UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE PREVENÇÃO E DETECÇÃO DE INTRUSOS EM UM AMBIENTE CORPORATIVO

TRABALHO DE GRADUAÇÃO

Caio Cezar Athayde Trevisan

Santa Maria, Rio Grande do Sul, Brasil
2015

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE PREVENÇÃO E DETECÇÃO DE INTRUSOS EM UM AMBIENTE CORPORATIVO

Caio Cezar Athayde Trevisan

Trabalho de Graduação apresentado ao Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Tecnólogo em Redes de Computadores.**

Orientador: Prof. Alfredo Del Fabro Neto

Santa Maria, Rio Grande do Sul, Brasil
2015

Universidade Federal de Santa Maria
Colégio Técnico Industrial de Santa Maria
Curso Superior de Tecnologia em Redes de Computadores

A Comissão Examinadora, abaixo assinada,
aprova o trabalho de conclusão de curso

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE PREVENÇÃO E DETECÇÃO DE INTRUSOS EM UM AMBIENTE CORPORATIVO

elaborado por
Caio Cezar Athayde Trevisan

como requisito parcial para obtenção do grau de
Tecnólogo em Redes de Computadores.

COMISSÃO EXAMINADORA

Prof. Alfredo Del Fabro Neto
(Presidente/Orientador)

Renato Preigschadt de Azevedo, Me. (UFSM)

Tiago Antônio Rizzetti, Me. (UFSM)

Santa Maria, 03 de Julho de 2015.

Dedico este trabalho a duas pessoas que foram a base da realização deste sonho: meu irmão Luiz Gustavo (in memorian) pela orientação, paciência e companheirsmo (sinto falta) durante os primeiros passos na escolha da profissão e a minha esposa Vanessa pelas incansáveis horas ao meu lado me apoiando incondicionalmente.

AGRADECIMENTOS

Primeiramente eu gostaria de agradecer a minha família: meus pais Gustavo e Rosane; meus irmãos Gustavo e Roberta; minha esposa Vanessa e meu cunhado Daniel pelo apoio, amor e carinho que recebo diariamente. Obrigado por sempre estarem ao meu lado quando eu precisei.

Aos meus amigos Diogo e Eliza que tem estado ao meu lado, me acompanhando neste desafio e diariamente se mostrando preocupados tentando ajudar, e também por me proporcionaram inúmeras oportunidades nestes últimos anos que eu nunca imaginei poder estar aproveitando.

Ao Prof. Alfredo por ter topado este desafio de orientar a distância, por estar sempre disposto a ajudar no que precise e pelas cobranças fundamentais na finalização deste projeto.

Aos professores do curso de Redes de computadores, principalmente ao amigo Murilo, que foi fundamental neste período longe da Universidade.

A todo mundo que não foi citado, mas que de alguma forma contribuiu para que este sonho pudesse se tornar realidade.

“O preço da liberdade é a eterna vigilância. “

- Thomas Jeferson

RESUMO

Trabalho de Conclusão de Curso
Curso Superior de Tecnologia em Redes de Computadores
Colégio Técnico Industrial de Santa Maria
Universidade Federal de Santa Maria

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE PREVENÇÃO E DETECÇÃO DE INTRUSOS EM UM AMBIENTE CORPORATIVO

AUTOR: CAIO CEZAR ATHAYDE TREVISAN
ORIENTADOR: Prof. ALFREDO DEL FABRO NETO

Data e Local da Defesa: Santa Maria, 03 de Julho de 2015.

O presente trabalho tem por objetivo realizar um estudo comparativo entre dois sistemas de detecção de intrusos em um ambiente corporativo. Este estudo define qual dos *softwares* é o mais documentado em relação a instalação e configuração, qual possui a melhor eficácia na detecção de possíveis ataques e qual obteve o melhor desempenho durante a realização dos testes. Os sistemas utilizados são o *Snort* e o *Suricata*, ambos são *Open-Source*. O estudo deu-se a partir de uma análise bibliográfica e testes reais com ambos os *softwares* implementados em diferentes computadores para evitar anomalias nos resultados de performance. Os dois *softwares* testados obtiveram bons resultados, entretanto, o *Snort* se mostrou melhor na identificação de possíveis ataques com requerimentos de hardware semelhante ao Suricata.

Palavras-chave: Segurança da Informação; Sistemas de detecção e prevenção de intrusos (IDS/IPS); Análise de performance; *Snort*; *Suricata*.

ABSTRACT

Capstone Project
Curso Superior de Tecnologia em Redes de Computadores
Colégio Técnico Industrial de Santa Maria
Universidade Federal de Santa Maria

COMPARATIVE RESEARCH OF INTRUSION DETECTION AND INTRUSION PREVENTION SYSTEMS

AUTHOR: CAIO CEZAR ATHAYDE TREVISAN
ADVISOR: Prof. ALFREDO DEL FABRO NETO

Place and Date: Santa Maria, July 03, 2015.

The main objective of this paper is to conduct a comparative research between two Intrusion Detection Systems. This study defines which of one of the software's is the most documented related with installation and configuration, has the best performance results and is more efficient on detecting threats on an intranet. The compared systems are Snort and Suricata, both Open-Source. The research started with a literature review and then some real tests with both tools implemented on separated computers to avoid any anomaly on the performance results. Both software's performed well, although, Snort had better results identifying possible attacks with similar hardware of Suricata.

Keywords: Information Security; Intrusion Detection Systems; Intrusion Prevention Systems; Perfomance Analysis; Snort; and Suricata.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ARPA - *Advanced Research Projects Agency*
HIDS - *Host Intrusion Detection System*
HTML - *Hyper Text Markup Language*
ICMP - *Internet Control Message Protocol*
IDS - *Intrusion Detection System*
IPS - *Intrusion Prevention System*
LAN - *Local Area Network*
LDAP - *Lightweight Directory Access Protocol*
LOG – Registro de evento
NIDS - *Network Intrusion Detection System*
NIST - *National Institute of Standards and Technology*
OWASP - *Open Web Application Security Project*
SNMP - *Simple Network Management Protocol*
SQL - *Structured Query Language*
TCP - *Transport Control Protocol*
UDP - *User Datagram Protocol*
UFSM - Universidade Federal de Santa Maria
VLAN - *Virtual Local Area Network*

SUMÁRIO

INTRODUÇÃO

Durante a Guerra Fria, em meados de 1942, foi necessário o desenvolvimento de um sistema de comunicação interligado que facilitasse a comunicação entre bases militares. Em 1969 a *Advanced Research Projects Agency* (ARPA), autorizada pelo Departamento de Defesa dos Estados Unidos da América (EUA), criou a ARPANET, uma rede de comunicações fechada, na qual tinham acesso somente os funcionários do governo (COSTA ALMEIDA, 1988). Costa Almeida (1988, p. 52-53) descreve a ARPANET "como um sistema de informações descentralizado e independente de Washington, para que fosse possível a comunicação entre cientistas e engenheiros militares".

Somente após muitos anos de mudanças, e com a dissociação da área militar é que surge o termo internet, uma vez que agora existiam duas redes diferentes para o meio militar e civil. A internet, segundo Paesani (2000, p. 25) "é vista como um meio de comunicação que interliga dezenas de milhões de computadores no mundo inteiro e permite o acesso a uma quantidade de informações praticamente inesgotáveis, anulando toda distância de lugar e tempo.

Os protocolos que compõe a Internet não foram projetados pensando em segurança, em uma época em que a comunicação era baseada na confiança entre os usuários envolvidos (Universidade). Então com o acentuado crescimento da internet e a facilidade de propagação de informações na rede, usuários maliciosos tiraram proveito desta arquitetura para aplicar golpes, ativismo, etc.

Para se defender, empresas começaram a utilizar filtros de pacote, como *firewall*, porém já não são mais suficientes para proteger devidamente os ambientes corporativos (intranet). *Firewalls* são excelentes ferramentas para o bloqueio de portas, porém com a necessidade do compartilhamento de informações através da internet, empresas necessitam abrir algumas destas portas, podendo assim gerar vulnerabilidades que podem ser exploradas.

Atualmente existem outras soluções em segurança da informação, são os sistemas de prevenção e detecção de intrusos (*Intrusion Prevention Systems*/IPS ou *Intrusion Detection Systems*/IDS). Segundo Sequeira (2012), o *firewall* pode bloquear alguns serviços impedindo que pacotes trafeguem por certos números de portas,

porém não é realizado uma análise no tipo de informação que está sendo trafegado pelas portas abertas. O IDS analisa o tráfego de pacotes na interface e o IPS pode realizar o bloqueio quando necessário.

O IDS é utilizado para monitorar todo o tráfego de entrada e saída de uma rede e averiguar os registros em busca de padrões que podem ser considerados ataques de rede ou sistema e que poderão comprometer a segurança das informações da empresa. Estes sistemas são responsáveis por avisar o administrador da rede.

A função do IDS é informar ao administrador que algo errado pode estar acontecendo. Por outro lado, o IPS após analisar algum registro de tráfego de rede e considerá-lo malicioso, pode tomar algumas medidas preventivas como buscar por ataques com características genéricas e tentar bloquear o acesso.

Iniciantes na área de invasão encontram facilmente uma grande variedade de *softwares* maliciosos. Essas ferramentas podem ser utilizadas para scanear, identificar e penetrar sistemas de computadores, tornando-se um perigo e um problema para os ambientes corporativos.

Cada vez mais as empresas buscam novos *softwares* que automatizam processos internos e o uso dos computadores e da rede tornam-se imprescindíveis. Segurança da informação ou da rede é extremamente importante, pois permite que a comunicação e as transferências de arquivos sejam realizadas de maneira segura. Quanto maior for a rede, mais difícil é mantê-la segura. Uma única solução é incapaz de proteger uma rede de computadores de diversos tipos de ataques diferentes. Diferentes tipos de *software* e até mesmo *hardware* devem ter seu uso planejado para que seja possível alcançar um nível aceitável de segurança.

Desse modo, a problemática da pesquisa é norteada pela questão: “Entre duas ferramentas IDS *Open-Source* (distribuídas gratuitamente) utilizadas em um ambiente corporativo, qual é a que possui resultados mais satisfatórios na instalação e configuração, na identificação de ataques e no desempenho?”.

Para responder a este questionamento, buscou-se comparar dois sistemas de detecção de intrusos, em um ambiente corporativo, e definir qual dos *softwares* é o mais documentado em relação a instalação e configuração, qual possui a melhor eficiência na detecção de possíveis ataques e o melhor desempenho na utilização de processador e memória RAM, durante a realização dos testes. Sendo assim, os objetivos específicos deste trabalho são:

a) Estudar dois sistemas de segurança contra intrusos em um ambiente corporativo;

b) Delimitar em quais aspectos serão comparados estes sistemas;

c) Realizar testes entre as ferramentas em computadores separados para evitar anomalias nos resultados;

d) Analisar os resultados.

O primeiro capítulo, Introdução, apresenta o estudo acerca de sistemas de segurança contra intrusos. O capítulo 2, Segurança da Informação, apresenta conceitos acerca da segurança da informação em ambientes corporativos. O terceiro capítulo intitulado Sistemas de Detecção e Prevenção de Intrusos (IDS/IPS) expõe a conceituação destes sistemas e a importância dos mesmos na detecção e prevenção de ataques em ambientes corporativos. O capítulo 4, Programas (Sensores), apresenta e descreve os dois sensores estudados no presente trabalho, *Snort* e *Suricata*. Os processos metodológicos são apresentados no quinto capítulo, denominado Metodologia. A análise dos resultados do estudo é apresentada no sexto capítulo, Resultados e Discussões. Por fim, o capítulo 7, Considerações Finais, expõe as conclusões do presente estudo.

2 SEGURANÇA DA INFORMAÇÃO

As intranets atuais apresentam uma grande complexidade pois possuem diferentes sistenas operando em conjunto como: sistemas Windows/UNIX, banco de dados e acessos entre filiais e matrizes. Assim, é cada vez mais difícil o gerenciamento e controle dados trafegados em ambientes empresariais.

Northcutt e Novak (2002) citam alguns pontos chaves para se ter uma segurança mínima. Estabelecer regras de segurança, analisar riscos, verificar a infraestrutura disponível e implementar um sistema de prevenção de intrusos e resposta a incidentes são vitais para o sucesso no combate a invasores. Também afirmam que o risco de problemas com segurança da informação é alto e deve ser aceito pelas empresas. Após ser aceito, deve ser estudado as possibilidades de invasão e desvio de informação privilegiada e compreender os problemas que podem ocorrer em caso de falhas.

Nakamura e Geus (2002) explicam que a segurança nesse tipo de ambiente se torna muito complexa e deve ser planejada para que apenas recursos necessários sejam utilizados e evitar acessos de terceiros. Deve ser feito uma análise sobre quais serviços devem ser disponibilizados *online*, para que assim possam ser tomadas medidas provisórias de controle de acesso, para dar maior segurança ao acesso a informação.

Ano após ano aumentam as ocorrências de problemas, como: redes inoperantes, vazamento de informações sigilosas, lentidões, botnets e entre outros. Estes causam inúmeros problemas para as empresas. Esse tipo de problema já não pode mais ser tratado por métodos ultrapassados de *firewall* (NSFOCUS, 2012).

A Figura 1 demonstra que, dentro de um ambiente corporativo, existem muitos problemas para serem enfrentados por um profissional de segurança da informação, sendo que o maior desafio é conseguir uma proteção confiável e eficaz contra ameaças digitais, problemas físicos e até mesmo funcionários mal-intencionados (SÊMOLA, 2003).

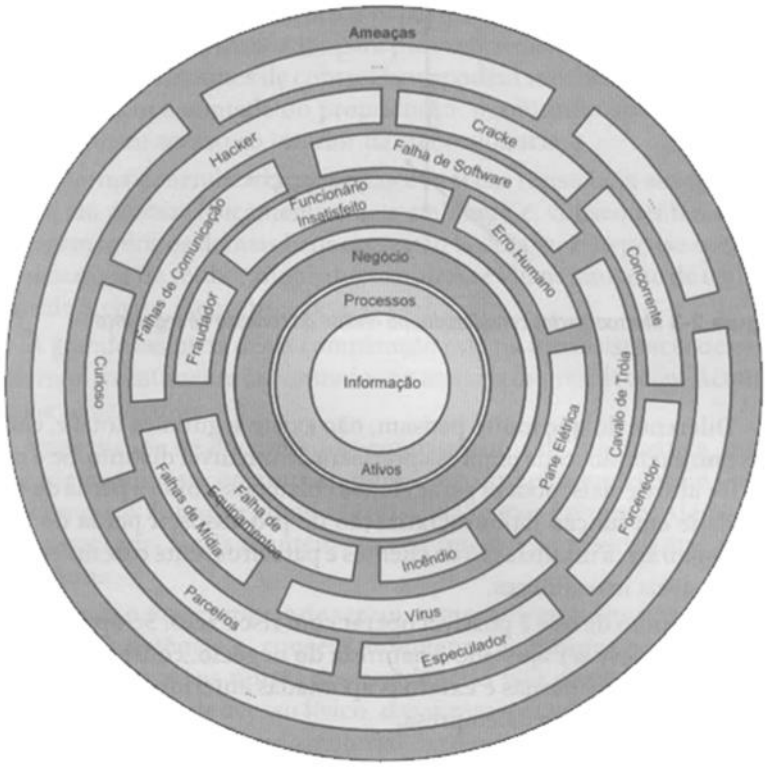


Figura 1: Ameaças e vulnerabilidades (SÊMOLA, 2003).

Na Figura 1 pode-se perceber que as ameaças e a vulnerabilidade estão presentes em todo e qualquer espaço na intranet. Dessa forma, é de extrema importância que o profissional conheça os perigos iminentes e as formas de defesa destes.

2.1 Ambientes Corporativos (intranet)

O termo intranet consiste em uma rede interna e privada em ambientes corporativos. Intranets são utilizadas para reduzir o desperdício de tempo (agilizando a troca de informações) e também para auxiliar no gerenciamento de tarefas, colaboração em equipe e produtividade.

Enquanto a internet é uma rede livre e sem controle, a intranet é gerenciada e somente acessos controlados e monitorados devem ser permitidos. Pinho (2003, p. 226) diz que “as intranets são, por natureza, *webs* privativas projetadas com base em fronteiras bem nítidas”. Intranets são delimitadas por empresas e estes limites se diferenciam conforme o objetivo de quem as planejaram.

Na atualidade a maior preocupação nas intranets é o compartilhamento de informação de uma maneira segura, isto é, ter certeza que o que está sendo transferido entre dispositivos internos e externos estejam devidamente seguros contra acessos indevidos. Ferramentas de varredura, tais quais o NMAP, facilmente encontram falhas na rede que podem vir a ser exploradas por pessoas mal-intencionadas.

2.2 Ameaças Digitais

Profissionais da segurança da informação enfrentam um conflito diário com diferentes tipos de ameaças digitais. Estas podem ser *softwares* maliciosos (*malwares*) que instalados em computadores podem permitir o acesso indevido de intrusos a rede corporativa, engenharia social realizada por hackers mal-intencionados e/ou entre outras.

Davis, Bodmer e Lemasters (2010) apontam a violação da relação de confiança do ser humano (engenharia social) como a mais antiga e, ainda, a mais eficiente forma de distribuição de *malwares* pela *internet*. Os autores ainda explicam que através desse método, o atacante consegue enganar o usuário e faz com que programas sejam instalados normalmente sem nem o usuário desconfiar.

Já os *malwares*, de acordo com o NIST:

> Malware refere-se a um programa que, inserido em um sistema, normalmente de maneira secreta, com a intenção de comprometer com a confidencialidade, integridade, ou disponibilidade da informação da vítima, aplicações, ou sistema operacional (SO) ou de outra forma somente atrapalhar a vítima[1] (NIST, 2005, p. ES-1, tradução nossa).

Hardikar (2008) explica que *malware* é um termo que representa qualquer programa de computador que possua um objetivo malicioso. Ele também cita que dentre os mais diversos tipos de malwares existentes, os mais comuns são: *virus*, *trojans* e *rootkits*.

[1] *Malware […] refers to a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system (OS) or of otherwise annoying or disrupting the victim.*

2.2.1 Vírus

Vírus são programas de computadores capazes de se autorreplicar (criar cópias de si mesmo) e distribuir estas cópias para outros arquivos e até mesmo para outros computadores em uma mesma rede. São formados basicamente por duas partes: o *payload*, onde esta contido o objetivo do *vírus* e o *trigger*, mecanismo responsável por ativar o *payload* quando as condições previstas forem cumpridas. *Vírus* também podem ser apenas para irritar o usuário, como bloqueios de tela ou reprodução de áudios, ou até mesmo encaminhamento de informações do usuário para o desenvolvedor do programa (NIST, 2005, p. 2-1).

2.2.2 Trojans

Inspirado na mitologia grega, *trojans* ou cavalos de Tróia são, contrários aos v*írus*, programas não autorreplicantes que parecem ser legítimos, mas também possuem um *payload* malicioso. Bloquear um trojan pode ser um trabalho bem desafiador, pois estes são desenvolvidos para esconderem-se dentro do código de programas genuínos e acabar passando despercebido por sistemas de monitoramento (NIST, 2005, p. 2-4).

2.2.3 Rootkits

Rootkits podem servir para capturar as teclas pressionadas no teclado, para escanear pacotes HTTP e inclusive modificar o código *html* de páginas *web*, principalmente páginas de bancos (Davis, Bodmer e Lemasters, 2010).

Também afirmam que um dos *rootkits* mais avançados, já descobertos, foi o *malware* chamado “*Mebroot*”. Este instalava arquivos de recuperação no “*Master Boot Record*” (MBR)[2], assim, cada vez que o programa fosse removido o mesmo instalava-se novamente quando o computador era reiniciado. O computador infectado passava a executar comandos executados pelo *hacker* e enviar informações diretamente para um servidor na *web*. O maior objetivo de todo esse complexo *software* era obter qualquer tipo de informação que ajudasse no roubo de dinheiro da vítima.

[2] Primeiros 512 bytes do disco-rígido primário do computador.

Além desses *softwares* há também diversas falhas de segurança já conhecidas. A *Open Web Application Security Project* (OWASP)[3] mantém um top 10 de um amplo consenso das mais críticas falhas de segurança em aplicações *web*. Atualmente a falha que está no topo da lista é conhecida por "*Injection*", o que nada mais é que, por meio de consultas, injetar códigos no servidor através de falhas em servidores SQL e/ou LDAP ou até mesmo em sistemas operacionais não devidamente configurados e fazer com que sejam executados comandos não intencionais.

Assim como os *malwares*, estas falhas de segurança em aplicações *web* podem ser exploradas e, através de ataques, serem utilizadas para que informações sigilosas sejam acessadas em um ambiente corporativo. Para proteger a rede contra estes tipos de ataques, não existe uma fórmula única de controle, é necessário, profissionais capazes, e sempre atualizados com novas tendências na área de segurança. E como complemento, é de suma importância a instalação e configuração de programas específicos para a segurança da rede, chamados de Sistemas de Detecção e/ou Prevenção de Intrusos (IDS/IPS), os quais serão abordados no próximo capítulo.

[3] Comunidade aberta, dedicada a ajudar organizações a desenvolverem aplicações confiáveis. Disponível em: https://www.owasp.org/index.php/Top_10_2013-Top_10. Acesso em: 8 maio 2015.

3 SISTEMAS DE DETECÇÃO E PREVENÇÃO DE INTRUSOS (IDS/IPS)

Piper (2011) aponta que até 1990, virtualmente, todos os ataques baseados em rede eram bloqueados apenas pelos *firewalls*. Já hoje em dia, com ataques mais complexos, *firewalls* e antivírus não conseguem mais manter uma proteção adequada. Assim, os IDS e/ou IPS se tornam essenciais para que se tenha um elevado nível de segurança na rede de uma empresa.

Piper (2011) também aponta que a principal diferença entre um *firewall* e um IDS/IPS é a maneira como o pacote é analisado. O autor explica que o *firewall* faz uma análise do pacote, identificando o endereço e a porta de origem e destino, independente do seu *payload* (conteúdo). Por outro lado, o IDS/IPS foi desenvolvido para analisar o *payload* do pacote e, baseado em configurações pré-estabelecidas, pode avisar ao administrador de rede, bloquear ou permitir o tráfego.

Bace e Mell (2011) definem Sistemas de Detecção e/ou Proteção como o processo de monitorar eventos ocorridos em uma rede e analisá-los em busca de sinais de intrusão. Estes eventos podem ser tentativas de comprometer a confidencialidade, integridade ou, até mesmo, a segurança da informação.

Estes sistemas podem ser, geralmente, configurados para gerarem alertas de duas maneiras distintas: baseados em regras ou anomalias. Sistemas baseados em regras comparam o pacote recebido com uma biblioteca de regras pré-configuradas e se cumprido as especificações, um alerta é gerado. Já os sistemas baseados em anomalias estabelecem um padrão de tráfego e o consideram como “normal”, e então a qualquer momento em que o tráfego monitorado sair deste padrão, um alerta será gerado.

Os IDS/IPS podem ser implementados em uma rede de computadores de diversas maneiras. Os modelos de configuração mais encontrados atualmente são os sistemas baseados em rede e os sistemas baseados em *hosts*. Há também os sistemas híbridos, que são os dois modelos combinados.

Burton, et al (2003) definem os Sistemas de Detecção de Intrusos baseados em rede (NIDS) como sistemas inteligentes, distribuídos pela rede, que passivamente inspecionam todo o tráfego que é transmitido entre computadores e equipamentos de

rede. Os NIDS podem ser *softwares* instalados em computadores ou até mesmo *hardwares* dedicados para tal função.

Os autores também explicam que com a complexidade das redes atuais, com VLANS[4] configuradas, empresas de telecomunicações precisaram desenvolver técnicas chamadas *port-mirroring* ou *SPAN ports*, ou seja, a replicação de todo tráfego de uma porta de um switch para outra. Assim os NIDS podem ser instalados em uma destas *SPAN ports* e analisar todo o tráfego da rede.

Já os Sistemas de Prevenção de Intrusos baseados em rede (do tipo *InLine*), são *softwares* ou *hardwares* dedicados que precisam ser instalados entre todo o tráfego que precisa ser monitorado, e não somente replicados através de *SPAN ports*. Assim, quando alertas são gerados, ações podem ser tomadas automaticamente, como o descarte imediato do pacote capturado.

Tanto os NIDS, quanto os IPS do tipo *InLine* podem ser criticamente afetados por redes com altas taxas de transferência. Em redes *gigabits*, é comum que os sistemas do tipo *software* ignorem muitos pacotes, permitindo a passagem sem que sejam analisados.

Segundo Davis, Bodmer e Lemasters (2010) os Sistemas de Detecção e/ou Prevenção baseados em Host (HIDS/HIPS), são basicamente *softwares* que necessitam ser instalados em cada computador da empresa, onde este, fica responsável pelo monitoramento apenas da máquina em si. Estes são responsáveis, basicamente, pela comunicação da interface de rede do computador com a rede, acesso a disco e gerenciamento de processos.

Garfinkel e Rosenblum (2003) aponta que configurações incorretas ou incompletas tanto do sistema operacional como do HIDS fazem com que a dificuldade de implementação seja muito grande. Um grande problema que os HIDS/HIPS enfrentam é a possibilidade de o atacante invadir o computador onde foi feita a instalação do *software*, e desabilitá-lo, comprometendo toda a segurança da rede.

Sistemas híbridos tendem a ser a opção com um maior nível de segurança em uma rede corporativa. Ainda que não exista uma solução perfeita, uma combinação de diferentes ferramentas, como mostra a Figura 2, pode elevar a segurança para níveis muito bem aceitos (STONESOFT, 2007, p. 22).

[4] Redes virtuais, logicamente independente e com diferentes domínios de broadcast.

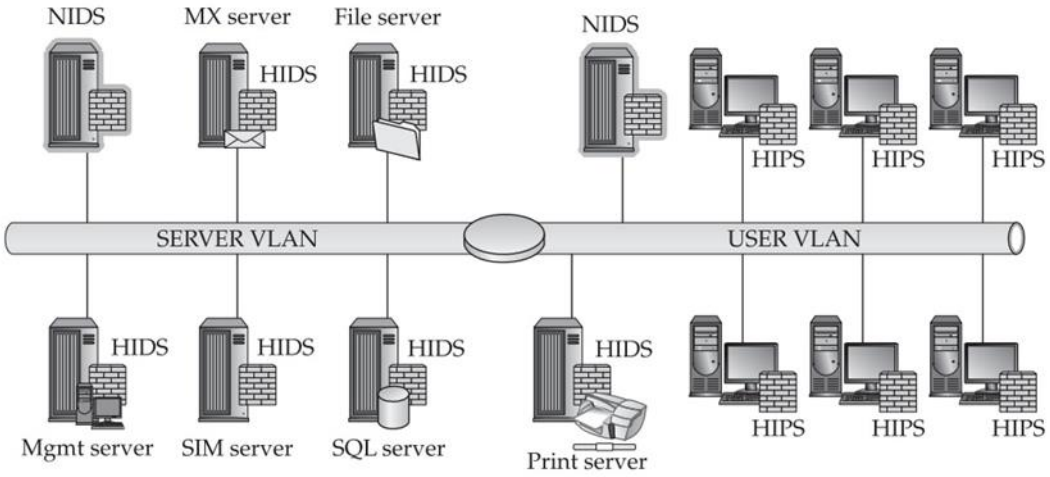


Figura 2: Sistema Híbrido de Prevenção de Intrusos. (DAVIS, BODMER e LEMASTERS, 2010).

A Figura 2, representa um modelo híbrido de Sistemas de Detecção e Prevenção de Intrusos. Nos computadores dos usuários foi instalado um HIPS para o gerenciamento local da máquina e proteção em tempo real. Já nos servidores foi instalado um HIDS para que o administrador da rede possa ter um total controle sobre o tráfego e tomar as atitudes necessárias quando preciso. Finalmente nas duas VLANS existentes, foram colocados NIDS na porta *SPAN* dos switches para que todo o tráfego seja monitorado.

Finalmente o funcionamento de um IDS baseado em regras se dá, genericamente, através de três etapas: monitoramento dos pacotes pelo sensor, análise dos pacotes e emissão de alertas. No caso do IPS, uma etapa extra é necessária: após a análise do pacote, este é liberado ou descartado e então é gerado o registro do acontecido.

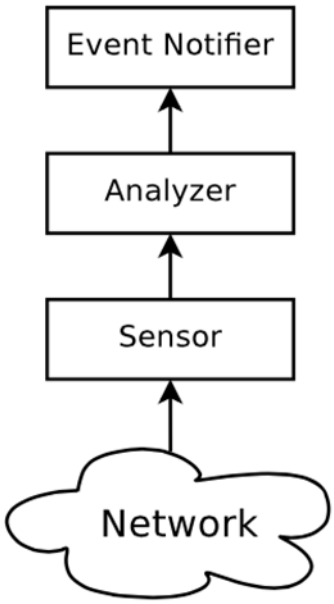


Figura 3: Etapas do funcionamento de um IDS (DI PIETRO e MANCINI, 2008).

O sensor, a primeira etapa do funcionamento de um IDS, representada na Figura 3, tem como função monitorar o tráfego na rede e extrair informações de todos os pacotes que trafegarem pela interface de modo que o sistema possa analisa-los. De acordo com Di Pietro e Mancini (2008), o sistema extrai e avalia as informações do cabeçalho do pacote e então compara com os critérios antes definidos.

Uma regra pode ser basicamente formada por duas partes: o cabeçalho, que contém a ação da regra, o protocolo e os endereços juntamente com as portas e máscaras de rede de origem e destino; e as opções da regra, que contém mensagens de alerta e informações sobre qual parte do pacote deve ser inspecionado para determinar se a regra deve tomar alguma ação ou não.

A regra citada na Figura 4 está configurada para gerar um alerta com a mensagem “PING DETECTADO” com o ID ”001” no caso de o pacote capturado for do tipo ICMP vindo de qualquer origem/porta em direção ao host “10.20.0.10/16” em qualquer porta. Após a análise do pacote capturado, se houver uma relação entre as informações coletadas e qualquer ponto (pelo menos um) e os critérios definidos, um alerta é gerado.

```
alert icmp any any -> 10.20.0.10/16 any  (msg:"PING DETECTADO";sid:001;)
```

Figura 4: Exemplo de uma regra do *Snort*.

Todas as regras ficam em arquivos de configuração específico, variando de acordo com o desenvolvedor, podendo possuir milhares de regras em um único arquivo. Normalmente existe um arquivo principal (*rules.conf*), onde este importa arquivos secundários (*virus.conf*, *downloaded.conf*, *webattacks.conf*, etc) para se obter uma melhor organização das especificações de cada regras. Estas configurações podem ser baixadas da internet de fontes gratuitas e proprietárias. A "*Emerging Threats*"[5], comunidade *Open-Source* que produz material científico sobre o surgimento de novas ameaças e pesquisas na área de segurança de redes, disponibiliza um arquivo para *download* gratuito no site oficial com aproximadamente 17.000 regras pré-configuradas, que são compatíveis com a maioria dos atuais sensores.

Por último, o alerta gerado pelo sensor recebe uma classificação de acordo com o nível de risco oferecido a rede, podendo ser de três tipos: baixo, médio e alto risco. Há também a possibilidade de o sensor gerar alertas falsos, chamados de falso positivos. Piper (2011) define falsos positivos quando o sensor gera um alerta de risco baseado em um falso alarme, normalmente causado por uma regra mal configurada ou uma anomalia mal definida. O autor salienta que um falso positivo não deve ser confundido com um ataque real que não obteve um resultado satisfatório contra o sistema. Como exemplo um ataque desenvolvido para a plataforma *Unix* ser executado em um computador com *Windows*. O evento irá registrar o ataque, mas o mesmo não oferecerá risco ao ambiente.

[5] Lista de regras para IDS/IPS *Open-Source* atualizada regularmente. Disponível em: http://www.emergingthreats.net/. Acesso: 4 Maio 2015.

4 SENSORES ANALISADOS

Existem diversos IDS/IPS *Open-Source* e proprietários no mercado. Dentre eles, o *Snort* e o *Suricata* são muito comparados em termos de configurações e eficiência. Ambos possuem uma grande quantidade de usuários e estão em constante desenvolvimento.

Segundo White, Fitzsimmons e Matthews (2013), qualquer empresa que se preocupa com segurança deve possuir um sistema de monitoramento de rede e o *Snort* é o sistema de detecção de intrusos mais utilizado no mercado. Os autores também afirmam, que o *Suricata* surge como uma alternativa a limitação do Snort em relação a utilização de apenas um núcleo do processador.

4.1 Snort

Snort é um IDS capaz de realizar análise de tráfego em tempo real e registro de pacotes em redes de computadores. Criado em 1998 por Martin Roesch é atualmente mantido pela Cisco. Basicamente, possui três principais funções: *sniffer* de pacotes; registrador de pacotes ou um completo sistema de prevenção de intrusos. A ferramenta pode ser obtida gratuitamente em versões para *Windows* ou *Linux* do site oficial sob uma licença *GNU GPL v.2* (*Open-Source*)[6].

Apesar de ser uma ferramenta poderosa, não há grandes requisitos de *hardware* para se obter um bom desempenho. Há versões para ambos processadores *x86* ou *x64* e suporta *IPv4* e *IPv6*, porém utiliza apenas um núcleo do processador.

Gaur (2001, p. 1) afirma que “a arquitetura do *Snort* é composta por três principais componentes.”, e os descreve da seguinte maneira:

1. *Packet Decoder* (Decodificador de Pacotes): tem como funções interpretar os pacotes capturados pelo sensor e preparar a informação coletada para que o mecanismo de detecção consiga realizar a análise.
2. *Detection Engine* (Mecanismo de Detecção): após a decodificação do pacote, as informações obtidas são passadas por uma série

[6] Disponível em: https://snort.org. Acesso em: 8 maio 2015.

de pré-processadores (mecanismo de detecção) que identificam algumas propriedades, estas que serão analisadas durante as regras configuradas. Este mecanismo separa as regras do *Snort* em duas partes: cabeçalho e opções.

A partir da Figura 5, é possível identificar que o cabeçalho da regra possui os detalhes de origem e destino do pacote (endereço IP e portas), ação a ser tomada (registrar o evento) e o tipo de protocolo. As opções da regra, por sua vez, definem o que será feito, argumentos, detalhes como TCP *flags*, tipos de códigos de ICMP, conteúdo específico para a regra, tamanho do *payload*, entre outros.

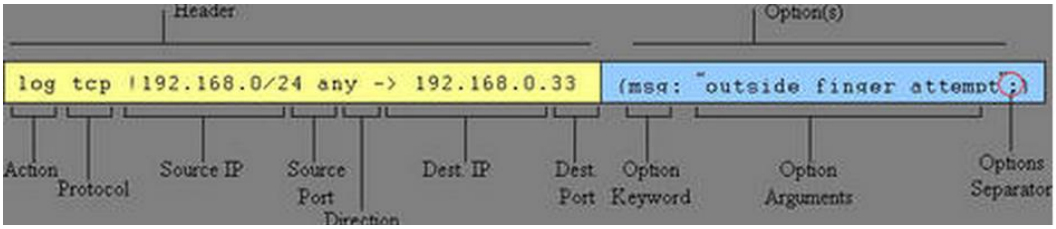


Figura 5: Componentes de uma regra (Gaur 2001).

Após a análise do pacote, o sensor o compara com as regras configuradas e em caso de correspondência, é executado o gatilho configurado nas opções da regra. Finalmente, um pacote que não combina com nenhuma regra, é simplesmente liberado pelo mecanismo de detecção.

3. *Logger/Alerter* (Arquivos de registros e Alertas): a informação coletada pelo decodificador de pacotes pode ser gravada em disco em diversas formas como texto ou no formato de *tcpdump* utilizando a ferramenta de log do sensor. Altamente flexível, seus *logs* podem ser exportados para praticamente qualquer tipo de banco de dados como *Oracle* e *MySQL*. Seus resultados também podem ser analisados por meio de ferramentas externas o que facilita sua customização. Já os alertas podem ser configurados para serem enviados por *e-mail* ou salvos em banco de dados ou arquivos de texto. Os alertas também podem ser opcionalmente desabilitados durante o uso do sensor em ambientes testes.

A Figura 6 representa os três principais componentes da arquitetura do *Snort*, onde o pacote é capturado, passa por uma decodificação de informações para que o mecanismo de detecção consiga realizar a análise e *logs* e alertas são gerados.

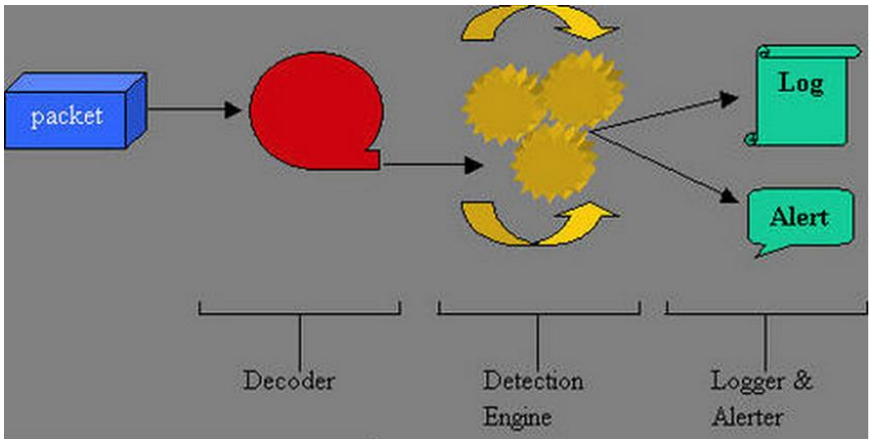


Figura 6: Arquitetura do *Snort* (Gaur, 2001).

4.2 Suricata

Assim como o *Snort*, o *Suricata* pode ser usado tanto como IDS ou como IPS. Foi lançado em janeiro de 2010 e é desenvolvido e mantido pela *Open Information Security Foundation* (OISF) e algumas empresas privadas. Também é disponibilizado no site oficial[7] para a maioria das plataformas, como *Linux* e *Windows*, sob licença *GPL v2* (*Open-Source*).

Apesar de ter bem menos tempo de mercado que o *Snort*, ambos os sistemas são muito comparados. O seu modo de funcionamento é exatamente igual ao do *Snort* e basicamente tem o mesmo sistema de regras. O *Suricata* utiliza um ou mais núcleos do processador ao mesmo tempo, o que é uma grande vantagem sobre o seu concorrente.

A partir da Figura 7, é possível notar que o *Suricata* possui a mesma estrutura de regras que o *Snort*, tendo um cabeçalho com informações de origem e destino do pacote, protocolo utilizado e ação a ser tomada e opções como conteúdo do *payload*, mensagem a ser exibida, etc. As regras criadas para o *Snort* geralmente são compatíveis com o *Suricata* sem a necessidade de alterações.

[7] Disponível em: http://suricata-ids.org/. Acesso em: 8 maio 2015.

rule header `alert tcp any any -> 192.168.1.0/24 111 \`
rule options `(content:"|00 01 86 a5|"; msg:"mountd access";)`

Figura 7: Estrutura de regras do Suricata (Svoboda, 2014).

Svoboda (2014) explica que diferente do *Snort* que suporta apenas protocolos da camada de rede, o Suricata ainda suporta diversos protocolos da camada de aplicação, como HTTP, FTP, TLS, SMB e DNS. O autor também salienta que a arquitetura do *Snort* difere em alguns pontos com a do *Suricata*, mas em geral são muito parecidas. O decodificador de pacotes e o mecanismo de detecção do *Suricata* são, ambos, divididos em outras partes. A Figura 8 demonstra a arquitetura do *Suricata* com algumas subdivisões que tem como objetivo permitir o processamento de padrões que originalmente não são suportados pelas regras.

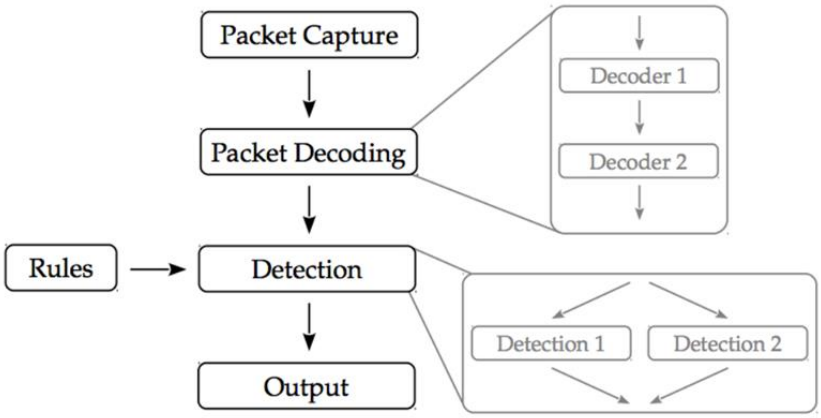


Figura 8: Arquitetura do *Suricata* (Svoboda, 2014).

O pacote é decodificado e então é gerado uma representação interna do pacote, permitindo que diferentes funções sejam chamadas uma-a-uma para análise do mesmo, estendendo assim, a capacidade de detecção. Após a decodificação, a detecção é possível seguindo as regras configuradas, com a diferença de que esse processo é subdividido em diferentes tarefas, podendo fazer a análise de vários pacotes ao mesmo tempo.

5 METODOLOGIA

Para o embasamento teórico realizou-se uma pesquisa bibliográfica, que segundo Gil (2008, p. 50) “é desenvolvida a partir de material já elaborado, constituído principalmente de livros e artigos científicos”, acerca de tecnologias utilizadas para a segurança da informação em ambientes corporativos. Também se utiliza da pesquisa documental, que Gil (2008, p. 51) afirma ser realizada a partir de “materiais que não receberam ainda um tratamento analítico”, tais como: documentos oficiais, relatórios de pesquisa, entre outros. A partir disso é possível estudar e compreender as questões ligadas aos sistemas de monitoramento em questão.

Após este entendimento, escolheu-se o *Snort* e o *Suricata* para que fosse realizado um estudo comparativo, a fim de ressaltar as diferenças e as similaridades entre os dois sistemas buscando o mais documentado em relação a instalação e configuração, o que apresenta a melhor eficiência na detecção de possíveis ataques e o melhor desempenho durante a realização dos testes. Para tal, as seguintes etapas, tornam-se fundamentais:

5.1 Montagem de um laboratório virtual e realização dos testes iniciais

A decisão de utilizar um laboratório virtual deu-se pelas facilidades que o mesmo possibilita, já que este permite retornar rapidamente o estado das máquinas através de *snapshots*, evitando a necessidade de reinstalação dos sistemas durante os testes realizados. Foram então criadas máquinas virtuais para que todos os sistemas fossem testados separadamente, buscando evitar qualquer anomalia nos resultados esperados.

Após a instalação e configurações básicas necessárias (*plug-ins* e atualização das regras) de ambos *Snort* e *Suricata*, a partir do computador com a distribuição *Kali Linux*, realizou-se alguns testes básicos como *ping* e escaneamento de portas, usando como alvo os próprios sensores (*Snort* e *Suricata*) a fim de certificar que os mesmos estavam identificando os possíveis ataques.

Posteriormente, foram realizados testes nos sistemas a partir de duas capturas de tráfego de rede em ambientes distintos: o primeiro, capturado na porta *SPAN* de

um *switch*, durante 24 horas em uma empresa com aproximadamente 50 funcionários; e o segundo, capturado durante a realização de um desafio de hackers em uma convenção anual realizada nos *EUA*, chamada *DEFCON*. Os arquivos *cap*[8] foram retransmitidos individualmente para o *Snort* e *Suricata* através da ferramenta *Tcpreplay*.

Para medir a utilização de CPU e memória dos computadores enquanto os testes foram realizados, utilizou-se o *Nagios*, configurado a partir de uma interface gráfica chamada *Centreon*. Para que o *Nagios* pudesse ter acesso as informações de utilização de *CPU* e memória dos computadores testados, foi necessário realizar a instalação do *SNMPD*.

5.2 Estabelecimento de um ambiente real e realização dos testes finais

Para a realização dos testes reais de configuração, análise de tráfego e medição de performance foram reservados computadores físicos para cada sistema com o intuito de evitar anomalias e obter o resultado mais apurado possível.

Assim, os seguintes computadores foram configurados:

PC 1 – Processador *Intel Core 2 Duo 3.0 GHz*, 4GB de memória RAM. Sistema *Ubuntu 12.04*. *Softwares* instalados: *Snort*.

PC 2 – Processador *Intel Core 2 Duo 3.0 GHz*, 4GB de memória RAM. Sistema *Ubuntu 12.04*. *Softwares* instalados: *Suricata*.

PC 3 – Processador *Intel Core 2 Duo 3.0 GHz*, 4GB de memória RAM. Sistema *Kali Linux*. *Softwares* instalados: *Tcpreplay*.

Máquina virtual – 1GB de memória RAM. Sistema Operacional *Centreon*. *Softwares* instalados: *Nagios* e *Centreon*.

O *Centreon*, instalado no computador 3, foi utilizado para configurar o *Nagios*, para que este monitore a utilização do processador e memória dos computadores 1 e 2 através do protocolo *SNMP*.

Primeiramente, foram realizados testes com o *Snort*. Para isso utilizou-se a distribuição *Ubuntu 12.04*, com o *Snort* instalado e apenas as regras *"Emerging Threa*ts" habilitadas, com o protocolo *SNMP* configurado para que o *Nagios* instalado na máquina 3 possa monitorar o desempenho, com um script perl de análise de *logs*.

[8] Pacote contendo tráfego de rede.

A partir da máquina 3 com a distribuição *Kali Linux*, foi replicado o tráfego armazenado nos pacotes com capturas de rede com a utilização da ferramenta *Tcpreplay*. Após o término de leitura dos pacotes, realizou-se a análise dos resultados obtidos pelo *Snort* e também dos gráficos de desempenho computados pelo *Nagios*.

Os mesmos testes foram realizados na máquina 2 com *Ubuntu 12.04* e o *Suricata*. Todos os procedimentos de instalação e configuração do *Ubuntu*, *Snort*, *Suricata*, *Nagios*, *Centreon* e *Kali Linux*, os comandos executados com as ferramentas *NMAP* e *Tcpreplay* podem ser encontrados no Apêndice II da presente pesquisa.

Para discutir o estudo foi elaborada uma análise minuciosa de todo o processo até alcançar os resultados finais. A partir disso, para facilitar o entendimento e compilar as informações foram elaboradas tabelas comparativas entre os sensores, *Snort* e *Suricata*, evidenciando suas diferenças e similaridades na identificação e configuração, desempenho e eficiência.

6 RESULTADOS E DISCUSSÕES

6.1 Montagem de laboratório virtual e realização dos testes iniciais

6.1.1 Montagem do laboratório virtual

A fim de realizar os testes iniciais foi instalado o *VirtualBox*[9] como *hypervisor*[10] em um *MacBook PRO* (computador portátil) com um processador *i7 2.2GHz* e 16GB de memória RAM. Neste computador foram criadas máquinas virtuais com configurações a partir dos modelos pré-estabelecidos para *Linux* 32 *bits*, com 10GB de disco rígido e 1GB de memória RAM. Separadamente foram instalados *Ubuntu 12.04*[11] e *Snort*; *Ubuntu 12.04* e *Suricata*; *Kali Linux*[12]; e *Centreon*[13], representadas na figura 9.

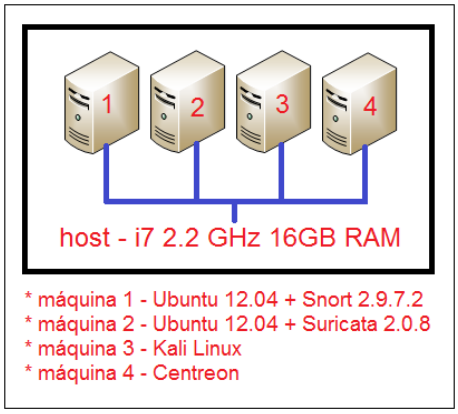


Figura 9: Representação do ambiente virtual de testes.

Nas máquinas virtuais 1 e 2 foram instalados o sistema operacional *Ubuntu*, versão 12.04 com basicamente todas as configurações padrões. A partir da instalação

[9] *VirtualBox* é um *software* de criação e gerenciamento de máquinas virtuais, podendo ser baixado gratuitamente. Disponível em: http://www.virtualbox.org. Acesso em: 11 maio 2015.
[10] *Software* necessário para a criação e configuração de máquinas virtuais.
[11] *Ubuntu* é uma versão de linux, *Open-Source*, baseado em *Debian*. Disponível em: http://www.ubuntu.com. Acesso em: 11 maio 2015.
[12] Sistema operacional *Open-Source*, com diversas ferramentas para testes de penetração. Disponível em: https://www.kali.org. Acesso em: 5 maio 2015.
[13] Sistema operacional *Open-Source*, com *Nagios* e *Centreon* já pré-configurados. Disponível em: https://www.centreon.com. Acesso em: 7 maio 2015.

limpa do *Ubuntu*, na máquina 1 foi instalado o *Snort* 2.9.7.2 juntamente com todos os pacotes requeridos pelo mesmo, e configurado para utilizar apenas as regras disponibilizadas pela "*Emerging Threats*".

Após isto, foram realizadas todos as atualizações disponíveis para o sistema operacional e para o *Snort*. Também foi instalado e configurado o programa *SNMPD* para compartilhar informações dos recursos da máquina através da rede local usando a comunidade "*public*" e versão "*2c*" do protocolo *SNMP* e o pacote *SAMBA*[14] para que o sistema pudesse ter acesso a arquivos da rede. Na máquina 2, foi instalado e atualizado o *Suricata* 2.0.8, seguido dos mesmos procedimentos com a configuração das regras, *SNMPD* e o *SAMBA*.

Na máquina 4 foi instalado o sistema operacional *Centreon* para que pudesse fazer a análise de performance das máquinas 1 e 2 durante a execução dos testes. Utilizando-se da plataforma do *Nagios*, porém com uma interface gráfica mais amigável, foram cadastrados os dois hosts e definidos dois plug-ins próprios do Centreon para o monitoramento do *CPU* e da memória RAM. Os plug-ins escolhidos foram: *check_centreon_snmp_mem* e *check_centreon_snmp_cpu*.

6.1.2 Testes iniciais

Os testes realizados foram os mesmos para os dois programas, *Snort* e *Suricata*, diferenciando apenas no nome dos diretórios e arquivos de configuração. Então, para fins de explicação dos testes executados, as instruções serão referenciadas ao IDS, já os resultados estarão devidamente identificados.

Primeiramente, o IDS foi configurado para não carregar nenhuma regra padrão (comentando todas as linhas que começam com "*include*" no arquivo de configuração) e foi adicionado apenas uma regra personalizada para a detecção de *ping* vinda de qualquer origem para qualquer destino.

A partir da máquina com o *Kali Linux*, foi executado o comando *ping* direcionado a máquina com o IDS instalado, podendo-se observar na Figura 10 a captura do pacote pelo IDS.

[14] *SAMBA* é um programa utilizado para interligar sistemas *UNIX* com ambientes *Windows*. Disponível em: https://www.samba.org/. Acessado em: 14 maio 2015.

RT	6	caio-snor...	2.105	2015-05-08 06:15:45	10.20.1.107	10.20.1.108	1	GPL ICMP_INFO PING BSDtype
RT	6	caio-snor...	2.106	2015-05-08 06:15:45	10.20.1.107	10.20.1.108	1	GPL ICMP_INFO PING *NIX

Figura 10: Alerta gerado pelo IDS no reconhecimento de um *ping*.

A Figura 10 representa a visualização a partir do programa *Sguil*[15], do alerta gerado no IDS após identificar um pacote *ICMP* com origem do *host* "10.20.1.107" com destino ao *host* "10.20.1.108". Nota-se que em se tratando de um protocolo *ICMP* (camada 3) não há portas (camada 4) no alerta. Um segundo teste foi realizado a partir do uso da ferramenta *NMAP*, executado do *Kali Linux*, tendo como alvo o IDS. Uma regra personalizada foi criada para que o sensor pudesse capturar o escaneamento de portas.

Diferente da Figura 10, é possível observar o registro das portas no resultado de um escaneamento de portas representado na Figura 11. Tendo certeza que os sensores estão devidamente configurados e que o *Nagios* está devidamente monitorando os computadores com os sensores instalados, pode-se realizar os testes reais tratados no capítulo 6.2.

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	2	caio-snor...	2.12634	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	3306	6	ET POLICY Suspicious inbound to mySQL port 3306
RT	1	caio-snor...	2.12636	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	5904	6	ET SCAN Potential VNC Scan 5900-5920
RT	2	caio-snor...	2.12637	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	1521	6	ET POLICY Suspicious inbound to Oracle SQL port 1521
RT	2	caio-snor...	2.12639	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	1433	6	ET POLICY Suspicious inbound to MSSQL port 1433
RT	1	caio-snor...	2.12641	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	5800	6	ET SCAN Potential VNC Scan 5800-5820
RT	2	caio-snor...	2.12642	2015-05-08 07:39:00	10.20.1.107	40058	10.20.1.110	5432	6	ET POLICY Suspicious inbound to PostgreSQL port 5432

Figura 11: Alerta gerado pelo IDS no reconhecimento de um *scaneamento* de portas.

6.2 Estabelecimento de ambiente real e realização dos testes finais

6.2.1 Montagem do laboratório para a realização da análise

Para a montagem do laboratório onde os testes deste estudo serão realizados, usou-se três máquinas reais, todas com o mesmo *hardware* para que não haja diferença na medição de desempenho, com a seguinte configuração:

- Processador *Intel Core 2 Duo 3.0 GHz* (*dual core*);
- 4GB de memória RAM;

[15] Programa de análise de *logs* para *Snort* e *Suricata* incluso pré-instalado na distribuição *Security Onion*.

- Interface de rede 10/100/1000.

Os seguintes sistemas operacionais e programas foram instalados nos computadores:

a) Máquina 1: *Ubuntu 12.04*, *Snort* 2.9.7.2, *Samba*, *SNMPD*;
b) Máquina 2: *Ubuntu 12.04*, *Suricata* 2.0.8, *Samba*, *SNMPD*;
c) Máquina 3: *Ubuntu 12.04*, Tcpreplay;
d) Máquina 4: *Centreon* e *Nagios*.

Todos os computadores foram ligados a um switch gerenciável, *gigabit*, *PoE*, modelo *DLINK DGS-1210-10P*, respectivamente nas portas 1, 2, 3 e 4. A cada teste realizado, o *switch* foi configurado para espelhar o tráfego da máquina com a ferramenta *Tcpreplay* para o computador com o sensor em teste. Por exemplo, nos testes realizados no *Snort*, durante a replicação do tráfego pela máquina 3, o *switch* foi configurado para espelhar todo o tráfego (Rx/Tx) na porta 1.

6.2.2 Conjuntos com captura de tráfego de rede (arquivos *cap*)

O objetivo do estudo é identificar qual dos dois sensores, a partir de configurações de fábrica (não personalizadas), identificam mais pacotes com possíveis ataques e com um melhor desempenho. Contudo, não há a necessidade de saber exatamente que tipos de ataques estão inclusos nos tráfegos capturados pois serão utilizados as mesmas capturas para ambos os sensores. Essa pesquisa também não se baseia em registros falso-positivos, e sim apenas na quantidade de registros gerados e classificação dos mesmos por ambos os sistemas.

Outro ponto importante é em relação ao tamanho de cada pacote contido nas capturas. De acordo com a configuração padrão das interfaces de rede, são aceitos pacotes com até 1.500 *bytes* de tamanho, e como o estudo utiliza apenas configurações padrões, todos os pacotes maiores que 1.500 *bytes* serão descartados pelas ferramentas. Com essas considerações, para a realização dos testes deste trabalho foram utilizados dois diferentes pacotes com capturas de tráfego de rede:

a) Conjunto 1: registrado a partir de uma competição de "*Capture The Flag*"[16] realizado na *DEFCON* em 2014:

A captura foi realizada pelo time vencedor da edição e possui apenas 3/10 do tráfego capturado durante a competição devido a limitações de espaço. Apenas esta porção do tráfego representa 16GB de espaço em disco e um total de 27.497.063 pacotes. Qualquer divergência de contabilidade em até 0.1% (27.497 pacotes) será desconsiderada.

Para este presente estudo apenas os protocolos mais comuns serão considerados. É possível observar a partir da Figura 12, que de um total de 27.497.063 pacotes contidos na captura do tráfego, 24.646.425 pacotes são do tipo TCP, 2.848.916 são UDP e 1.722 são ICMP, representando respectivamente 89,63%, 10,36% e 0,01%.

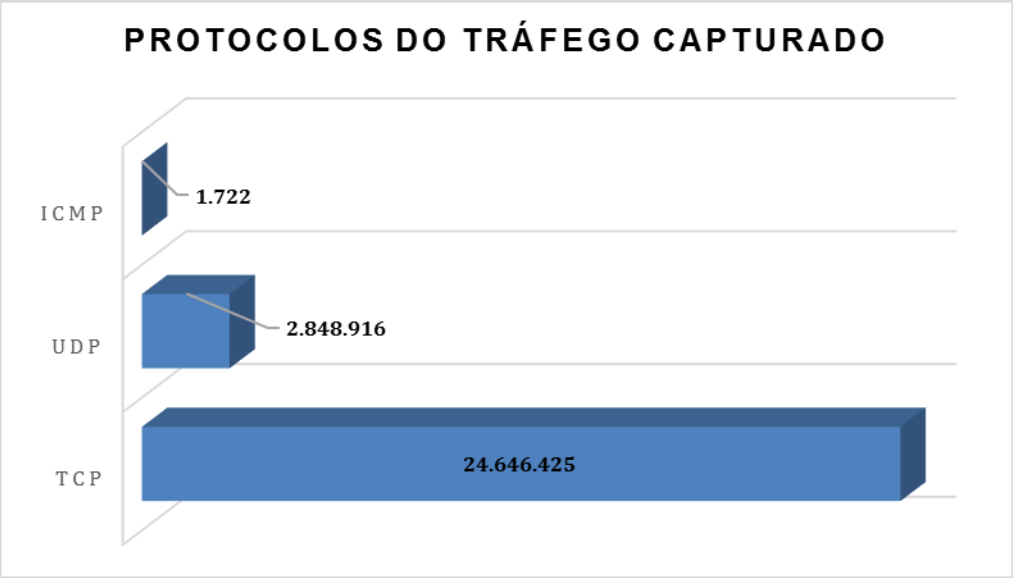


Figura 12: Divisão de pacotes por protocolo do tráfego capturado na *DEFCON*.

A Figura 13 representa o tamanho dos pacotes capturados e percebe-se que pacotes maiores que 2.559 *bytes* foram omitidos do gráfico, por não serem utilizados neste estudo, descritos a seguir:

- 0-19 *bytes*: 0 pacotes;
- 20-39 *bytes*: 0 pacotes;

[16] Competição entre *hackers* onde o objetivo é atacar e defender uma série de computadores e/ou redes.

- 40-79 *bytes*: 15.492.753 pacotes – 56,34%;
- 80-159 *bytes*: 5.865.237 pacotes – 21,33%;
- 160-319 *bytes*: 3.453.120 pacotes – 14,09%;
- 320-639 *bytes*: 301.015 pacotes – 1,09%;
- 640-1279 *bytes*: 2.114.484 pacotes – 7,68%;
- 1280-2559 *bytes*: 359.432 pacotes – 1,3%.

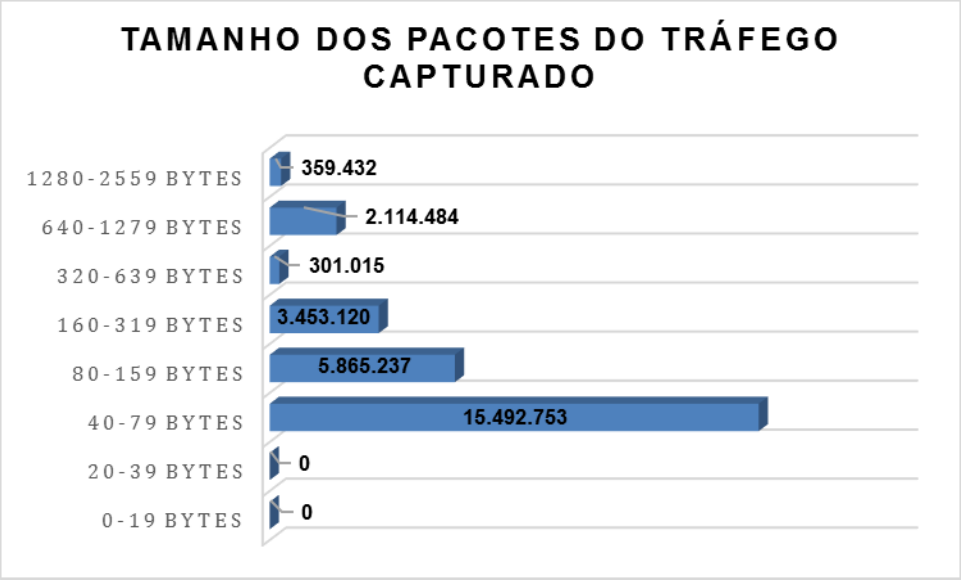


Figura 13: Tamanho dos contidos no tráfego capturado na *DEFCON*.

b) Conjunto 2: registrado a partir de uma rede corporativa:

A captura foi realizada durante 24 horas a partir de uma ferramenta chamada *Tshark*, instalada em um *Ubuntu 12.04*, este conectado a uma porta *SPAN* em um *switch* gerenciável *DLINK DGS-1210-10P*. O tráfego capturado resultou em aproximadamente 35 milhões de pacotes, necessitando ser removidos aproximadamente 12 milhões de pacotes por motivos de espaço em disco e tempo de processamento. O arquivo final possui 23.644.015 pacotes e será desconsiderado qualquer divergência de contabilidade em até 0.1% (23.644 pacotes).

Para este presente estudo apenas os protocolos mais comuns (TCP, UDP e ICMP) serão considerados. É possível observar a partir da Figura 14, que de um total de 27.000.000 pacotes contidos na captura do tráfego, 22.940.995 pacotes são do

tipo TCP, 612.494 são UDP e 90.526 são ICMP, representando respectivamente 97,02%, 2,59% e 0,38%.

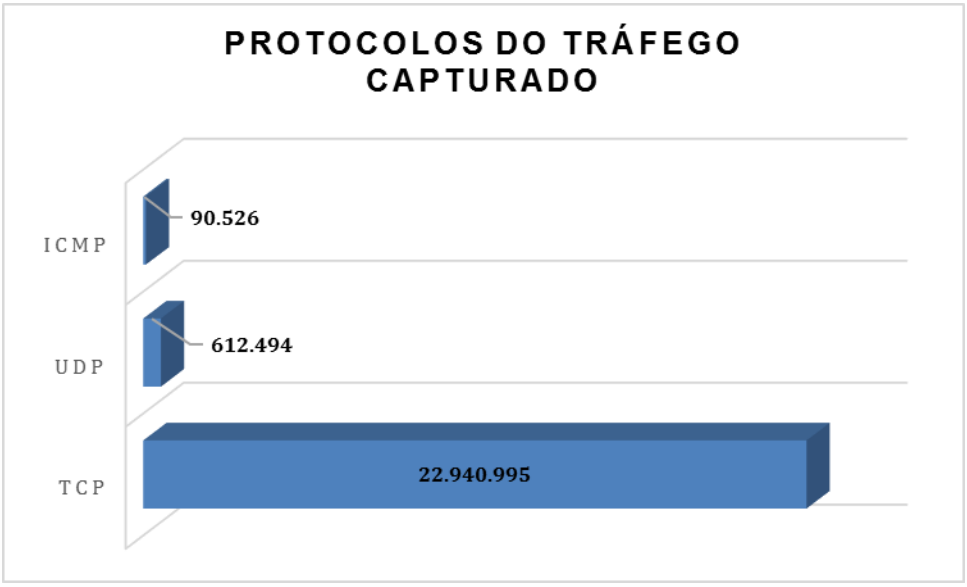


Figura 14: Divisão de pacotes por protocolo do tráfego capturado em um ambiente corporativo.

A Figura 15 representa o tamanho dos pacotes capturados e percebe-se que pacotes maiores que 2.559 *bytes* foram omitidos do gráfico, por não serem utilizados neste estudo, descritos a seguir:

- 0-19 *bytes*: 0 pacotes;
- 20-39 *bytes*: 0 pacotes;
- 40-79 *bytes*: 8.753.624 pacotes – 37,02%;
- 80-159 *bytes*: 1.891.586 pacotes – 8%;
- 160-319 *bytes*: 442.560 pacotes – 1,87%;
- 320-639 *bytes*: 755.712 pacotes – 3,19%;
- 640-1279 *bytes*: 11.686.455 pacotes – 49,42%;
- 1280-2559 *bytes*: 469.828 pacotes – 1,98%.

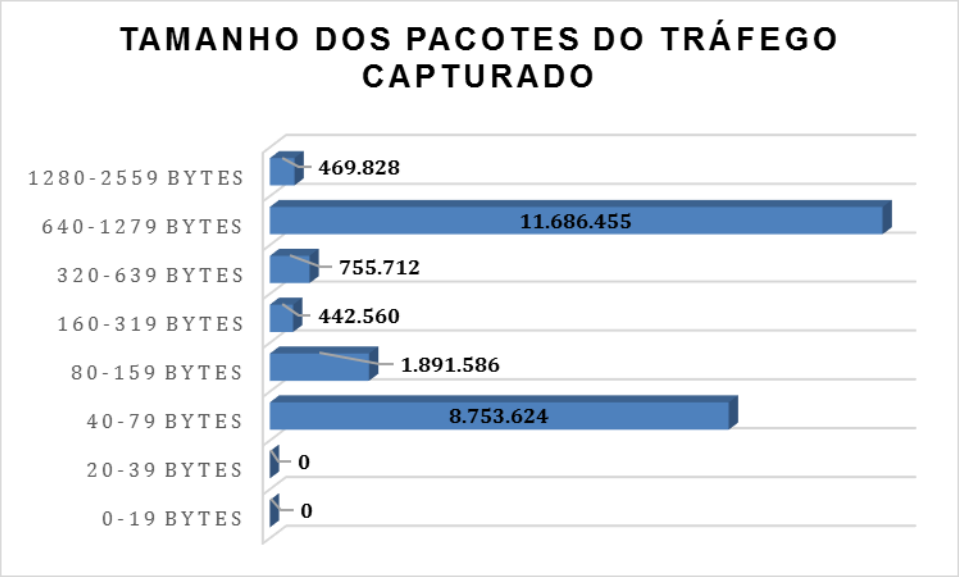


Figura 15: Divisão de pacotes por protocolo do tráfego capturado no ambiente corporativo.

Para cada pacote apresentado foram realizados dois testes com diferentes velocidades durante a reprodução do tráfego. No primeiro teste foi simulado uma rede 10/100 com a taxa de transmissão variando entre 40 e 50 MBit/s. Já no segundo teste foi representado uma rede 10/1000/1000 com a taxa de transmissão alterando entre 600 e 900 MBit/s.

Os sensores deverão receber mais pacotes do que a ferramenta *Tcpreplay* reproduzir pois os testes serão realizados em um laboratório real dentro de um ambiente corporativo. A seguir serão apresentados os resultados obtidos nos testes com os sensores, *Snort* e *Suricata*.

6.2.3 Testes com *Snort*

Para a realização dos testes com o *Snort*: primeiramente, a máquina 1, no qual o sensor estava instalado, foi conectada a porta *SPAN* do *switch* gerenciável. A máquina 3, com a ferramenta *Tcpreplay*, foi ligada a uma outra porta qualquer do mesmo *switch*. Esse processo está representado pela Figura 16, abaixo.

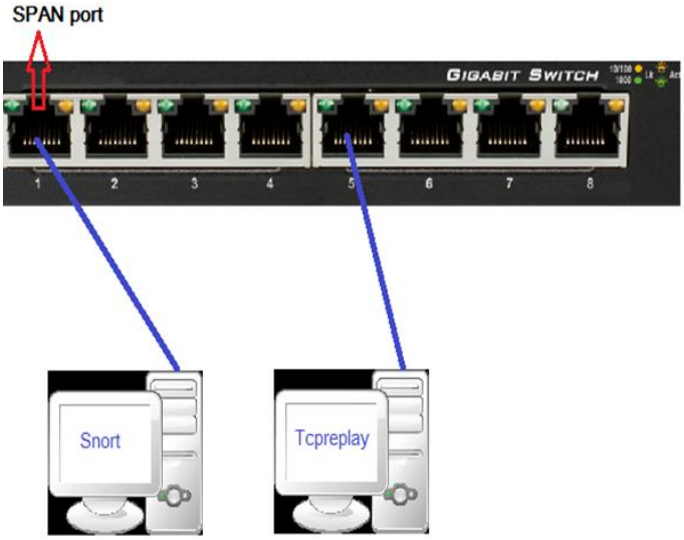


Figura 16: *Snort* conectado a porta *SPAN* do *switch*.

Após a configuração do ambiente representado pela Figura 16, o *Snort* foi habilitado para realizar o monitoramento da rede com um total de 15.714 regras habilitadas e, a partir da ferramenta *Tcpreplay*, foram realizadas as seguintes simulações:

O primeiro teste foi realizado com a reprodução do Pacote 1 (*DEFCON*) a uma taxa de 40 a 50 MBit/s (rede 10/100) durante 31 minutos. A partir disso, foi possível observar que o *Snort* recebeu um total de 28.115.820 pacotes e todos foram processados, não havendo descartes. Durante o monitoramento, 11.663 alertas foram criados e destes, 0,57% foram de alto risco (66 alertas), 15,04% de baixo risco (1.754 alertas) e 84,40% de médio risco (9.843 alertas), é o que representa a Figura 17.

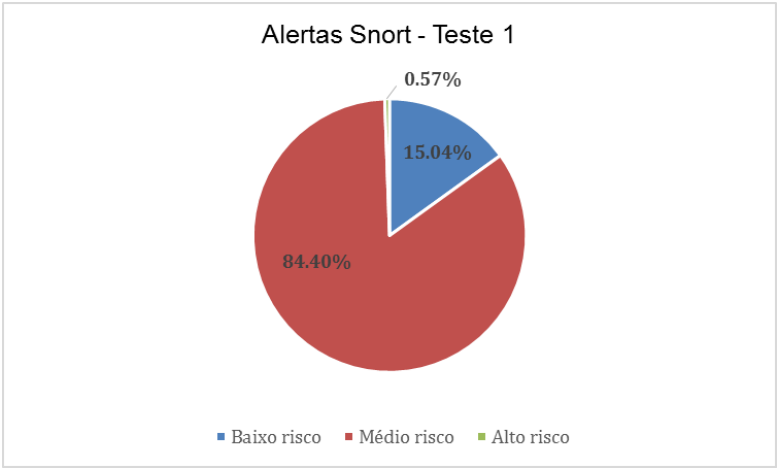


Figura 17: Categoria dos alertas gerados pelo *Snort* após a reprodução do Pacote 1 a 100 MBit/s.

No segundo teste, o Pacote 1 foi reproduzido a uma velocidade de 600 a 900 MBit/s (rede 10/100/1000) durante 3 minutos. Com esta configuração foi possível representar uma rede gigabit com tráfego intenso, resultando em uma perda considerável de pacotes. De um total de 29.647.027 pacotes, 8.506.735 destes foram analisados (28,70%), 21.140.291 foram descartados (41,62%) e 2.918 alertas foram gerados. Do total, 0,62% dos pacotes são de alto risco (18 alertas), 6,68% foram de baixo risco (195 alertas) e 92,7% de médio risco (2.705 alertas), conforme representado na Figura 18.

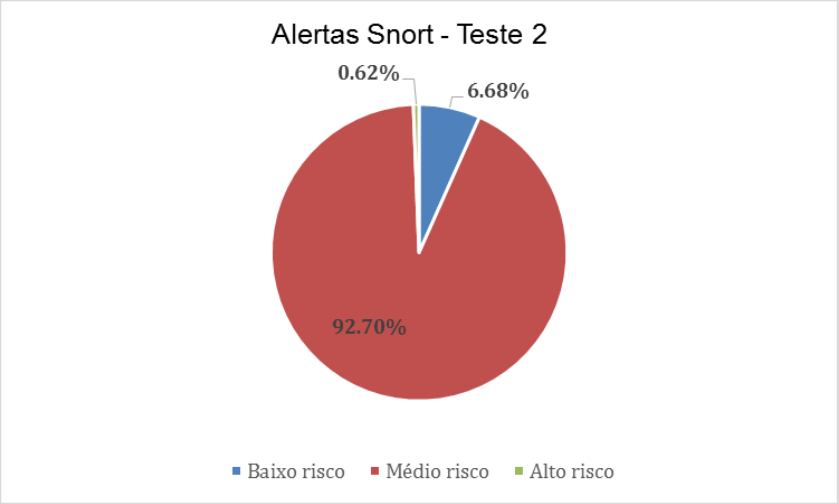


Figura 18: Categoria dos alertas gerados pelo *Snort* após a reprodução do Pacote 1 a 1000 MBit/s.

O terceiro teste apresenta o primeiro experimento com o Pacote 2 (capturado em um ambiente corporativo). Reproduzido a uma taxa de transmissão entre 40 e 50 MBit/s (rede 10/100) durante 32 minutos e 26 segundos, novamente o *Snort* conseguiu realizar o processamento de todos os 24.607.820 pacotes enviados e foram gerados 2.828 alertas. Dos alertas, 15,62% foram de alto risco (441 alertas), 59,15% foram de baixo risco (1673 alertas), e 25,21% foram de médio risco (713 alertas) representado na Figura 19.

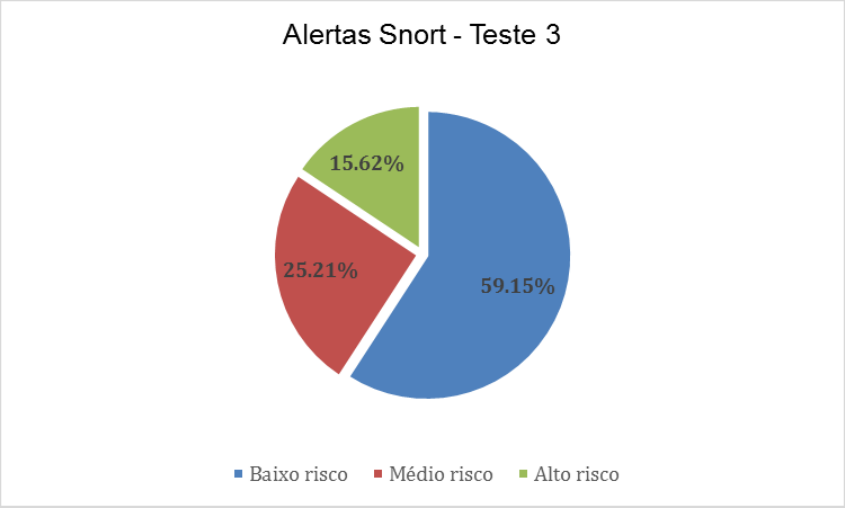


Figura 19: Categoria dos alertas gerados pelo *Snort* após a reprodução do Pacote 2 a 100 MBit/s.

No último teste realizado com o *Snort*, foi reproduzido o Pacote 2 a uma taxa de transmissão de 600 a 900 MBit/s (rede 10/100/1000) durante 1 minuto e 20 segundos. Como resultado o *Snort* foi capaz de processar apenas 38.490 pacotes, equivalente a 0.15% do total de pacotes enviados. Durante a reprodução de tráfego 501 alertas foram gerados e destes, 74 foram de alto risco (14,77%), 244 foram de baixo risco (48,70%) e 183 foram de médio risco (36,53%), ilustrado na Figura 20.

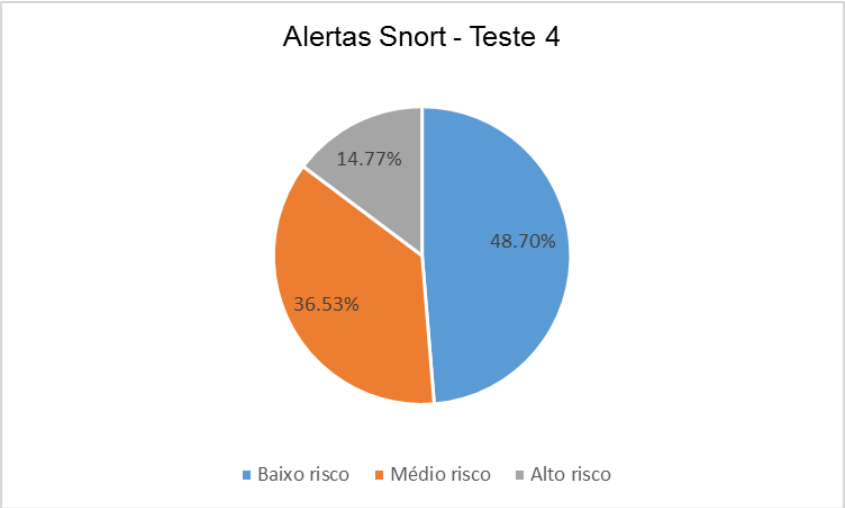


Figura 20: Categoria dos alertas gerados pelo *Snort* após a reprodução do Pacote 2 a 1000 MBit/s.

Após a realização de todos os testes com o *Snort*, todos os registros gerados pelo sensor foram reunidos e estão apresentados na Tabela 1.

Número de Pacotes	Nome/Descrição da regra ativada
17618	ET INFO UPnP Discovery Search Response vulnerable UPnP device
8790	GPL MISC UPnP malformed advertisement
1646	GPL ICMP_INFO PING BSD
429	GPL IMAP delete overflow attempt
208	GPL SNMP public access udp
178	ET POLICY FTP Login Successful
127	ET SCAN Rapid IMAP Connections - Possible Brute Force Attack
106	ET SCAN Rapid IMAP Connections - Possible Brute Force Attack
92	GPL MISC UPnP malformed advertisement
90	ET POLICY Suspicious inbound to mySQL port 3306
79	ET POLICY Suspicious inbound to PostgreSQL port 5432
56	ET SCAN Potential VNC Scan 5900-5920
49	ET SCAN Potential VNC Scan 5800-5820
45	ET POLICY Suspicious inbound to MSSQL port 1433
38	ET SCAN Potential SSH Scan
36	ET POLICY Dropbox DNS Lookup - Possible Offsite File Backup in Use
31	ET POLICY DNS Update From External net
25	GPL ICMP_INFO PING *NIX
22	ET ATTACK_RESPONSE Output of id command from HTTP server
19	ET POLICY Dropbox Client Broadcasting
17	ET POLICY Executable and linking format (ELF) file download
17	ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management
14	ET INFO HTTP Request to a *.tc domain
14	GPL WEB_SERVER 403 Forbidden
12	ET POLICY Suspicious inbound to mSQL port 4333
12	GPL NETBIOS name query overflow attempt UDP
11	ET POLICY Dropbox.com Offsite File Backup in Use
9	ET INFO SUSPICIOUS SMTP EXE - ZIP file with .exe filename inside (Inbound)
7	GPL WEB_SERVER python access attempt
6	ET POLICY Suspicious inbound to Oracle SQL port 1521
5	ET INFO DYNAMIC_DNS Query to *.dyndns. Domain
5	ET SCAN Potential SSH Scan OUTBOUND
4	ET CHAT Skype User-Agent detected
3	ET POLICY iTunes User Agent
3	ET POLICY Possible External IP Lookup ipinfo.io
3	ET SCAN Behavioral Unusually fast Terminal Server Traffic
3	ET TROJAN Win32.Chroject.B Retrieving encoded payload
2	ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second DoS Attempt
2	ET MALWARE Mozilla User-Agent (Mozilla/5.0) Inbound Likely Fake
2	ET POLICY Data POST to an image file (gif)

2	ET POLICY PE EXE or DLL Windows file download HTTP
2	ET POLICY RDP connection confirm
2	ET POLICY RDP connection request
2	ET SCAN Behavioral Unusual Port 135 traffic
2	ET SCAN NMAP SIP Version Detect OPTIONS Scan
2	ET WEB_SERVER Possible HTTP 404 XSS Attempt (Local Source)
2	GPL FTP large PWD command
1	ET CURRENT_EVENTS Possible TLS HeartBleed Unencrypted Request Method 3 (Inbound to Common SSL Port)
1	ET NETBIOS Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference
1	ET P2P Edonkey Publicize File
1	ET POLICY Outdated Windows Flash Version IE
1	GPL NETBIOS SMB IPC$ unicode share access

Tabela 1: Relação de todos os registros gerado pelo Snort.

.

6.2.4 Testes com Suricata

Os testes realizados com o *Suricata* seguem o mesmo padrão já apresentados. O computador 2, com o *Suricata* instalado, foi conectado a porta *SPAN* do *switch* e a máquina 2, com o *Tcpreplay* foi conectado a uma outra porta do mesmo *switch*. Esse processo está representado pela Figura 21, abaixo.

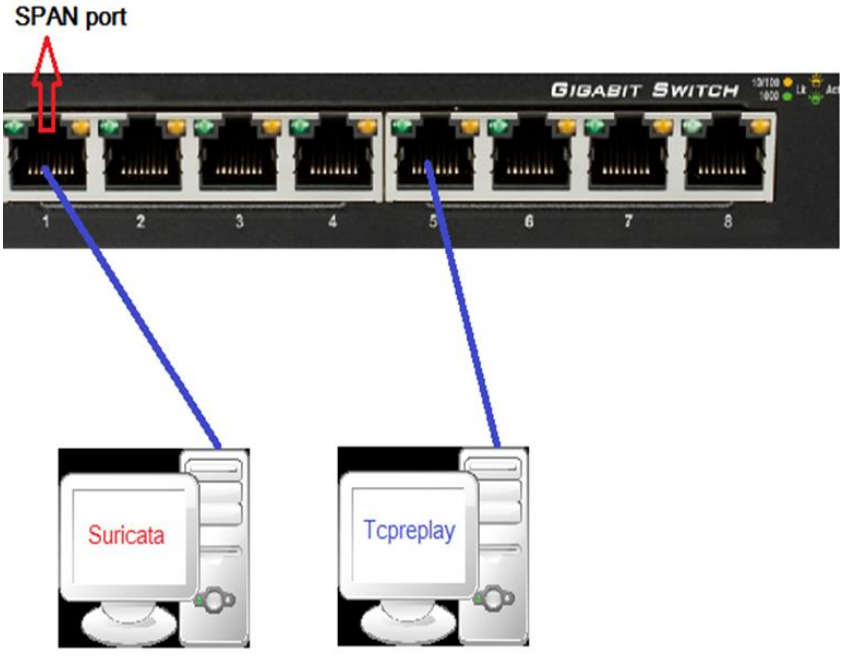


Figura 21: *Suricata* conectado a porta *SPAN* do *switch*.

Após a configuração do ambiente representado pela Figura 20, o *Suricata* foi habilitado para realizar o monitoramento da rede com um total de 15.735 regras habilitadas e, a partir da ferramenta *Tcpreplay*, foram realizadas as seguintes simulações:

Durante o primeiro teste realizado com o *Suricata*, foi reproduzido na rede o pacote 1 (*DEFCON*) a uma taxa de transmissão de 40 a 50 MBit/s (simulando uma rede 10/100) durante 30 minutos e 15 segundos. Após o término da reprodução do tráfego de rede, foi analisado o log do Suricata e constatado que o mesmo analisou 27.774.261 pacotes (99,77%) de 27.837810 pacotes enviados, totalizando um descarte de 63.549 pacotes (0,23%). Destes pacotes, 2.944 alertas foram gerados, especificamente 19 alertas de alto risco (1%), 2.823 alertas de baixo risco (96%) e 102 alertas de médio risco (3%), representado na Figura 22.

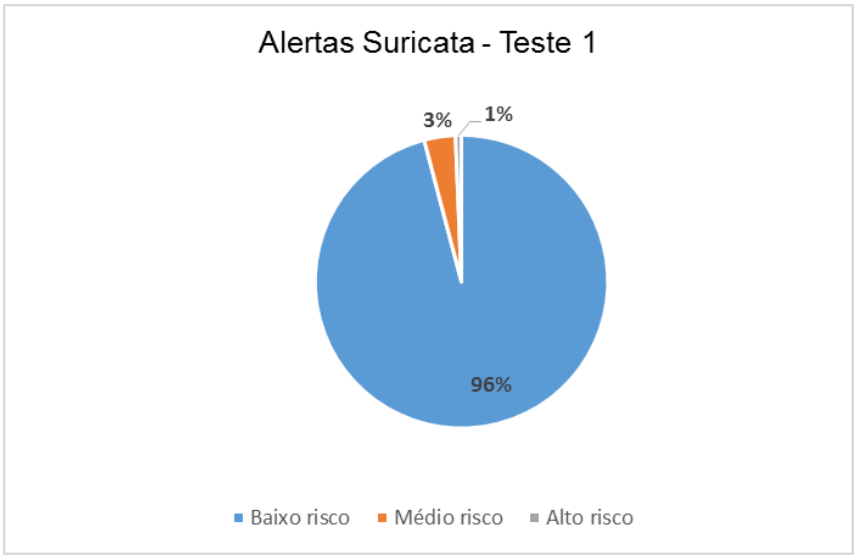


Figura 22: Categoria dos alertas gerados pelo *Suricata* após a reprodução do Pacote 1 a 100 MBit/s.

Em um segundo teste, foi utilizado para transmissão de tráfego o mesmo pacote 1, porém a uma velocidade de 600 a 900 MBit/s durante 2 minutos e 30 segundos. O *Suricata* analisou apenas 27% do tráfego enviado (5.057.667 pacotes), representando 22.828.431 pacotes descartados, 83% da totalidade de pacotes enviados. Destes pacotes capturados 1.013 alertas foram gerados. Dos alertas, 12 foram de alto risco (1,18%), 960 foram de baixo risco (94,77%) e 41 foram de médio risco (4,05%), representado na Figura 23.

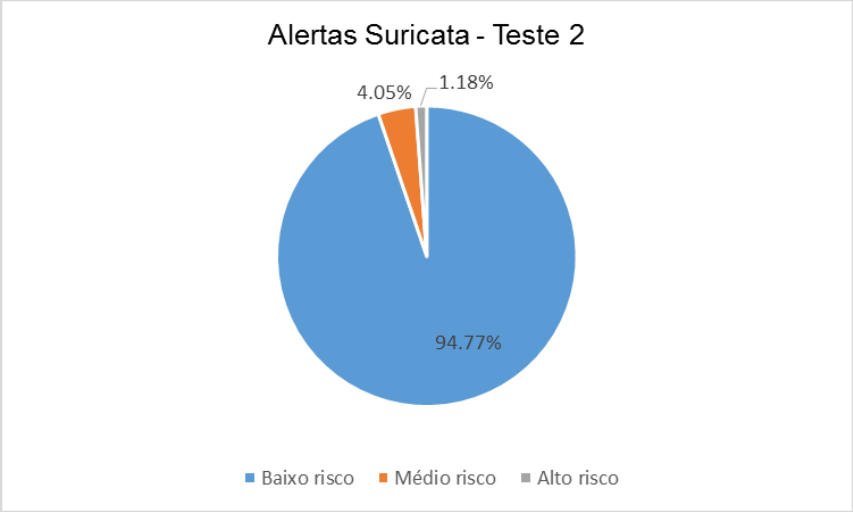


Figura 23: Categoria dos alertas gerados pelo *Suricata* após a reprodução do Pacote 1 a 1000 MBit/s.

No terceiro teste, o *Suricata* foi utilizado para monitorar a reprodução do tráfego do pacote 2 (rede corporativa) a uma velocidade de transmissão entre 40 e 50 MBit/s durante 25 minutos e 10 segundos. O *Suricata* foi capaz de realizar a análise de aproximadamente 99.99% (23.999.589) dos pacotes recebidos 24.000.000, tendo descartado apenas 411 pacotes (menos de 0.01%), e gerado 11.108 alertas. A Figura 24 representa que, destes alertas, 8 foram de alto risco (0.07%), 11.080 de baixo risco (99.75%) e 20 de médio risco (0.18%).

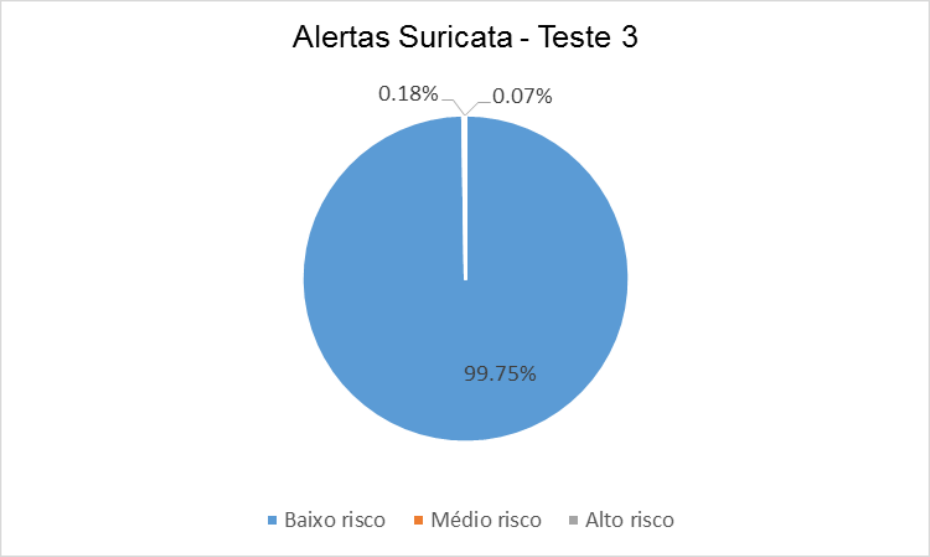


Figura 24: Categoria dos alertas gerados pelo *Suricata* após a reprodução do Pacote 2 a 100 MBit/s.

O último teste realizado com o *Suricata* reproduziu o pacote 2 a uma taxa de transmissão entre 600 e 900 MBit/s durante 1 minuto e 25 segundos. O *Suricata* foi capaz de analisar apenas 32.932 pacotes (0,13% do total de pacotes) e gerou 9.731 alertas, descartando 23.967.068 pacotes (99.87%). Dos alertas, 6 foram de alto risco (0,06%), 9.718 foram de baixo risco (99,87%) e 7 de médio risco (0,07%), representado na Figura 25.

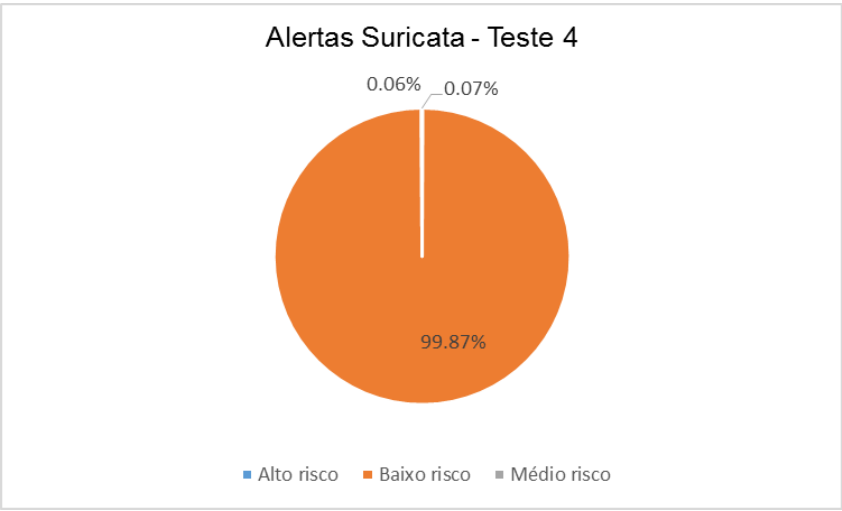


Figura 25: Categoria dos alertas gerados pelo *Suricata* após a reprodução do Pacote 2 a 1000 MBit/s.

Aproximadamente 90% dos registros gerados pelo *Suricata* são do tipo *Upnp discover*, que nada mais são que solicitações vindas de diversos tipos de *hardware* em uma rede de computadores tentando identificar outros componentes. A Figura 26 apresenta a maioria dos registros gerados pelo *Suricata* durante os 4 testes, e a fim de melhor visualização, foram retirados desta figura os registros de *Upnp discovery*.

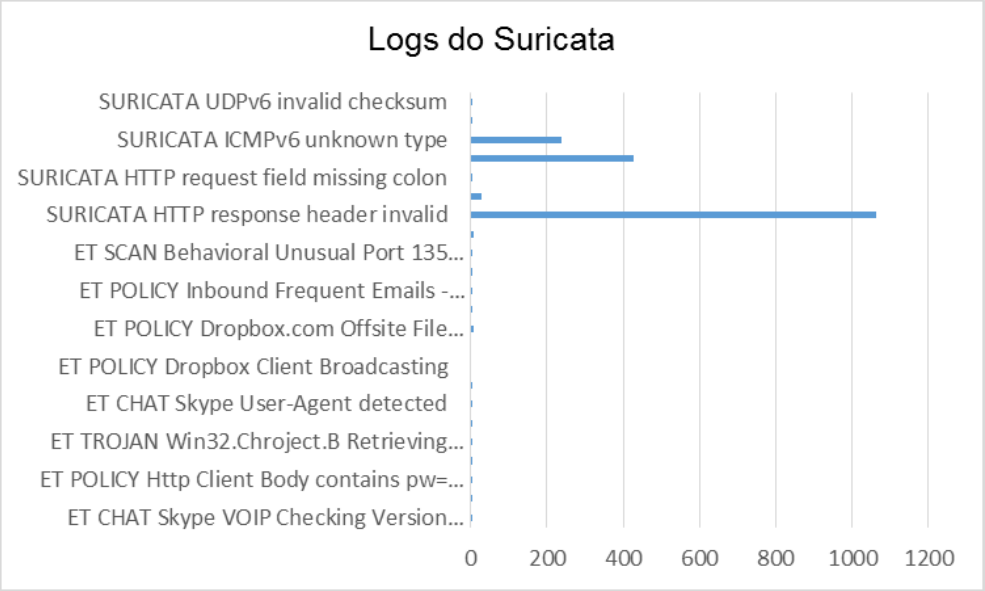


Figura 26: Logs gerados pelo *Suricata* durante os 4 testes.

6.2.5 Desempenho durante os testes

Durante a realização de cada um de todos os testes com o *Snort* e o Suricata foi monitorado toda a utilização de processamento da máquina com o sensor instalado (não do processo em si) e da utilização de memória RAM.

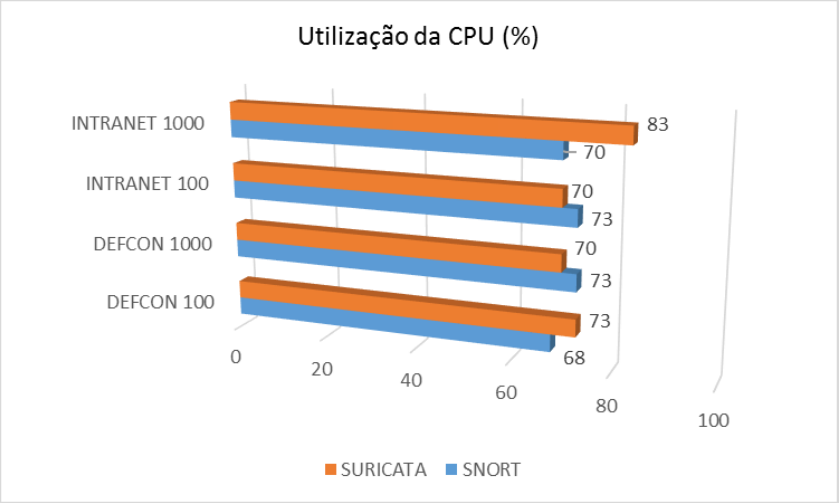


Figura 27: Utilização da CPU do computador com os sensores instalados, durante os testes.

A Figura 27 representa a utilização total da CPU das máquinas com os sensores instalados durante cada teste realizado. É possível verificar que o *Suricata* possui um desempenho ligeiramente melhor do que o *Snort*. Isso se dá pelo fato do *Suricata* utilizar mais de um núcleo do processador durante o monitoramento, porém, o ganho de desempenho não chega a ser justificável para afirmar que o mesmo é mais eficiente que o *Snort*.

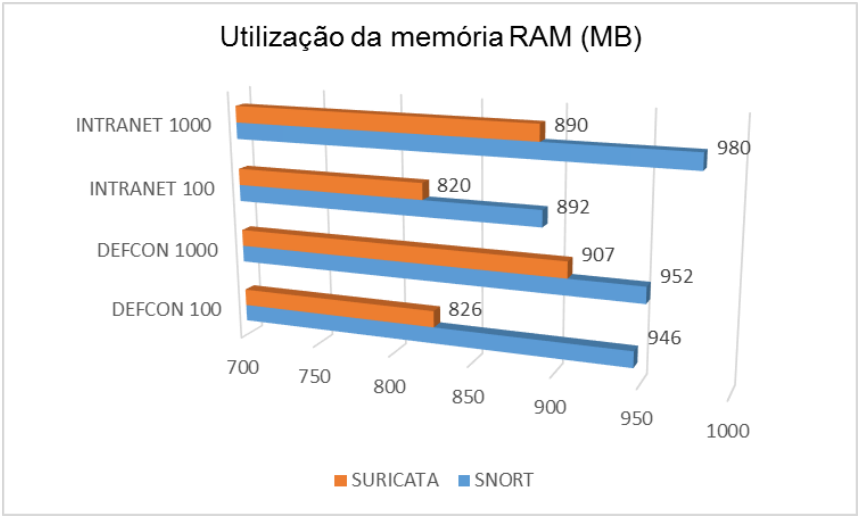


Figura 28: Utilização da memória RAM do computador com os sensores instalados, durante os testes.

A partir da Figura 28 é possível notar que a utilização da memória RAM dos computadores com os sensores instalados durante a execução dos testes é

praticamente irrelevante. Considerando que 1GB de memória RAM não é uma quantidade a ser considerada na configuração de um servidor, a utilização de 980MB de RAM pelo *Snort* no monitoramento do pacote 2 a uma taxa de transmissão de até 900 MBit/s, é totalmente aceitável. É possível notar que a diferença de utilização entre os dois sensores é de praticamente 10% na maioria dos casos, o que, novamente, não é o suficiente para afirmar que o *Suricata* possui um melhor desempenho.

6.3 Snort x *Suricata*

Para facilitar a análise das similaridades e diferenças entre os sensores *Snort* e *Suricata*, foi elaborada a Tabela 2 de uma forma comparativa.

	Snort	**Suricata**
Documentação	A página oficial do sensor é atualizada frequentemente e possui informações suficientes para que qualquer iniciante na área comece seus estudos.	A própria página oficial do desenvolvedor possui muita informação confusa e dificilmente é encontrado material científico a respeito do sensor.
Versões	Principalmente para sistemas *Unix*, mas também possui versões para *Windows* e *Mac*.	*Unix* e *Windows*.
Instalação	A partir da compilação do código fonte, *ppa* para *Ubuntu* ou executável para Windows.	A partir do código fonte, executável para *Windows* e distribuído nativamente junto com o *Ubuntu*.
Recursos	Single-thread, diferentes arquivos de regras disponibilizados gratuitamente e também para colaboradores através de cadastro, suporte a Ipv6	*Multi-thread*, apenas duas fontes de arquivos com regras disponibilizados nativamente mas possui grande compatibilidade com outros IDS, suporte a Ipv6 nativo.

	somente se for compilado do código fonte.	
Eficiência na detecção de possíveis ataques durante os testes a partir de configurações padrão da instalação	O *Snort* demonstrou um alto nível de detecção, provavelmente muitos falsos positivos, mas a partir destes registros é possível realizar uma análise e promover melhorias ao sensor.	O *Suricata* praticamente não foi capaz de detectar tráfego malicioso, o que pode dificultar o trabalho do administrador da rede durante a sua implementação.
Desempenho durante os testes realizados	O computador com o *Snort* instalado teve uma média de utilização do processador de 80% e 900MB de utilização da memória RAM.	O computador com o *Suricata* instalado teve uma média de utilização do processador de 70% e 800MB de utilização da memória RAM.
Resumo	O *Snort* é um sensor confiável, poderoso e robusto, muito bem documentado e que *out-of-box*[17] já consegue demonstrar boa parte de sua excelente capacidade de detecção de intrusos.	A falta de material didático dificulta a exploração do software que tem como objetivo ser melhor que o *Snort*. Mas ainda há a necessidade de muito estudo em cima de suas capacidades.

Tabela 2: Tabela comparativa dos resultados encontrados durante os testes entre o *Snort* e o *Suricata*.

[17] Configurações padrões após a instalação.

7 CONSIDERAÇÕES FINAIS

A segurança de informação nos ambientes corporativos é de suma importância para que não ocorram acessos indevidos aos sistemas internos da empresa e/ou vazamento de informações sigilosas. Existem diversas ferramentas que facilitam o acesso a dados e/ou computadores com acesso restrito. Por outro lado, há também uma grande variedade de programas e medidas a serem tomadas para evitar estes tipos de ataques.

O presente trabalho apresentou a análise de dois sistemas de detecção de intrusos, em um ambiente corporativo, definindo qual dos *softwares* é o mais documentado em relação a instalação e configuração, qual possui a melhor eficiência na detecção de possíveis ataques e o melhor desempenho na realização dos testes. Para tal foram utilizados os sensores: *Snort* e *Suricata*, ambos distribuídos gratuitamente sob licença *GNU GPL v.2* (*Open-Source*).

Snort possibilita a análise de tráfego em tempo real, comparando-o com regras pré-estabelecidas em busca de padrões e registrando eventos e alertas em caso de correspondência. Apesar de ser uma excelente ferramenta para o monitoramento de redes, é possível realizar a instalação do mesmo em praticamente qualquer computador.

Suricata, apesar de ter menos tempo de existência, surgiu como alternativa às limitações do *Snort* em relação a utilização de apenas um núcleo de processamento durante o monitoramento da rede. Utilizando as mesmas regras criadas para o *Snort*, o Suricata possui um funcionamento muito semelhante. Pode-se observar como uma função adicional a análise de protocolos da camada de aplicação, já que o *Snort* monitora apenas protocolos da camada de rede.

Para a análise dos sensores e a realização dos testes comparativos, fez-se as seguintes etapas: o estudo da documentação disponibilizada para a instalação e configuração dos *softwares*, a montagem de um laboratório virtual e os testes iniciais e o estabelecimento de um ambiente real e os testes finais. A primeira etapa permitiu entender o funcionamento dos programas, a fim de realizar a instalação e a configuração dos mesmos. Já a segunda etapa, auxiliou na tentativa e erro durante a

instalação e configuração dos sensores. Finalmente, na terceira etapa, com o ambiente real estabelecido, foi possível a obtenção dos resultados finais.

A partir da realização destas etapas, foi possível concluir que:

a) Após a leitura da documentação disponibilizada oficialmente pelos distribuidores dos programas, fóruns e publicações acadêmicas, encontra-se maior facilidade na instalação e configuração do *Snort*. Isto pode estar relacionado ao *Snort* possuir mais de 10 anos de mercado e desenvolvimento que o *Suricata*. Dificilmente é encontrado alguma publicação com o foco no *Suricata*.

b) Durante a instalação e configuração dos sensores nota-se uma grande semelhança no funcionamento dos mesmos. Apesar de o *Suricata* ser capaz de monitorar protocolos da camada de apresentação, diferente do *Snort* que apenas monitora a camada de rede, e o mesmo utilizar mais de um núcleo do processador, estes não são motivos importantes o suficiente para diferenciá-lo do *Snort*.

c) Quando montado o laboratório virtual é possível estabelecer a correta instalação dos sensores e configurá-los como desejado. Também é possível instalar com sucesso a ferramenta utilizada para a replicação dos tráfegos de rede (*Tcpreplay*) e da avaliação do desempenho (*Nagios*).

d) Os testes iniciais tiveram como objetivo a completa instalação e configuração dos programas a fim de preparar os mesmos para a realização dos testes finais. Dessa forma, os testes iniciais atingiram o esperado.

e) O estabelecimento do ambiente real possibilitou a representação de uma rede corporativa para que os resultados fossem os mais próximos do esperado.

f) A partir dos testes finais, o programa que foi mais eficiente na detecção de possíveis ataques, utilizado a partir de configurações padrões da instalação, foi o *Snort*. Apesar de existir a incerteza em relação a falsos positivos, o *Snort* identificou inúmeros possíveis ataques durante a realização dos testes, possibilitando o administrador realizar uma análise detalhada do que está sendo trafegado pela rede. O *Suricata*, por sua vez, registrou apenas alguns acessos suspeitos, não transmitindo confiança em seus resultados.

g) A utilização de processamento do *Suricata* foi de aproximadamente 10% a menos que a do *Snort*, o que, apesar de precisar menos processamento para realizar o mesmo trabalho, não chega a ser um número relevante.

h) Durante a medição do desempenho dos sensores, enquanto monitoravam a rede, a utilização de memória RAM não é um fator importante para

avaliação de um IDS, pois ambos os sensores tiveram uma utilização de aproximadamente 1GB de memória RAM durante todos os testes realizados.

A partir dos pontos analisados e dos resultados encontrados, pode-se afirmar que o *Snort* é superior ao *Suricata*, pois é muito mais documentado e possui inúmeras publicações científicas sobre o seu funcionamento. O mesmo também obteve resultados muito superiores em relação ao *Suricata* durante a análise do tráfego a partir de configurações padrões da instalação. Reforçando que o *Snort* utiliza basicamente a mesma capacidade de processamento e memória que o *Suricata*.

Este trabalho respondeu ao objetivo principal e abriu a possibilidade de novas pesquisas acerca do mecanismo de detecção do IDS. Este leque de possibilidades aberto para novos estudos deu-se ao realizar todos os testes, porém fugiria do objetivo geral desta pesquisa e não teria tempo hábil para realiza-los.

O que mais se sobressaiu, deixando margens para a continuação deste estudo, é a incerteza de alarmes emitidos pelo sensor (falsos positivos). Quando realizados os testes, observou-se que se as capturas de tráfego utilizadas nesta pesquisa tivessem todos os pacotes devidamente identificados, seria possível realizar uma análise mais profunda dos programas. Almejando melhores resultados na detecção de possíveis ataques, tem-se a possibilidade de modificar as configurações padrões dos sensores. A ineficiência do *Suricata* a partir de configurações da instalação, também foi algo considerável, resultando em questionamentos em relação a possibilidade de melhorias no monitoramento caso seja devidamente configurado.

REFERÊNCIAS BIBLIOGRÁFICAS

BACE, R.; MELL P. **Intrusion Detection Systems**. 2011. Disponível em: http://cryptome.org/sp800-31.htm. Acesso 15 Junho 2015.

BURTON; et al. **Guide to Secure Intrusion Detection Systems**. Michigan: Syngress Publishing, 2003.

COSTA ALMEIDA, A. A. L. **A Internet e o Direito. Revista Consulex**, Ano II, nº. 24, Dezembro/1998.

DAVIS, M. A.; BODMER, S.; LEMASTERS, A. (2010). **Hacking exposed malware & rootkits: malware & rootkits security secrets & solutions**. New York, McGraw Hill.

GARFINKEL, T.; ROSENBLUM M. **A Virtual Machine Introspection Based Architecture for Intrusion Detection**. In: Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS), 2003.

GAUR, N. **Snort: Planning IDS for Your Enterprise**. Linux Jornal, 2001. Disponível em: http://www.linuxjournal.com/article/4668. Acesso em: 15 Junho 2015.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 6º Edição. São Paulo: Atlas, 2008.

HARDIKAR, A. M. **Malware 101 – Viruses**. SANS Institute, 2008. Disponível em: https://goo.gl/kY1bcc. Acesso em: 1 Junho 2015.

NAKAMURA, E. T.; GEUS, P. L. **Segurança de redes em ambientes cooperativos**. São Paulo: Berkeley Brasil, 2002.

NIST. **Guide to Malware Incident Prevention and Handling**. Recommendations of the National Institute of Standards and Technology, 2005. Disponível em: http://goo.gl/aLd91. Acesso em: 1 Junho 2015.

NORTHCUTT, S.; NOVAK, J. **Network Intrusion Detection**. 3rd Edition. New Riders Publishing, September 2002.

NSFOCUS. **Network Intrusion Prevention System**. White Paper, 2012. Disponível em: http://goo.gl/cE0JCU. Acesso em: 1 Junho 2015.

PAESANI, L. M. **Direito e internet: liberdade de informação, privacidade e responsabilidade civil**. São Paulo: Editora Atlas, 2000.

PINHO, J.B. **Relações Públicas na Internet: Técnicas e Estratégias para informar e influenciar públicos de interesse**. São Paulo: Summus, 2003.

PIPER, S. **Intrusion Prevention Systems for Dummies**. New Jersey: Wiley Publishing, Inc, 2011.

SÊMOLA, Marcos. **Gestão da segurança da informação: visão executiva da segurança da informação**. Rio de Janeiro:Campus, 2003.

SEQUEIRA, D. **Intrusion Prevention Systems** – Security's Silver Bullet?. GSEC Version 1.4B, Option 1, U.S., SANS Institute, 2002.

STONESOFT. **StoneGate Reference Guide**. Stonesoft Corporation, 2007. Disponível em: https://goo.gl/bqMXLq. Acceso em: 15 Junho 2015.

SVOBODA, J. **Network Traffic Analysis with Deep Packet Inspection** Method. 2014. Tese de Mestrado. Masaryk University. Faculty of Informatics.

WHITE, J.; FITZSIMMONS, T.; MATTHEWS, J. **Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata.** Cyber Sensing 2013, Proceedings of SPIE (8757). Disponível em: http://goo.gl/m9t9YQ. Acesso em: 15 Junho 2015.

APÊNDICE I DOWNLOAD DOS PROGRAMAS UTILIZADOS NESTE ESTUDO

Este anexo tem por objetivo disponibilizar o link de download de todas as ferramentas utilizadas durante os testes deste trabalho.

1) Sistemas operacionais:
   a. Ubuntu 12.04: http://releases.ubuntu.com/12.04/
   b. Kali Linux: https://www.kali.org/downloads/
   c. Centreon: https://download.centreon.com/

2) Sensores:
   a. Snort: https://snort.org/downloads
   b. Suricata: http://suricata-ids.org/download/

3) Ferramentas:
   a. VirtualBox: https://www.virtualbox.org/wiki/Downloads
   b. Samba: A partir de uma janela de terminal do Ubuntu: **apt-get install samba**.
   c. SNMPD: A partir de uma janela de terminal do Ubuntu: **apt-get install snmpd**
   d. Tcpreplay: A partir de uma janela de terminal do Ubuntu: **apt-get install tcpreplay**

APÊNDICE II GUIA DE INSTALAÇÃO E UTILIZAÇÃO DAS FERRAMENTAS

1) Configuração do Ubuntu
   a. Rede

```
sudo nano /etc/network/interfaces
```

```
auto eth0
iface eth0 inet static
        address x.x.x.x
        netmask x.x.x.x
        gateway x.x.x.x
```

```
sudo echo 8.8.8.8 > /etc/network/resolv.conf
/etc/init.d/networking restart
```

2) Ferramentas
   a. Tcpreplay

```
sudo apt-get install tcpreplay
## reproduzir captura a uma taxa de 100mbit/s
sudo tcpreplay –M100 –ieth0 sample.cap
## reproduzir captura a uma taxa de 1000mbit/s
sudo tcpreplay ---topspeed –ieth0 sample.cap
```

   b. Ssh

```
sudo apt-get install openssh-server
```

c. Samba

```
sudo apt-get install samba

sudo mkdir /shared
sudo chmod 777 /shared

sudo nano /etc/samba/smb.conf

## no final do arquivo, adicionar (não recomendado para ambientes de produção):

[Shared]
        path = /shared
        guest ok = yes
        read only = no
        browseable = yes
        writable = yes

## salvar e sair

sudo service smbd restart
```

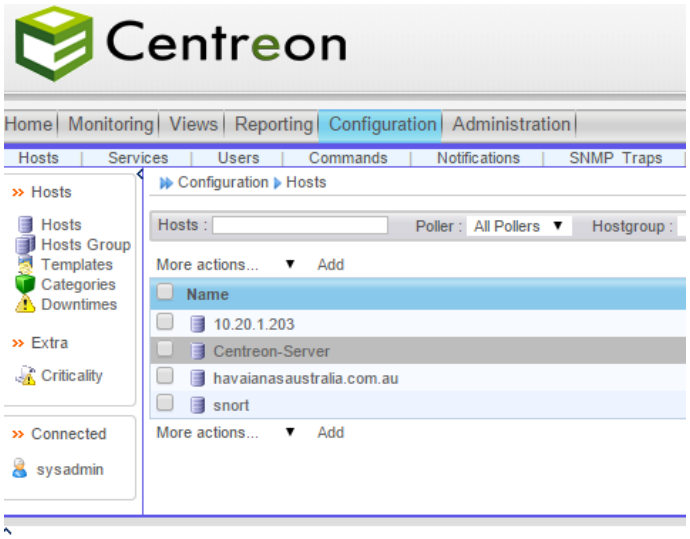
e. SNMPD

```
sudo apt-get install snmpd
sudo nano /etc/snmp/snmpd.conf
## mudar a linha: agentAddress  udp:127.0.0.1:161
## para: agentAddress  udp:*:161

## e a linha: rocommunity public  default  -V systemonly
## mude para: rocommunity public  default
```

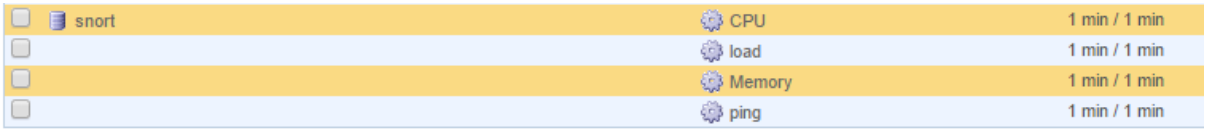
```
## salve e saia

sudo service snmdp restart
```

f. Centreon/Nagios: O Centreon é distribuído como um sistema operacional completo, bastante somente iniciar o computador a partir do disco e instalar com configurações padrão. Após a instalação, configurar a rede e então realizar o acesso ao sistema através do browser. A seguir, será mostrado uma sequencia de figuras das telas de configuração do mesmo.

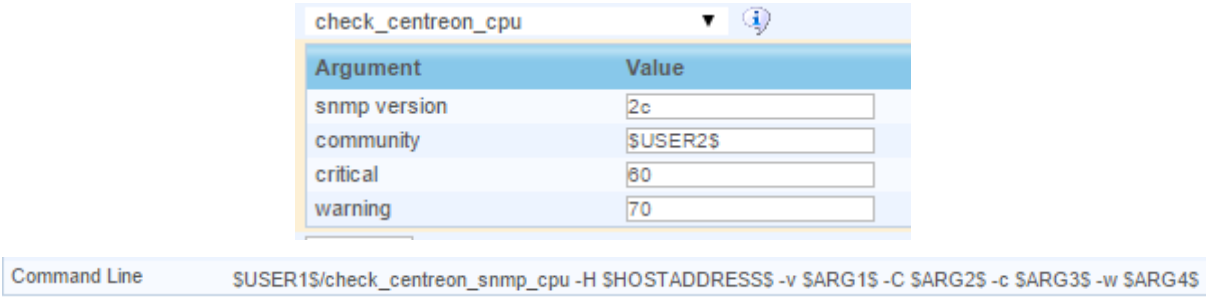
- Tela de cadastro de hosts:



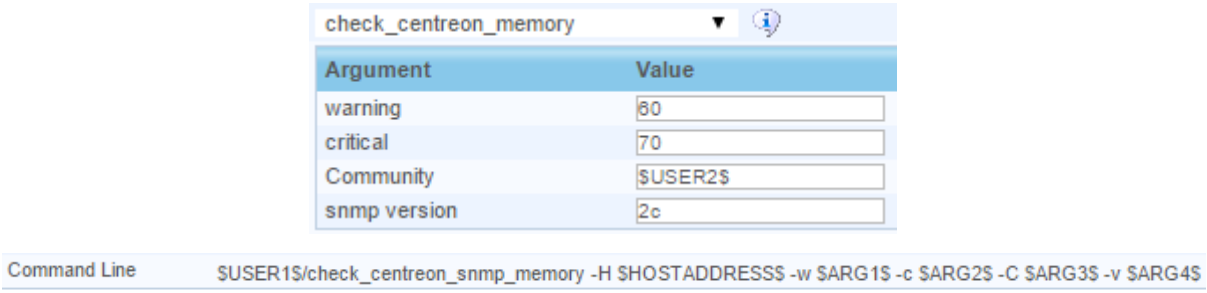
- Tela de cadastro de serviços para o host:



- Configurações para o monitoramento do processador:



- Configurações para o monitoramento da memória RAM:



3) Sensores

a. Snort: O snort pode ser instalado no Ubuntu apenas com apt-get install snort, ou através do código fonte, como mostrado a seguir:

- Primeiro instalar as dependências;

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install mysql-server nmap nbtscan apache2 php5 php5-mysql php5-gd libpcap0.8-dev libpcre3-dev g++ bison flex libpcap-ruby make zlib1g-dev libmysqld-dev libdnet libdnet-dev libpcre3 libpcre3-dev gcc make flex byacc bison linux-headers-generic libxml2-dev libdumbnet-dev zlib1g zlib1g-dev
```

- Configure uma pasta para baixar o código fonte e realizar a compilação;

```
mkdir /usr/local/src/snort
```

```
cd /usr/local/src/snort
wget https://snort.org/downloads/snort/daq-2.0.5.tar.gz
tar –xvzf daq-2.0.5.tar.gz
cd daq-2.0.5
./configure
make
make install
cd ..
wget https://snort.org/downloads/snort/snort-2.9.7.3.tar.gz
tar –xvzf snort-2.9.7.3.tar.gz
cd snort-2.9.7.3
./configure
make
make install

ldconfig

ln -s /usr/local/bin/snort /usr/sbin/snort

mkdir /etc/snort
mkdir /etc/snort/rules/
mkdir /etc/snort/preproc_rules
mkdir /var/log/snort

cp ~/snort_src/snort-2.9.7.0/etc/*.conf* /etc/snort
cp ~/snort_src/snort-2.9.7.3/etc/*.map /etc/snort

sudo nano /etc/snort/snort.conf

## procure pela linha: ipvar HOME_NET any
## mude o “any”a o endereço de rede local
## mudar tambem:
## var RULE_PATH /etc/snort/rules
```

```
## var SO_RULE_PATH /etc/snort/so_rules
## var WHITE_LIST_PATH /etc/snort/rules
## var BLACK_LIST_PATH /etc/snort/rules
## procure pelas linhas de regras (include $rule_path...)
## comente todas estas linhas
## deixe ou adicione apenas as linhas que queira

## regras open-source
# http://rules.emergingthreatspro.com/open/

## salve e saia
## iniciar o Snort em modo NIDS
snort -A console -q -c /etc/snort/snort.conf -i eth0
```

b. Suricata

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata htp
# baixar as regras do suricata no website
# https://rules.emergingthreatspro.com/open/suricata/

sudo nano /etc/suricata/suricate.yaml
# Comentar todas as linhas de regras existentes
# manter somente as do “Emerging Threats”
# iniciar o Suricata
sudo suricata -c /etc/suricata/suricata.yaml -i eth0
```