Análise Conceitual e Comparativa entre Sistemas de Detecção de Intrusão

Lucian Beraldo[1] **, Diego Pedroso**[1] **, Felipe Gobo Bruno**[1] **, Fernando Tadao Ito**[1]

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil

{mnvlucian,pedrosodiego,fgobobruno,f.tadaoito}@gmail.com

Abstract. *In recent years, the amount of known incidents of computer security is rapidly increasing, so as their damage inflicted on the organizations, with the financial matter posing as the main focus for the attackers. To fill this gap, this paper shows the field of Intrusion Detection Systems (IDS), providing the necessary theoretical background and the tools to analyze the computer networks traffic looking for suspicius activities. For this project experimentation, a local network was implemented at the Computer Science Department at the Federal University of São Carlos (UFSCar), where an attacker's system attacks an user's system, with an intermediary system capturing the network packets, which are mixed with legitimate traffic from the lab users. By the end of this paper, the generated alerts and the processing and memory resources used by the Snort, Suricata and Bro IDS are correlated, resulting in a better performance for the Suricata system, it's also provided a comparison between them, enlighting their main characteristics.*

Resumo. *Nos últimos anos, a quantidade de incidentes envolvendo segurança cibernética tem aumentado vertiginosamente, assim como os prejuízos causados por tais incidentes, junto ao crescente foco em ganhos financeiros por parte dos envolvidos na execução de ataques cibernéticos. Nesse sentido, o presente artigo tem por objeto de estudo os Sistemas de Detecção de Intrusão, ou Intrusion Detection Systems (IDS), estabelecendo um referencial teórico sobre a área focando no entendimento das ferramentas utilizadas para analisar tráfego suspeito em redes computacionais. Para a experimentação do projeto, foi estabelecida uma rede local, implementada no Departamento de Ciência da Computação da Universidade Federal de São Carlos (UFSCar), realizando ataques cibernéticos de conhecimento público a partir de um sistema atacante direcionados para um sistema alvo a fim de capturar os pacotes de ataques misturados com pacotes legítimos de tráfego, identificando atividades maliciosas através dos sistemas de detecção testados. Ao final do artigo, a quantidade de alertas gerados e os recursos de processamento e memória utilizado pelos IDS Snort, Suricata e Bro são correlacionados, mostrando um uso mais eficiente de recursos pelo sistema Suricata, também são destacados os aspectos mais importantes de cada ferramenta.*

1. Introdução

Nos últimos anos a quantidade de incidentes envolvendo a segurança cibernética tem aumentado vertiginosamente, assim como os prejuízos causados por tais incidentes

[Websense Security Labs 2015], movidos pelo crescente foco em ganhos financeiros por parte dos envolvidos na execução de ataques cibernéticos. De acordo com [Panes 2011], conforme a utilização de sistemas de informação cresce, a proteção dos dados gerados por tais sistemas requer ferramentas e conceitos de segurança cada vez mais eficientes e sofisticados. Diante desse cenário, cria-se a necessidade de se proteger as informações e os sistemas que permeiam o espaço cibernético, visto o risco ao qual eles estão sujeitos, por manipulação não autorizada de mensagens com dados sensíveis entre clientes e sistemas, negação de serviços essenciais por meio de inundação de mensagens de controle, etc. As organizações têm percebido tal crescimento na ameaça de cibercrimes e atentados contra seus sistemas, evidenciado pelo constante crescimento no número de vagas para postos de trabalho no ramo de segurança cibernética [Burning Glass Technologies 2015], estando dispostas a investir cada vez mais em meios de proteção. De acordo com [Lobato et al. 2015], quanto mais tempo se leva para detectar as invasões aos sistemas computacionais, mais substanciais serão os riscos de ocorrerem danos irreparáveis aos ativos de indivíduos e organizações atacados.

Em contrapartida, todos os ataques deixam vestígios, permitindo que muitas vezes sejam detectados antes de causar danos consideráveis. Tentando sanar tal problema da detecção tardia das ameaças, o monitoramento e análise passivos permitem identificar vestígios de transações suspeitas no tráfego da rede, permitindo um plano de ação antes que os impactos causados por essa ameaça sejam irreparáveis. Organizações e pesquisadores se empenham para desenvolver novas técnicas de detecção e aperfeiçoar as já existentes, porém os atacantes também desenvolvem novos mecanismos para driblar a detecção e não serem encontrados, constituindo uma disputa constante entre organizações e atacantes.

Uma das ferramentas mais populares para análise e proteção de redes são os Sistemas de Detecção de Intrusão, ou *Intrusion Detection Systems* (IDS), que utilizam de análise passiva e ativa, permitindo encontrar comportamentos suspeitos no tráfego de rede antes mesmo de haver assinaturas mapeadas descrevendo tais formas de invasão (*zero-day exploit*), ou ainda quando há uma nova forma de *malware* que ainda não foi totalmente explorada por técnicas de engenharia reversa [Syversen 2008]. Seguindo tal tendência, o artigo apresentado tem por intento a análise conceitual e comparativa entre Sistemas de Detecção de Intrusão, dentro de uma rede controlada e com tráfego real, focando no entendimento do processo e monitoração de fluxos e tráfego de rede. Para a experimentação do projeto, foi estabelecida uma rede local controlada no laboratório *Asgard* do Departamento de Ciência da Computação da Universidade Federal de São Carlos (UFSCar), realizando ataques amplamente conhecidos, a partir de um sistema atacante e direcionados para um sistema alvo, a fim de capturar os pacotes trafegados e identificar tais ataques maliciosos.

O artigo apresentado está organizado da seguinte forma: A seção 1 mostra a introdução desse trabalho, evidenciando o contexto que motivou a escolha do tema; a seção 2 apresenta conceitos de sistemas de detecção de intrusão e diferencia os seus diversos tipos; a seção 3 descreve os métodos utilizados para a implementação do comparativo de sistemas de detecção e, juntamente, descreve as particularidades de cada uma das ferramentas que foram usadas em conjunto; na seção 4 são correlacionados o resultados gerados pelos sistemas de detecção, destacando os aspectos mais importantes de cada

ferramenta e suas particularidades; por fim, a seção 5 conclui o artigo.

2. Sistemas de Detecção de Intrusão

Os sistemas de detecção de intrusão têm por função primordial, detectar em tempo real, comportamentos não autorizados dentro de uma rede ou dentro de um mesmo sistema [Northcutt et al. 2004]. Para chegar a tal objetivo, são necessários a coleta e o processamento de todo o tráfego dos dados que transitam pela rede, comparando-os com assinaturas ou comportamentos conhecidos, a fim de encontrar atividades maliciosas e acessos não autorizados aos recursos da rede, através de um serviço de monitoramento.

O conceito de Sistemas de Detecção de Intrusão começou a ser cunhado em meados dos anos 80, em resposta aos crescentes números de invasões nos sistemas conectados à rede *Advanced Research Projects Agency Network* (ARPANET) utilizada na época. No contexto dos sistemas de informação, o termo intrusão se referencia a indivíduos e sistemas que utilizam recursos computacionais sem autorização, assim como indivíduos legítimos, ou seja, autenticados, que abusam de seus privilégios.

Os mecanismos e técnicas de ataque cibernético evoluem na mesma proporção que o desenvolvimento de novas tecnologias para a defesa desses ataques. [Mitnick and Simon 2003] apontam em seu trabalho que, para que a política de segurança de uma organização esteja sempre em dia, é essencial que sejam estabelecidos procedimentos rotineiros que visem identificar novas ameaças e, dessa forma, combatê-las com procedimentos e ferramentas adequados. Inseridos em tal cenário, mostra-se necessária a substanciação de ferramentas como os Sistemas de Detecção de Intrusão como um meio de proteção para os ativos computacionais de indivíduos e organizações. Os sistemas detectores de intrusão podem ser classificados pelas suas funcionalidades em três grupos, como será apresentado na seção 2.1 a seguir.

2.1. Sistema de Detecção de Intrusão Baseado em Rede (*Network Intrusion Detection* - NID)

Os Sistemas de Detecção de Intrusão Baseados em Rede capturam em modo promíscuo todos os pacotes que circulam através da rede à qual tem acesso, nos vários meios de comunicação e protocolos, configurando uma proteção de perímetro na rede. Após a captura, os pacotes são então analisados utilizando-se várias regras, chamadas de assinaturas, que descrevem quais podem ser os comportamentos de uma intrusão ou anomalia no cenário monitorado [Innella 2016].

Alguns exemplos de Sistema de Detecção de Intrusão Baseado em Rede podem ser encontrados em [The Snort Team 2016], [Chung et al. 1995], [Handley et al. 2001], [Paxson 1999], [Ptacek and Newsham 1998]. A seguir são descritas algumas vantagens e desvantagens desses sistemas detectores:

Prós:

- Pouco ou nenhum impacto nos sistemas ou na rede monitorada, por ser um equipamento dedicado que faz a análise;
- Possibilidade de blindagem contra invasão pelos *hosts* sendo observados na rede;
- Possibilidade de uso em modo invisível, sem atribuição de endereço na interface de rede em modo promíscuo.

Contras:

- Nem todas as interfaces de rede possuem o modo promíscuo, necessário para esse método;
- Existe a necessidade da utilização de conexão do tipo *Switched Port Analyser* (SPAN) nos *switchs* para captura de todo o tráfego. Tal funcionalidade permite que o tráfego que passa em um *switch* seja clonado e direcionado para apenas uma porta de comunicação do mesmo, evitando que alguns pacotes sejam perdidos;
- Carga de processamento pode ser excessiva em sistemas com tráfego de rede maior que *Gigabit*, o monitoramento torna-se muito complexo quando o tráfego é criptografado e também pode conter restrições de legislação quanto ao monitoramento de determinadas redes.

2.2. Sistema de Detecção de Intrusão Baseado em *Host* (*Host-based Intrusion Detection* - HID)

Os Sistemas de Detecção de Intrusão Baseados em *Host* possuem como propósito principal o monitoramento, detecção e resposta em um sistema (*host*) ou conjunto de sistemas específicos. Esse tipo de sistema oferece gerenciamento das permissões de acesso, centralização dos alertas emitidos, possibilidade de aquisição de dados para análise forense, evidências criminais, análises estatísticas e controle de acesso. Alguns sistemas baseados em *host* comparam as assinaturas dos arquivos no sistema analisado com uma base de assinaturas conhecidamente maliciosas, outros analisam o comportamento dos pacotes de rede procurando por anomalias [Innella 2016] que possam caracterizar ataques internos na rede [Northcutt et al. 2004].

Alguns exemplos de sistemas baseados em host podem ser encontrados em [Lee and Stolfo 1998], [Forrest et al. 1996], [Hofmeyr et al. 1998], [Ghosh et al. 1999], [Warrender et al. 1999]. Seguem algumas características a serem observadas:

Prós:

- Nem todas interfaces de rede possuem modo promíscuo e esse método não precisa do modo promíscuo;
- Menor carga de execução para a Central de Processamento Unitário (CPU) e memória em comparação a um NID, que processa pacotes de outros sistemas em toda a rede analisada;
- Configuração específica para cada *host*, tendo menor chance de ocorrer falsos positivos;
- Detecta modificações nos arquivos do sistema, monitorando mudanças no *checksum*, processos, carga de processamento e atividades de rede do *host*;
- Capacidade de analisar dados que trafegam na rede criptografados, pois os dados geralmente são descriptografados no *host*.

Contras:

- Deve possuir suporte ao sistema operacional de cada *host* e pode possuir carga considerável para recursos do sistema local, uma vez que irá disputar por recursos com outras aplicações do sistema;
- Alta complexidade para redes com muitos sistemas monitorados, a quantidade de alertas pode ser um problema se não for bem gerenciada.

2.3. Sistema de Detecção de Intrusão Híbrido ou Distribuído (*Hybrid Intrusion Detection*

Os sistemas de detecções híbridos, ou distribuídos, oferecem as formas de ação tanto do Sistema de Detecção de Intrusão Baseado em Rede quanto do Sistema de Detecção de Intrusão Baseado em *Host* e são distribuídos por toda a rede de forma a disseminar o monitoramento e o tornar mais eficaz [Innella 2016]. Tal categoria de sistema agrega os prós dos dois formatos descritos anteriormente, porém necessita de gerenciamento eficiente, devido a quantidade excessiva de alertas, o que é comumente feito concentrando-se os alertas gerados em uma central unificada.

Alguns exemplos de sistemas de detecção híbridos podem ser encontrados em [Hwang et al. 2007], [Aydın et al. 2009], [Gómez et al. 2009], [Kim et al. 2014], [Zekrifa 2014]. A seguir são descritas algumas vantagens e desvantagens de tais sistemas detectores:

Prós:

- Pode descentralizar a detecção de invasões distribuindo sensores em toda a rede, aumentando a eficácia;
- Pode ter menor carga de processamento que existiria em abordagens com um único servidor centralizado, barateando o *hardware* utilizado;
- Potencial para maior eficácia na detecção, uma vez que pequenas anomalias nas redes locais de uma organização podem ser detectadas.

Contras:

- Maior quantidade de equipamentos dedicados à detecção de intrusões, dificultando o gerenciamento;
- Grande quantidade de alertas gerados pelos muitos sensores de detecção;
- Maiores custos de operação com equipe treinada para análise do grande volume de alertas;
- Maior complexidade de implementação e manutenção.

2.4. Sistema de Detecção de Intrusão em Linha (*Inline*)

Os Sistemas de Detecção de Intrusão em Linha (*inline*), comercialmente conhecidos como Sistemas de Prevenção de Intrusão (*Intrusion Prevention Systems* - IPS), são sistemas que podem aplicar medidas de bloqueio no tráfego malicioso após a geração de um alerta de ataque, com uma resposta ativa. Tais sistemas são geralmente aplicados nos dispositivos de rede em linha, como *Firewall* e *switchs*, que possuem a capacidade de descartar pacotes individuais suspeitos de serem ataques, enquanto os mesmos passam pela interface do dispositivo [Northcutt et al. 2004].

As funções de resposta ativa a ataques não precisam necessariamente estar implementadas nos dispositivos em linha, algumas sessões podem ser interrompidas por um pacote de reinício enviado pelo sistema de detecção de intrusão, ou também podem ser modificadas nas Listas de Controle de Acesso (*Access Control Lists* - ACLs) de roteadores, a fim de bloquear o endereço de origem ou destino.

O objetivo em comum de tais funcionalidades é tomar ações automáticas de resposta a ataques detectados, consequentemente tentando minimizar ou, idealmente extinguir, os prejuízos causados, sem haver a necessidade de um administrador de sistemas especialista que identifique o problema e tome a ação necessária [Northcutt et al. 2004].

Alguns exemplos de sistemas de detecção *inline* podem ser encontrados em [Zhang et al. 2004], [Tajalli et al. 2003], [Jain 2004b], [Jain 2004a], [Brock et al. 2001]. Seguem algumas características a serem observadas:

Prós:

- A detecção e a ação necessárias para as invasões em andamento são tomadas automaticamente;
- Possibilidade de ações serem tomadas antes que danos significativos sejam causados;
- Menor dependência de especialistas para a análise dos alertas.

Contras:

- Em casos de falsos positivos, serviços importantes podem ser interrompidos pelas ações automáticas;
- Complexidade elevada de implementação, necessitando descrever ações para cada tipo de ataque.

3. Método

O objetivo do presente artigo é comparar a eficácia de três sistemas de detecção de intrusão diferentes em um mesmo ambiente de rede. Para tanto, o modelo foi arquitetado para analisar os pacotes trafegados em uma rede local implementada no laboratório Asgard do departamento de Ciência da Computação da Universidade Federal de São Carlos (UFSCar), no campus de São Carlos, durante o período de uma hora e vinte e seis minutos. Os comandos usados para a simulação dos ataques foram escritos na linguagem de *scripts Shell* para o sistema operacional Ubuntu, esses programas são oriundos da tese de mestrado em ciência da computação de [Ferreira 2016] que fazem a automatização de um conjunto de ataques pré-programados, operando em ordem aleatória de tempo e evitando *bias* pelo sistema detector.

Os ataques utilizados nos experimentos do projeto incluíram ataques de Negação de Serviço (*Denial of Service* - DoS) através da inundação de pacotes com protocolo *Internet Control Message Protocol* (ICMP); ataques de força bruta utilizando a ferramenta Arachni para descobrir as credenciais de autenticação de um servidor Web de testes implementado na linguagem de programação Python versão 3; e por fim, ataques de varredura de portas de conexão utilizando a ferramenta NMAP versão 0.3.1-1.

A topologia de rede utilizada na execução dos experimentos, apresentada na Figura 1, mostra os seguintes componentes integrantes:

- O item “Usuário” representa um sistema que recebe os ataques gerados pelo sistema “Atacante”. Para tal função, foi utilizado um servidor com processador Intell Xeon de 8 núcleos de 3.5GHz, 16GB de memória RAM DDR3 e sistema operacional Ubuntu 16.04.4;
- “IDS-Server” é o servidor que contém os sistemas de detecção de intrusão executando em paralelo. Para tal, foi usado um servidor com processador Intel Xeon de 4 núcleos de 3.5GHz, 4GB de memória RAM DDR3 e sistema operacional Ubuntu Server 16.04.4;
- “Router 1” é um roteador comum sem fio que conecta a rede local do experimento a outros computadores do laboratório que estão conectados à “*Cloud* Asgard”;

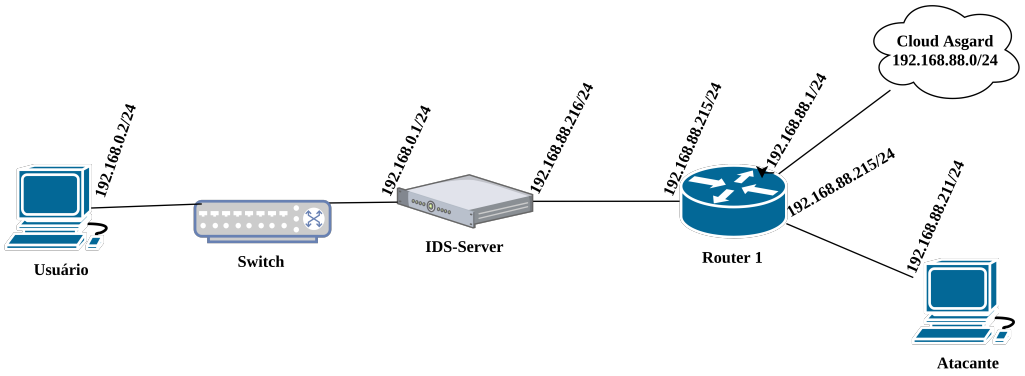

Figura 1. Topologia utilizada para a realização dos experimentos.

- "Atacante" é um servidor Intel Corei7, com 12 núcleos de 3.30GHz e 64GB de memória RAM DDR4, que gera os ataques direcionados para "Usuário";
- "Switch" é um *switch Ethernet* com 1Gbps de largura de banda.

Os sistemas de detecção de intrusão instalados no sistema "IDS-Server" que foram utilizados para a verificação são apresentados nas seções subsequentes.

3.1. Snort

O Snort é um Sistema de Detecção de Intrusão baseado em Rede (NID), foi criado em 1998 por Martin Roesch da Sourcefire, hoje parte da empresa Cisco, é compatível com a maioria dos sistemas operacionais disponíveis no mercado, sendo a versão 2.9.7.0 GRE (*Build* 149) utilizada no projeto. É construído na linguagem de programação C devido à necessidade de boa performance na análise de grande volume de pacotes de rede, o mesmo é composto por três módulos básicos [Northcutt et al. 2004]:

- Farejador de pacotes (*package sniffer*): Esse módulo simplesmente lê os pacotes da rede e os direciona para a saída padrão (*Standard Output*) do sistema na forma de texto;
- Registrador de pacotes (*package logger*): Armazena os pacotes capturados pelo farejador em disco local ou remoto, em arquivos binários ou, através de uma interface de programação para aplicações (*Application Programming Interface* - API), com a extensão ".pcap";
- Sistema detector de intrusão baseado em rede: É o módulo mais complexo e configurável do sistema, permitindo a exibição de alertas em tempo real através da análise de tráfego comparando com assinaturas maliciosas.

É um sistema robusto e de código aberto baseado na Licença Geral Pública do GNU (*General Public License* - GPL), o que possibilita o desenvolvimento de módulos adicionais personalizados para gerenciamento de *logs* e alertas, também conhecidos como *add-ons*. O Snort tem o Protocolo de Transmissão de Controle (*Transmission Control Protocol* – TCP) e o Protocolo de Internet (*Internet Protocol* - IP) como principal linguagem de comunicação, porém também analisa pacotes *User Datagram Protocol* (UDP) e ICMP, suportando também outros protocolos menos comuns, como *Address Resolution Protocol* (ARP), 802.11 de comunicação sem fio, *Hypertext Transfer Protocol* (HTTP), entre outros, através de extensões criadas por terceiros [The Snort Team 2016].

As regras utilizadas no Snort são divididas em duas partes lógicas: o cabeçalho e a opção de regra. A estrutura do cabeçalho, apresentada na Figura 2, contém a ação da regra, o protocolo a ser verificado, os endereços IP de origem e destino, as máscaras de rede aplicadas, as portas de origem e destino verificadas.

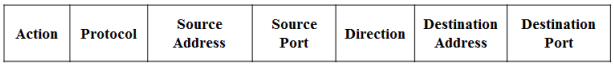

Figura 2. Estrutura do cabeçalho regra Snort
Fonte: [Khamphakdee et al. 2016]

Além disso, o campo ação do cabeçalho da regra também é capaz de definir o tipo de ação a ser tomada, como ignorar o pacote sem exibir alerta, exibir o alerta mesmo o pacote não sendo suspeito, etc. As opções de regra seguem o cabeçalho de regra e estão dentro de um par de parênteses. Em geral, cada opção de regra consiste em duas partes: uma palavra-chave e um argumento.

A Figura 3 mostra um exemplo da regra no formato do módulo Haxixe utilizada no Snort. Esta regra vai gerar um alerta se o IP de origem 172.16.115.50 é detectado comunicando a partir de qualquer porta, direcionando o pacote para qualquer endereço IP de destino e o número da porta de destino for igual a 53 (protocolo de nomes de domínio ou *Domain Name Server* DNS). Assim que as regras do cabeçalho forem aplicadas com sucesso, será exibido o alerta "tentativa de solicitação de DNS" e o número identificador da regra será "sid: 1.000.010".

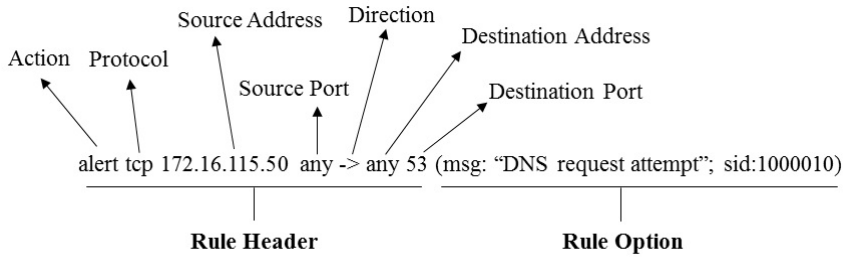

Figura 3. Exemplo de regra do Snort
Fonte: [Khamphakdee et al. 2016]

Devido ao fato de ser um sistema de código aberto e possuir relativa facilidade para criar novas assinaturas de intrusões, o Snort possibilita a criação de assinaturas em poucos dias ou até algumas horas após a divulgação de uma vulnerabilidade, permitindo aos seus usuários a rápida atualização do inventário de assinaturas maliciosas, aumentando a eficácia do sistema. Em alguns casos, é possível até a detecção dos sinais da propagação de um *malware* antes de a assinatura para o mesmo estar disponível pois, dependendo da forma de ação do software malicioso, o mesmo pode ativar os alarmes do sistema de detecção [Northcutt et al. 2004].

3.2. Bro

Embora voltado para o monitoramento de segurança de rede, o sistema baseado em rede (NID) Bro fornece uma plataforma abrangente para análise de tráfego de rede, tem su-

perado com sucesso a diferença tradicional entre a academia e o mercado desde a sua criação. Atualmente o sistema é utilizado de forma operacional por muitos ambientes científicos para proteger sua infraestrutura cibernética, foi escolhida a versão 2.4.1 nos experimentos.

A comunidade de usuários do Bro inclui grandes universidades, laboratórios de pesquisa, centros de supercomputação e comunidades de ciência aberta. A ferramenta foi originalmente desenvolvida por Vern Paxson, que continua a liderar o projeto agora em conjunto com uma equipe de núcleo de pesquisadores e desenvolvedores do Instituto Internacional de Ciência da Computação em *Berkeley, CA*; e do Centro Nacional de Aplicações de Supercomputação em *Urbana-Champaign, IL*. O Projeto Bro é um membro do projeto *Software Freedom Conservancy*, que é uma organização sem fins lucrativos criada com a finalidade de apoiar e proteger projetos *Free/Libre/Open Source Software* (FLOSS). Nos dizeres de [Lobato et al. 2015], o Bro consiste de uma linguagem para a detecção de ameaças por anomalias, porém as formas utilizadas para se detectar tais ameaças devem ser implementadas pela comunidade de usuários. O Bro também permite a integração com Redes Definidas por Software (*Software Defined Networks* - SDN), permitindo a detecção de intrusões em redes virtuais utilizadas em vários *Data Centers* [Paxson 1999].

É uma ferramenta de monitoramento e análise em tempo real do tráfego de rede sumarizado em fluxos (*flows*), onde os pacotes são agrupados de acordo com o endereço e porta de origem e destino, se mostrando eficiente em ambientes com redes de alta velocidade. O Bro possui uma linguagem de programação orientada a redes, o que facilita a abstração dos fluxos. Em suma, ele atua como um caracterizador de fluxos, sintetizando as informações dos pacotes que entram pelas interfaces de rede do sistema e agrupando as informações em janelas de tempo.

3.3. Suricata

O sistema baseado em rede (NID) Suricata é um mecanismo de monitoramento de segurança de redes, de código aberto (*open source*) e de propriedade de uma fundação sem fins lucrativos, a Fundação de Segurança da Informação Aberta (*Open Information Security Foundation* - OISF). É altamente escalável e *multi threaded*, isso significa que é possível executar uma instância do programa e equilibrar a carga de processamento em cada processador em um sensor Suricata configurado. Isso permite que o hardware atinja velocidades de até 10 Gigabit sobre o tráfego real, sem sacrificar a cobertura conjunta de regras [Open Information Security Foundation 2016].

É possível combinar palavras-chave específicas em campos de protocolo, para aumentar o leque de protocolos suportados, que vão desde o HTTP a identificadores de certificado *Secure Socket Layer* (SSL). Os protocolos mais comuns são automaticamente reconhecidos pelo sistema quando o fluxo começa a ser capturado, isso faz do Suricata um excelente detector de *malwares*.

O Suricata também pode identificar milhares de extensões de arquivos durante a varredura na rede, é possível então marcar esse arquivo para a extração e o mesmo será gravado em disco como um arquivo de dados, seus metadados são descritos na captura em flows e um identificador único (*checksum*) do arquivo é calculado em tempo real. Com esse processo, caso exista uma lista de arquivos mapeados a partir de seus identi-

ficadores que se deseja manter na sua rede sem que haja alteração do mesmo, ou caso queira manter esses arquivos fora da rede, o sistema de detecção poderá encontrá-los. A documentação do sistema [D. and B. 2011] explicita que o Suricata é implementado utilizando a aceleração da placa gráfica (*Graphics Processor Unit* - GPU) da máquina através das plataformas de desenvolvimento de aplicações (API) *Computer Unified Device Architecture* (CUDA) e também o *framework Open Computing Language (OpenCL)*. O Suricata também possui implementação para o uso de descompressão de pacotes com o formato Gzip, detecção automática dos protocolos e variáveis de fluxos trafegados pela rede, suporta bibliotecas HTTP independentes e compreende o protocolo *Fast Local Internet Protocol* (FLIP).

De acordo com [Kasinathan et al. 2013], o Suricata desencadeia alertas com base nas regras programadas, então, pode-se desenvolver regras específicas para detectar ataques de inundações e outros tipos específicos. O sistema foi arquitetado para detectar ataques a redes tradicionais *Ethernet* e, portanto, não foi concebido para interpretar redes sem fio ou protocolos mais recentes como o IPV6.

4. Resultados

Foram executadas as ferramentas Bro, Snort e Suricata em conjunto, em um mesmo ambiente de rede controlado, durante um período pré definido de uma hora e vinte e seis minutos, utilizando a topologia apresentada na Figura 1. De acordo com a Tabela 1 apresentada a seguir, a quantidade total de pacotes trafegados na rede durante o período de análise, contabilizando os pacotes de ataques e de navegação dos usuários da "Cloud Asgard", foi estimada em 68.915.167 pacotes, a quantidade de alertas gerados pelo IDS Bro foi de 164.824, do IDS Snort foi de 63.241.244 alertas e do IDS Suricata 6.804 alertas. A quantidade de alertas gerada pelo Snort e o Suricata, devido ao fato deles utilizarem a mesma base de assinaturas, são ordens de magnitude diferentes, pois o Snort exibe alertas para cada pacote suspeito que ele encontra, já o Suricata e o Bro sumarizam os alertas agrupando alertas similares.

A média de consumo de CPU apresentada na Tabela 1 compreende a CPU em sua totalidade, contabilizando o consumo de todos os 4 núcleos de processamento em conjunto, isso pode ser observado pela soma do consumo dos sistemas de detecção exceder 100%. Pode-se observar que o Suricata é o sistema que menos usa recursos de CPU e RAM, talvez devido à sua forma de verificação em *threads*, utilizando mais eficientemente os núcleos do sistema. Devido ao fato inesperado de o Bro e o Suricata agruparem os alertas similares de pacotes suspeitos, a abordagem utilizada para medir a eficácia dos Sistemas de Detecção de Intrusão pela quantidade de alertas gerados não foi uma boa escolha.

Tabela 1. Comparativo de processamento, memória RAM e alertas capturados.

	Bro		**Snort**		**Suricata**	
	CPU %	**RAM %**	**CPU %**	**RAM %**	**CPU %**	**RAM %**
Média	106,9	3,2	91,3	3,0	33,2	2,5
Desvio Padrão	7,3	0,5	8,9	0,4	4,8	0
Alertas Capturados	164.824		63.241.244		6.804	
Total Pacotes Capturados	68.915,167					

O gráfico apresentado na Figura 4 apresenta a porcentagem de memória RAM

consumida por cada uma das ferramentas utilizadas ao longo de cerca de uma hora e vinte e seis minutos (5271 segundos) de coleta e análise dos pacotes trafegados pela rede. É possível notar que o Snort diminui gradualmente a quantidade de memória utilizada ao longo do tempo, já o Suricata apresenta um uso constante de memória.

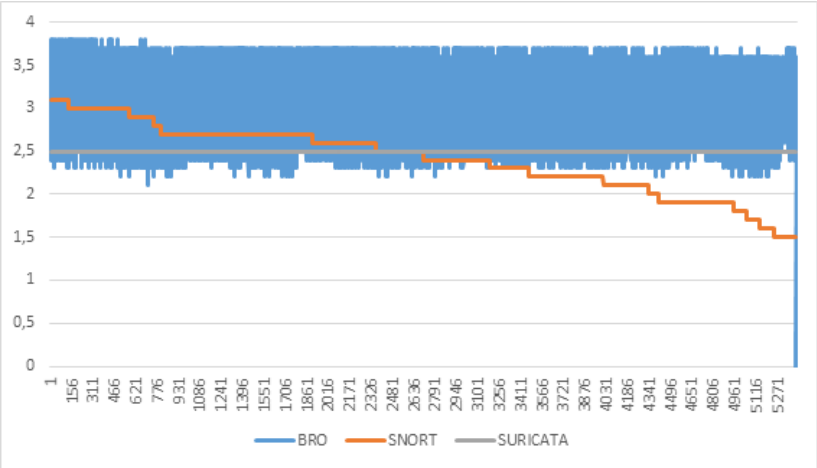

Figura 4. Comparativo de memória RAM consumida durante execução das ferramentas.

A Figura 5 mostra a evolução do consumo de recursos da CPU durante o período de cerca de uma hora e vinte e seis minutos (5249 segundos) de testes. É possível observar que a ferramenta Snort apresenta um consumo de CPU bastante variável, porém nunca ultrapassa os 100% de consumo, mostrando que o mesmo não é capaz de utilizar recursos de múltiplos núcleos de processamento na versão utilizada sem *plugins* adicionais; o Bro mostra a capacidade de utilização de mais de um núcleo de processamento do sistema, porém é o que mais consumiu recursos de CPU; por fim o Suricata é o IDS que melhor gerencia os núcleos de processamento, dividindo as tarefas e consumindo menos recursos. Analisando-se a Figura 5, nota-se que em determinados períodos de tempo, há um declínio no consumo da CPU por parte de todos os sistemas de detecção, isso acontece devido a uma função implementada em todos os *scripts* de ataque chamada de *sleep*, ela faz com que os ataques sejam cessados em períodos de tempo aleatórios a fim de evitar que os sistemas de detecção prevejam quando um ataque ocorrerá. Desta forma, não havendo ataques nesses intervalos, o consumo de recurso computacional tem uma queda parcial e, quando esse intervalo termina, o sistema volta a consumir os mesmos recursos de processamento que consumia anteriormente.

De acordo com os resultados obtidos com o experimento, conclui-se que o Suricata cumpre a tarefa de alertar sobre os ataques realizados, exigindo menos processamento e memória RAM e também emitindo menos alertas ao usuário do que os outros IDS testados. O Bro, por sua vez, oferece mais generalização para captura de pacotes utilizando sua linguagem proprietária de *scripts*. A utilização do Bro torna-se mais complexa, pois há necessidade de utilizar os *scripts* para uma análise mais detalhada dos pacotes e, como não existe uma base unificada de assinaturas, tem-se uma tarefa adicional de procurar em fontes externas. O Suricata gera menos alertas pois, ao invés de tratar cada pacote suspeito como um alerta de comportamento malicioso, ele interpreta a sequência de pacotes similares como uma única sequência maliciosa.

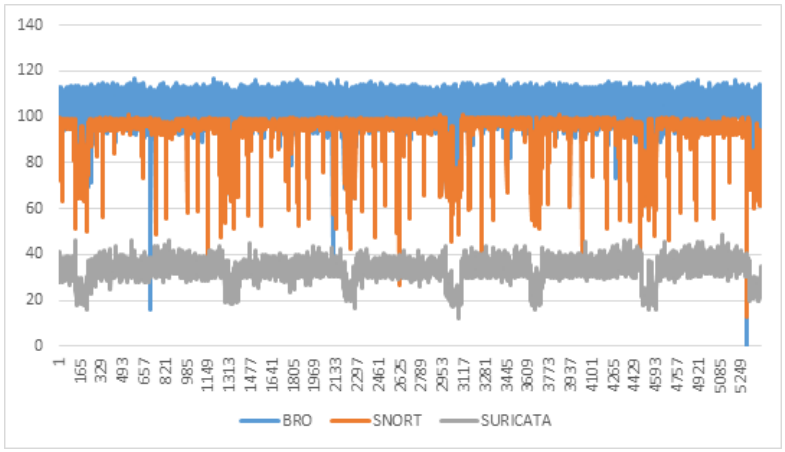

Figura 5. Comparativo de uso da CPU durante execução das ferramentas.

5. Conclusões

A abordagem utilizada para medir a eficácia dos IDS pela quantidade de alertas gerados e os pacotes de ataque enviados, não gerou uma boa base de comparação da eficácia dos sistemas. Em contrapartida, o trabalho apresentado contribui para o avanço do estado da arte, elucidando a viabilidade do uso em conjunto dos três sistemas citados, assim como são listados os recursos computacionais necessários para tal. A execução conjunta dos IDS poderia melhorar a detecção de ameaças em uma organização, se comparado ao cenário onde apenas um sistema é executado isoladamente, devido à variedade de bases de assinaturas que podem ser utilizadas. Como continuidade do projeto planeja-se diversificar ainda mais a quantidade de ataques cibernéticos, usando como parâmetro da eficácia de cada IDS, a habilidade de detecção de cada ataque através do parâmetro de *timestamp* coletado no início de cada ataque. Existe também a pretensão de incluir outras ferramentas de detecção de intrusão como o Hogzilla [Angelo 2014], aumentando o escopo da análise.

Referências

Angelo, P. (2014). Hogzilla ids. `http://ids-hogzilla.org/`. Acesso: Janeiro/2017.

Aydın, M. A., Zaim, A. H., and Ceylan, K. G. (2009). A hybrid intrusion detection system design for computer network security. *Computers Electrical Engineering*, 35(3):517 – 526.

Brock, A. A., Kim, N. W., and Lingafelt, C. S. (2001). Dynamic intrusion detection for computer systems.

Burning Glass Technologies (2015). Job market intelligence: Cybersecurity jobs, 2015. `http://burning-glass.com/wp-content/uploads/Cybersecurity_Jobs_Report_2015.pdf`. Acesso: Janeiro/2018.

Chung, M., Puketza, N., Olsson, R. A., and Mukherjee, B. (1995). Simulating concurrent intrusions for testing intrusion detection systems: Parallelizing intrusions. In *Proc., 18th National Information Systems Security Conference*, pages 173–183.

D., D. and B., B. (2011). A performance analysis of snort and suricata network intrusion detection and prevention engines. *ICDS 2011 : The Fifth International Conference on Digital Society*.

Ferreira, V. (2016). Classificação de anomalias e redução de falsos positivos em sistemas de detecção de intrusão baseados em rede utilizando métodos de agrupamento. Master's thesis, Universidade Estadual Paulista Júlio de Mesquita Filho, São José do Rio Preto.

Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 120–, Washington, DC, USA. IEEE Computer Society.

Ghosh, A. K., Schwartzbard, A., and Schatz, M. (1999). Learning program behavior profiles for intrusion detection. In *Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1*, ID'99, pages 6–6, Berkeley, CA, USA. USENIX Association.

Gómez, J., Gil, C., Padilla, N., Baños, R., and Jiménez, C. (2009). Design of a snort-based hybrid intrusion detection system. In Omatu, S., Rocha, M. P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., and Corchado, J. M., editors, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 515–522, Berlin, Heidelberg. Springer Berlin Heidelberg.

Handley, M., Paxson, V., and Kreibich, C. (2001). Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, pages 9–9, Berkeley, CA, USA. USENIX Association.

Hofmeyr, S. A., Forrest, S., and Somayaji, A. (1998). Intrusion detection using sequences of system calls. *J. Comput. Secur.*, 6(3):151–180.

Hwang, K., Cai, M., Chen, Y., and Qin, M. (2007). Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *IEEE Transactions on Dependable and Secure Computing*, 4(1):41–55.

Innella, P. (2016). Evolution of intrusion detection systems. `http://www.symantec.com/connect/articles/evolution-intrusion-detection-systems`. Acesso: Fevereiro/2018.

Jain, H. (2004a). System and method for integrated header, state, rate and content anomaly prevention with policy enforcement.

Jain, H. K. (2004b). System and method for integrated header, state, rate and content anomaly prevention for domain name service.

Kasinathan, P., Costamagna, G., Khaleel, G., Pastrone, C., and A. Spirito, M. (2013). In *DEMO: An IDS Framework for Internet of Things Empowered by 6LoWPAN*. Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security.

Khamphakdee, N., Benjamas, N., and Saiyod, S. (2016). Improving intrusion detection system based on snort rules for network probe attack detection. *IEEE*.

Kim, G., Lee, S., and Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4, Part 2):1690 – 1700.

Lee, W. and Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 6–6, Berkeley, CA, USA. USENIX Association.

Lobato, A., Lopez, M., and Duarte, O. (2015). In *Um Sistema Acurado de Detecção de ameaças em Tempo Real por processamento de Fluxos*. Universidade Federal do Rio de Janeiro, UFRJ.

Mitnick, K. D. and Simon, W. L. (2003). A arte de enganar: Ataques de hackers: Controlando o fator humano na segurança da informação. *Editora Makron*.

Northcutt, S., Alder, R., Babbin, J., Doxtater, A., Foster, J. C., Kohlenberg, T., and Rash, M. (2004). Snort 2.1 intrusion detection. 2. ed. *Syngress Publishing*.

Open Information Security Foundation (2016). Suricata ids about. `https://suricata-ids.org/`. Acesso: Janeiro/2018.

Panes, G. (2011). Firewall dinâmico: Uma implementação cliente/servidor. *Dissertação de Mestrado em Ciência da Computação – Universidade Estadual Paulista Júlio de Mesquita Filho*.

Paxson, V. (1999). Bro: A system for detecting network intruders in real-time. *Comput. Netw.*, 31(23-24):2435–2463.

Ptacek, T. H. and Newsham, T. N. (1998). Insertion evasion and denial of service eluding network intrusion detection.

Syversen, J. (2008). Method and apparatus for defending against zero-day worm-based attacks. US Patent App. 11/632,669.

Tajalli, H., Graham, J. J., and Fraser, T. J. (2003). Behavior-based host-based intrusion prevention system.

The Snort Team (2016). Snort users manual: 2.9.8.2. *Cisco And/or Its Affiliates*.

Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting intrusions using system calls: alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*, pages 133–145.

Websense Security Labs (2015). Theat report. *Websense Inc*.

Zekrifa, D. M. S. (2014). *Hybrid Intrusion Detection System*. Theses, School of Information Technology & Mathematical Sciences.

Zhang, X., Li, C., and Zheng, W. (2004). Intrusion prevention system design. In *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, pages 386–390.