crítico na cirurgia de preservação de membro em cães. 2011. 128 f. Tese (Livre-Docência) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2011.

Folha	**Linha**	**Onde se lê**	**Leia-se**
1	10	auto-conclavo	autoconclavo

Glenon Mateus Barbosa Araújo

Análise de IDPSs

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

Trabalho aprovado. Brasil, 24 de novembro de 2012:

Dr. Roberto Samarone dos Santos Araújo
Orientador

Brasil
2017

Agradecimentos

Resumo

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

Abstract

Keywords: Security, 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*
MB	*Megabytes*
GB	*Gigabytes*
SO	*Sistema Operacional*
JSON	*JavaScript Object Notation*

Sumário

Introdução

Objetivos

Trabalhos Relacionados

Motivação

1 Segurança de Redes de Computadores

1.1 Cenário Geral

1.2 Ataques

1.2.1 Varredura de Redes

1.2.2 Exploração de Vulnerabilidades

1.2.3 Força Bruta

1.2.4 Desfiguração de páginas

1.2.5 Negação de Serviços

1.2.6 Worm

1.2.7 Trojan

1.2.8 Fraudes - Direitos Autorais

2 Sistemas de Detecção e Prevenção de Intrusão

2.1 Tipos de IDS/IPS

2.2 Snort

2.3 Suricata

3 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 3.2 será abordado a infraestrutura usada para os testes, ferramentas utilizadas e as configurações feitas. Na seção 3.3 será descrito os testes realizados com suas respectivas justificativas. Na seção 3.4 será apresentado os resultados esperados e obtidos, problemas encontrados e a comparação das ferramentas e por último, na seção 3.5, uma breve conclusão.

3.1 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

3.2 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 e tipo do capturador de tráfego para dimensionar os recursos, aplicado especificamente ao Snort. A

mesma regra foi aplicada ao Suricata.

Para o *host* conseguir pegar o pacotes destinados a rede selecionada foi necessário uma configuração de espelhamento que consiste na copia dos pacotes que saem pela porta dessa rede no *switch* para a porta conectada no *host*. A interface de rede do *host* precisou ser configurado 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 pode ser visualizada na Figura 1.

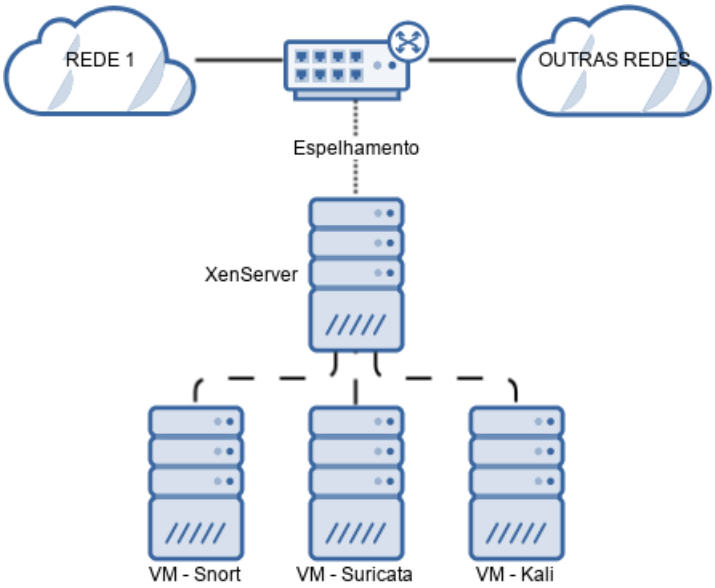

Figura 1 – Infraestrutura do ambiente de teste

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 (IDSTOOLS, 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, 2017a), uma plataforma de análise e visualização desenhada para trabalhar com os índices do Elasticsearch (ELASTIC, 2017b), 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 2). Dessa forma centralizou-se os *logs*, facilitando a visualização das ocorrências dos IDSs.

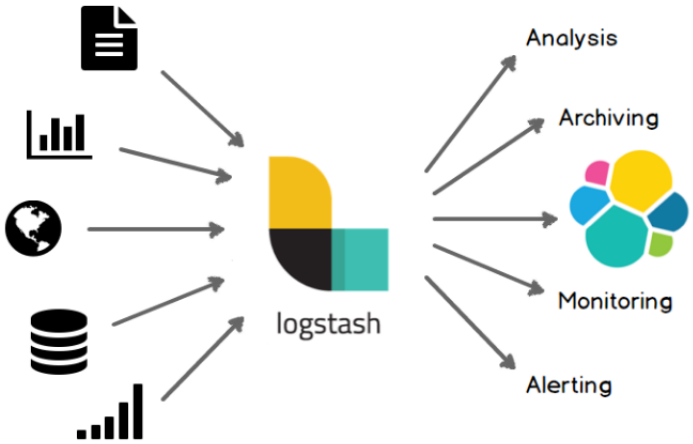

Figura 2 – Busca e união dos dados de diferentes fontes

3.3 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 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).

3.3.1 NMAP

```
nmap -F 200.239.82.0/24
```

No primeiro teste de Scan, usou-se o parâmetro -F, habilitando a modo *fast* do Nmap (NMAP, 2017). Nesse modo, são verificadas apenas a portas especificadas no arquivo nmap-services, que por padrão, são descritas 27372 portas descritas nesse arquivo. Isso é muito mais

rápido que verificar todas as 65535 portas existentes em um *host*. Nesse teste nenhum IDs conseguiu detectar os *scan* na rede.

```
nmap -sV 200.239.82.0/24
```

3.3.2 Pytbull

3.3.3 Metasploit Framework

3.4 Resultados

3.5 Conclusão

3.6 Métricas de Comparação

4 Considerações Finais

Referências

COLLECTD. 2017. Disponível em: <https://collectd.org/>. Citado na página 32.

DEBIAN. 2017. Disponível em: <http://www.debian.org/>. Citado na página 31.

ELASTIC. 2017. Disponível em: <https://www.elastic.co/guide/en/kibana/current/introduction.html>. Citado na página 33.

ELASTIC. 2017. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html>. Citado na página 33.

ELASTIC. 2017. Disponível em: <https://www.elastic.co/guide/en/logstash/current/introduction.html>. Citado na página 33.

IDSTOOLS. 2017. Disponível em: <https://github.com/jasonish/py-idstools>. Citado na página 32.

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

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

LOCOCO, M. *Capacity Planning for Snort IDS*. 2011. Disponível em: <http://mikelococo.com/2011/08/snort-capacity-planning/>. Citado na página 31.

NMAP. 2017. Disponível em: <https://nmap.org/>. Citado na página 33.

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

ZABBIX. 2017. Disponível em: <http://www.zabbix.com/>. Citado na página 32.