UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO/TECNÓLOGO EM REDES DE COMPUTADORES

GEYCIANE TAVARES JORGE

UM ESTUDO COMPARATIVO DA CAPACIDADE DE DETECÇÃO E DESEMPENHO DOS SISTEMAS DE DETECÇÃO E PREVENÇÃO DE INTRUSÕES SNORT, SURICATA E OSSEC

QUIXADÁ
2016

GEYCIANE TAVARES JORGE

UM ESTUDO COMPARATIVO DA CAPACIDADE DE DETECÇÃO E DESEMPENHO DOS SISTEMAS DE DETECÇÃO E PREVENÇÃO DE INTRUSÕES SNORT, SURICATA E OSSEC

Monografia apresentada ao curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Tecnólogo em Redes de Computadores. Área de concentração: Computação.

Orientador: Prof. MSc. Michel Sales Bonfim

QUIXADÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

J7le Jorge, Geyciane Tavares.
Um estudo comparativo da capacidade de detecção e desempenho dos Sistemas de Detecção e Prevenção de Intrusão Snort, Suricata e OSSEC. / Geyciane Tavares Jorge. – 2016.
40 f..: il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2016.
Orientação: Prof°. Ms. Michel Sales Bonfim.

1. Redes de Computadores - Segurança. 2. Desempenho. 3. Segurança da Informação. I. Título.

CDD 004.6

GEYCIANE TAVARES JORGE

UM ESTUDO COMPARATIVO DA CAPACIDADE DE DETECÇÃO E DESEMPENHO DOS SISTEMAS DEDETECÇÃO E PREVENÇÃO DE INTRUSÕES SNORT, SURICATA E OSSEC

Monografia apresentada ao curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Tecnólogo em Redes de Computadores. Área de concentração: Computação.

Aprovada em: 21/ Dezembro / 2016.

BANCA EXAMINADORA

_____
Profº.MSc Michel Sales Bonfim (Orientador)
Universidade Federal do Ceará (UFC)

_____
Prof. MSc. Francisco Helder Candido dos Santos Filho
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Paulo Antonio Leal Rego
Universidade Federal do Ceará (UFC)

Dedico este trabalho aos meus pais, Aldenir Tavares e Luis Pereira, irmão, Wagner Tavares e aos amigos pelo apoio e incentivos ao longo desta fase da minha vida.

AGRADECIMENTOS

Ao professor e orientador Michel Sales Bonfim pela, paciência e dedicação para a realização deste trabalho.

Aos amigos da turma de Redes de Computadores pelo apoio e amizade ao longo desses anos e aos demais amigos que fiz na UFC durante estes anos.

Agradeço aos professores que tive a honra de conhecer durante o curso de Redes de computadores na UFC Quixadá, por sua dedicação e comprometimento.

À minha família, que sempre me incentivou à buscar conhecimento e me deu força para realizar este sonho.
.

"Nós somos o que fazemos repetidas vezes. Portanto, a excelência não é um ato, mas um hábito."

(Aristóteles)

RESUMO

É notável o crescimento exponencial nos últimos anos de dispositivos conectados à Internet e isso consequentemente gera muita insegurança sobre estar ou não protegido contra ataques maliciosos, e ferramentas como essas auxiliam na identificação de possíveis ataques, muitas não são capazes de conter qualquer ataque que a rede ou *host* sofra, pois atuam em sua maioria em modo passivo, apenas notificando ao administrador sobre atividades que geram alertas no ambiente monitorado. Sistema de Detecção e Prevenção de Intrusão (*Intrusion Detection and Prevention System* - IDPS) são sistemas já consolidados e presentes na realidade das organizações que possuam uma pequena ou grande rede de computadores interligados. A função principal desse tipo de ferramenta, tem como o propósito realizar a detecção de intrusos que queiram realizar atividades maliciosas como roubo de informações e realizar alguma instrução interromper qualquer ataque. O funcionamento de uma ferramenta IDPS é basicamente verificar e detectar comportamentos maliciosos que ocorram dentro de uma rede ou um simples host que esteja sendo monitorado. Para a implementação de IDPS é preciso conhecer o seu funcionamento e as configurações mínimas, no entanto se essas informações básicas não forem consideradas há o risco utilizar o Sistema de Detecção e Prevenção de Intrusão incorreto gerando detecções incorretas ou fazer excessivo uso dos recursos computacionais. Este trabalho visa compara os IDPS Snort, Suricata e OSSEC para avaliar o desempenho e taxa de detecção de cada um deles, executando diversos testes de intrusão e com inúmeros usuário gerando tráfego ao mesmo tempo. Para tal, utilizamos as ferramentas Pytbull, que geral uma grande quantidade de ataques de intrusão e o iPerf usado para criar todo o tráfego na rede. Após a realização dos identificamos que os IDPSs OSSEC e Suricata apresentaram resultados satisfatórios para a taxa de detecção e desempenho, o OSSEC por sua vez se destacou mais e obteve os melhores resultados nos dois testes realizados.

Palavras-chave: Sistema de Detecção e Prevenção de Intrusão. Segurança. Pytbull. iPerf.

ABSTRACT

It is notable the exponential growth of Internet-connected devices in recent years and this consequently generates a great deal of insecurity about whether or not it is protected against malicious attacks, and such tools assist in identifying possible attacks, many are not capable of containing any attack that the network or host suffers because they mostly act in passive mode, only notifying the administrator about activities that generate alerts in the monitored environment. Intrusion Detection and Prevention System (IDPS) are systems already consolidated and present in the reality of organizations that have a small or large network of interconnected computers. The main function of this type of tool is to detect intruders who want to perform malicious activities such as information theft and perform some instruction to stop any attack. The operation of an IDPS tool is basically to check and detect malicious behavior that occurs within a network or a simple host being monitored. For the implementation of IDPS it is necessary to know its operation and the minimum configurations, however, if this basic information is not considered there is a risk to use the incorrect Intrusion Detection and Prevention System, generating incorrect detections or making excessive use of computational resources. This work compares the Snort, Suricata, and OSSEC IDPS to evaluate the performance and detection rate of each of them, performing several intrusion tests and with numerous users generating traffic at the same time. To do this, we use the Pytbull tools, which generally a lot of intrusion attacks and the Iperf used to create all the traffic on the network. With the results, it was possible to identify how the IDPS behaved and which one presented more satisfactory results.
Keywords: Intrusion Detection System. Security. Pytbull. iPerf

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

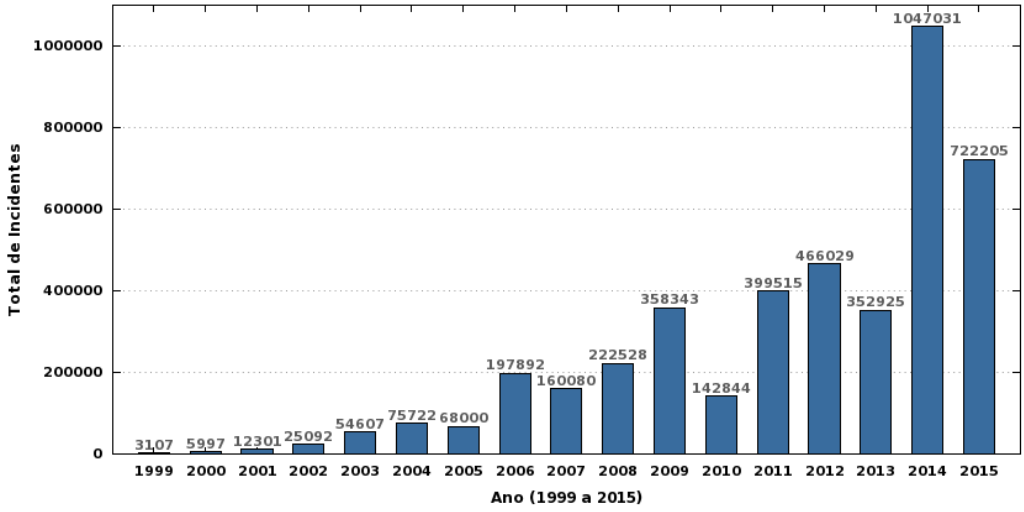
1 INTRODUÇÃO

O crescente aumento de dispositivos conectados à Internet certamente vem contribuindo para o crescimento de crimes cibernéticos. De acordo com Park e Anh (2016), a facilidade de realização dos ataques pode se justificar pelo grande número de usuário que simplesmente ignoram a importância de alguma ferramenta que faça identificação e toma alguma ação sobre atividades maliciosas. Existem vários tipos de ataques, que incluem:

- Ataque *Scanning* - usando técnicas de exploração, o atacante pode obter informações sobre as configurações do sistema e nível de segurança, ao dispor destas informações o atacante pode atacar o sistema.
- Ataque Negação de Serviço - Neste tipo de ataque, os invasores tentam tornar os recursos indisponível de um sistema ou organização para que os usuários não consigam acessar,
- Ataque de Penetração - Possui todos os ataques, nos quais o atacante invade o sistema como um super usuário para acessar tudo com mais privilégios.

O atual crescimento no número de ataques cibernéticos pode ser visualizado no gráfico da figura 1a, reportado pelo CERT.br. O CERT.br é o Grupo de Resposta a Incidentes de Segurança para a Internet brasileira, mantido pelo NIC.br, do Comitê Gestor da Internet no Brasil. É responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à Internet brasileira (CERT.br, 2016). De acordo com a figura 1 há um aumento significativo de incidentes ao longo dos anos.

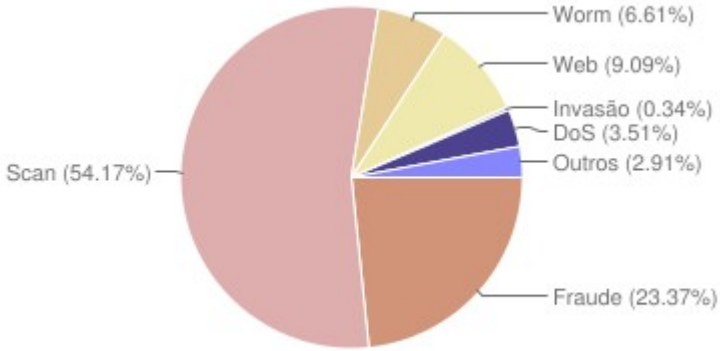
Figura 1: Estatísticas de Incidentes Reportados pelo CERT.br anualmente



Fonte: http://www.cert.br/stats/incidentes/

Na figura 2 podemos ver os ataques que ocorreram com mais frequência no ano de 2015, os *Scans* representaram mais da metade dos ataques, seguido de fraudes e crimes na Web.

Figura 2: Tipos de Ataques reportados pelo CERT.br



Fonte: http://www.cert.br/stats/incidentes/2015-jan-dec/tipos-ataque.html

Whilte, et al. (2013) afirma ser indispensável o uso de ferramentas, que contribuíam para segurança da informação dentro das organizações e que elas devem ser cada vez mais eficientes e capazes de acompanhar o crescimento e a diversificação dos crimes. Uma destas ferramentas são os Sistemas de Detecção de Intrusão (*Intrusion Detection and Prevention Systems* - IDPS) que monitoram várias informações das redes ou sistemas finais, como: pacotes de rede, análise de *rookit* ou mesmo *logs* do sistema, relatando eventos para um servidor principal ou mantendo registros localmente. Um IDPS mantém controle sobre a rede para monitorar os ataques maliciosos (como *Scans*) que podem intervir no funcionamento do sistema.

Para detectar os ataques descritos acima, além de IDPS também podemos usar *firewalls* mas eles não são de natureza dinâmica e tem regras simples para permitir ou negar protocolos, IDPS é usado no tratamento de ataques mais complexos e também é dinâmico por natureza (SINGH; SINGH, 2014).

Portanto, utilizar um IDPS para o aumento da segurança em uma organização é de extrema importância, porém existem uma grande variedade de ferramentas deste tipo, como: Snort, Suricata e OSSEC, o que dificulta a decisão de um administrador de rede. Wang et al. (2013) afirma ser esse um dos problemas enfrentados na gestão de TI, pois é necessário analisar objetivamente os produtos existentes nas perspectivas do sistema e da administração de rede. Os autores afirmam que muitos têm um desempenho semelhante, no entanto, na prática da gestão de TI, além de considerar a efetividade de um produto, precisamos saber suas exi-

gências para recursos de computação, espaço de armazenamento e largura de banda de rede, bem como outros fatores relacionados ao gerenciamento de TI.

Neste trabalho propomos uma análise comparativa de 3 (três) IDPSs sendo eles: o Snort (KUMAR; SANGWAN, 2012), Suricata (DAY; BURNS, 2011) e OSSEC (MEHTA; VISHAL, 2015), provendo assim resultados que possam auxiliar na escolha de uma ou mais destas ferramentas. Para tanto, avaliamos o impacto da utilização desses sistemas sobre os recursos computacionais e as suas capacidades de detecção de ataques. Tais ferramentas foram selecionadas devido â sua popularidade e por serem *Open Source*.

Para tanto, montamos um cenário composto por dois *hosts* que realizam a troca de informações durante a execução dos testes e também uma máquina *gateway* com os IDPSs instalados, sendo utilizados para monitorar o tráfego entre estes dois *hosts*. Assim como Park e Ahn (2016), utilizamos o *Pytbull* para avaliar a capacidade de detecção de ataques, ele simula ataques de intrusão e reporta o total de detecções realizadas pelos IDPSs. Por outro lado, também utilizamos o iPerf para gerar todo o tráfego nas máquinas durante o teste de avaliação de desempenho.

Este trabalho está organizado nas seguintes seções. Na Seção 2, apresentamos os trabalhos relacionados. Na Seção 3, descrevemos os objetivos gerais e específico. A Seção 4, é composta pela fundamentação teórica onde são abordados os principais pontos deste trabalho. A Seção 5, aborda a descrição dos experimentos e as análises dos resultados obtidos no trabalho. Por fim, na Seção 6, apresentamos as conclusões.

2 OBJETIVO

2.1 Objetivo Geral

Realizar uma avaliação dos IDPS (*Intrusion Detection and Prevention Systems*) Snort, Suricata e OSSEC, através de uma análise comparativa baseada na capacidade de detecção de ataques e desempenho (uso de recursos computacionais).

2.2 Objetivos Específicos

- Definir os experimentos;
- Realização dos experimentos e coleta dos dados;
- Análise dos resultados.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta sessão irei abordar os principais conceitos que fundamentam a base deste trabalho, englobando os principais sistemas utilizados e suas definições.

3.1 Sistema de Prevenção de Intrusão - (*Intrusion Prevention System* - IPS)

Sistemas de prevenção de intrusão funcionam como um *sniffer*, capturando e analisando a comunicação do segmento de rede (NAKAMURA e de GEUS, 2007).

Os sistemas de prevenção de intrusão, ou IPSs, são dispositivos ou programas que são usados para detectar sinais de intrusões em redes, ou sistemas, e agir de acordo com as instruções. Essa ação consiste em gerar alarmes e/ou bloquear ativamente intrusões (PIPER, 2011).

Firewalls e IPSs são ferramentas essenciais para proteger uma empresa de intrusões. Ambos são necessários, principalmente porque eles são cada um projetado para olhar para coisas diferentes:

- Um *firewall* foi projetado para bloquear todo o tráfego de rede, exceto o que é explicitamente permitido;
- Um sistema de prevenção de intrusões é projetado para permitir tudo, exceto o que é explicitamente proibido;
- Um *firewall* foi projetado para permitir (ou bloquear) pacotes de rede com base em sua origem, destino e número de porta, independentemente do conteúdo da carga útil de cada pacote (o conteúdo da mensagem);
- Um sistema de prevenção de intrusão é projetado para permitir (ou bloquear) pacotes de rede com base na carga útil do pacote (PIPER, 2011).

3.2 Sistema de Detecção de Intrusão - (*Intrusion Detection System* - IDS)

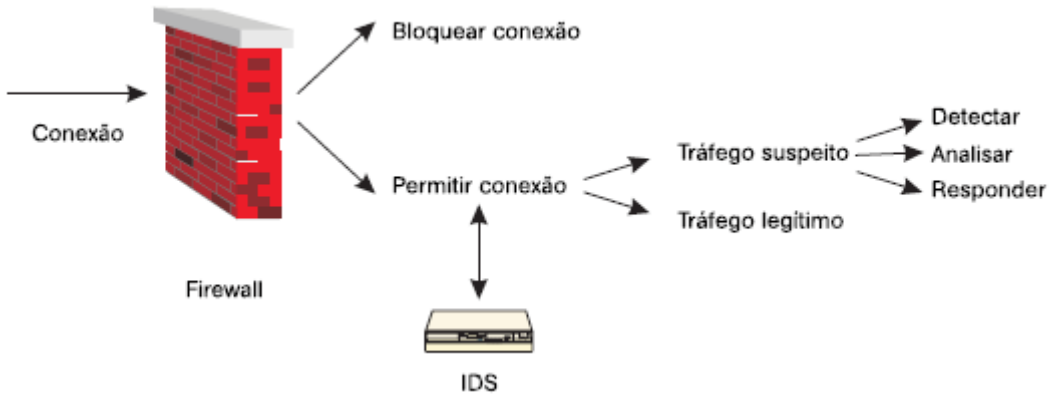
Pode-se afirmar que o uso de IDS em ambientes coorporativos é de fundamental importância para que os administradores de redes possam monitorar e acompanhar em tempo real tudo que se passa dentro da rede monitorada e assim tomar medidas preventivas ou corretivas sobre possíveis ataques que ocorram.

Basicamente, podemos definir IDS como uma ferramenta inteligente capaz de detectar tentativas de invasão em tempo real. Esses sistemas podem atuar de forma a somente alertar as tentativas de invasão, como também em forma reativa, aplicando ações necessárias contra o ataque (SNORT, 2016). Aplicar ações de forma reativa é a principal função do IPS, o

IDS irá fazer isso se estiver configurado junto a um IPS, dessa forma o IDS gera os alertas e o IPS aplica alguma resposta para interromper o ataque

Os IDS são usados para detectar vários tipos de comportamentos maliciosos que podem comprometer a segurança e a confiabilidade de um sistema. Entre eles incluem ataques pela rede contra serviços vulneráveis, ataques baseados em uma estação, aumento de privilegio, *logins* não autorizados, *malware* (GTA, 2016). Na figura 3 podemos ver como o funcionamento do IDS acontece, o *firewall* é responsável por liberar conexões de acordo com suas regras e o IDS tem a função de analisar tudo o que passa pelo *firewall* e detectar se o tráfego é suspeito, caso seja ele irá gerar o alerta para o IPS, que nesses casos irá executa alguma ação para barrar o tráfego malicioso.

Figura 3: Funcionamento de um Sistema de Detecção de Intrusão - IDS



Fonte: Nakamura, de Geus. (2007)

Um IDPS é composto de diversos componentes: sensores, que geram eventos de segurança; console para monitorar eventos e alertas e controlar os sensores e um mecanismo central que grava os eventos registrados pelo sensor na base de dados e usa um sistema de regras para gerar alertas a partir dos eventos recebidos na maioria dos IDSs essa base de dados trata-se de arquivos de logs contento as principais informações sobre os alertas gerados, como por exemplo IP de origem e destino. Uma vez que é detectado alguma instrução, alertas são enviados, e podem haver dois tipos de resposta, ativa ou passiva. Na ativa, respostas aos incidentes são geradas pelo próprio sistema, enquanto que na passiva, são gerados apenas relatórios para que o administrador venha a tomar as medidas que julgar necessária (GTA, 2016).

Eles são divididos em alguns métodos de detecção, como: baseados em assinatura e detecção de anomalias.

3.2.1 Baseada em Assinatura

Intrusos tem assinaturas, como vírus de computadores, que podem ser detectados por software. Tenta-se achar pacotes de dados que contenham quaisquer assinaturas conhecidas relacionadas a intrusos ou anomalias relacionadas a protocolos de Internet. Baseado em um conjunto de assinaturas e regras, o sistema de detecção pode achar atividades suspeitas e gerar alertas.

3.2.2 Baseada em Anomalia

Detecção de intrusos baseadas em anomalias geralmente dependem de anomalias nos pacotes presentes nos cabeçalhos dos pacotes. Em alguns casos esses métodos podem produzir resultados melhores comparados aos IDS baseados em assinaturas. Geralmente, IDS capturam dados da rede e aplicam suas regras a esses dados ou detectam anomalias neles (SINGH; SINGH, 2014)

Os IDS geralmente são apenas passivos, somente observando o sistema e gerando alertas para os responsáveis por este sistema. Porém, em alguns casos, o sistema pode reagir em caso de uma intrusão ser detectada, como por exemplo fechar a conexão, bloquear a partir do firewall ou até mesmo desabilitar uma conta. (GTA, 2016)

Além da divisão pelas técnicas de reconhecimento de ataque, os IDSs podem ser também classificados em dois tipos principais:

3.2.2.1 ***Sistemas Baseados em Redes***

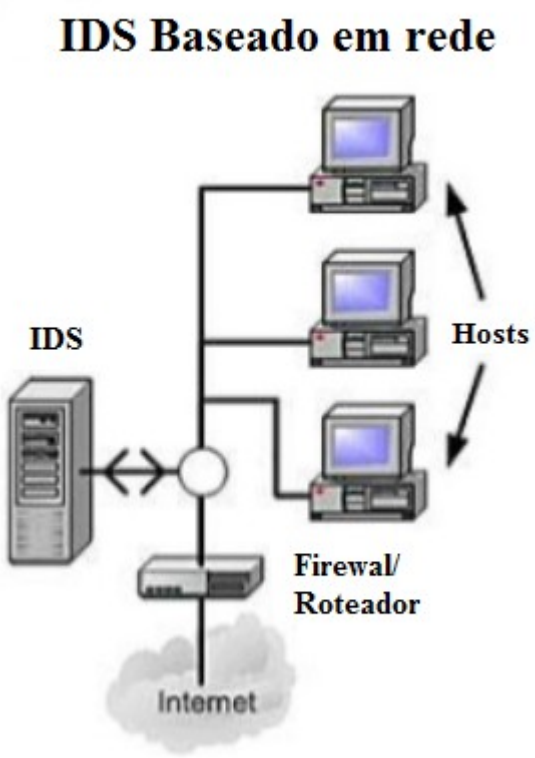
Este tipo de sistema é capaz de monitorar uma grande quantidade de máquinas que estejam conectadas na rede em que o NIDS está examinando, pois ele analisa todo o tráfego da rede (SNORT, 2016).

Os NIDS também podem consistir em um conjunto de sensores ou estações espalhadas por vários pontos da rede. Essas unidades monitoram o tráfego da rede, realizando análises locais do tráfego e reportando os ataques a um console central. A implementação de um NIDS tem pouco impacto sobre a performance da rede. Eles geralmente ficam em modo passivo, apenas escutando o tráfego da rede sem interferir no seu funcionamento. Os NIDSs podem ter dificuldade em processar todos os pacotes em uma rede que possua um grande tráfego de dados.

A maioria dos NIDS não podem reconhecer se um ataque foi bem-sucedido. Eles apenas apontam que um ataque foi iniciado. Dessa maneira eles apenas detectam um ataque,

sendo que o administrador de sistemas deve verificar se o host apontado foi atacado. (SNORT, 2016). Na figura 4 podemos observar de forma resumida como funciona um NIDS, é possível notar a presença do NIDS que apenas está recebendo informações de todo o tráfego da rede, no sistema de detecção em rede não é necessária nenhuma instalação nos *hosts* da rede, basta apenas estarem conectados em rede.

Figura 4: Sistema NIDS



Fonte: Park e Ahn (2016)

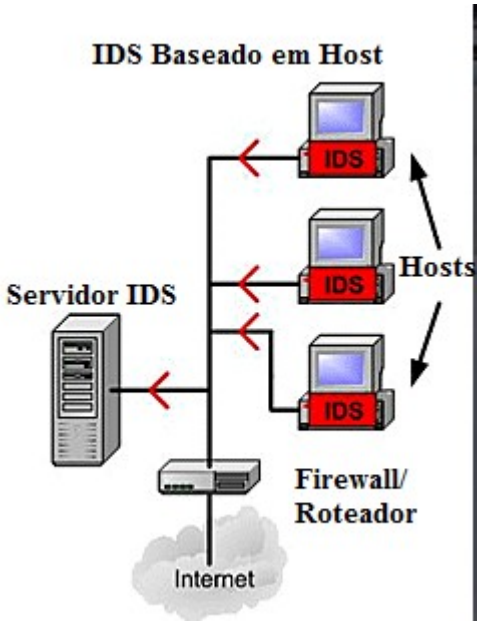
3.2.2.2 ***Sistema baseados em Host***

Nesse tipo de IDS ele realiza a detecção apenas na máquina em que está instalado, não observa tráfego de rede e faz apenas a verificação de evento e registros de *logs*.

Através disso os HIDS podem analisar as atividades das estações com confiança e precisão, determinando exatamente quais processos e usuários estão envolvidos em um tipo particular de ataque no sistema operacional. Além disso, ao contrário dos sistemas baseados em rede, os baseados em *host* (estação) podem ver as consequências de uma tentativa de ataque, como eles podem acessar diretamente e monitorar os arquivos e processos do sistema usualmente alvos de ataques. Esse tipo de IDS tem a capacidade de monitorar eventos locais de um *host*, podendo detectar ataques que não poderiam ser detectados por um IDS de rede. Eles podem operar em um ambiente onde o tráfego de rede é criptografado, a informação é

analisada antes de ser criptografada na origem, ou depois de ser decriptada no destino. (SNORT, 2016). Na figura 5 visualizamos como é o funcionamento de um HIDS, nelas observamos que cada *host* possui um elemento do IDS que ser o responsável por reportar ao servidor IDS qualquer eventualidade que aconteça nos *hosts* monitorados.

Figura 5: Sistema HIDS



Fonte: Park e Ahn (2016)

Neste trabalho foram utilizados apenas IDSs em modo passivo e como já mencionado anteriormente são eles Snort, Suricata e OSSEC, por padrão de instalação este último possui um sistema de *active response* capaz gerar "respostas" para eventuais alertas, no entanto ele foi desabilitado durante os testes para não interferir no resultado final.

3.3 Sistema de Detecção e Prevenção de Intrusão – (*Intrusion Detecction and Prevention System* – IDPS)

O Sistema de Detecção de Intrusão (IDS) é o processo de monitoramento de eventos que ocorrem em um sistema ou rede e reporta as possíveis invasões, enquanto o Sistema de Prevenção de Intrusão (IPS) tem a capacidade de tentar parar essas possíveis intrusões. A combinação dos dois sistemas resultará em Sistema de Prevenção e Detecção de Intrusão (IDPS), é um sistema híbrido, que não só detecta os ataques, mas também evitar que tais ataques ocorram nas redes (Poongodi e Bose, 2013).

3.4 Snort

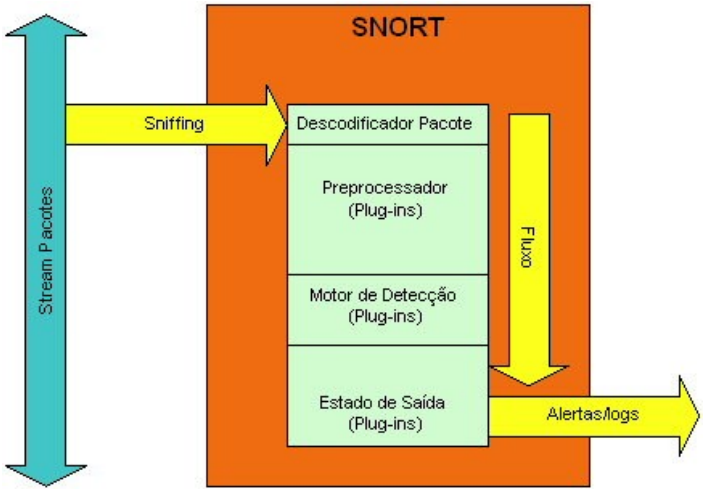
Snort é um Sistema de Detecção de Intrusão popular (IDPS), que utilizam para proteger o sistema dos riscos de um atacante. É um *software* leve de código aberto baseado em rede, foi desenvolvido por Martin Roesch com a linguagem C em 1998. Snort pode ser instalado em praticamente qualquer arquitetura de computador e plataforma de sistema operacional. Além disso, o IDPS Snort também gera alerta em tempo real (KHAMPHAKDEE et al. 2014).

Ele é capaz de verificar análise de protocolo e pode detectar vários tipos de ataque. As regras do Snort podem ser escritas em qualquer idioma, a sua estrutura também é simples e pode ser facilmente lidas ou modificadas (KUMAR; SANGWAN, 2012).

O Snort é basicamente a combinação de vários componentes como pode ser visto na figura 6. Todo o trabalho dos componentes em conjunto é para encontrar um ataque em particular. Basicamente, consiste em seguir os seguintes componentes:

- **Decodificado de pacotes:** Decodifica e recolhe os pacotes da interface de rede e envia para o pré-processador;
- **Pré-processadores:** Modifica ou organiza os pacotes antes do motor de detecção aplicar alguma operação no pacote se ele estiver corrompido. Realiza desfragmentação pois às vezes o intruso quebra a assinatura em dois pacotes e assim a assinatura é encontrada;
- **Motor de Detecção:** Seu trabalho principal é descobrir as saídas das atividades de intrusão nos pacotes e com a ajuda das regras dos Snort encontrar e aplica as regras apropriadas;
- **Logs e Alertas do Sistema:** Seja qual for o mecanismo de detecção ele irá gerar alertas das atividades de intrusão;
- **Módulos de Saídas:** Módulos de Saídas ou *plug-ins* salva as saídas geradas pelos logs e os alertas dos Snort (KUMAR; SANGWAN, 2012);

Figura 6: Arquitetura de Funcionamento do Snort



Fonte: *http://paginas.fe.up.pt/~mgi98020/pgr/snort.htm*

3.5 Suricata

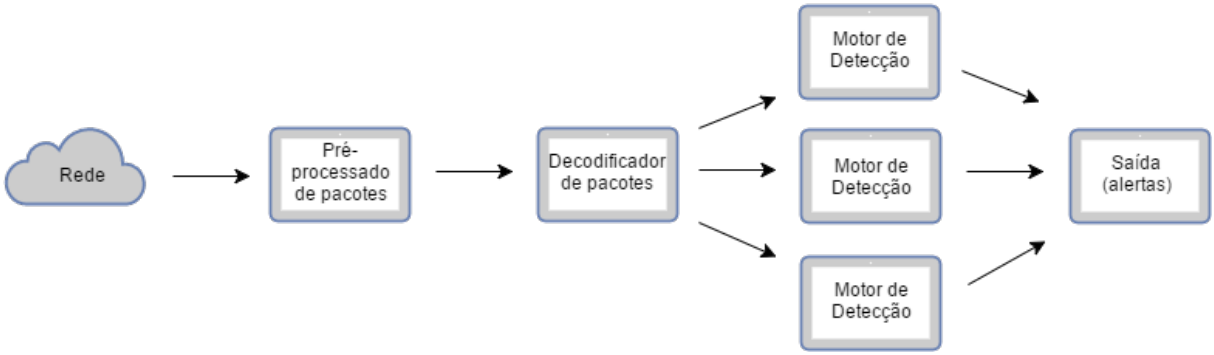
Suricata foi desenvolvida em 2010 pelo OISF (*Open Information Security Foundation*), que recebeu apoio financeiro do Departamento de Segurança Interna dos EUA. A arquitetura do Suricata é semelhante ao do Snort, mas ao invés de usar uma única *thread*, o Suricata implementa vários *threads* para processar os pacotes. Isto permite característica únicas ao Suricata e maximiza a capacidade de receber e tratar os pacotes (Park; Ahn, 2016).

Suricata é um sistema de detecção de intrusão baseado em regras, que utiliza vários conjuntos de regras desenvolvidas exatamente para monitorar o tráfego de rede e fornecer alertas para o administrador do sistema quando ocorrem eventos suspeitos. Projetado para ser compatível com os componentes de segurança de rede existentes, Suricata possui funcionalidade de saída unificada e opções de biblioteca conectáveis para aceitar chamadas de outros aplicativos. Como um motor de detecção *multi-threaded* o Suricata oferece maior velocidade e eficiência na análise de tráfego de rede. Além de aceleração de hardware, o motor é construído para utilizar o maior poder de processamento oferecido pelos conjuntos de chips CPU *multi-core* (OISF, 2016).

Suricata e Snort foram configurados para serem executados utilizando de modo quase que idênticos conjuntos de regras. Suricata usa uma classificação de diferente configuração para Snort, que usa 134 decodificadores e 174 pré-processador regras. Ambos NIDPSs estavam usando o registo idêntico métodos, ou seja, *Barnyard*, MySQL e *acidbase* (DAY; BURNS, 2011). Assim como o Snort ele possui pré-processador de pacotes, responsável reco-

lher os pacotes que passam na rede, há também o decodificador de pacotes que organiza os pacotes e analisa se eles estão corrompidos, seu motor de detecção é *multi –threading* o que oferece maior capacidade e agilidade na análise do tráfego recebido e por fim temos o componente saída que envia os alertas do motor de detecção (PARK; AHN, 2016).

Figura 7: Arquitetura Suricata



Fonte: Park e Ahn (2016)

3.6 OSSEC

OSSEC é um sistema *open source* de detecção de intrusão baseado no *host* (HIDS), ele detecta intrusões examinando logs gerados por vários aplicativos. OSSEC tem dois modos de trabalho, local (monitoramento um único sistema) e agente/servidor (recolhe e monitora registos de várias fontes em toda a rede) (MEHTA; VISHAL, 2015).

OSSEC pode ler e analisar arquivos de log de mais de 40 programas e dispositivos diferentes. Ele possui componentes para recolher, ler, analisar logs e enviar e-mails de alertas. Os agentes coletam os registros em dispositivos e envia para o servidor central para análise. Quando os logs são recebidos, os valores de campo são extraídos e informações importantes contidas dentro de logs são identificadas e comparados com regras predefinidas ou criadas manualmente (OSSEC, 2016)

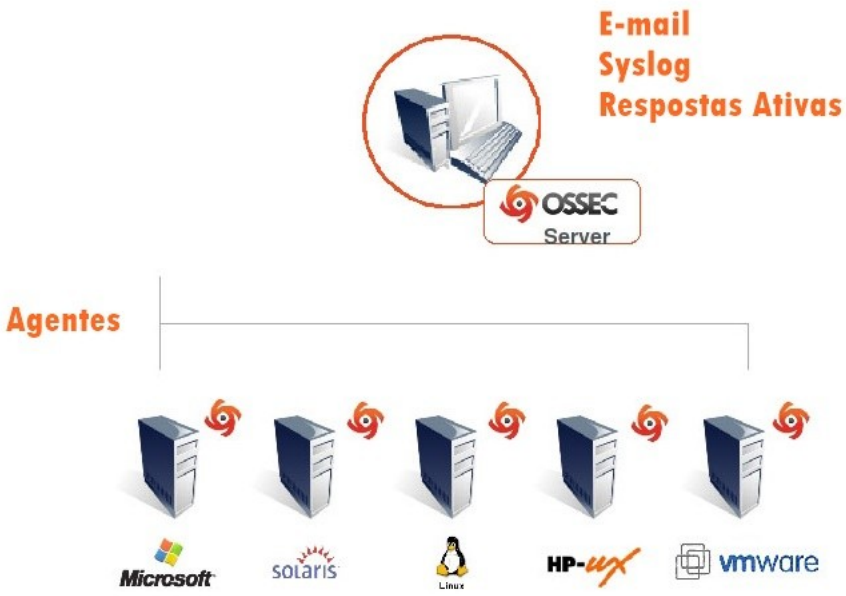
O gerente é a peça fundamental para implantação do OSSEC. Ele armazena a integridade do arquivo verificando bancos de dados e os logs. Todas as regras e opções de configuração importantes são armazenados centralmente no gestor, tornando-o fácil de administrar, mesmo um grande número de agentes (OSSEC, 2016)

O agente é um pequeno programa ou conjunto de programas, instalado nos sistemas a serem monitorados. O agente irá recolher informação e transmiti-la para o gerente analisar. Algumas informações são coletadas em tempo real, outros periodicamente. O OSSEC suporta agentes em diferentes sistemas, ele pode receber e analisar os eventos *syslog* a partir

de uma grande variedade de *firewalls*, *switches* e roteadores, mais exigirá que o servidor execute em sistema UNIX (SINGH; SINGH, 2014).

Na figura 8 podemos ver um exemplo da arquitetura do OSSEC. Os agentes como visto na figura podem ser executados em diferentes sistemas operacionais e são responsáveis por reportar ao servidor OSSEC qualquer alteração que ocorra no *host* monitorado. O servidor OSSEC possui restrições para o sistema operacional a ser utilizado só podendo ser executado em sistemas UNIX, o servidor fica responsável por receber os dados coletados pelos agentes e executar alguma instrução, como enviar e-mail sobre as ocorrências para o administrador ou mesmo realizar alguma resposta para as atividades maliciosas de acordo com as regras e se assim for configurado.

Figura 8: Arquitetura de funcionamento OSSEC



Fonte: ossec.github.io/docs/manual/ossec-architecture.html (2016)

3.7 Ferramentas Utilizadas

3.7.1 VirtualBox

VirtualBox é um aplicativo de virtualização de plataforma cruzada. O que isso significa? Significa que ele estende as capacidades existente do seu computador para que ele possa executar vários sistemas operacionais ao mesmo tempo. Assim, por exemplo, você pode executar Windows e Linux no seu Mac, execute o Windows Server 2008 em seu servidor

Linux, rodar o Linux em seu PC com Windows, e assim por diante, todos ao lado de seus aplicativos existentes. Você pode instalar e executar o maior número de máquinas virtuais que você quiser os únicos limites práticos são o espaço em disco e memória (VirtualBox, 2016).

Como descrito acima o VirtualBox tem a capacidade de executar máquinas de diferentes sistemas operacionais em uma mesma máquina, sendo assim, neste trabalho utilizamos ele para virtualizar as máquinas do cenário proposto e assim realizar os testes.

3.7.2 *Pytbull*

Pytbull é um *framework* de teste para Sistemas de Detecção e Prevenção de Intrusão (IDPS) como o Snort, Suricata e qualquer IDPS que gera arquivos de alertas. Ele pode ser usado para testar a capacidade de detecção e de bloqueio de um IDPS, para comparar IDPS, para comparar alterações de configuração e para verificar/validar configurações *Pytbull* fornece mais de 300 testes, agrupados em 10 módulos, cobrindo um vasto leque de ataques (*clientSideAttacks*, *testRules*, *badTraffic*, *fragmentedPackets*, *multipleFailedLogins*, *evasionTechniques*, *shellcodes*, *denialOfService*, *pcapReplay*).

- **badTraffic**: pacotes não compatível são enviados para o servidor para testar como os pacotes são processados;
- **bruteforce**: testa a capacidade do servidor para controlar ataques de força bruta (por exemplo, FTP). Aproveita-se de regras personalizadas sobre Snort, Suricata ou outros;
- **clientSideAttacks**: este módulo usa um *shell* reverso para fornecer ao servidor instruções para baixar arquivos maliciosos remotamente. Este módulo testa a capacidade do IDPS de proteger contra ataques do lado do cliente;
- **denialOfService**: testa a capacidade do IDPS para proteger contra tentativas de negação de serviço;
- **evasionTechniques**: várias técnicas de evasão são usados para verificar se o IDPS pode detectá-los;
- **fragmentedPackets**: várias cargas úteis fragmentados são enviados para o servidor para testar a sua capacidade de recompô-los e detectar os ataques;
- **ipReputation**: testa a capacidade do servidor para detectar o tráfego de/para servidores de baixa reputação;
- **normalUsage**: Cargas que correspondem a uma utilização normal;
- **pcapReplay**: permite reproduzir arquivos pcap;

- **shellcodes**: enviar vários *shellcodes* para o servidor na porta 21/tcp para testar a capacidade do servidor para detectar/rejeitar *shellcodes*;
- **testRules**: testando regras básicas. Estes ataques são supostamente para ser detectado pelos conjuntos de regras fornecidas pelo IDPS (PYTBULL, 2016).

Basicamente, cada um dos módulos tem capacidades especificas para executar comandos que simulem possíveis ataques, coletar informações, agrupar e analisar essas informações para ao final fornecer um relatório de todas as atividades coletadas. O relatório pode ser criado a partir de alterações nos arquivos de configuração, onde deve ser indicada as máquinas analisadas, ele irar indicar se as detecções foram totais, parciais. As detecções totais indicar que o IDS conseguiu identificar como um ataque tudo que foi considerado suspeito, ou seja, todos as atividades identificadas como suspeitas foram analisadas de acordo com as regras do IDPS e de fato continha alguma tentativa de ataque, da mesma forma acontece para as detecções parciais, porém, se a tentativa não se confirmar depois de analisada pelo IDPS, ele irá identificar apenas como uma possibilidade de ataque, sendo assim uma detecção parcial.

Neste trabalho a função do *Pytbull* foi exatamente como descrita anteriormente, depois de configurada a ferramenta foi executado o *script* com parâmetros responsáveis por ativar os módulos que executaram os testes.

3.7.3 iPerf

iPerf é uma ferramenta para medir a largura de banda e a qualidade de um *link* de rede, utiliza as diferentes capacidades de TCP e UDP para fornecer estatísticas sobre os *links* de rede, para cada teste ele relata a largura de banda, perda e outros parâmetros. Pode ser instalado facilmente em qualquer sistema UNIX/Linux ou Microsoft Windows e um anfitrião tem de ser definido como cliente, o outro como servidor (IPERF, 2016).

4 TRABALHOS RELACIONADOS

White et al. (2013) realizaram uma análise quantitativa entre os dois Sistemas de Detecção de Intrusão em Redes mais amplamente difundidos: Snort e Suricata. Na execução dos testes foi examinado o desempenho de ambos os sistemas, especificando nos testes um número máximo pacotes e variando a quantidade de núcleos de processamento de 1 até 24. A ferramenta *Pytbull* foi utilizada para analisar o desempenho do Snort e Suricata com diferen-

tes conjuntos de regras em cada IDPS. A análise compara o Suricata e o Snort tanto em uma única instância como em múltiplas instâncias. Nos resultados foi identificado como as mudanças nas regras e o número de pacotes afetam os resultados dos testes. Ao final o Suricata apresentou melhor desempenho com mais núcleos conforme variou a quantidade de pacotes.

Park e Ahn (2016) apresentaram uma análise sobre a taxa de detecção e taxa de desempenho usando pelo Snort e Suricata. Neste trabalho foi utilizado o S*ecurity Onion*, um sistema operacional desenvolvido para a utilização de Sistemas de Detecção e Prevenção de Intrusão (*Intrusion Detection and Prevention Systems* – IDPS). Nesta análise foi utilizada a versão 12.04 com 3GB de memória RAM e para obter resultados precisos, foi desativado todos os *firewalls* e instalaram a vítima e o agressor na mesma rede. Os resultados apresentados neste trabalho mostram que o Snort obteve melhor desempenho (baixo uso de CPU) do que Suricata. Este resultado mostra que, se o usuário está tentando implementar IDPS em um ambiente de baixa CPU, Snort é a solução melhor para escolher. No entanto, Suricata possui algumas vantagens sobre o Snort com seu bom desempenho nas detecções simples e *multi-core*.

Singh e Singh (2014) realizam uma análise entre o IDPS de *host* OSSEC e o de rede Snort. No trabalho foi configurado um ambiente OSSEC com um agente e um servidor, nos testes ele conseguiu gerar vários alertas como envio de e-mail, alerta *syslog*, saída para o banco de dados, etc. No OSSEC eles destacam que a melhor característica desta ferramenta é a de enviar notificações de e-mail imediatas se qualquer tipo de ameaça for detectada. Além disso, foi criado um ambiente para teste com o Snort, utilizando quatro computadores conectados na mesma LAN. Uma máquina foi colocada em modo *sniffing* e as outras trocando pacote ARP, de acordo com os testes ele foi capaz de detectar e analisar todos os pacotes transferidos. Ao final concluiu que o uso de um IDPS depende dos requisitos e dos resultados que se busca, pois são flexíveis e podem ser utilizados para vários fins.

Os dois primeiros trabalhos listados relatam experimentos com o Suricata e o Snort. No primeiro caso é realizada uma análise quantitativa em um cenário mais robusto onde os IDPSs são submetidos a intensos testes com grandes quantidades de cargas de trabalhos. No segundo trabalho a análise é realizada em um cenário menos complexo e visa identificar qual dos dois IDPSs, Snort e Suricata, faz mais uso de recursos computacionais como memória RAM e o uso de CPU, tratando-se apenas de uma análise de desempenho. O último trabalho listado realiza uma análise comparativa entre os sistemas de detecção de intrusão baseado em redes, o Snort, e o sistema baseado em *host*, o OSSEC, este trabalho ele visa apenas identificar as diferenças entre os dois sistemas e avaliar como é o comportamento de cada um quando submetido aos mesmos testes.

Neste trabalho, usamos as principais características dos outros artigos aqui listados e realizamos uma análise mais completa com um comparativo da capacidade de detecção e o desempenho de cada IDPS durante seu funcionamento. Utilizamos dois IDPSs de rede e um de *host*: Snort, Suricata e OSSEC, respectivamente. A escolha de dois IDPSs de rede e apenas um de *host* se justifica pelo fato de os de rede possuírem diferenças significativas como fato do Suricata ser *multi-threading* que lhe possibilita trabalhar com maior quantidade de dados ao contrário do Snort. Com relação a escolha do OSSEC, que é de *host*, se justifica pelo fato dele possuir um sistema de notificação de alertas com maior precisão e com mais detalhes sobre o que acontece nos *hosts*. Buscamos identificar a capacidade de detecção a ataques maliciosos de cada um dos IDPSs.

5 EXPERIMENTOS

Este projeto propões uma análise da taxa de detecções e desempenho de 3 Sistemas de Detecção e Prevenção de Intrusão (*Intrusion Detection and Prevention System* - IDPS). Realizamos os testes nos IDPSs Snort, Suricata e OSSEC, para avaliar a taxa detecção utilizamos com *Pytbull* e nos testes de desempenho usamos o iPerf.

Para a criação do cenário e a realização dos testes utilizamos uma máquina física do **Laboratório de Redes da Universidade Federal do Ceará - Campus Quixadá**. Na tabela 1, estão listadas as configurações da máquina.

Tabela 1: Configuração da máquina física

Configurações	
Sistema Operacional	Ubuntu Desktop 14.04 64-bits
Memória	8 GB
Disco Rígido	500 GB
Processador	Intel Core i7
Versão VirtualBox	5.0

Fonte: Elaborada pela autora.

Com o VirtualBox instalado na máquina física foi possível criar e configurar as 2 máquinas cliente e 1 servidor onde foram realizados todos os testes. Na tabela 2, está ilustrada todas as configurações do *host* virtual servidor onde estão os IDPS.

Tabela 2*: Configuração da máquina virtual (Servidor)*

Configurações	
Sistema Operacional	Ubuntu Desktop 14.04 64-bits
Memória	3 GB
Disco Rígido	20 GB

Versão dos IDPS	Snort 2.9 | Suricata 3.1 | OSSEC 2.8.2

Fonte: Elaborada pela autora.

Por fim, na tabela 3, temos a configuração do *host* virtual cliente onde foram instaladas as ferramentas utilizadas nos testes.

Tabela 3: *Configuração da máquina virtual (cliente)*

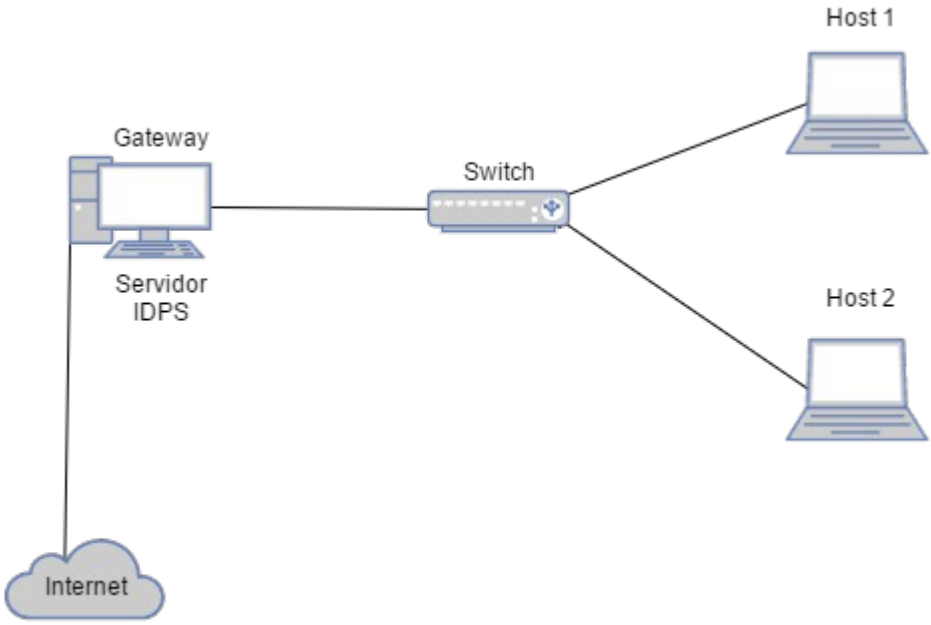
Configurações	
Sistema Operacional	Ubuntu Desktop 14.04 64-bits
Memória	1 GB
Disco Rígido	20 GB
Versão iPerf	2.0.8
Versão Pytbull	2.1

Fonte: Elaborada pela autora.

Com o cenário configurado foi possível realizar dois experimentos, um para testar capacidade de detecção de eventuais intrusões em cada IDPS e o segundo visava avaliar o desempenho de cada um dos IDPS e assim identificar qual deles fez mais uso dos recursos computacionais.

A topologia criada para a realização dos experimentos foi composta por 4 elementos sendo, *host* 1 e *host* 2 os clientes, um servidor e um switch. No *host* 1 estavam instaladas as ferramentas Pytbull e iPerf, no *host* 2 não havia nenhum programa específico sendo executado, essa máquina era responsável apenas por receber os ataques do Pytbull e o tráfego enviado pelo iPerf, já no servidor estavam instalados os 3 IDPSs que ficou em modo *sniffing* apenas coletando todas as informações da rede e do *host* durante a execução dos testes, outro elemento que está presente é um switch que representa um *bridge* criada pelo VirtualBox, na configuração de rede utilizamos uma rede interna apenas interligando as máquinas utilizadas. A topologia descrita pode ser vista na figura 9

Figura 9: Topologia utilizada nos experimentos



Fonte: Elaborada pela autora.

5.1 Experimento 1 – Capacidade de Detecção

Os primeiros testes realizados foram os de taxa de detecção de cada IDPS, esses foram realizados com o *Pytbull*, trata-se de um *framework* de testes que faz uso de diversas ferramentas como Hping3 para tentativas de Ataques de Negação de Serviços, *Ncrack* que realiza ataques de força bruta, entre outras. O *Pytbull* é composto por 10 diferentes módulos responsáveis por executar comandos referentes a cada um dos ataques realizados como por exemplo, existe um módulo chamado *denielOfService* que por sua vez irá executar comandos do Hping3 para lançar Ataques de Negação de Serviços.

O Pytbull funciona da seguinte forma, em seu arquivo de configuração é adicionado o IP do host onde se encontra o IDPS e é passado o caminho do arquivo de alertas gerados por cada um, desta forma sempre que houver alterações nesse arquivo o *Pytbull* será capaz de identificar qual tipo de alerta foi gerado e assim ler esta informação e adicionar ao seu relatório geral produzido ao final dos testes, essa ferramenta realiza aproximadamente 300 testes com seus 10 módulos.

O método de detecção usado neste trabalho foi baseado em assinatura e o conjunto de regras utilizadas nos IDPS foram obtidos da base de dados de cada uma das comunidades responsáveis pelo desenvolvimento e criação de novas regras dos IDPS, como é o caso Snort e Suricata que tem suas regras atualizadas diariamente. As regras utilizadas Pelo Snort foram atualizadas até a data de 7 junho de 2016 e as do Suricata até a data de 13 de junho de 2016,

no caso do OSSEC utilizamos seu conjunto de regras padrão que acompanha a ferramenta durante o processo de instalação. As métricas utilizados na avaliação da capacidade de detecção de cada IDPS podem ser vistas na tabela 4.

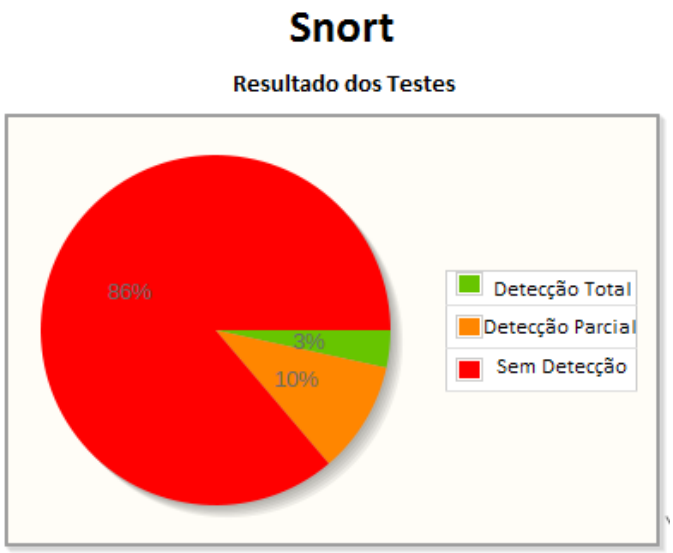
Tabela 4*: Métricas Utilizadas*

Métricas Utilizadas
Percentual de Detecções Totais
Percentual de Detecções Parciais
Percentual de não Detecções

Fonte: Elaborada pela autora

Como pode ser visto na figura 10 os resultados do Snort não foram muito satisfatórios para detecção total ou mesmo parcial dos testes realizados, identificando como tentativa de intrusão apenas com 3% de todos os ataques realizados. Já para detecção parcial soma-se só 10 % do total, mesmo assim não representa muito se comparado aos 86% de tentativas que não foram identificadas por ele.

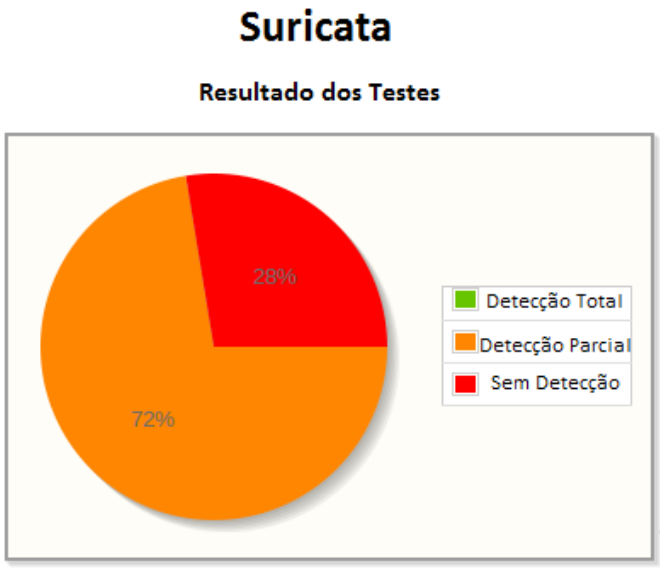
Figura 10: Taxa de detecção do Snort



Fonte: Elaborada pela autora.

Na figura 11 pode ser visto os resultados obtidos pelo Suricata, ele obteve 72% de detecção parcial e 28% sem detecções, apesar de não ter realizado nenhuma detecção total se o compararmos como os resultados obtidos pelo Snort ele apresenta uma enorme vantagem por identificar 72% como possível intrusão de todos os ataques realizados, em um cenário real o Suricata seria o indicado entre os dois.

Figura 11:Taxa de detecção do Suricata

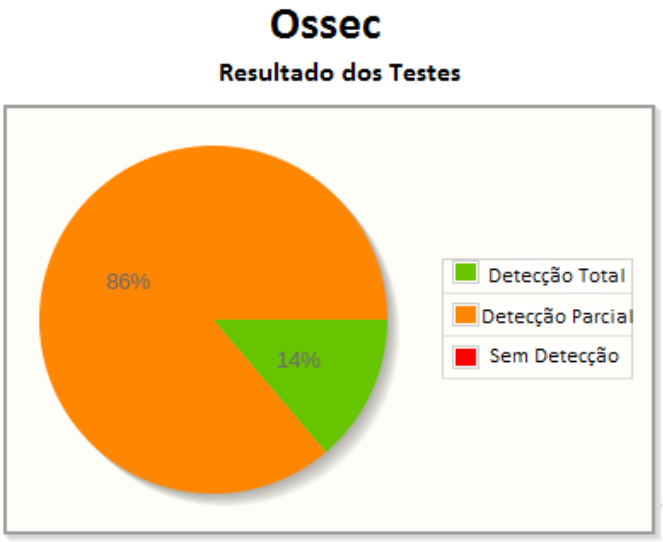


Fonte: Elaborada pela autora.

O OSSEC, que diferente dos outros dois IDPS, é baseado em *host*, executa avaliando qualquer alteração que possa representar algum tipo de intrusão, ou mesmo a tentativa, no *host* que está sendo monitorado. Ele identifica alterações em arquivos logs, por exemplo, e isso permite que ele obtenha bem mais informações e identifique uma maior quantidade de atividades maliciosas que esteja ocorrendo dentro do *host* monitorado o que contribuiu para que ele obtivesse taxas de detecção bem mais significativas.

Como mostra na figura 12, foram identificados em 86% das ocorrências atividades parcialmente suspeitas e que neste caso poderiam de fato não representar qualquer intrusão, ou representar apenas alterações em arquivos. Ou mesmo poderiam representar algum tipo de tentativa real, porém mascarada fazendo com que a ferramentas identificasse apenas como parcialmente suspeita e 14% para aquelas que de fato representavam a ocorrência real de ataques de intrusão no *host*.

Figura 12: Taxa de detecção do OSSEC



Fonte: Elaborada pela autora.

5.2 Experimento 2 – Desempenho

A analise de desempenho feito com todos Sistemas de Detecção de Intrusão foi realizada com o auxilio da ferramenta iPerf. A execução destes testes foi composta pelos seguintes passos, as duas máquinas Ubuntu foram responsáveis por gerar o tráfego na rede onde se encontram os IDPS, assim seria possível que o Snort e o Suricata conseguisse pegar tudo que acontecia nesta rede, já que são baseados em rede, e também em uma das máquinas foi instalado um agente do OSSEC responsável por enviar aos servidor OSSEC toda e qualquer o modificação ou tentativas de intrusão que ocorrece.

O iPerf foi responsável por gerar tráfego entre as máquinas e então os IDPSs seriam forçados a executar um monitoramento, nesta etapa o iPerf lançou 1000 pacotes TCP de 1500 *bytes* variando o número de usuários entre 1, 3, 5, 7, 9 e 11 em cada execução, em seguida dois pequenos *script* é executado coletando as informações de consumo de CPU e de memória de cada IDPS ao final dos testes. Na fase de gerar os resultados foi contabilizada a média de consumo de CPU dos IDPS em relação aos usuários como mostra a figura 12. Esses scripts estão disponíveis em https://github.com/Geyciane/Script. Na tabela 5 estão descritos os fatora e níveis utilizados nos testes de desempenho.

Tabela 5*: Fatores e Níveis*

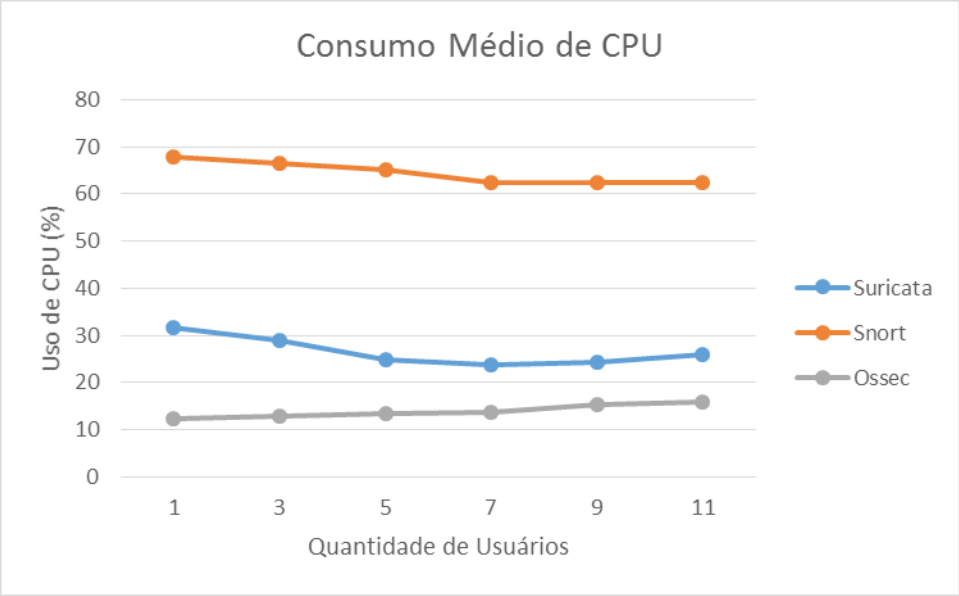
Fatores	Níveis
Número de Usuários	1, 3, 5, 7, 9, 11

Quantidade de Pacotes	1000
Tamanho dos Pacotes	1500 *bytes*

Fonte: Elaborada pela autora.

O consumo de CPU dos *hosts* pode ser visto na figura 13 abaixo. O IDPS Snort obteve o maior percentual de consumo de CPU, utilizando mais de 60% dos recursos, o Suricata se manteve entre 30 % a 25%, por ser *multi-threading* e dividir suas atividades em vários processo ele consegue manter o consumo dos recursos em uma média bem inferior ao Snort, por fim no OSSEC os dados foram coletas na máquinas onde os agentes estavam instalados e apresentou o menor consumo dentre os 3 IDPS analisado ficando entre 12 % a 15% do total.

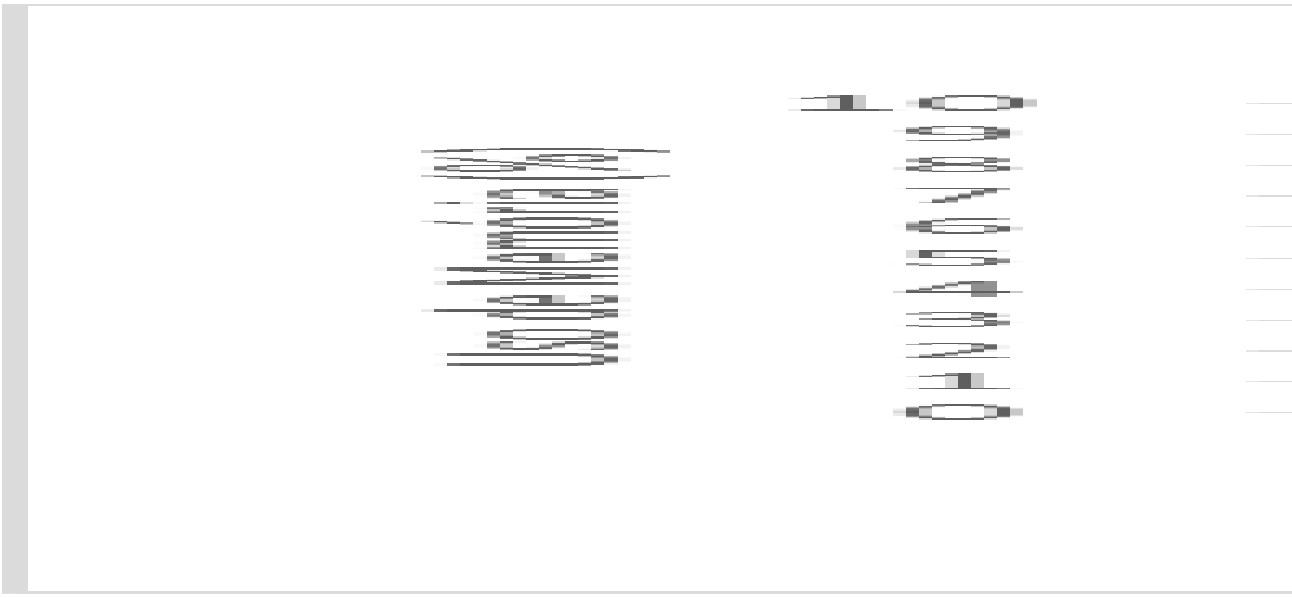
Figura 13: Média de consumo de CPU por usuário



Fonte: Elaborada pelo Autor.

Durante a execução dos testes notou-se que o consumo de memória se manteve constante, mesmo com a variação do número de usuários mantendo em 9% o Suricata, 5% o Snort e entre 3 % e 4% para o OSSEC conforme varia a quantidade de usuários, a figura 14 mostra como ficou o resultado final.

Figura 14: Média de consumo de memória por usuário



Fonte: Elaborada pelo Autor.

O consumo de memória dos IDPSs no decorrer dos testes não apresentou tanta variação, pois durante a execução eles estavam realizando basicamente atividade de processamento como indica figura 12 em que o uso de CPU foi muito intenso em alguns momentos durante a realização dos testes.

5.3 Discussão

Fazendo uma análise dos resultados obtidos, foi possível identificar que o uso das ferramentas se mostraram eficientes, onde todos conseguiram identificar a existência de alguma tentativa de intrusão. O OSSEC conseguiu realizar a maior quantidade detecções, a justificativa para essa alta taxa da ferramenta é porque ele monitorar qualquer modificação em arquivos de log ou atividades que pareçam suspeita o que pode ser positivo se a intenção do administrador da rede for monitorar essas alterações, do contrário é desnecessário que ele gere tantos alertas, nos testes de desempenho também mostrou bons resultados exigindo pouco dos recursos computacionais tanto de memória como de CPU.

O Snort, ao contrário do OSSEC, obteve um alto percentual de não detecções, dessa forma podemos entender que é mais criterioso ao tentar identificar tentativas de intrusão, não basta apenas que haja alterações em arquivos para que o alerta seja lançado, nos testes de desempenho foi o que mais fez uso de recursos computacionais ficando sempre acima de 60% no uso de CPU, neste caso o consumo de memória foi um fator positivo variando entre 5% e 7% conforme foram adicionados novos usuários. O Snort nesse caso

pode ser recomendado para uso em ambientes mais complexos e que disponha de uma grande infraestrutura para processamento da ferramenta.

O Suricata, por sua vez, conseguiu identificar uma grande quantidade ataques parciais o que é positivo, a ferramenta possui em seu conjunto de regras a capacidade identificar algumas atividades maliciosas e que não necessariamente precisam ser lançada como um alerta de possível ataques, mostra-se apenas como uma notificações de que há atividade suspeita, nos testes de desempenho para uso de CPU obteve bons resultados e se manteve sempre em torno dos 25%. Esse menor costume se justifica porque o Suricata é *multi-treading* podendo dividir o seu trabalho em outros processos. Em relação ao uso de memória, o Suricata foi o que mais consumiu entre os 3 IDPS, porém não foi um consumo muito expressivo, atingindo apenas 9%.

As considerações feitas sobre os resultados obtidos é que cada IDPS tem características específicas, realizam o mesmo trabalho mais podem exigir ambientes diferentes por exemplo, o OSSEC pode ser utilizado em ambiente onde não exista muitos *hosts*, uma vez que não é nada prático configurar várias máquinas com um agente OSSEC. O Snort pode ser utilizado em grandes ambientes, pois consegue monitorar tudo que acontece em rede a partir de uma máquina central onde o IDPS esteja configurado, mas vai precisar de uma máquina poderosa já que faz muito uso de CPU. Suricata pode ser usado tanto em ambientes robustos como em ambientes mais simples, pois nos testes mostrou não precisar de um *hardware* muito potente para realizar suas atividades oferecendo assim melhor escalabilidade, por ser *multi-treading.*

6 CONCLUSÕES

Objetivo do nosso trabalho foi realizar uma análise da taxa de detecção de Sistemas de Detecção de Intrusão baseado em rede e *host*. Para os testes de taxa de detecção utilizamos uma ferramenta capaz de executar aproximadamente 300 testes diferentes no ambiente monitorado pelos IDPS. Assim, foi possível gerar diferentes tipos de alertas e constatar se as ferramentas estavam realizando as detecções como previsto. Ao final foi possível identificar que dentre os 3 IDPS, o OSSEC apresentou melhores resultados e foi capaz de detectar uma maior quantidade de atividades maliciosas.

Já nos testes de desempenho a ideia foi usar o iPerf para criar tráfego entre as máquinas para que os IDPS pudessem identificar as atividades na rede, fixamos uma

quantidade de pacotes e seu tamanho em cada e nos testes variamos na quantidade de usuário, o que nos possibilitou acompanhar a variação dos resultados monitorados. Nos testes o OSSEC mostrou bom desempenho não chegando a resultados não muito expressivos para o uso dos recursos computacionais, podemos atribuir isso ao fato de que dependo do tipo de informação recebida no servidor OSSEC haverá casos ele executar mais tarefas e em outros casos irá realizar checagens de tempos em tempos, ou seja o agente não está constantemente averiguando o tráfego e isso justifica o baixo uso de memória e CPU, mais pode ser um ponto negativo já que irá demorar para reportar uma ocorrência..

Entre o Suricata e o Snort, o primeiro apresentou melhores resultados, pois foi capaz de gerar mais alertas de intrusão parcial do que os gerados pelo Snort, que por sua vez nos testes de taxa de detecção não apresentou bons resultados. Os dois IDPS apresentaram resultados bem distintos para uso de CPU, o Suricata se manteve utilizando sempre algo entorno de 25% a 30% aproximadamente sem grandes variações mesmo com muitos usuários essa sua vantagem pode ser explicada pelo fato dele utilizar *multi-threads* fazendo com que, ele seja capaz executar muitos processos sem sobrecarregar as atividades de processamento da máquina. O resultado para CPU do Snort mostrou que, diferente do Suricata, faz muito uso de processamento ficando em torno de 60 a 70% no total. Com relação ao uso de memória RAM eles tiveram sempre os mesmos resultados para todos os usuários sendo entre 5% a 7% o Snort e 9% o Suricata, pois eles executam mais atividades de processamento e não de memória.

Em resumo, cada um possui características próprias, o que faz da escolha entre um deles depender particularmente do ambiente que será monitorado, por exemplo, se for um ambiente mais robusto e com uma grande quantidade de máquinas interligadas a melhor escolha seria os IDPSs rede como o Snort e o Suricata, pois eles são capazes de identifica tudo que ocorre em uma rede sem necessidade de instalações nas máquinas monitoras, no entanto se o ambiente for mais simples e composto por menos máquinas o OSSEC é uma boa escolha, ele também é capaz de trabalhar em ambientes mais complexos, todavia a instalação e configuração dos agentes em cada máquina seria uma desvantagem.

Escolher um dos IDPS requer conhecer todas as necessidades e limitações do local que se deseja monitorar, optar por um que obteve os melhores resultados não significa dizer que ele irá fazer um bom trabalho se suas necessidades de configuração não estiverem de acordo com o que oferece o IDPS ou necessita para o bom funcionamento.

Como trabalhos futuros recomendamos a realização de testes com outras ferramentas IDPS em conjuntos com as que foram utilizadas, também indicamos o uso de

mais fatores e níveis, com uma maior variações nas quantidades de pacotes ou quantidades mais expressivas no número de usuário e por fim recomendamos também o uso de um conjunto de regras mais específicas para determinados tipos de ataques que pode contribuir para melhores resultados nas detecções totais nos resultados dos testes.

Esperamos que este trabalho possa ser útil para aqueles que desejam estudar ou implantar um dos detectores avaliados, auxiliando na escolha por um ou outro para o ambiente que deseja monitorar.

REFERÊNCIAS

CERT.br. Disponível em: < http://www.cert.br/> Acessado em Novembro de 2016.

DAY, David; BURNS, B. **A performance analysis of snort and suricata network intrusion detection and prevention engines**. In: Fifth International Conference on Digital Society, Gosier, Guadeloupe. 2011. p. 187-192.

GTA. IDS **- Sistema de detecção de Intrusão.** Disponível em http://www.gta.ufrj.br/grad/07_2/rodrigo_leobons/deteccao.html. Acesso em: 28 jun 2016

IPERF. < https://iperf.fr/>. Acessado em Novembro de 2016

KHAMPHAKDEE, Nattawat; BENJAMAS, Nunnapus; SAIYOD, Saiyan. **Improving Intrusion Detection System based on Snort rules for network probe attack detection**. In Information and Communication Technology (ICoICT), 2014 2nd International Conference on. IEEE, 2014. p. 69-74.

KUMAR, Vinod; SANGWAN, Om Prakash**. Signature based intrusion detection system using snort**. In International Journal of Computer Applications & Information Technology, 2012, p. 35-41.

MEHTA, Vishal et al. **Threat prediction using honeypot and machine learning.** In: Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), In International Conference on. IEEE, 2015. p. 278-282.

NAKAMURA, Emilio Tissato, de GEUS, Paulo Lício. **Segurança de Redes em Ambientes de Cooperativos. Novatec**. 2007. 278 p.

OISF. **What is Suricata.** Disponível em: <

https://redme.openinfosecfoundation.org/projects/suricata/wiki/whatissuricata>. Acessado em: Dezembro 2016

OSSEC. **Arquitetura Ossec**.. Disponível em: <http://ossec.github.io/docs/manual/ossec-architecture.html>. Acesso em: junho. 2016.

PARK, Wonhyung; AHN, Seongjin**. Performance Comparison and Detection Analysis in Snort and Suricata Environment.** Wireless Personal Communications, p. 1-12, 2016.

PIPER, Steve **Intrusion Prevention Systems For Dummies**. Indianapolis: Wiley Publishing, 2011. 3 - 5 p.

POONGODI, M., BOSE, S. **Design of Intrusion Detection and Prevention System (IDPS)using DGSOTFC in collaborative protection networks**. In 5th International Conference on Advanced Computing, ICoAC 2013, p. 172–178.

PYTBULL. Disponível em: <http://pytbull.sourceforge.net/index.php?page=documentation>. Acessado em Novembro de 2016.

SINGH, Amrit Pal; SINGH, Manik Deep. **Analysis of Host-Based and Network-Based Intrusion Detection System**. In International Journal of Computer Network and Information Security, 2014, p. 41.

SNORT**. O que é e como funciona uma ferramenta IDS**. Disponível em <http://www.snort.org.br/comofuncionaids.php>. Acesso em jun 2016.

WANG, Xinli et al. **Administrative evaluation of intrusion detection system.** In Proceedings of the 2nd annual conference on Research in information technology**.** ACM, 2013. p. 47-52.

WHTIE, Joshua S, FITZSIMMONS, Thomas, MATTHEWS, Jeanna N. **Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata.** In SPIE Defence, Security and Sensing Internetional Society for Optics and Photonics, 2013.

VIRTUALBOX. Disponível em: <https://www.virtualbox.org/manual/ch01.html>. Acessado em Novembro de 2016.

VUKALOVIĆ, J.; DELIJA, D. **Advanced Persistent Threats-detection and defense. Information and Communication Technology, Electronics and Microelectronics (MIPRO),** In 38th International Convention on IEEE, 2015. p. 1324-1330.