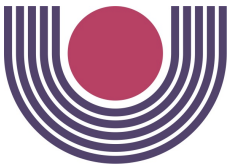

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Comparação de Softwares de Detecção de Intrusão em Rede de Computadores

Marco Antonio Paixão Néia

CASCAVEL
2019

MARCO ANTONIO PAIXÃO NÉIA

COMPARAÇÃO DE SOFTWARES DE DETECÇÃO DE INTRUSÃO EM REDE DE COMPUTADORES

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Luiz Antonio Rodrigues

CASCAVEL
2019

MARCO ANTONIO PAIXÃO NÉIA

COMPARAÇÃO DE SOFTWARES DE DETECÇÃO DE INTRUSÃO EM REDE DE COMPUTADORES

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Luiz Antonio Rodrigues (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Marcio Seiji Oyamada
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Ivonei de Freitas
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 5 de dezembro de 2019

DEDICATÓRIA

Dedico este trabalho a minha família, cujo suporte ao longo dos anos foi imprecindível para sua conclusão.

AGRADECIMENTOS

A toda a minha família, pelo apoio emocional e pelas lições de vida fornecidas durante toda a minha vida.

Aos professores do curso de Ciência de Computação, pelas lições durante meu período nesta instituição de ensino.

E a meus colegas de curso, que me acompanharam durante a jornada.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Mbps	Mega bits por segundo
IDS	*Intrusion Detection System*
IPS	*Intrusion Prevention System*
ARPA	*Advanced Research Projects Agency*
DARPA	*Defense Advanced Research Projects Agency*
CERT	*Computer Emergency Response Team*
RFC	*Request For Comments*
NIDS	*Network Based Intrusion Detection System*
HIDS	*Host Based Intrusion Detection System*

Sumário

Resumo

Este trabalho teve por objetivo estudar e comparar as principais ferramentas de detecção de intrusão em redes de computadores. Para esta avaliação foram utilizadas três ferramentas de código aberto (*open source*) de detecção de intrusão, Snort, Suricata e Zeek. Estas ferramentas foram avaliadas por meio de análise de *traces* contendo tráfego de rede com diferentes tentativas de intrusão, com objetivo de comparar a eficiência delas na detecção de diferentes tipos de ataques. Foi comparado o desempenho das ferramentas Zeek, Snort e Suricata em diferentes velocidades de rede. Foi obtido como resultado que as três ferramentas são equivalentes na detecção de ameaças. Foi observado ainda que a ferramenta Zeek perde menos pacotes que a ferramenta Snort, sendo que a ferramenta Zeek apresentou perda de pacotes a partir de redes com taxas de transmissão de 761 Mbps e a ferramenta Snort apresentou perda de pacotes a partir de redes com taxas de transmissão de 388 Mbps.

Palavras-chave: Segurança de redes, Sistema de Detecção de Intrusão, Snort, Suricata, Zeek.

Capítulo 1

Introdução

Segurança de redes de computadores é uma área que vem ganhando importância com a maior dependência em sistemas computacionais com o passar do tempo, e com seu resultante maior número de incidentes. Vale destacar que de acordo com o publicado pela empresa de segurança de computadores Kaspersky em relatório (2017), 29,4% dos computadores avaliados na pesquisa foram vítimas de pelo menos um ataque do tipo *Malware* em 2017, pesquisa essa que abrangeu milhões de computadores em 213 países. Com a evolução das ameaças se tornou necessário desenvolver ferramentas capazes de diferenciar o tráfego normal da rede de tentativas de invasão.

Softwares de detecção de intrusão (IDS) possuem o papel de alertar os administradores de determinada rede de computadores para a possibilidade de ocorrência de atividade desonesta no tráfego de sua rede (KACHA; SHEVADE, 2012), possibilitando a tomada das medidas necessárias para evitar maiores danos. Estas ferramentas tendem a ser utilizadas no lugar de uma ferramenta de prevenção de intrusão (IPS), pois ao contrário desta, um IDS não bloqueia tráfego, portanto evitando que falsos positivos causem o bloqueio de acesso à rede por um usuário válido. Entre as ferramentas IDS open-source disponíveis três foram escolhidas por sua ampla utilização na área (GHAFIR et al., 2016), sendo elas, Snort, Suricata e Zeek.

Snort (ROESCH, 1999) é um sistema de detecção de invasão através de análise de padrões (GHAFIR et al., 2016). É uma ferramenta leve criada para monitorar pequenas redes TCP/IP detectando uma grande variedade de tráfego suspeito, assim como ataques (ROESCH, 1999). A análise de padrões é realizada através de regras que são definidas em um arquivo de texto descrevendo tráfego de rede de interesse. Essas regras são as responsáveis por detectar tentativas de invasão e definem quais os tipos de alertas a serem fornecidos em qual situação. A ferramenta

Snort foi desenvolvida usando a linguagem de programação C++.

Suricata é um sistema de detecção de intrusão através de análise de padrões (GHAFIR et al., 2016), utiliza sistema de regras similar ao Snort. Duas vantagens que esta ferramenta possui sobre a anterior consistem de suporte a especificação de mais tipos de protocolos (L3 - rede e L7 - aplicação), assim como ser multithreaded nativamente (GHAFIR et al., 2016). Também é capaz de analisar uma quantidade de tráfego superior a seu antecessor (KACHA; SHEVADE, 2012). A ferramenta Suricata foi desenvolvida usando a linguagem de programação C.

Zeek (PAXSON, 1999), originalmente nomeado Bro, foi inicialmente desenvolvido como um sistema de pesquisa em detecção de invasão e análise de tráfego por Vern Paxson do Centro de Pesquisa da Internet em Berkeley (KAUR, 2008). Zeek é um monitor de segurança de rede que realiza inspeção profunda de pacotes (*Deep Packet Inspection*) utilizando de análise baseada em eventos. Ao contrário das ferramentas Snort e Suricata, Zeek não é dirigido por regras sendo seus filtros de detecção definidos através de scripts implementados em linguagem de programação própria (GHAFIR et al., 2016). É importante ressaltar que por ter sido originalmente criada como uma ferramenta de pesquisa, a ferramenta Zeek possui uma maior complexidade em sua utilização, tendo seu uso indicado para redes precisando de detecção de invasão flexível e altamente customizável (SING, 2008). A ferramenta Zeek atualmente usa em seu desenvolvimento as linguagens de programação C++ e Zeek em proporções aproximandamente iguais.

Diversos estudos vêm sendo realizados na área de IDS, tais como (ALBIN; ROWE, 2012), (ROS; CARELA-ESPAñOL; BUJLOW, 2013), (ROS; CARELA-ESPAñOL; BUJLOW, 2014), e (MURINI, 2014). Estudos estes que têm como objetivo validar o funcionamento destas e de outras ferramentas, de maneira a melhor avaliar suas respectivas vantagens e desvantagens.

Um dos problemas de avaliar ferramentas da área de segurança é que, conforme as ameaças evoluem, os estudos realizados tendem a perder a sua validade, sendo assim necessário realizar este tipo de estudo periódicamente. O resultado esperado neste trabalho é a obtenção de dados quantitativos e atuais que permitam comparar a eficácia das três ferramentas escolhidas na detecção de tráfego malicioso, perante tipos variados de tentativas de intrusão, considerando especialmente o desempenho na detecção de anomalias e a eficiência de análise dos dados em tempo real.

1.1 Objetivo

1.1.1 Geral

O objetivo deste trabalho foi comparar três ferramentas *open-source* de detecção de intrusão quanto o seu desempenho na detecção de tentativas de intrusão diferentes, assim como testar a eficiência desta detecção sob velocidades de tráfego de rede diferentes através de software simulador de tráfego de rede.

1.1.2 Específicos

- Escolher e classificar os tipos de tentativa de intrusão a serem usados na avaliação.
- Usar *traces* com tráfego de rede com tentativa de intrusão para comparar a eficácia na detecção de intrusão das três ferramentas.
- Usar *traces* com tráfego de rede inofensivo para testar a presença de falsos positivos por parte das três ferramentas na detecção de intrusão.
- Comparar a detecção de tentativa de intrusão das ferramentas Snort, Suricata e Zeek em redes com diferentes taxas de transmissão.

1.2 Metodologia

Para atingir os objetivos do projeto foram escolhidas as ferramentas *open source* Snort, Suricata e Zeek. Estas ferramentas têm a capacidade de agir como IDS (*Intrusion Detection System*), sendo portanto capaz de identificar tráfego suspeito em uma rede a partir de leitura dos pacotes de informação que passam por esta.

Para facilitar a realização da comparação entre as ferramentas os *traces* com as tentativas de intrusão foram classificados nos seguintes grupos:

- *Malware*: propagação de pacotes contendo malware de tipos variados.
- *Botnet*: dispositivos infectados e controlados remotamente pra realizar ataques (SALMON; MACLAFFERTY; LAVESQUE, 2017).
- *SQL Injection*: Inserção de consulta SQL maliciosa.

- *Port Scan*: Varredura em rede de computadores, com o intuito de identificar quais computadores estão ativos e quais serviços estão sendo disponibilizados por eles.

- *Packet Injection*: Ataque no qual pacotes criados pelo atacante são inseridos no fluxo de dados já existente (SALMON; MACLAFFERTY; LAVESQUE, 2017).

Uma categoria extra foi adicionada a esta classificação, *inofensivo*, que corresponde a *traces* sem tentativa de intrusão, possibilitando a análise de possíveis falsos positivos na detecção de intrusão por parte das ferramentas.

Para a realização do experimento foram utilizados *traces* obtidos em repositórios públicos, sendo que todas as ferramentas utilizaram os mesmos *traces* e que cada categoria deverá teve o mesmo número de *traces* para facilitar a visualização dos resultados. Para melhores resultados é ideal a obtenção do maior número possível de *traces*, mas por consequência da pouca disponibilidade destes, esse número foi limitado a 7 *traces* por categoria.

Para a realização do teste das ferramentas cada *trace* (no formato *pcap*) foi analisado de maneira individual pelas ferramentas, sendo todas as ferramentas instaladas na mesma máquina. Vale ressaltar que estas ferramentas realizam a classificação de tráfego, em seguro e suspeito, de acordo com regras internas, regras estas que podem ser modificadas pelo usuário. Para este experimento serão utilizadas as regras configuradas automaticamente no processo de instalação da ferramenta, ou quando isto não ocorre, serão utilizadas as regras recomendadas na documentação da ferramenta. No caso de ferramenta Zeek serão utilizados os *scripts* configurados durante a instalação.

Os resultados foram então registrados de acordo com as categorias já definidas para facilitar a visualização dos resultados e a comparação das ferramentas.

Para a manipulação da velocidade do tráfego de rede foi usada a ferramenta Tcpreplay, ferramenta esta que possibilita a simulação e edição de tráfego de rede a partir de arquivo do tipo *pcap* capturado previamente, possibilitando o teste das ferramentas sob condições diferentes de tráfego de rede.

Capítulo 2

Contextualização e Trabalhos Correlatos

2.1 Contextualização

Este capítulo apresenta duas seções que visam fornecer um entendimento comum do estado atual da pesquisa em softwares do tipo IDS e como se chegou a este estado.

Na seção 2.1 é feita uma rápida contextualização da área de redes de computadores e da área de segurança de redes em específico.

Na seção 2.2 são mencionados alguns trabalhos executados em linha de pesquisa similar ao corrente trabalho.

2.1.1 Redes de Computadores e a Internet

De acordo com Tanenbaum (2011) uma rede de computadores é composta por um conjunto de computadores autônomos interconectados sendo capazes de trocar informação, sendo este o principal motivo de sua utilização.

O exemplo mais conhecido, e indiscutivelmente o mais relevante, de uma rede de computadores é a internet, que consiste de uma rede de redes (TANENBAUM, 2011). No entanto, a troca de informações é apenas a descrição de mais baixo nível do que a internet é capaz de realizar. De fato a internet pode ser melhor descrita como "uma infraestrutura que provê serviços para aplicações"(KUROSE; ROSS, 2012). Para entender como a internet chegou a esse nível e quais os problemas que seu uso desta maneira causa, é necessário analisar a internet desde a sua criação.

A internet nasceu como um projeto dentro da ARPA (*Advanced Research Projects Agency*) que era a agência do governo dos Estados Unidos da América responsável por controlar os pro-

jetos de pesquisa avançada relacionados a defesa. Naquela época (1966) computadores não só eram caros mas também incompatíveis entre sí, o que tornava necessária a criação de algo novo que possibilitasse a comunicação entre as máquinas. A internet, que seria então chamada de ARPANET, foi um projeto que nasceu com o objetivo de estabelecer um canal de comunicação entre as equipes que estavam desenvolvendo pesquisa para a agência (ARPA) em diferentes universidades americanas (NAUGHTON, 2011).

Apesar de o projeto para o que se tornaria a ARPANET ter recebido fundos em 1966, foi só em 1967 que a ideia recebeu apoio das universidades em questão. A preocupação das universidades era que seus preciosos computadores teriam de gastar recursos, então escassos, para gerenciar a comunicação na rede. Esse problema foi sanado com a adição de um computador especializado (roteador) entre a rede e o computador host que ficaria responsável por lidar com essa tarefa, tendo o *host* apenas que se comunicar com este computador. Mas foi apenas em 1969 que a rede começou a funcionar, com apenas dois nodos, um na universidade UCLA e outro na universidade de Stanford (NAUGHTON, 2011).

Em 1975 a ARPANET teve seu controle transferido da DARPA (antiga ARPA) para outra agência governamental, o que livrou a agência para se focar no seu próximo projeto, conhecido então como *internetting*, que tinha como objetivo possibilitar a comunicação entre redes de computadores diferentes. Eventualmente o conjunto de protocolos estabelecidos tanto pela DARPA como por outras organizações se tornaria o que hoje é conhecido como o modelo TCP/IP (NAUGHTON, 2011).

Foi apenas no início da década de 80, com o aumento de tráfego na rede, que o pentágono começou a se preocupar com a segurança da rede, que estava sendo usada principalmente para tráfego militar e científico. A sugestão era separar a rede em duas, com uma delas (MILNET) sendo reservada para uso militar, mas como as duas redes ainda teriam de se comunicar se tornou urgente a implantação dos protocolos de *internetworking*. Em 1982 foi decidido que todos os nodos da ARPANET trocariam seus protocolos para o modelo TCP/IP. A troca foi realizada em 1983 (NAUGHTON, 2011).

A utilidade de uma rede de computadores é algo que se tornou óbvia com o tempo, mas como o acesso a ARPANET era controlada, outras redes foram criadas, tais como a USENET, para possibilitar outros a desfrutar das vantagens de ter acesso a uma rede de computadores

(NAUGHTON, 2011). É a junção destas redes que eventualmente criaria o que hoje chamamos de internet.

A internet possui um modelo de arquitetura que usa o conceito de camadas (ver a figura 2.1), com cada camada tendo responsabilidades bem definidas. Esta arquitetura facilita o gerenciamento de um sistema tão complexo como a internet, permitindo com que cada camada se preocupe apenas com suas responsabilidades, tratando as demais camadas como uma caixa preta. Essa modularidade torna mais simples atualizar os componentes deste sistema (KUROSE; ROSS, 2012).

Figura 2.1: Arquitetura da Internet. Fonte: Adaptado pelo autor.

A camada de aplicação consiste das aplicações de rede e de seus protocolos. Todas as aplicações de um computador que desejem utilizar a internet utilizam os serviços fornecidos por esta camada para fazê-lo. A camada de transporte é a responsável por entregar as mensagens da camada de aplicação, geradas no *host* remetente, para a camada de aplicação do destinatário. A camada de rede é responsável pela entrega da mensagem ao seu endereço de destino, definindo inclusive qual caminho essa mensagem deve seguir pela rede, essa camada é caracterizada pelo uso do protocolo IP. A camada de enlace é responsável por levar um pacote de um nó a outro na rede, esses nós podem ser *hosts* ou outros dispositivos de rede como roteadores, por exemplo. A camada física tem como responsabilidade a movimentação dos *bits* da mensagem de um nó ao outro (KUROSE; ROSS, 2012).

A comunicação entre os *hosts* se dá através da troca de pacotes, que nada mais são que pedaços de informação, que juntos constroem a mensagem enviada. O processo de comunicação

se dá através de protocolos que têm seus passos definidos em um documento do tipo RFC (*Request for Comments*).

2.1.2 Segurança de Redes

De acordo com Kurose e Ross (2012), segurança de redes consiste em garantir comunicação segura, que por sua vez pode ser dividida em, confidencialidade, integridade de mensagem, autenticação de *end-point* e segurança operacional. De modo similar Tanenbaum (2011), considera que segurança de redes em sua forma mais simples consiste em garantir que pessoas mal intencionadas não possam ler ou modificar mensagens destinadas a outros usuários. Pode-se ver então que segurança de redes consiste em proteger informação, não sendo por coincidência então que a área de segurança de redes é também muitas vezes chamada de segurança de informação.

Como citado no capítulo anterior foi na época da ARPANET que a comunidade começou a se preocupar com segurança. No entanto naquela época, a rede ainda era algo fechado com poucos usuários e desse modo a preocupação era mais voltada a parte operacional e não a parte técnica da rede. Aquele que é considerado como o evento mais marcante do inicio da área ocorreu no ano de 1988 e é conhecido como o *Morris Worm*. Esse *Worm* consistia de um programa capaz de auto-replicação que rapidamente se espalhou pela rede. Nota-se que este código não possuía outra função além da capacidade de se espalhar pela rede, capacidade essa que por si só foi suficiente para parar o funcionamento da rede por dias. A rede na época possuía cerca de 60 mil computadores conectados estimando-se que 10% destes foram infectados, com um custo de recuperação por máquina que variou de $200 a $53.000 (DENARDIS, 2007). É interessante observar que na época a internet estava tão no início que no julgamento do responsável pelo ataque foi necessário para as cortes de justiça criar primeiramente uma definição do que é internet (DENARDIS, 2007).

Provavelmente a maior consequência do *Morris Worm* foi aumentar a importância da área de segurança de redes. Após o ataque, e as discussões que seguiram, a DARPA anunciou a criação de uma nova organização, o *Computer Emergency Response Team* (CERT), que seria responsável por responder a problemas relacionados a segurança de computadores (DENARDIS, 2007). Nesta época a internet ainda consistia de redes menores interconectadas, com essas

redes menores (ARPANET, Milnet, NSFnet e Bitnet) sendo comunidades isoladas que apenas se comunicavam entre sí, não sendo ainda portanto a internet como funciona hoje. Isso levou a CERT ser no momento de sua criação uma agência que se focava na ARPANET e na Milnet. Era esperado na época que cada comunidade estabelecesse a sua própria CERT (DENARDIS, 2007).

Na década de 1990, no entanto, a internet começou a se espalhar pela sociedade em geral. No início foram as empresas que adotaram o uso da rede, o que as expos a um possível ataque, gerando o surgimento dos primeiros produtos comerciais relacionados a segurança. Os primeiros produtos que começaram a ser comercializados no início da década de 90 foram softwares de *Firewall* que eram implementados dentro dos roteadores comercializados. O surgimento de transações monetárias realizadas via internet foi um outro fator que aumentou a preocupação das empresas com segurança (DENARDIS, 2007).

Com a maior adoção pela sociedade da internet, os governos começaram a se posicionar. Isso se deve a importância que a rede tem para a economia. Um exemplo de ação tomada por um governo foi a Ordem Executiva 13010 do governo dos Estados Unidos da América que identificou *CyberTerrorism* como uma maneira auxiliar de terrorismo físico. Essa maior preocupação levou a transformação do CERT, que até então era considerada uma organização privada fundada pelo governo, em uma organização federal sob direto controle do governo americano. No entanto por mais que os governos do mundo tenham aumentado seu interesse na área, grande parte da responsabilidade em enfrentar vulnerabilidades permanece sob o setor privado (DENARDIS, 2007).

2.1.3 Sistemas de Detecção de Intrusão

Dentro dos softwares que visam ajudar na proteção de uma rede de computadores os IDS são softwares que visam alertar os administradores da rede a respeito de atividade possivelmente desonestas (KACHA; SHEVADE, 2012). A decisão sobre qual atitude deve ser tomada perante esta atividade permanece nas mãos do administrador. Vale ressaltar no entanto que um IDS pode ser integrado com outros sistemas possibilitando a automatização de proteção ativa contra intrusão.

Softwares do tipo IDS podem ser categorizados em duas classes, NIDS (*Network Based*

Intrusion Detection System) que operam através da análise do tráfego da rede, e HIDS (*Host Based Intrusion Detection System*) que analisam o sistema operacional e suas ações (KACHA; SHEVADE, 2012). Outra maneira pela qual os IDS se diferenciam é a maneira pela qual eles detectam o que é ameaça e o que não é. Algumas das maneiras de detecção são: detecção baseada em padrão ou assinatura, que identifica comportamento suspeito através de detecção de técnicas de intrusão conhecidas, e detecção baseada em anomalia, que cria um perfil do que é considerado padrão de uso normal em dado período de tempo e apontam as atividades que ferem esse padrão (KACHA; SHEVADE, 2012).

Muitos IDS, começando com o Snort, utilizam de um sistema de regras, que são arquivos que definem o comportamento da rede que deve ser considerado suspeito. Esse sistema permite não só a customização da ferramenta por parte dos usuários para melhor atender às suas necessidades, como permitem a atualização da ferramenta para melhor detectar novas ameaças. No entanto essa abordagem exige grande conhecimento por parte dos usuários para que possam criar regras novas e caso o usuário decida trocar de IDS, pode ser necessário aprender um novo modelo de declarar regras.

Exemplo de regra da ferramenta Snort que gera um alerta quando o conteúdo do pacote possui o padrão especificado. O campo flow estabelece que esta regra apenas se aplica quando o fluxo dos bits é uma requisição de um cliente (*to_server*) e somente em conexões TCP estabelecidas (*estabilished*). O campo msg define a mensagem a ser mostrada ao usuário. O campo metadata server para adicionar informação adicional a este regra em particular. O campo sid serve como identificador único da regra. Por fim o campo *content* é onde se define o conteúdo do pacote que se está procurando, normalmente este conteúdo é a característica pricipal que define o que é uma tentativa de intrusão e o que não é, na regra abaixo as palavras entre acentos de interogação são variáveis definidas pela ferramenta em um de seus arquivos de configuração, *within* e *http_uri* são modificadores de como o conteúdo é avaliado.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(
        msg: ?Snort 3 http_uri sticky buffer Example?;
        flow:to_server,established;
        content:?var=1?; http_uri;
        content:?malicious?; within:20; http_uri;
        metadata: service http;
```

```
        sid:1;
    )
```

Exemplo de regra do Suricata que descarta o pacote quando seu conteúdo possui o padrão especificado, observa-se que a ferramenta suricata pode ser utilizada como um IPS, mas neste trabalho apenas o modo IDS nos interessa. Os campos já citados na regra do Snort se comportam da mesma maneira. O campo flowbits permite que a regra rastreie o estado de um protocolo de aplicação, no caso da regra abaixo está sendo usado para identificar se o estado especificado em *content* está setado (*isset*). O campo PCRE permite que uma regra de detecção seja escrita usando expressão regular compátivel com PERL. O campo *reference* é usado para referenciar um sistema externo de identificação de ameaças. O campo *classtype* é usado para informar que essa ameaça faz parte de uma categoria de ameaças, no caso desta regra a ameaça sendo detectada é um *trojan*. Por fim o campo rev significa que esta é a segunda versão desta regra (ou seja esta regra foi atualizada uma vez).

```
    drop tcp $HOME_NET any -> $EXTERNAL_NET any
    (
        msg:?ET TROJAN Likely Bot Nick in IRC (USA +..)?;
        flow:established,to_server;
        flowbits:isset,is_proto_irc;
        content:?NICK ?;
        pcre:?/NICK .*USA.*[0-9]{3,}/i?;
        reference:url,doc.emergingthreats.net/2008124;
        classtype:trojan-activity;
        sid:2008124;
        rev:2;
    )
```

Exemplo de script do Zeek, gerando alerta após detectar *malware*:

```
    when ( local MHR_result = lookup_hostname_txt(hash_domain) )
{
    # Data is returned as "<dateFirstDetected> <detectionRate>"
    local MHR_answer = split_string1(MHR_result, / /);
    if ( |MHR_answer| == 2 )
    {
        local mhr_detect_rate = to_count(MHR_answer[1]);
        if ( mhr_detect_rate >= notice_threshold )
        {
            local mhr_first_detected =
```

```
                    double_to_time(to_double(MHR_answer[0]));
                local readable_first_detected =
                    strftime("%Y-%m-%d %H:%M:%S",
                    mhr_first_detected);
                local message = fmt(
                    "Malware Hash Registry Detection rate:
                    %d%% Last seen: %s", mhr_detect_rate,
                    readable_first_detected);
                local virustotal_url =
                    fmt(match_sub_url, hash);
                # We don't have the full fa_file
                # record here in order to
                # avoid the "when" statement
                # cloning it (expensive!).
                local n: Notice::Info =
                    Notice::Info($note=Match,
                    $msg=message, $sub=virustotal_url);
                Notice::populate_file_info2(fi, n);
                NOTICE(n);
            }
        }
    }
```

2.2 Trabalhos Correlatos

Kacha e Shevade (2012) em seu trabalho realizou uma comparação entre as ferramentas Snort e Suricata, focando na comparação dos recursos computacionais necessários para a utilização da ferramenta. O trabalho concluiu que a arquitetura *multithread* do Suricata gasta mais recursos, mas que o Snort apresenta problemas em redes com taxa de transmissão maior do que 200 - 300 Mbps. Concluiu ainda que as ferramentas são similarmente eficientes por poder usar os mesmos tipos de regras e as interpretar de maneiras semelhantes.

Day e Burns (2011) em seu trabalho teve como objetivo comparar as ferramentas Snort e Suricata para observar os efeitos que a implementação da arquitetura *multithread* no Suricata trouxe quando comparado ao padrão da indústria (Snort). Como resultado obteve a conclusão que o Suricata apresentou uma menor perda de pacotes do que o Snort. No entanto quando os softwares foram executados em um único *core* operando no limite o Snort perdeu menos pacotes por requerer menos recursos do sistema. O autor termina seu trabalho concluindo que a melhor solução ao analisar redes com taxas de transmissão maiores do que o Snort consegue processar

com um *core* seria utilizar de várias instâncias do Snort, com um core para cada. Conclui ainda que na época a ferramenta Suricata demonstrava potencial, mas seu desenvolvimento ainda não estava completo.

Thongkanchorn e Visoottiviseth (2013) em seu trabalho teve como objetivo comparar as ferramentas Snort, Suricata e Zeek, concluindo que as três ferramentas apresentam comportamento diferente perante os mesmos ataques, concluindo ainda que a presença de tráfego comum junto a tentativa de intrusão não afeta a capacidade de detecção das ferramentas e que o uso de um conjunto de regras diferentes modifica o número de alertas gerado pelas ferramentas.

Lakkaraju e Slagell (2008) em seu trabalho avaliou a utilidade de *traces* anonimizados para detecção de intrusão, concluindo que a utilidade do *trace* depende de qual o campo que foi anonimizado, sendo os campos de endereço IP e de PORTA os dois campos que causam o maior impacto na capacidade de um IDS de detectar uma tentativa de intrusão.

Foi possível observar que a literatura é rica em comparações entre as ferramentas Snort e Suricata. No entanto a ferramenta Zeek possui menos estudos a comparando com outras ferramentas. Um possível motivo disso é a diferença de paradigma seguido, enquanto que tanto Snort como Suricata fazem a detecção baseando-se em regras, a ferramenta Zeek não se prende a nenhum paradigma para monitorar redes como as outras ferramentas o fazem (GHAFIR et al., 2016). Deste modo este trabalho visa tanto enriquecer a literatura disponível com comparação envolvendo a ferramenta Zeek assim como fornecer uma comparação atualizada das ferramentas Snort e Suricata.

Capítulo 3

Desenvolvimento do Experimento

3.1 Descrição do Experimento

Para comparar as três ferramentas deste trabalho o seguinte experimento foi realizado. Foram obtidos de repositórios públicos *traces* (arquivos *pcap*) de tráfego de rede com tentativas de intrusão registradas. Esses *traces* foram divididos em 5 categorias de acordo com o tipo de tentativa de intrusão, essas categorias são:

- *Malware*: Propagação de pacotes contendo *Malware*.
- *Botnet*: Tráfego de computador comprometido, conectado a outros computadores na mesma situação, para realizar determinada tarefa.
- *SQL Injection*: Inserção de consulta SQL maliciosa.
- *Port Scan*: Varredura em rede de computadores, com o intuito de identificar quais computadores estão ativos e quais serviços estão sendo disponibilizados por eles.
- *Packet Injection - Man on the middle*: Inserção de pacotes manipulados em tráfego normal de rede.

Como os *traces* usados no experimento são anonimizados eles não podem ser usados para indicar o quão eficazes as ferramentas são ao avaliar tráfego não anonimizado (LAKKARAJU; SLAGELL, 2008). Mas como as ferramentas estão avaliando-os sob condições idênticas, os *traces* podem ser usados para comparar as ferramentas entre si, indicando o quão eficazes as ferramentas são quando comparadas com as outras.

Tabela 3.1: Descrição dos traces usados no experimento 1 (fontes especificadas no Apêndice A)

Trace (.pcap)	**Nº Pacotes**	**Tipo de Intrusão**	**Fonte**
metasploit-ms017-010-win7x64	608	Malware	1
eternalromance-success-2008r2	205	Malware	1
eternalromance-doublepulsar-meterpreter	740	Malware	1
eternalblue-success-unpatched-win7	576	Malware	1
eternalblue-failed-patched-win7	82	Malware	1
esteemedaudit-failed-XPSP2	3016	Malware	1
doublepulsar-backdoor-connect-win7	94	Malware	1
botnet-capture-20110810-neris	323154	Botnet	2
botnet-capture-20110811-neris	176064	Botnet	2
botnet-capture-20110812-rbot	495056	Botnet	2
botnet-capture-20110815-fast-flux	45853	Botnet	2
botnet-capture-20110815-rbot-dos	256712	Botnet	2
obfuscated_20596	8	Botnet	3
2017-12-29-Necurs-Botnet-malspam-traffic	308	Botnet	3
sqlserver-select-insert-update	33	SQL-Injection	4
SQL_MSSQL_2000_Hello_Buffer_Over	30	SQL-Injection	4
Vacation_Estate_Listing_Blind_SQL	96	SQL-Injection	5
SQL_Injection_in_Mobile_Space	11	SQL-Injection	5
joomla_sql_injection1	10	SQL-Injection	5
CVE-2008-0514_JoomlaMambo	27	SQL-Injection	5
Active_Business_Directory_RemoteBlind	22	SQL-Injection	5
portscan01	29	Port Scan	6
nmap_ACK_scan_on_port_80	6	Port Scan	7
nmap_ACK_scan_on_port_80_2	20	Port Scan	7
nmap_OS_scan	2056	Port Scan	7
nmap_OS_scan_succesful	2052	Port Scan	7
nmap_standard_scan	2004	Port Scan	7
nmap_zombie_scan	42	Port Scan	7
id1-cn_packet-injection	155	Packet Injection	8
hao123-com_packet-injection	202	Packet Injection	8
id1.cn-inject	9	Packet Injection	9
Uyan-1512	8	Packet Injection	10
Uyan-1511	7	Packet Injection	10
Tupian-2282	14	Packet Injection	10
Taobao-145	28	Packet Injection	10

Para a avaliação foram obtidos 7 *traces* em cada categoria de intrusão, com os mesmos *traces* sendo usados para todas as ferramentas. As três ferramentas possuem a capabilidade de avaliar arquivos no formato pcap, possibilitando a execução do experimento sem o uso de

ferramentas auxiliares. A tabela 3.1 traz a descrição dos traces usados para a comparação das ferramentas. Observa-se que a avaliação de detecção será feita por *trace* e não por pacote, isso se deve ao fato de que a quantidade de pacotes em cada trace não é padronizado.

Os comandos usados para a execução das ferramentas encontram-se no Apêndice B.

As ferramentas Snort e Suricata fornecem alertas de maneira semelhante, classificando estes alertas de acordo com a sua severidade. Essa severidade varia seu valor de 3 até 1 com 3 sendo a severidade mais baixa e portanto, menos importante. Para esse experimento apenas os alertas com severidade 2 e 1 foram considerados como detecção de sucesso.

A ferramenta Zeek no entanto não gera alertas mas sim arquivos que descrevem o tráfego analisado, estes arquivos são divididos na documentação da ferramenta em categorias, os arquivos na categoria *detection* foram considerados para avaliar se a ferramenta conseguiu ou não detectar a tentativa de intrusão.

Para avaliar as ferramentas em relação aos falsos positivos foi realizado o mesmo processo que na parte relativa à avaliação de detecção de intrusão, mas desta vez utilizando *traces* que apenas possuem pacotes inofensivos. Os *traces* utilizados estão descritos na tabela 3.2.

Tabela 3.2: Descrição dos traces usados no experimento 2 (fontes especificadas no Apêndice A)

Trace (.pcap)	**Nº Pacotes**	**Descrição**	**Fonte**
BitTorrent_Transfer	13	arquivo contendo pacotes de tráfico da aplicação BitTorrent	7
http	43	requisição http (download) e sua resposta via TCP	7
multicast	226	pacotes transmitidos via multicast	4
Obsolete_Protocols_Packets	10949	arquivo com pacotes de protocolos obsoletos (Banyan VINES, AppleTalk e DECnet)	7
dhcp-auth	1	oferta dhcp	11
smtp	60	pacotes do protocolo smtp	7
TeamSpeak2	24	pacotes do aplicativo TeamSpeak	7

Para avaliar a perda de pacotes em redes com altas velocidades for realizada a simulação da transmissão de um arquivo .pcap usando a ferramenta TCPReplay, que possibilita o controle da velocidade de transmissão de um arquivo do tipo .pcap para uma interface de rede. O arquivo pcap utilizado possui 495 mil pacotes, sendo este mesmo pacote sido utilizado para todas as velocidades simuladas. Esta avaliação foi realizada com as ferramentas Zeek, Snort e Suricata.

O processo de avaliação consistiu do envio do arquivo para a interface de rede em que a ferramenta (Zeek, Snort e Suricata) estava escutando. Para cada velocidade, em cada ferramenta, foram executados 10 testes de maneira a obter a média da perda de pacotes.

Foi realizado ainda um teste semelhante ao realizado com a ferramenta TCPReplay descrito acima. A diferença é que neste teste o mesmo arquivo contendo 495 mil pacotes foi enviado de um computador para outro através de cabo de rede do tipo par trançado. Esse envio foi realizado utilizando a ferramenta TCPReplay a uma velocidade de 100 Mbps.

O computador utilizado durante o experimento possui um processador de 64 bits da Intel, modelo i5-6200U 2.3 GHz, com 4 núcleos (cada núcleo com 2 threads), 8 GB de memória RAM e um cartão de interface de rede com suporte a taxa de transmissão de até 100 Mbps.

A versão da ferramenta Snort utilizada é a 2.9.7, configurada com as regras originais obtidas durante a instalação. A versão da ferramenta Suricata utilizada é 4.1.4 com as regras do *ruleset Emerging Threats Open*, essas regras foram obtidas em 05 de dezembro do ano de 2018. A versão da ferramenta Zeek utilizada é 2.6-130, configurada com os *scripts* de detecção obtidos durante a instalação da ferramenta. A versão da ferramenta TCPReplay utilizada é a 4.2.6.

3.2 Ameaças à Validade do Experimento

Para a análise dos resultados do experimento foi feito uso de análise estátistica, uma das características deste tipo de análise é que o resultado se torna mais representativo da realidade conforme o tamanho da amostra aumenta. Como já mencionado anteriormente o número de *traces* obtidos foi limitado, isso é uma ameaça à validade do experimento pois faz com que os resultados obtidos sejam representativos da amostra usada, mas menos representativa da população como um todo (realidade).

Outra ameaça é a falta de padronização dos *traces*, estes foram obtidos de diversas fontes e, portanto, passaram por processos diferentes de criação. Uma destas diferenças é o processo de anonimização dos *traces* e outra é a quantidade de pacotes que compõem cada *trace*. Essa falta de padronização pode ter influenciado o resultado final por fornecer informação não padronizada para a ferramenta de análise, dificultando a análise de alguns *traces* mais do que outros.

Vale ressaltar ainda que a pouca disponibilidade de *traces* torna difícil a obtenção de trá-

fego de rede com as ameaças mais atuais, o que pode causar com que os resultados obtidos neste experimento não reflitam a performance das ferramentas avaliadas perante ameaças mais recentes.

3.3 Resultado do Experimento

Através da execução do experimento foram obtidos os seguintes resultados.

Tabela 3.3: Porcentagem dos pacotes não analizados (*dropped*)

Velocidade(Mbps)	**Snort**	**d. padrão**	**Zeek**	**d. padrão**	**Suricata**	**d. padrão**
100	0	0	0	0	0	0
200	0	0	0	0	0	0
300	0	0	0	0	0	0
400(388)	0.97	1.05	0	0	0	0
500(468)	2.05	1.35	0	0	0	0
600(574)	2.69	1.61	0	0	0	0
700(624)	3.25	0.31	0	0	0	0
800(694)	5.03	0.41	0	0	0	0
900(761)	6.76	0.84	0.01	0	0	0
1000(826)	8.12	0.79	0.02	0.02	0	0

Na tabela 3.3 os valores entre parênteses na coluna Velocidade representam a velocidade real de envio, apesar de ferramenta TCPReplay permitir o controle da velocidade de envio, esta velocidade pode ser afetada por fatores externos (troca de contexto no processador por exemplo), o que acaba por afetar a velocidade real de envio. Desse modo o valor fora dos parênteses é a velocidade escolhida para envio e o valor dentro dos parênteses é a média da velocidade real obtida durante as 10 execuções em cada ferramenta.

Ouve um conflito quando realizado a execução do experimento envolvendo a ferramenta TCPReplay com a ferramenta suricata, esta ferramenta apresentou problemas ao escutar a interface de rede usada com as outras ferramentas. Desse modo o seu teste foi realizado de maneira diferente, ao invés de utilizar a ferramenta TCPReplay instalada na máquina, foi utilizado uma imagem docker desta ferramenta. A imagem enviou os pacotes de teste e a ferramenta suricata recebeu os pacotes pela interface de rede própria do docker. Vale observar que o uso da ferramenta docker não trouxe influência ao experimento, visto que a ferramenta suricata conseguiu

analisar todos os *traces* sem perda alguma. Na tabela 3.4 estão detalhados quais *traces* foram detectados corretamente pelas ferramentas.

Tabela 3.4: Detecção dos traces pelas ferramentas (X = conseguiu detectar)

Trace (.pcap)	**Snort**	**Suricata**	**Zeek**
metasploit-ms017-010-win7x64	X	X	
eternalromance-success-2008r2	X	X	
eternalromance-doublepulsar-meterpreter	X	X	
eternalblue-success-unpatched-win7	X	X	
eternalblue-failed-patched-win7	X		
esteemedaudit-failed-XPSP2	X		
doublepulsar-backdoor-connect-win7	X	X	
botnet-capture-20110810-neris	X		X
botnet-capture-20110811-neris	X		X
botnet-capture-20110812-rbot	X	X	X
botnet-capture-20110815-fast-flux	X		X
botnet-capture-20110815-rbot-dos	X	X	X
obfuscated_20596			
2017-12-29-Necurs-Botnet-malspam-traffic		X	X
sqlserver-select-insert-update			
SQL_MSSQL_2000_Hello_Buffer_Over	X	X	X
Vacation_Estate_Listing_Blind_SQL			
SQL_Injection_in_Mobile_Space			
joomla_sql_injection1			
CVE-2008-0514_JoomlaMambo			
Active_Business_Directory_RemoteBlind			
portscan01	X		X
nmap_ACK_scan_on_port_80			
nmap_ACK_scan_on_port_80_2			
nmap_OS_scan	X		X
nmap_OS_scan_succesful	X		X
nmap_standard_scan	X		X
nmap_zombie_scan			
id1-cn_packet-injection	X		X
hao123-com_packet-injection	X		X
id1.cn-inject	X	X	
Uyan-1512			
Uyan-1511			
Tupian-2282			
Taobao-145			X

Na tabela 3.5 está descrito a porcentagem de *traces* corretamente identificados durante o

experimento envolvendo a detecção de tentativas de intrusão.

Tabela 3.5: Porcentagem dos traces com intrusão detectados corretamente

Categoria	**Snort**	**Suricata**	**Zeek**
Malware	100%	71,43%	0%
Botnet	71,43%	42,86%	85,71%
SQL Injection	14,29%	14,29%	14,29%
Port Scan	57,14%	0%	57,14%
Packet Injection	42,86%	14,29%	42,86%

No teste realizado através do envio entre computadores diferentes utilizando cabo de rede par trançado, não houve perda de pacotes na velocidade de 100 Mbps. Outras velocidades não foram testadas por causa da limitação dos cartões de rede disponíveis, que são capazes de receber redes com taxas de transmissão de no máximo 100 Mbps.

3.4 Análise do Resultado

Para comparação entre as ferramentas em relação a detecção de intrusão será utilizado o teste estatístico de Kruskal-Wallis que indica se existe ou não diferença estatística entre os grupos sendo avaliados, mesmo quando estes grupos não necessariamente seguem uma distribuição normal (WITTE; WITTE, 2017). Neste caso os grupos avaliados são as três ferramentas sendo avaliadas, e os parâmetros usados para a execução do teste são o número de *traces* corretamente classificados em cada categoria de tentativa de intrusão.

Sumário do cálculo:

$$H = (12/(n(n+1)) * (\sum Ri^2/ni) - 3(n+1) \tag{3.1}$$

Onde *n* é a soma do número de amostras nos grupos, *Ri* é a soma dos valores das amostras do grupo i e *ni* é o número de amostras do grupo i.

$H = 0.05 * 998.7 - 48$

$H = 1.935$

O valor H obtido é de 1.935 (2, N = 15), com 2 (número de grupos menos 1) sendo o grau de liberdade (um dos parâmetros para se obter a conclusão) e com N sendo o número total de amostras. Ao utilizar a tabela X^2(qui-quadrado) esse valor H nos leva a conclusão de

que o resultado não é significativo com 95% de confiança, ou seja não há diferença entre as ferramentas.

Ao executar o teste de Kruskal-Wallis foi obtido como resultado que não há diferença entre as ferramentas com 95% de certeza. Isso não significa que as ferramentas são igualmente eficazes em detectar todos os tipos de intrusão, mas sim que no geral a quantidade de tentativas de intrusão que serão detectadas pelas ferramentas é estaticamente equivalente.

Quanto às categorias observa-se que a ferramenta Zeek não foi capaz de detectar nenhum dos traces da categoria *Malware* enquanto que a média da detecção das outras ferramentas foi de 85,71%. De mesmo modo a ferramenta Suricata não foi capaz de detectar nenhum dos traces da categoria *Port Scan* enquanto que a média da detecção das outras ferramentas foi de 57,14%.

Observa-se ainda que, ao definir qual ferramenta foi melhor em cada categoria, houve empate em três das cinco categorias (SQL Injection, port scan e packet injection), com a ferramenta Snort sendo a melhor ao detectar *traces* da categoria *malware* e a ferramenta Zeek sendo a melhor em detectar *traces* da categoria *Botnet*.

Quanto aos falsos positivos, nenhuma das ferramentas apresentou falso positivo. Nota-se no entanto que a ferramenta Snort não foi capaz de analisar o arquivo que continha pacotes de protocolos obsoletos, pois a ferramenta não mais fornece suporte a análise de pacotes destes protocolos.

No teste realizado para verificar o comportamento das ferramentas *Single threaded* em diferentes velocidades de rede, a ferramenta Zeek apresentou uma perda muito menor de pacotes quando comparado com a ferramenta Snort. A ferramenta Snort apresentou perda de pacotes a partir da velocidade de transmissão de 388 Mbps, já a ferramenta Zeek apenas apresentou perda de pacotes na velocidade 761 Mbps. A ferramenta Suricata não apresentou perda de pacotes.

Capítulo 4

Conclusão

Através dos resultados obtidos durante a execução dos experimentos algumas conclusões podem ser obtidas.

Quanto à detecção de ameaças a análise estatística demonstrou empate entre as três ferramentas, com nenhuma delas se diferenciando das outras estatisticamente, análise essa possuindo 95% de confiança.

Em relação aos falsos positivos, nenhuma das ferramentas apresentou detecção falsa em nenhum momento. Observa-se no entanto que a ferramenta Snort não mais oferece suporte a determinados protocolos, desse modo pacotes que usam esses protocolos não serão analisados pela ferramenta.

No teste realizado em diferentes velocidades de transmissão simuladas, a ferramenta Zeek demonstrou uma perda muito menor de pacotes, com perda acontecendo apenas em redes com altas taxas de transmissão (a partir de 761 Mbps). Já a ferramenta Snort demonstrou grande perda de pacotes (apenas 8% em uma rede com velocidade de 826 Mbps), começando a apresentar perda de pacotes em velocidades acima de 300 Mbps, o que corrobora a conclusão semelhante obtida por Kacha e Shevade (2012). A ferramenta Suricata não apresentou perda de pacotes, o que é o comportamento esperado desta ferramenta graças a sua arquitetura *multithread*.

Foi constatado durante a realização da pesquisa dificuldade na obtenção de *traces* corretamente catalogados e com a configuração necessária para a execução dos experimentos propostos. Isso acarretou na pequena quantidade de *traces* usados em cada uma das categorias de ameaças ao realizar o teste de detecção de tentativa de intrusão. A análise estatística apresenta resultados mais conclusivos conforme o número de amostras aumenta, dessa maneira a pequena

quantidade de *traces* utilizados acabou por enfraquecer a conclusão da análise estatística.

Assim sendo o desenvolvimento de uma base de dados com *traces* devidamente catalogados seria algo extremamente positivo para futuras pesquisas na área. Diversas fontes já existem, mas se encontram fragmentadas e, muitas vezes, mal catalogadas.

Futuros estudos podem ainda explorar uma comparação dos recursos computacionais utilizados pelas ferramentas sob condições variadas, tais como análise de *traces* de tamanhos diferentes e sob velocidades de transmissão diferentes. Uma outra oportunidade seria o uso de serviços baseados em *cloud* para fazer a análise dos *traces*, já que tais serviços disponibilizam interfaces de rede que suportam velocidades de transmissão mais altas.

Apêndice A

Lista de fontes usadas no experimento

A seguir estão especificadas as fontes usadas para obter os traces usados no experimento.

- 1: https://www.dropbox.com/sh/kk24ewnqi9qjdvt/AACj7AHJrDHQeyJTuo1oBqeQa
- 2: https://www.stratosphereips.org/datasets-ctu13
- 3: http://www.malware-traffic-analysis.net/
- 4: http://www.pcapr.net
- 5: http://www.pcapanalysis.com
- 6: https://github.com/markofu/pcaps/blob/master/PracticalPacketAnalysis/ppa-capture-files/portscan.pcap
- 7: https://wiki.wireshark.org/SampleCaptures
- 8: https://www.netresec.com/?page=PcapFiles
- 9: https://github.com/fox-it/quantuminsert/blob/master/presentations/brocon2015/pcaps/id1.cn-inject.pcap
- 10: http://www.cs.technion.ac.il/ gnakibly/TCPInjections/samples.zip
- 11: https://packetlife.net/captures/

Apêndice B

Comandos utilizados para a execução das ferramentas

A seguir estão especificados os comandos utilizados para executar as ferramentas durante os experimentos.

- 1: Comando usado para ler arquivo .pcap com a ferramenta Snort:

  snort -dev -l ./log -r <path para arquivo pcap> -c /etc/snort/snort.conf

- 2: Comando usado para ler arquivo .pcap com a ferramenta Suricata:

  suricata -k none -c /etc/suricata/suricata.yaml -r <path para arquivo pcap>

- 3: Comando usado para ler arquivo .pcap com a ferramenta Zeek:

  bro -C local -r <path para arquivo pcap>

- 4: Comando para executar a ferramenta TCPReplay:

  tcpreplay -i <interface de rede escolhida> -M <velocidade de Mbps a ser simulada> <path para arquivo pcap>

- 5: Comando para executar a ferramenta TCPReplay através de imagem Docker:

  sudo docker run –rm -t -v $(pwd):/data -i dgarros/tcpreplay /usr/bin/tcpreplay -M 1000 –intf1=eth0 botnet-capture-20110812-rbot.pcap

Para fazer com que as ferramentas (Snort, Suricata e Zeek) analizem os pacotes entrando via uma interface de rede, ao invés de ler de um arquivo .pcap, modificar a flag -r para -i e <path para arquivo pcap> pela interface de rede escolhida.

Referências Bibliográficas

ALBIN, E.; ROWE, N. C. A realistic experimental comparison of the suricata and snort intrusion-detection systems. In: *Anais da 26th International Conference on Advanced Information Networking and Applications Workshops*. Fukuoka: [s.n.], 2012.

DENARDIS, L. *A History of Internet Security*. 2007. Publicado no livro The History of Information Security: A Comprehensive Handbook.

GHAFIR, I. et al. A survey on network security monitoring sustems. In: *International Conference on Future Internet of Things and Cloud Worhshops*. Viena: [s.n.], 2016.

KACHA, C.; SHEVADE, K. A. Comparison of different intrusion detection and prevention dydtems. *International Journal of Emerfing Technology and Advanced Engineering*, n. 2, p. 243–245, 2012.

KAUR, S. *Design and Development of Policy Scripts to Detect Network Intrusions Using Bro*. Dissertação (Dissertação de Mestrado) — COPIN – Thapar University, Patiala - India, 2008.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach*. 6. ed. New Jersey: Pearson, 2012.

LAKKARAJU, K.; SLAGELL, A. *Evaluating the Utility of Anonymized Network Traces for Intrusion Detection*. 2008. Consultado na INTERNET: https://dl.acm.org/citation.cfm?id=1460899, acesso em 17 de Abril de 2018.

MURINI, C. T. *Análise dos sistemas de detcção de invasão em redes: Snort e Suricata comparando com dados da DARPA*. Dissertação (Trabalho de Conclusão de Curso) — COPIN – Universidade Federal de Santa Maria, Santa Maria - RS, 2014.

NAUGHTON, J. *A Brief History of the Future: The origins of the internet*. 1. ed. London: Phoenix, 2011.

PAXSON, V. Bro: a system for detecting network intruders in real-time. *Computer Networks*, n. 31, p. 2435–2463, 1999.

ROESCH, M. Snort - lightweight intrusion detection for networks. In: *Anais da 13th Systems Administration Conference*. Washington: [s.n.], 1999.

ROS, P. B.; CARELA-ESPAñOL, V.; BUJLOW, T. *Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification*. 2013. Consultado na INTERNET: https://www.ac.upc.edu/app/research-reports/html/RR/2013/37.pdf, acesso em 16 de Abril de 2018.

ROS, P. B.; CARELA-ESPAñOL, V.; BUJLOW, T. *Independent comparison of popular DPI tools for traffic classification*. 2014. Consultado na INTERNET: https://www.sciencedirect.com/science/article/pii/S1389128614003909, acesso em 16 de Abril de 2018.

SALMON, A.; MACLAFFERTY, M.; LAVESQUE, W. *Applied Network Security*. 1. ed. Birmingham: Packt Publishing, 2017.

SING, N. *Intrusion Prevention Scripts to Detect Malicious Traffic using Bro*. Dissertação (Dissertação de Mestrado) — COPIN – Thapar University, Patiala - India, 2008.

TANENBAUM, A. S. *Computer Networks*. 5. ed. Boston, Massachusetts: Prentice Hall, 2011.

WITTE, R. S.; WITTE, J. S. *Statistics*. 11. ed. Hoboken, NJ: John Wiley I& Sons, 2017.