CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

EZEQUIEL COSER

AUTOMATIZAÇÃO DO PROCESSO DE CONTENÇÃO DE AMEAÇAS BASEADA EM FERRAMENTA DE IDS/IPS (SISTEMA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO)

Lajeado
2011

EZEQUIEL COSER

AUTOMATIZAÇÃO DO PROCESSO DE CONTENÇÃO DE AMEAÇAS BASEADA EM FERRAMENTA DE IDS/IPS (SISTEMA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO)

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES como parte dos requisitos para a obtenção do título de bacharel em Engenharia de Controle e Automação.
Área de concentração: Segurança de Redes

ORIENTADOR: Luís Antônio Schneiders

Lajeado

2011

EZEQUIEL COSER

AUTOMATIZAÇÃO DO PROCESSO DE CONTENÇÃO DE AMEAÇAS BASEADA EM FERRAMENTA DE IDS/IPS (SISTEMA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO)

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia de Controle e Automação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Luís Antônio Schneiders, Univates

Mestre pela UFRGS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Alexandre Stürmer Wolf, Univates

Mestre pela PUC-Rio – Rio de Janeiro, Brasil

Prof. Fabrício Pretto, Univates

Mestre pela PUCRS – Porto Alegre, Brasil

Coordenador do curso de Engenharia de Controle e Automação: _____

Prof. Rodrigo Wolf Porto

Lajeado, novembro de 2011

Dedico este trabalho à minha namorada, Carine Paula Walter, aos meus pais, Agemir Coser e Neisi Bonfanti Coser e ao meu irmão, Ederson Coser, pelo grande incentivo e apoio.

AGRADECIMENTOS

A minha família, por todo apoio e ajuda prestada durante o período que estive cursando a faculdade, pois sem eles a concretização deste sonho não seria possível.

A minha namorada Carine Paula Walter, pelo seu apoio e compreensão nos momentos de minha ausência, e por ser paciente comigo quando encontrava-me estressado com minha monografia.

Ao meu coordenador Tiago Giovanaz da Silva, que sempre foi prestativo e solidário nos momentos de minha ausência no trabalho e a todos meus colegas.

Ao meu orientador, que viu futuro no meu trabalho e aceitou guiar-me para sua implementação e a UNIVATES pela confiança por disponibilizar a sua estrutura de rede para que fosse possível a aplicação do mesmo.

Por fim, agradeço a Deus que tornou tudo isso possível.

RESUMO

A área de segurança de redes não era tão importante na década de 90 como tem sido nos dias atuais, pois encontrava-se disponível somente em universidades e grandes empresas, e apenas pessoas que possuíam vínculos com essas instituições podiam usufruir desse benefício. Com o passar do tempo, essa realidade tem se alterado, pois com o avanço da tecnologia e a queda nos custos de implantação, a rede de dados está cada vez mais interconectada. Essa evolução tem facilitado a rotina de pessoas que fazem uso de uma rede de computadores, mas, em contrapartida, isso trouxe diversos problemas quanto à segurança da rede de dados. Sendo assim, automatizar um processo que detecte uma intrusão é extremamente importante, pois pode-se evitar um ataque antes que este afete a integridade da rede. Diante desse contexto este estudo apresenta e implementa ferramentas que podem elevar a segurança de uma rede de dados, e tornar possível o processo automático de bloqueio de intrusos e tráfegos não desejados por meio de um sistema de detecção e prevenção de intrusão.

Palavras-chaves: Segurança à rede. Sistema de detecção e prevenção de intrusão, Ataques. Ameaças.

ABSTRACT

The area of network security was not as important in the 90's as it has been today, it was available only in universities and big companies, and only people who had links with these institutions could take advantage of this benefit. Over time, this reality has changed, because with the advancement of technology and falling costs of implementation, the data network is increasingly interconnected. This development has facilitated people who make routine use of a computer network, but this in turn brought several issues regarding the security of network. Therefore, automating a process that detects an intrusion is extremely important as it can prevent an attack before it affects network integrity. Against this background this paper presents and implements tools that can increase the security of a data network, and make possible the automatic process of blocking unwanted traffic and intruders through a detection system and intrusion prevention.

Keywords: Network security. Detection system and intrusion prevention. Attack. Threat.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API: Application Programming Interface

ATM: Asynchronous Transfer Mode

CIA: Confidentiality, Integrity, Availability

DAC: Discretionary access control

DdoS: Distributed Denial of Service Attack

DNS: Domain Name System

DoS: Denial of Service

FDDI: Fiber Distributed Data Interface

FTP: File Transfer Protocol

HIDS: Host Intrusion Detection System

HIPS: Host Intrusion Prevention System

HTML Hyper Text Markup Language

HTTP: Hypertext Transfer Protocol

IANA: Internet Assigned Numbers Authority

ICMP: Internet Control Message Protocol

IDS: Sistema de Detecção de Intrusão

IGMP: Internet Group Management Protocol

IP: Internet Protocol

IPS: Internet Protocol Suite

IPS: Sistema de Prevenção de Intrusão

IPSec: Internet Protocol Security

LAN: Local Area Network

MAC: Mandatory Access Control

MTU: Maximum Transmit Unit

NFS: Network File System

NIDS: Network Intrusion Detection System

NIPS: Network Intrusion Prevention System

NTFS: New Technology File System

OSI: Open Systems Interconnection

OSPF: Open Shortest Path First

RBAC: Role Based Access Control

RPC: Remote Procedure Calls

SMTP: Simple Mail Transfer Protocol

SO: Sistema Operacional

SQL: Structured Query Language

SSH: Secure Shell Client

TCP: Transmission Control Protocol

TI: Tecnologia da Informação

UDP: User Datagram Protocol

WAN: Wide Area Network

WEP: Wired Equivalent Privacy

SUMÁRIO

1 INTRODUÇÃO

As redes de computadores surgiram da necessidade da troca de informação, caracterizando uma nova realidade para uso dos computadores e nos sistemas computacionais. Com ela foi possível ter acesso a um determinado dado que está fisicamente localizado a centenas ou milhares de quilômetros de distância (TORRES, 2009).

No princípio as redes eram restritas às universidades, centros de pesquisas e departamentos governamentais, que possuíam mais recursos e possibilidades de implementação, sendo, dessa maneira, o acesso era restrito a um pequeno grupo de pessoas que possuíam vínculo com as instituições. Além disso, os dados raramente circulavam ou eram acessíveis de outras redes (TANENBAUM, 1997).

Atualmente, com a queda de custos de implementação de rede, é impossível pensar em um ambiente de trabalho onde os computadores não estejam interligados, efetuando troca de dados, compartilhando recursos ou acessando à *internet* (TORRES, 2009). Com toda essa disponibilidade, garantir a segurança dos dados passou a ser uma grande preocupação, pois possuindo acesso à *internet* possibilita-se que a rede privada seja atingida pelo mundo exterior. Então, se métodos de segurança não forem implementados, todo sistema de informação corre o risco de ser explorado, oriundo de várias fontes e de várias formas, podendo causar perda de conectividade ou divulgação de dados valiosos (BHARDWAJ, 2007).

Segundo Harrington (2005), segurança da informação pode ser definida como proteção contra ataques ou divulgação não intencional dos dados que estão armazenados ou os que trafegam através de uma rede de computadores. Ainda, conforme este, uma forma de facilitar a compreensão do que é segurança da informação, é diferenciar segurança de privacidade: privacidade é a necessidade de restringir o acesso aos dados e informações, e segurança é garantir esta privacidade.

Brenton e Hunt (2001) observam que diariamente surgem notícias de ataques contra sistemas de organizações governamentais, sistemas privados, políticos e sociais, não sendo nem todos os ataques divulgados, pois empresas podem ter prejuízos ao divulgarem que foram atacadas, seja por ter sua imagem denegrida por mostrar a fragilidade de seus sistemas computacionais ou por prejuízos financeiros diretos, como, por exemplo, para instituições financeiras e *sites* de compras *on-line*.

As redes de computadores estão cada vez mais interconectadas. Com isso a estabilidade e a segurança tornam-se imprescindíveis, já que o mundo dos negócios está

fortemente apoiado na Tecnologia da Informação (TI), podendo, em caso de falha, todo um negócio ser prejudicado (LOCKHART, 2006).

Bhaiji (2008) apresenta os três princípios de segurança, conforme visto na Figura 1.

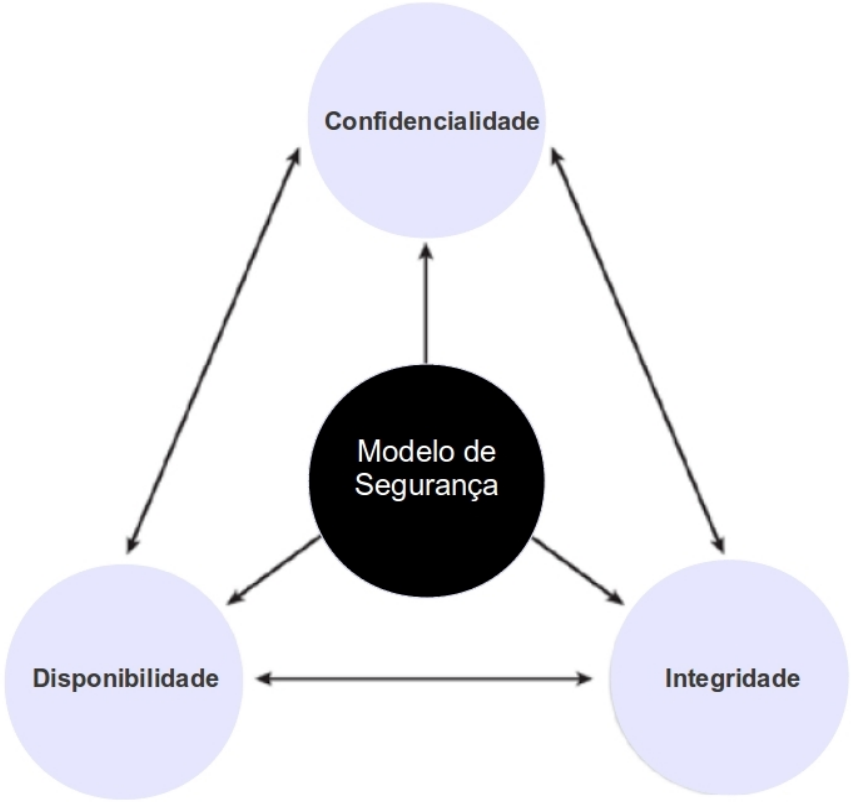

Figura 1 - Princípios da segurança
Fonte: Do autor com base em Bhaiji (2008).

- confidencialidade: impede a divulgação de informações confidenciais a pessoas não autorizadas;
- integridade: impede a modificação não autorizada de dados, sistemas e informações, proporcionando assim garantia de dados íntegros;
- disponibilidade: é a prevenção contra perda de acesso a recursos e informações, garante que a informação está disponível para uso quando necessário.

Para Brenton e Hunt (2001), um administrador de rede tem que pensar como um atacante, pois assim pode determinar a melhor forma de proteger seus recursos, efetuando uma análise da rede para avaliar o nível de proteção de que realmente necessita.

Com a crescente sofisticação nas ferramentas de ataques, sem contar com novas pragas que são liberadas diariamente, e a inclusão digital possibilitando o fácil acesso à *internet*, estão surgindo novos consumidores alavancando de forma exponencial fontes vulneráveis de ataques, tornando cada vez mais complexa a atividade de manter uma rede segura. Então, o fator segurança está cada vez mais presente no nosso cotidiano, e mantê-lo

tem se tornado uma tarefa árdua e desafiadora. Para tal, são incentivadas a conscientização aos usuários das redes de computadores, a utilização de software antivírus, de ferramentas de detecção e de prevenção de intrusão (IDS/IPS), a implantação de *firewalls*, entre outras ferramentas para auxiliar na proteção da rede (BHARDWAJ, 2007).

1.1 Objetivos

Considerando-se as questões levantadas, o presente trabalho tem por objetivo identificar e automatizar o processo de contenção de ameaças na rede *wireless* do Centro Universitário UNIVATES com o auxílio de ferramenta de IDS/IPS (Sistema de Detecção e Prevenção de Invasão), com isso efetuar uma análise qualitativa dos bloqueios gerados a fim de viabilizar a implementação da ferramenta. Como objetivos secundários, visa a atribuir assinaturas existentes aos processos de contenção e identificar novos padrões de assinaturas.

A manutenção da segurança de uma rede de dados é dificultada quando esse ambiente é híbrido e heterogêneo. O ambiente da grande maioria das instituições de ensino é assim, no caso da rede de dados do Centro Universitário UNIVATES – ambiente utilizado para realização deste trabalho – não poderia ser diferente. Sendo assim, para auxiliar no árduo processo de manter a segurança da rede de dados, implementar-se-á um sistema automatizado de monitoramento, detecção e prevenção de tentativas de intrusão utilizando um conjunto de aplicações de códigos abertos *(open source)* desenvolvidas para monitoramento do tráfego de pacotes da rede IP *(Internet Protocol)*, realizando análises em tempo real sobre os protocolos e conteúdos, e fazendo uso de metodologias de detecção multidimensional por meio da análise de assinaturas de ataques, anomalias no protocolo e no comportamento do tráfego.

1.2 Organização do trabalho

O presente trabalho está dividido em seis capítulos. Os dois primeiros fazem uma introdução a redes de computadores, enquanto os outros três são voltados para áreas de segurança e sua aplicação. O Capítulo 2 descreve o modelo OSI e TCP/IP, em que é conceituada cada camada por este composta. E sua finalização é feita com uma correlação entre os dois modelos, exibindo a não compatibilidade de suas camadas. O Capítulo 3 aborda os principais problemas quanto à segurança da rede de dados, tipos de ataques, dificuldade para contê-los e principais atacantes. Além disso, aborda as principais dificuldades para conter uma ameaça e boas práticas para minimizar os problemas relativos à segurança. No Capítulo 4 são apresentados os principais mecanismos de defesa, contemplando *firewall*, controle de

acesso, entre outros, os quais têm por função garantir e aumentar o nível de segurança lógica da rede. O Capítulo 5 aborda o sistema de detecção e prevenção de intrusão, compondo o tema principal do trabalho proposto. Descreve seus conceitos, tipos de implantação e formas de funcionamento. A contextualização deste capítulo serve como base para a estruturação de implementação a ser realizada no capítulo seguinte. O Capítulo 6 parte de uma análise de um ambiente base, determinação do problema, apresentação física e lógica de um ambiente ideal e descrição das ferramentas utilizadas no caso do IDS/IPS. Finalizando o capítulo, são apresentadas as considerações finais do autor quanto aos propósitos e resultados obtidos com o estudo de forma a automatizar o processo de identificação de ameaças na rede de dados analisada.

2 REVISÃO DE LITERATURA

Neste capítulo, são apresentados tópicos sobre rede de computadores, modelo OSI, modelo TCP/IP e um comparativo entre os dois modelos. Os assuntos mencionados são descritos resumidamente, pois entende-se que eles são pré-requisitos para o entendimento do tema segurança de redes de computadores, pois as ferramentas de segurança do tipo IDS/IPS vão interagir diretamente nas camadas desses modelos.

2.1 Redes de computadores

Segundo McQuerry (2008) e Sportack (2004), uma rede são diversos dispositivos conectados em um sistema final, como computadores e servidores, os quais se comunicam entre si, compartilhando recursos físicos e lógicos. Mais especificamente, uma rede de computador é uma infraestrutura que fornece a conectividade com múltiplos sistemas de computação autônoma, a fim de comunicar (por exemplo, *e-mail*) e compartilhar recursos. Esses recursos incluem hardware, como mídia de armazenamento e periféricos (por exemplo, discos rígidos, impressoras), software (ou seja, programas de computador), e as informações/dados (ou seja, arquivos de computador - por exemplo, um banco de dados).

As redes de computadores, salientam Soares, Lemos e Colcher (1995), surgiram para viabilizar a troca e o compartilhamento de informações e dispositivos periféricos, preservando a independência das várias estações de processamento e permitindo a integração em ambientes de trabalho cooperativos.

Uma rede local refere Castelli (2004), envia e recebe pacotes a uma velocidade muito maior do que conexões telefônicas, porém, existe limitação quanto à distância da estação até o concentrador. Tendo em vista essas características de velocidade e distância, a utilização das redes locais disseminou-se rapidamente em ambientes geograficamente agrupados, como em universidades e empresas, que se enquadram neste agrupamento e necessitam de velocidade maior na comunicação de dados. A Figura 2 ilustra o comportamento de uma rede local com compartilhamento de recursos.

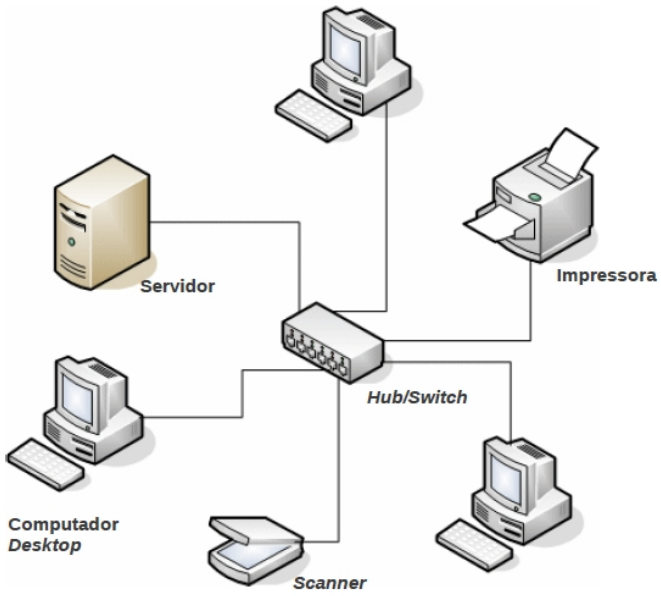

Figura 2 - Rede de computadores local
Fonte: Do autor com base em Sportack (2004).

Segundo Castelli (2004), três características diferenciam uma rede de outra:

- redes podem ser configuradas em diferentes topologias, que é a forma de interligação dos dispositivos. As duas topologias mais conhecidas são anel, em que um dispositivo conecta-se a outro até fechar o anel, e estrela, em que cada dispositivo é conectado ao concentrador;
- as redes seguem diferentes protocolos, que são as normas de envio e recebimento de pacotes;
- as redes são conectadas por algum meio físico: fios de cobre, fibra óptica ou *wireless*, com características específicas para cada meio.

2.2 Modelo de referência OSI *(Open Systems Interconnection)*

O modelo OSI trata da interconexão de sistemas que estão abertos à comunicação com outros sistemas (TANENBAUM, 1997). Compreende um modelo de sete camadas, como é possível observar na Figura 3.

Figura 3 - Modelo OSI
Fonte: Do autor com base em Torres (2009).

2.2.1 Aplicação – camada 7

A camada de aplicação faz a interface entre o protocolo de comunicação e o aplicativo que irá trafegar por meio da rede, como, por exemplo, navegadores de *internet*, softwares de mensagens instantâneas, leitores de *e-mail*, entre outros. Esta é a camada que faz a interface com os usuários por meio dos programas utilizados pelo usuário e que fazem uso da rede. Importante salientar que a maioria dos aplicativos pode estar em uso ao mesmo tempo acessando dados diferentes, como é o caso dos navegadores, em que é possível, ao mesmo tempo, o acesso a mais de um *site* (TORRES, 2009).

2.2.2 Apresentação – camada 6

Esta camada é responsável por apresentar os dados de e para a aplicação e a sessão, convertendo-os para que cada uma das camadas os entenda, ou seja, estabelece um formato comum a ser usado na transmissão desse dado. Esta camada é comumente chamada de tradutora. Exemplos de arquivos são formatos gráficos: *jpg*, *png*, *bmp*; formatos de som e vídeo: *mp3*, *divx*, *mp4*. Nesta camada é possível utilizar algum esquema de criptografia em que os dados serão decriptografados na camada 6 do destino (NOONAN; DUBRAWSKY, 2006).

2.2.3 Sessão – camada 5

Permite que duas aplicações em computadores diferentes estabeleçam e mantenham uma sessão de comunicação. Possibilita que em uma rede de computadores diversos dispositivos possam conectar-se a diversos outros dispositivos e, ao mesmo tempo, em diversos aplicativos, assegurando que os dados serão entregues à aplicação correta. Alguns exemplos de protocolos são o *Remote Procedure Calls* (*RPC*), no qual o servidor executa procedimentos que são requisitados pelo cliente; o *NetBios*, que é uma *Application Programming Interface* (*API*) utilizada basicamente por sistemas Windows; e a *Structured Query Language* (*SQL*), que são conexões a servidores de banco de dados (DOHERTY; ANDERSON; MAGGIORA, 2007).

2.2.4 Transporte – camada 4

Responsável por adquirir os dados enviados pela camada de sessão e dividi-los em pacotes que serão transmitidos através da rede, por meio de um método que independe de aplicação, executando os procedimentos de segmentação e remontagem. Os dois principais exemplos de protocolos desta camada são T*ransmission Control Protocol* (*TCP*) e *User Datagram Protocol* (*UDP*) (HELD, 2003*).*

2.2.5 Rede – camada 3

Responsável pelo endereçamento dos pacotes, convertendo endereços lógicos em físicos, fazendo com que o pacote chegue ao destino. É nela a maioria dos protocolos de comunicação funcionam, confiando nas camadas 1 e 2 para enviar e receber mensagens de e para outros dispositivos de rede. Nesta camada, é acrescentado cabeçalho que identifica os endereços *Internet Protocol* (*IP*) de origem e de destino do pacote. É neste nível que ocorre o roteamento IP (TANENBAUM, 2003).

2.2.6 *Link* de dados – camada 2

Também chamada de camada de enlace, pega os pacotes de dados recebidos da camada de rede e os transforma em quadros que serão trafegados pela rede. É responsável por estabelecer a mais elementar forma de comunicação entre dois dispositivos de rede, para que possam trocar dados em protocolos da camada 3. Nesta camada os dispositivos são

identificados pelo endereço conhecido como *Media Access Control* (*MAC*) (TANENBAUM, 2003).

2.2.7 Física – camada 1

Faz a transformação dos quadros enviados pela camada de enlace em sinais compatíveis com o meio de transmissão, que pode ser cobre, fibra óptica, *wireless*, entre outros. Esta camada também fornece um método para que o dispositivo receptor faça a validação considerando se os dados estão ou não corrompidos (TANENBAUM, 2003).

2.3 Modelo TCP/IP

TCP/IP é um padrão aberto de protocolos de comunicação que é o padrão utilizado na *internet*, escolhido para esta finalidade pois contém todos os mecanismos fundamentais para suportar todo e qualquer tipo de comunicação de rede (CASTELLI, 2004). Segundo Sportack, (2004), foi o protocolo de rede que mudou o mundo, revolucionou a forma como as pessoas se comunicam, e trocam informações e fazem negócios.

O nome *TCP/IP* se refere a um conjunto de protocolos de comunicação, recebendo o modelo o nome de dois dos protocolos que pertencem a ele, o *Transmission Control Protocol* (*TCP*) e o *Internet Protocol* (*IP*). Também encontra-se na literatura o nome *Internet Protocol Suite* (*IPS*), que, apesar de aceito, não é comumente utilizado (HUNT, 2002).

A grande popularidade do protocolo *TCP/IP*, conforme Hunt (2002), pode ser explicada por quatro características principais:

- toda a implementação e todas as normas são abertas, ou seja, independente de hardware ou software, é possível implementar o protocolo *TCP/IP*, podendo a comunicação ser estabelecida entre quaisquer dispositivos que implementem o protocolo;
- independente de hardware de rede, ou seja, o *TCP/IP* pode integrar diferentes tipos de rede, sejam sem fio, redes de cobre, ópticas, linhas *dial-up*, redes *ethernet*, enfim, qualquer meio físico ou tipo de transmissão;
- um sistema de endereçamento comum, que permite a comunicação entre dispositivos independente do tamanho da rede, seja uma rede com apenas dois computadores ou a *internet*, na qual existem incontáveis dispositivos conectados e trocando informações ao mesmo tempo;

- protocolos padronizados, consistentes e amplamente disponíveis para serviços aos usuários.

Atualmente, implementações do *TCP/IP* estão presentes em computadores de todos os portes, aparelhos celulares, *storages*[1], impressoras, eletrodomésticos, enfim o *TCP/IP* é cada vez mais encontrado em dispositivos que passam a comunicar-se, o que permite novos benefícios à população em geral.

Segundo Hunt (2002), não existe consenso em descrever o *TCP/IP* em camadas, porém, normalmente, divide-se o *TCP/IP*, assim como o modelo OSI, em camadas. Enquanto o modelo OSI é dividido em sete camadas, o *TCP/IP* é dividido em apenas quatro, como é possível observar na Figura 4.

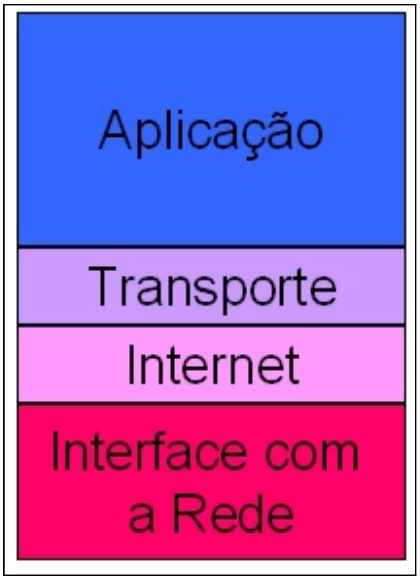

Figura 4 - Modelo *TCP/IP*
Fonte: Do autor com base em Torres (2009).

2.3.1 Aplicação – camada 4

É a camada que interage com o usuário, ou seja, nela encontram-se aplicativos como clientes de *e-mail*, navegadores de *internet*, enfim, diversos softwares que utilizam a rede e interagem com o usuário final (HUNT, 2002).

Para fácil entendimento do funcionamento, é importante destacar a existência de duas entidades na comunicação: o emissor e o receptor. Para o envio, os dados do usuário emissor são combinados com os dados do aplicativo, sendo encapsulados com a adição de algumas informações, como a porta de origem. Os dados são passados para a camada de transporte.

[1] *Storage* é um dispositivo exclusivamente dedicado ao armazenamento de informação. Disponível em: <http://www.kerkeberos.net/2009/01/07/solucoes-para-a-criacao-do-seu-servidor-nas-network-attachedstorage-domestico/>. Acesso em: 10 jun. 2011.

Quando do recebimento, o receptor, na camada de aplicação, remove o cabeçalho da aplicação, provê o tratamento necessário para finalizar a transação e confirma que o processo foi concluído (TORRES, 2009).

A Tabela 1 exemplifica alguns protocolos da camada de aplicação:

Tabela 1 - Protocolos da camada de aplicação

Protocolo	Função
Telnet	*Login* remoto
File Transfer Protocol (*FTP*)	Transferência de arquivos
Simple Mail Transfer Protocol (*SMTP*)	Entrega de correio eletrônico
Hypertext Transfer Protocol (*HTTP*)	Acesso a *web sites*
Domain Name System (*DNS*)	Resolução de nomes de domínios
Open Shortest Path First (*OSPF*)	Protocolo de roteamento
Network File System (*NFS*)	Compartilhamento de diretórios

Fonte: Do autor com base em Hunt (2002).

2.3.2 Transporte – camada 3

A camada *Host-to-Host Transport Layer* é geralmente abreviada e traduzida como camada de transporte. Os dois protocolos mais importantes desta camada são o *Transmission Control Protocol* (*TCP*) e o *User Datagram Protocol* (*UDP*). A diferença entre o *TCP* e o *UDP* é que o primeiro tem conexão orientada a serviço, ou seja, o destinatário confirma o recebimento do dado. Quando da implementação de aplicações, o desenvolvedor escolhe qual protocolo melhor se enquadra aos softwares, tendo em vista que o *TCP* é confiável e possui mecanismos de detecção e correção de erros, enquanto o *UDP* tem menor *overhead* pois dá acesso direto a um serviço de entrega de datagramas (HUNT, 2002).

2.3.2.1 TCP

Segundo Tanenbaum (2003), o protocolo *TCP* foi concebido para ser um protocolo de transporte confiável, fim a fim, em redes interconectadas, que apresentam diferentes topologias, banda, tipos e tamanho de pacotes, entre outros. O *TCP* é orientado à conexão, ou seja, cria um circuito virtual, *full duplex* (tem a capacidade de enviar e receber dados simultaneamente) entre duas aplicações, sendo todos os *bytes* numerados para que seja possível a retransmissão em caso de falhas.

Na Figura 5 é possível observar, de maneira ilustrativa, o túnel virtual criado pelo protocolo TCP, com sua característica ponto a ponto e *full duplex*.

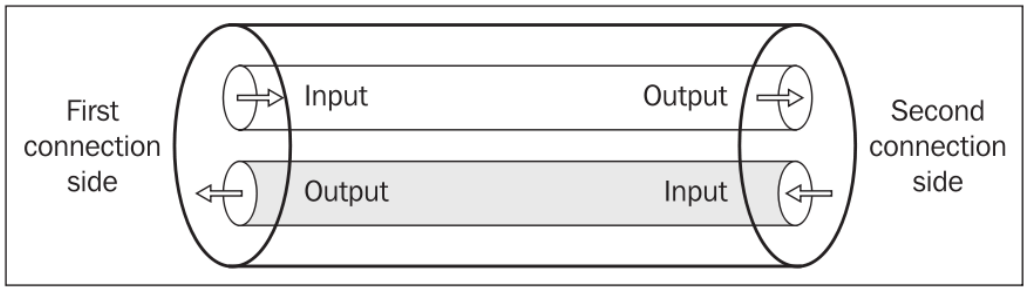

Figura 5 - Ligação ponto a ponto bidirecional
Fonte: Dostálek e Kabelová (2006, p. 239).

O detalhamento da estrutura de um segmento *TCP* pode ser visto na Figura 6, na qual se identificam alguns campos: porta de origem e de destino, usada para identificar as portas no emissor e no receptor; o número de sequência, utilizado para o controle do *TCP*, visto que os pacotes devem ser enviados e recebidos em ordem; *checksum*, que é utilizado para verificar a integridade dos dados, entre outros campos.

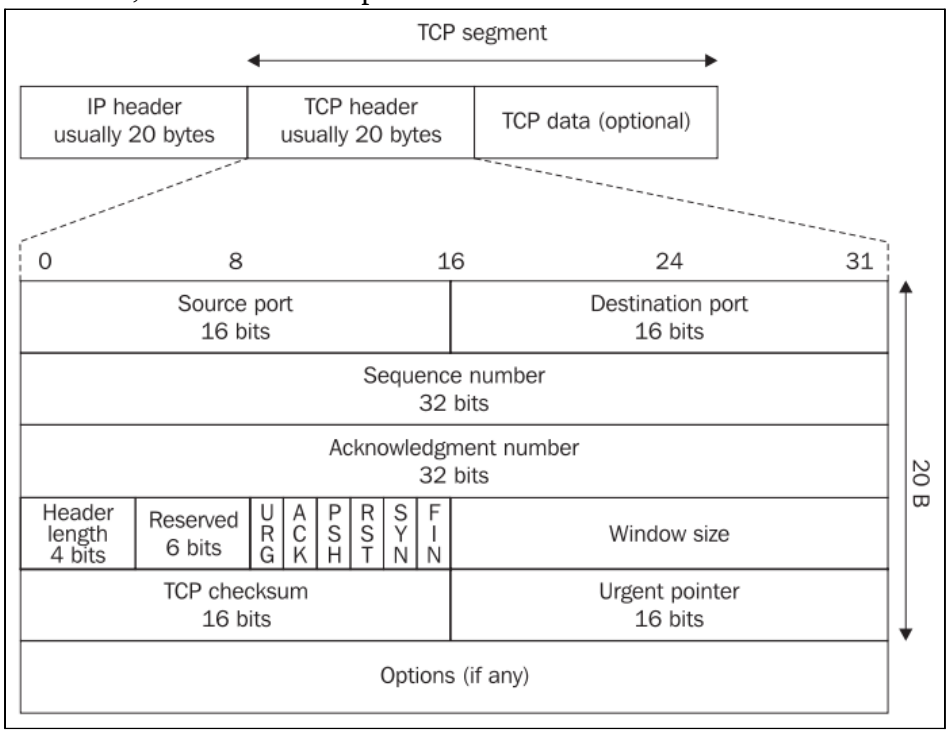

Figura 6 - Fragmento *TCP*
Fonte: Dostálek e Kabelová (2006, p. 241).

O estudo dos protocolos de rede pode confundir as funções dos dois protocolos que formam o nome do modelo *TCP/IP*. Segundo Dostálek e Kabelová (2006), o *IP* transmite dados entre dispositivos, enquanto o *TCP* transfere dados entre as aplicações nestes dispositivos, utilizando para tanto a porta na qual o serviço está sendo executado.

Carne (2004) explica que o protocolo *TCP* somente é utilizado com endereços *unicast* (entre dois *hosts*), não podendo ser usado para endereços *multicast* (grupo de computadores) ou *broadcast* (todos os dispositivos no mesmo domínio de *broadcast*).

Todas as aplicações que utilizam o protocolo *TCP* têm a garantia de que os dados serão entregues. Porém, destacam Dostálek e Kabelová (2006), que essa proteção garantida pelo *TCP* não oferece proteção contra ataques a rede de dados. A garantia do protocolo se limita à entrega dos dados a outra ponta.

Os pontos de origem e destino na conexão são identificados por um número de porta que, no caso do protocolo *TCP*, pode variar de 0 a 65535. No caso da *internet*, a aplicação de destino é endereçada por meio de um endereço *IP*, um número de porta e o protocolo, transferindo o *IP* e os pacotes para um determinado dispositivo, que pode executar vários aplicativos simultaneamente, visando entender o sistema operacional para tanto, é utilizado porta de destino, que entende para qual aplicação deve ser entregue o pacote *TCP* (DOSTÁLEK; KABELOVÁ, 2006).

Conforme Tanenbaum (2003), as portas de 0 até 1.024 são conhecidas e reservadas, com regulação feita pela *Internet Assigned Numbers Authority* (*IANA*). A Tabela 2 relaciona alguns serviços e portas.

Tabela 2 - Algumas portas atribuídas

Porta	Protocolo	Uso
21	FTP	Transferência de arquivos
23	TELNET	*Login* remoto
25	SMTP	Entrega de correio eletrônico
69	TFTP	Transferência de arquivos
79	FINGER	Verificação de informações de usuários
80	HTTP	Acesso a *web sites*
110	POP3	Acesso remoto a e-mails
119	NNTP	Notícias

Fonte: Do autor com base em Tanenbaum (2003).

2.3.2.2 User Datagram Protocol (UDP)

O *UDP* é um protocolo simples, destinado a aplicações que não exigem confirmação de entrega dos pacotes, ou seja, serviços não confiáveis. Ao enviar, o protocolo recebe os

dados da camada superior, adiciona o número da porta de destino e calcula o *checksum*[2] (opcional), para que o receptor valide os dados e faça o processo contrário no recebimento. O cabeçalho é bastante pequeno (8 *bytes*), como pode ser visto na Figura 7, e o restante são dados. Sem as informações de porta, seria impossível o tráfego de pacotes a camada de transporte não teria o que fazer com eles explicam Carne (2004) e Tanenbaum (2003).

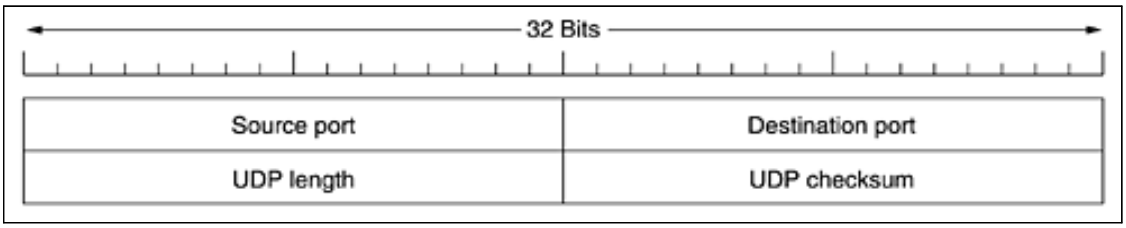

Figura 7 - O cabeçalho UDP
Fonte: Tanenbaum (2003, p. 434).

Segundo Tanenbaum (2003) é importante deixar claro algumas funcionalidades que o *UDP* não implementa, como controle de fluxo, controle de erros ou retransmissão de pacotes.

2.3.3 *Internet* – camada 2

Também chamada de camada de rede, consiste em todos os protocolos utilizados para que os dispositivos de origem e destino se encontrem, independente de sua localização física, já que os datagramas da camada de transporte são analisados para definir a rota que será utilizada. No presente trabalho o objetivo restringe-se a dar uma visão sucinta de quatro protocolos pertencentes a esta camada.

2.3.3.1 *Internet Protocol* (IP)

É o protocolo mais importante nesta camada, considerado o protocolo responsável pelo funcionamento da *internet*. *Internet Protocol*, ou simplesmente *IP* corresponde à camada de rede, que é responsável pelo envio de datagramas entre dispositivos, contendo cada datagrama as informações necessárias para que o mesmo seja entregue ao destino (DOSTÁLEK; KABELOVÁ, 2006; CARNE, 2004).

Carne (2004) explica um datagrama *IP* como sendo formado pela combinação da camada de transporte e o cabeçalho adicionado pelo protocolo *IP*, contendo os endereços de origem e de destino. Cada tipo de rede define sua unidade máxima de transmissão, *Maximum Transmit Unit* (*MTU*), que, em redes *ethernet*, tem por padrão o tamanho 1.500 *bytes*, sendo o protocolo *IP* que fragmenta os pacotes no emissor para adequar-se ao tipo da rede e monta os

[2] *Checksum* é um dado de tamanho fixo calculado a partir de um bloco arbitrário de dados digitais com a finalidade de detectar erros acidentais que possam ter sido introduzidas durante a sua transmissão ou armazenamento. Disponível em: <http://thefreedictionary.com/checksum>. Acesso em: 14 jun. 2011.

datagramas quando estes são recebidos pelo destinatário. Esse endereço é único na rede, seja *internet* ou redes internas, que permite aos dispositivos conseguir estabelecer a comunicação, ou seja, que os *hosts* comuniquem-se entre si.

Na Figura 8 os campos do cabeçalho *IP* são citados, dentre os quais se tem: Versão, que indica o uso da versão 4 ou 6 do *IP*; tipo de serviço, um campo que pode ser utilizado para priorizar o tráfego conforme o tipo de serviço; t*ime to live*, que define o tempo de vida de um pacote, para que não fique trafegando eternamente na rede; c*hecksum*, utilizado para a verificação apenas do cabeçalho *IP*; endereço *IP* de origem e destino, que identifica os *hosts* envolvidos na troca de informações, entre outros.

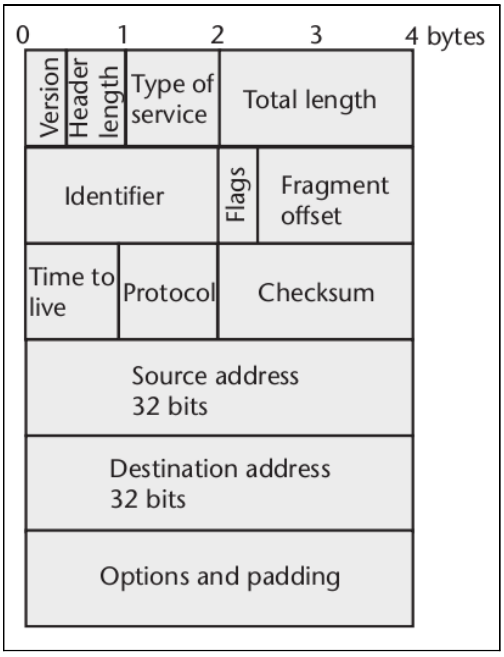

Figura 8 - Cabeçalho *Ipv4*
Fonte: Carne (2004, p. 16).

No presente trabalho o termo *IP* refere-se à versão 4 do protocolo.

2.3.3.2 *Address Resolution Protocol* (ARP)

Segundo Held (2003) e Carne (2004), o *ARP* resolve o endereço *IP* de um nó, que é configurado em alto nível, em seu endereço *MAC*, que fica gravado geralmente em um *chip* na placa de rede, em uma rede local, criando assim uma tabela que é gerada a partir de duas mensagens: *ARP request*, que envia *frames* via *broadcast* de nível *MAC*, e *ARP reply*, em que o endereço que responde ao *IP* solicitado responde com seu endereço *MAC*. Por meio dessa tabela e do endereço *MAC*, os pacotes podem ser transportados na *LAN*.

A Figura 9 ilustra um pacote *ARP*. O valor do campo hardware, definido para 1, significa *Ethernet*. O campo *Protocol* identifica o endereço do protocolo, em hexadecimal,

identificando o valor 0800 o uso de endereços *IP*. Os campos hardware *length* (*HLEN*) e *protocol length* (*PLEN*) definem o tamanho, em *bytes*, dos endereços a serem utilizados (hardware e protocolo). *Operation* indica *ARP request* (1) ou *ARP reply* (2). Os demais campos referem-se ao protocolo e ao hardware do emissor e ao hardware do destinatário.

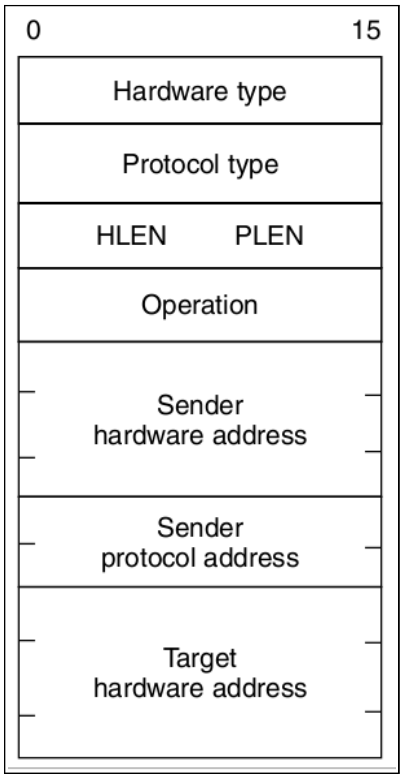

Figura 9 - Formato de um pacote ARP segundo
Fonte: Held (2003, p. 254).

2.3.3.3 *Internet Control Message Protocol* (ICMP)

O protocolo *ICMP* fornece um método para dispositivos *IP* trocarem informações sobre problemas com a rede que estejam impedindo a troca de pacotes. Mesmo que o protocolo *IP* não seja confiável e não garanta a entrega, é importante por ser uma maneira de informar ao remetente quando a entrega não é possível (HALL, 2000).

De acordo com Held (2003), uma mensagem *ICMP* é formada prefixando um cabeçalho *IP* à mensagem *ICMP*, sendo cada mensagem formada por quatro campos, como pode ser observado na Figura 10.

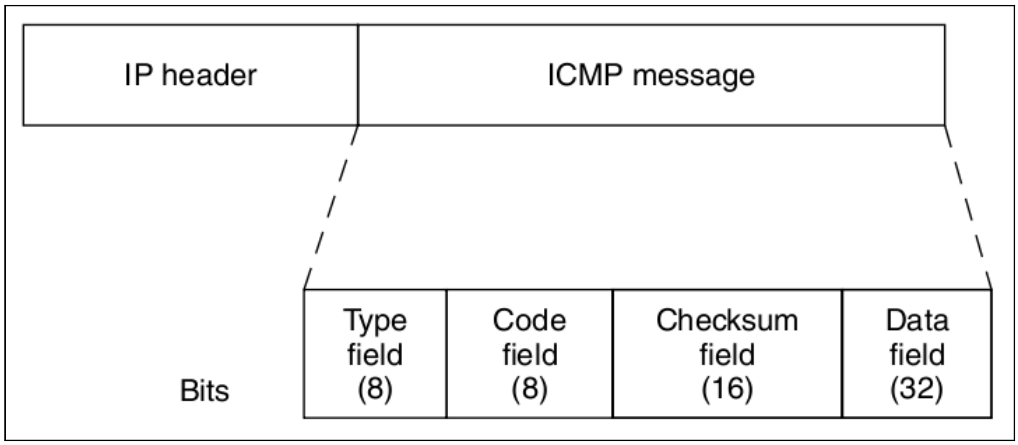

Figura 10 - Formação e composição de uma mensagem *ICMP*
Fonte: Held (2003, p. 250).

Segundo Tanenbaum (2003), a *internet* é acompanhada de perto por roteadores e, quando ocorre algo inesperado, o protocolo *ICMP* é utilizado para reportar os problemas. Este protocolo define cerca de uma dúzia de tipos de mensagens, como “destino não encontrado”, “tempo excedido”, entre outras. Hall (2000) divide essas mensagens em três famílias: *“Query (Reply)”* e *“Query (Request)”*, com quatro mensagens em cada família, e *“Error”* com cinco.

2.3.3.4 *Internet Group Management Protocol* (IGMP)

A necessidade de transferência de dados de forma simultânea para diversos nós em uma rede local evidenciou a necessidade de tráfego *IP multicast*[3], como é o caso de videoconferência, em que existe o envio de pacotes de um para muitos. Para que isso seja possível, o protocolo *IGMP* envia um único datagrama para os nodos locais e o faz por meio de roteadores aos nodos distantes interessados em receber o tráfego. Para que isso seja possível, o protocolo *IGMP* oferece um mecanismo para que os *hosts* informem seu interesse em receber o tráfego ou interromper o recebimento deste (CARNE, 2004).

A Figura 11 diferencia o envio de pacotes *unicast* e *multicast*. Enquanto o primeiro envia uma cópia para cada destinatário, o segundo encaminha apenas uma cópia para todos os membros do grupo.

[3] *Multicast* é a entrega de uma mensagem ou informações a um grupo de computadores de destino simultaneamente em uma única transmissão. Disponível em: <http://thefreedictionary.com/multicast>. Acesso em: 14 jun. 2011.

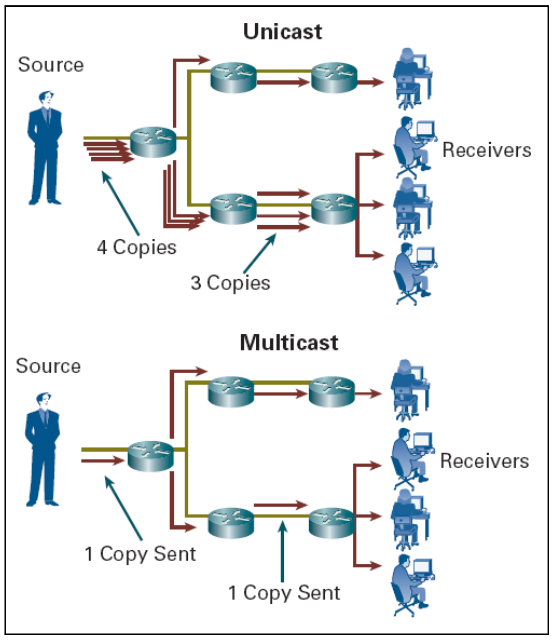

Figura 11 - Diferença entre o envio de pacotes unicast e multicast
Fonte: Doherty, Anderson e Maggiora (2008, p. 307).

O protocolo *IGMP* utiliza somente dois tipos de pacotes: consulta e resposta, cada um contendo algumas informações simples para controle. Sendo assim, é vagamente análogo ao protocolo ICMP (TANENBAUM, 2003).

2.3.4 Interface com a rede – camada 1

É a camada de mais baixo nível no modelo *TCP/IP*, estando intimamente ligada ao hardware através do *driver*[4] do dispositivo. Os datagramas recebidos por esta camada são convertidos para serem transmitidos utilizando o meio físico disponível, seja *wireless*, óptico, cobre, entre outros.

Segundo Palma e Prates (2000), esta camada baseia-se em três conjuntos de protocolos: redes W*ide Area Network (*WAN), redes *Local Area Network* (LAN) e utilizados em redes discadas.

2.4 Comparativo entre os modelos (OSI e TCP/IP)

Uma questão que normalmente surge é a correlação entre as normas ISO OSI e o protocolo *TCP/IP*. Segundo Dostálek e Kabelová (2006), ambos são divididos em camadas e em cada camada são definidos os protocolos utilizados. De maneira geral, os modelos são incompatíveis.

[4] *Driver* (de dispositivo): é um conjunto de rotinas que permitem ao sistema operacional acessar o periférico. Disponível em: <http://wiki.softwarelivre.org/bin/view/PCLivre/GlossariO>. Acesso em 17 jul. 2011.

Semelhante ao modelo OSI da ISO, no modelo *TCP/IP* os dados são passados para baixo na pilha no momento do envio e são levados até o topo quando do recebimento. (SPORTAK, 2004). Em cada camada são adicionadas informações do protocolo e de controle a fim de garantir a entrega dos dados no destino (HUNT, 2002). Esta adição de informações que ocorre em cada camada é chamada de encapsulamento, como retrata Figura 12:

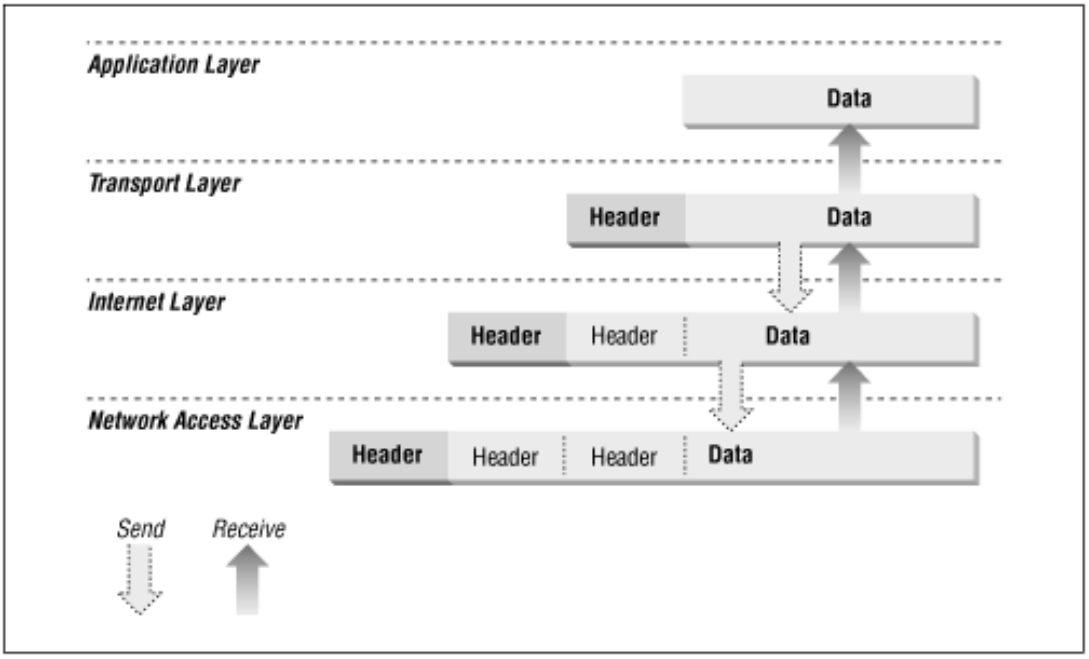

Figura 12 - Encapsulamento de dados
Fonte: Hunt (2002, p. 22).

A Figura 13 traça comparativo entre os dois modelos, conforme a maioria dos autores os trata, porém, é importante ficar claro que esses modelos não são compatíveis.

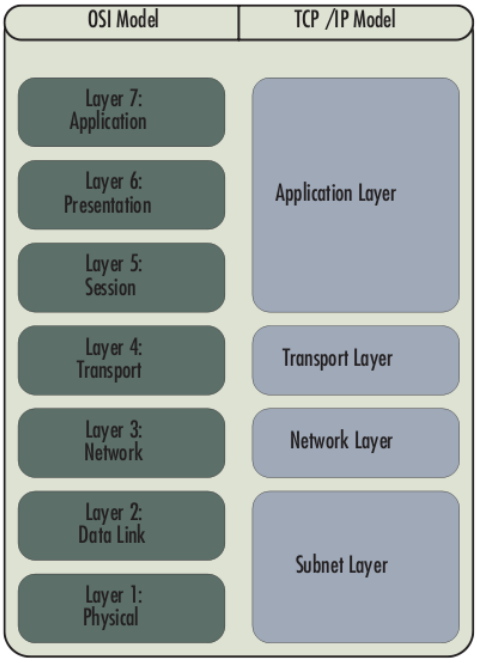

Figura 13 - Os modelos OSI e *TCP/IP*
Fonte: Noonan e Dubrawsky (2006, p. 46).

No próximo capítulo é dada enfase à segurança de redes. Nele são apresentados as principais ameças, riscos e vulnerabilidades encontradas em uma rede de computadores, assim como os tipos de ameaças e seus atacantes.

3 SEGURANÇA DE REDES

Segundo Harrington (2005), segurança de redes é um termo muito amplo, isso significa proteção dos dados que são armazenados ou dos que trafegam em uma rede de computadores contra a divulgação não autorizada, modificação acidental ou intencional. Sendo assim, neste capítulo são apresentadas definição e composição sobre riscos e ameças, assim como vulnerabilidades e ataques sofridos em uma rede de computadores.

3.1 Riscos e ameaças

Nakamura e Geus (2009) definem risco em segurança da informação como a probabilidade de um agente de ameaça explorar uma vulnerabilidade, comprometendo dados da confidencialidade, integridade e disponibilidade (CIA, *confidentiality, integrity, availability*). Teixeira et al. (1999) consideram que uma ameaça consiste em uma pessoa ou uma organização querer atentar contra a segurança da rede.

Observam Nakamura e Geus (2007) que, no momento que uma rede de dados passa a ser parte importante em uma organização, existem algumas considerações que devem ser feitas quanto aos riscos existentes.

- as informações que trafegam pela rede estão sujeitas a serem capturadas;
- os *e-mails* podem ser capturados, lidos, modificados e/ou falsificados;
- a *internet* deve ser considerada um ambiente hostil e portanto, não confidenciável;
- novas tecnologias significam novas vulnerabilidades;
- a interação entre diferentes ambientes resulta na multiplicação dos pontos vulneráveis;
- a segurança é complexa.

De acordo com Alcapan e Basar (2010), é essencial que se execute uma análise de riscos durante a fase inicial de um projeto de rede, pois é quando se identificam os bens que se pretende proteger contra as mais diversas ameaças, estas envolvendo desastres naturais, ataques *hackers*, erros acidentais de configuração, brechas na política de segurança, *spam*, negligência, entre outros. A Figura 14 ilustra alguns exemplos que caracterizam fatores de riscos.

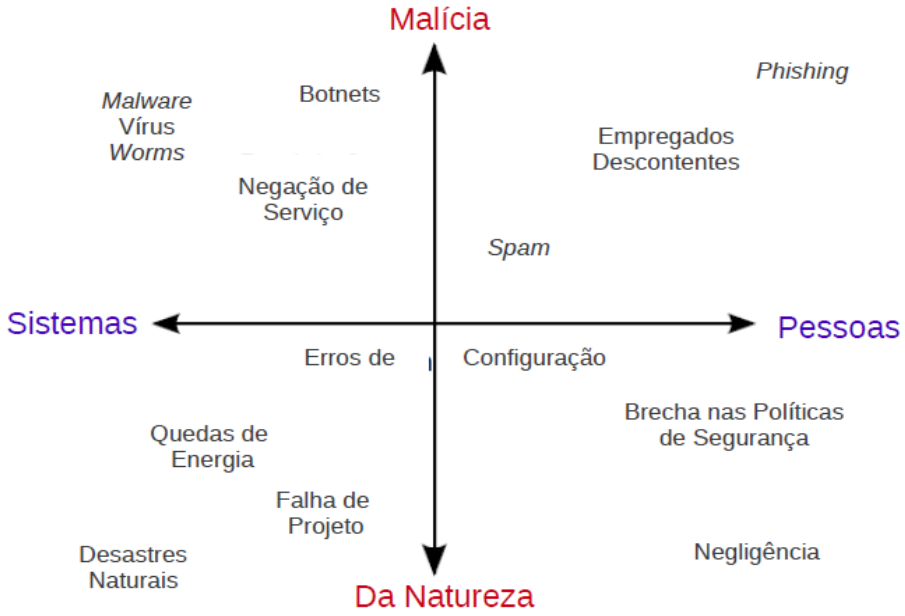

Figura 14 - Fatores de riscos: natureza *versus* malícia e sistemas versus indivíduos
Fonte: Do autor com base em Alpcan e Basar (2010).

A identificação das ameaças em uma rede de computadores é extremamente importante, pois elas fazem uso das vulnerabilidades para atingir um sistema de segurança. As ameaças e os riscos de segurança estão fortemente interligados, pois identificando-se os riscos podem-se identificar as ameaças (BHARDWAJ, 2007).

Usuários de sistemas computacionais geralmente não possuem conhecimento dos impactos e consequências de suas ações sobre a segurança da rede de dados. Muitas vezes as falhas de segurança não partem de uma ação mal intencionada, mas sim, são causadas pela ignorância ou por erro humano. Consequentemente inúmeros administradores de sistemas não mantêm uma base de treinamento contínuo, com orientações e conscientização da forma de uso adequado dos recursos de tecnologia da informação (VACCA, 2010).

As ameaças geralmente incluem incidentes envolvendo vandalismo, roubo de equipamento ou dados, intrusão, entre outras situações, variando de uma organização para outra (BHARDWAJ, 2007). Inúmeras vezes a principal ameaça à segurança da rede de computadores parte de dentro da própria empresa, origina-se dos próprios usuários, promovendo possíveis vulnerabilidades que podem ser exploradas pelos atacantes (BRENTON; HUNT, 2001).

3.2 Vulnerabilidades e ataques

Segundo Harrington (2005), a *internet* tem sido uma maravilha e um problema para as empresas que possuem redes de computadores. Com ela, empregados e clientes passaram a ter

acesso a uma organização por meio de seu site. Em contrapartida, com esse novo acesso surgiram enormes problemas causados por indivíduos que tentam burlar a segurança da rede ilegalmente, por meio das vulnerabilidades, sendo uma vulnerabilidade, em termos de segurança, é um ponto fraco de projeto de rede, que pode ser explorado com más intenções (TEXEIRA et al., 1999).

Segundo Alpcan e Basar (2010), ataque pode ser definido como uma ou mais tentativas de comprometer a confidencialidade, integridade e disponibilidade da rede de dados para obter um controle parcial ou total de dispositivos que se comunicam em rede.

Ataques à rede fazem uso de vários métodos para explorar as vulnerabilidades dos sistemas de rede e de seus usuários. Essas vulnerabilidades envolvem a presença de falhas ou *bugs* encontrados nos mais diferentes sistemas, sistemas operacionais, aplicativos sem receber atualizações, correções contra vírus e *spywares*, utilização de senhas que podem ser quebradas facilmente, entre outros (ALPCAN; BASAR, 2010).

Segundo Smith e Marchesini (2008), as principais falhas de segurança são causadas pela complexidade nas aplicações desenvolvidas, isso devido ao avanço computacional, que proporcionou que os códigos fossem cada vez mais amplos e complexos, alguns desses, executados em dispositivos de rede, podem conter de centenas a milhões de linhas. Com a alta complexidade dos códigos somadas às pressões econômicas sofridas durante o desenvolvimento do projeto, alavancou-se o número de vulnerabilidades a serem exploradas por *crackers*.

Para um observador casual manter uma rede segura parece ser simples, podendo imaginar que, para tal, seria necessário simplesmente criar senhas mais longas, e usufruir de antivírus. Mas não existe uma solução simples para proteger uma rede de computadores. Tal fato percebe-se devido a diferentes tipos de ataques sofridos pelos usuários em seu cotidiano. Em contrapartida, são gastos milhões de dólares anualmente em segurança de informação, não diminuindo a taxa de sucesso de ataques.

A maioria dos ataques não é considerada aleatória. O invasor acredita que há algo a ganhar em atacar a segurança de um sistema. Esses ataques podem ser oriundos de fontes internas ou externas, ou seja, pode ser originado dentro da própria instituição, por algum usuário desavisado, desatento, ou mesmo mal intencionado, ou vinda de fora, por um ex-funcionário insatisfeito com a demissão ou por algum atacante desconhecido (BRENTON; HUNT, 2001).

Algumas invasões são efetuadas simplesmente para que seja percebido que um determinado *site* ou *host* foi atacado, desconfigurando o *site* com textos ou imagens

grosseiras. Outros são mais maliciosos, buscando extrair informações importantes, implantando parasitas que desviam todos os dados possíveis que estão trafegando na rede. Em outros casos a implantação do código acontece cuidadosamente, quebrando senhas, ou até mesmo clonando um *site* inteiro, fazendo com que os usuários sejam direcionados para um local desconhecido (CIAMPA, 2009).

Conforme Doherty, Anderson e Maggiora (2008), os ataques à rede podem ser classificados como ativos ou passivos:

- ataques ativos incluem injeção de arquivos maliciosos, alterações de dados, ou congestionamento da rede, tendo a intenção de causar danos ou prejuízos a vítima. Esse tipo de ataque somente é possível ser identificado através dos rastros deixados pelo invasor;
- ataques passivos na maioria das vezes não tem intenção de prejudicar o funcionamento da rede, mas sim obter informações ou dados sigilosos, deste modo, torna-se difícil detectá-lo.

3.3 Tipos de ataques

A seguir, são apresentados os principais tipos de ataques sofridos em uma rede de computadores, suas definições, as dificuldades para conseguir contê-los e a caraterização de cada atacante. Por final, são apresentadas algumas implementações de defesas que uma rede de computadores deve apresentar para se tornar mais segura.

3.3.1 Malware

Harrington (2005) resume o termo *malware* como softwares maliciosos: vírus, *worms* e cavalos de tróia. Códigos maliciosos, ou *malwares*, são projetados, segundo Bhardwaj (2007), com a finalidade de infiltrar-se no computador do usuário, sem que este conheça ou permita. As três principais categorias de *malware* são: programas que não visam a ganhos comerciais; programas que têm por objetivo ocultar a identidade dos invasores; e softwares que visam ao roubo de dados e consequente ganho comercial.

A palavra *malware* vem do termo mal, de maldade. Na definição de Smith e Marchesini (2007), muitos desenvolvem este tipo de aplicativo com o intuito de brincadeira, outros como forma de vandalismo, de protesto ou mesmo de roubo. Observam os autores ainda que *malware* deve ser visto como um software que faz com que, intencionalmente, o sistema passe a não se comportar de maneira aceita como correta.

3.3.2 Força bruta

Este tipo de ataque caracteriza-se por simples tentativa de tentar acertar a senha testando todas as possibilidades, de forma exaustiva, ou seja, o atacante não tenta invadir o sistema ou parar o serviço, apenas confia que tentando todas as possibilidades terá sucesso (BRENTON; HUNT, 2001). Levando-se em consideração que serão testadas todas as possibilidades, nenhum algoritmo de criptografia é imune a este ataque, porém, o atacante considera o que pode ganhar com o tempo necessário para quebrar a criptografia, já que, dependendo da senha utilizada, o tempo para que um ataque de força bruta tenha sucesso pode ser de dias, semanas, anos ou até mesmo ser considerado computacionalmente incalculável com o recurso computacional existente.

Tendo em vista a relação entre tamanho da senha e tempo necessário para obter êxito, Brenton e Hunt (2001), apresentam a Tabela 3, na qual é possível observar que, com o aumento no tamanho da chave, o número de combinações possíveis aumenta de maneira exponencial; e, consequentemente, o tempo e os recursos computacionais exigidos para decifrar a senha são cada vez maiores.

Tabela 3 - Métodos de criptografia

Criptografia	*Bits* na chave	Número de possibilidades
Netscape	40	$1.1x10^{6}$
Data Encryption Standard (DES)	56	$72.1x10^{6}$
Triple DES (2 keys)	112	$5.2x10^{33}$
International Data Encryption Algorithm (IDEA)	128	$3.4x10^{38}$
Rivest Cipher (RC4)	128	$3.4x10^{38}$
Triple DES (3 keys)	168	$3.7x10^{50}$
Blowfish	Up to 448	
Advanced Encryption Standard (AES)	128, 192, 256	$3.4x10^{38}$

Fonte: Brenton e Hunt (2001, p. 196).

Segundo Harrington (2005), geralmente os ataques são feitos utilizando um nome de usuário conhecido. Esses ataques são mais facilmente percebidos pelos administradores, já que as tentativas de *login* são registradas.

3.3.3 Ataques *Denial of Service* (*DoS*)

Segundo Vyncke e Paggen (2008) e Bhardwaj (2007), um ataque *DoS* é caracterizado como uma tentativa explícita de impedir que os usuários que devem ter acesso a determinado serviço não o consigam acessar. As técnicas mais utilizadas para alcançar os objetivos são: tentativas de inundar a rede e impedir o tráfego legítimo, tentativas de derrubar um servidor

pelo envio de inúmeras requisições até que ele trave por não conseguir responder a todos, envio de pacotes mal formados para servidores ou serviços; e tentativas de interromper um serviço como um todo ou para determinado usuário.

Geralmente os ataques de *DoS*, salienta Bhardwaj (2007), não são capazes de causar falhas em todos os serviços da rede, já que são ataques direcionados a serviços, como o *Domain Name System* (*DNS*), em que os usuários não conseguiriam utilizar o serviço atacado. Nesse sentido, os ataques de *DoS* podem resultar em consumo exagerado de recursos da rede ou de um servidor, em modificação ou alteração de configuração em dispositivos ou falhas em serviços como bases de dados ou servidores *web*.

Harrington (2005) explica que ataques de *DoS* não podem ser evitados, sendo apenas possível detectar este tipo de ataque quando ele estiver em curso e tomar as medidas, para que os ataques não surtam efeito.

Uma variação dos ataques *DoS* são os *Distributed Denial of Service Attack* (*DDoS*), em que, conforme Vyncke e Paggen (2008), vários atacantes disparam tentativas contra um mesmo alvo, geralmente um serviço válido, com a tentativa de executar mais requisições que possam ser atendidas. Os atacantes são máquinas *zombies*, em que os usuários geralmente não têm conhecimento de que seu *host* esteja participando do ataque. A Figura 15 ilustra os agentes de um ataque *DDoS*.

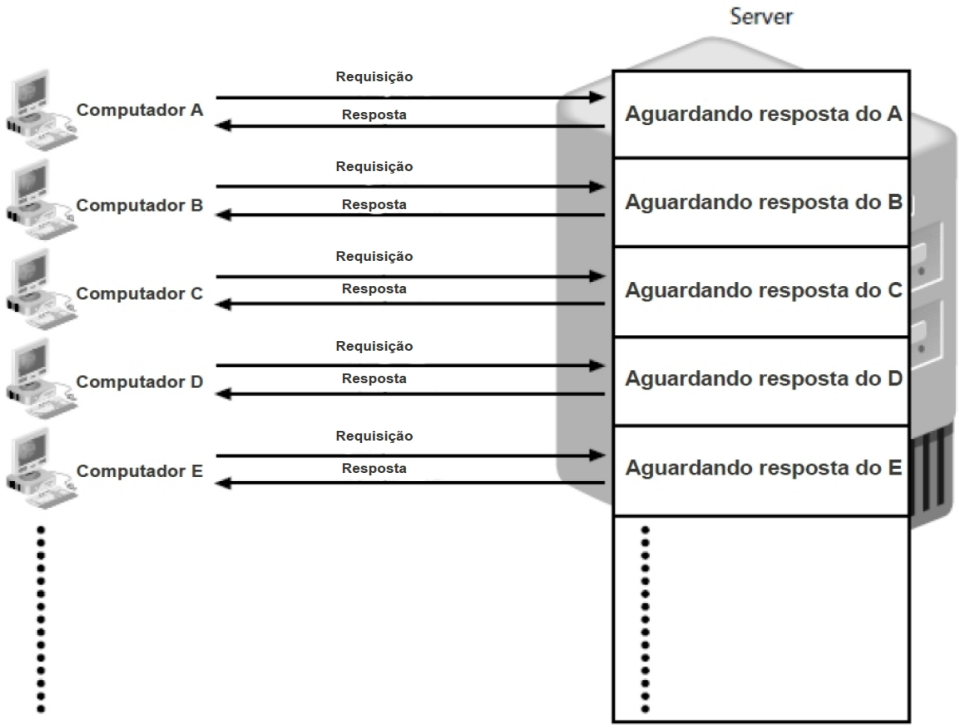

Figura 15 - Formato de um ataque DDoS
Fonte: Do autor com base em Ciampa (2009).

3.3.4 Spoofing

Segundo Ciampa (2009), o termo *spoofing* é a representação, ou seja, o atacante está fingindo ser alguém ou outra coisa, apresentando informações falsas. Para Harrington (2005), um ataque *spoofing* envolve a falsificação do endereço de origem. É o ato de usar uma máquina para representar o papel de outra. A maioria das aplicações e ferramentas no *Unix*[5] baseiam-se na autenticação do IP de origem.

Há uma variedade de ataques diferentes que usam *spoofing*, assinala Ciampa (2009), como:

- como a maioria dos sistemas de rede mantêm registros de atividade do usuário, um invasor pode falsificar seu endereço para que suas ações maliciosas seriam atribuídas a um usuário válido;
- um invasor pode falsificar o seu endereço de rede utilizando um endereço confiável e conhecido, a fim de enganar o computador de destino;
- pode-se exibir uma tela de login fictícia pedindo o nome de usuário e senha, permitindo que o invasor capture as credenciais válidas de um usuário.

A Figura 16 ilustra um ataque do tipo *spoofing*.

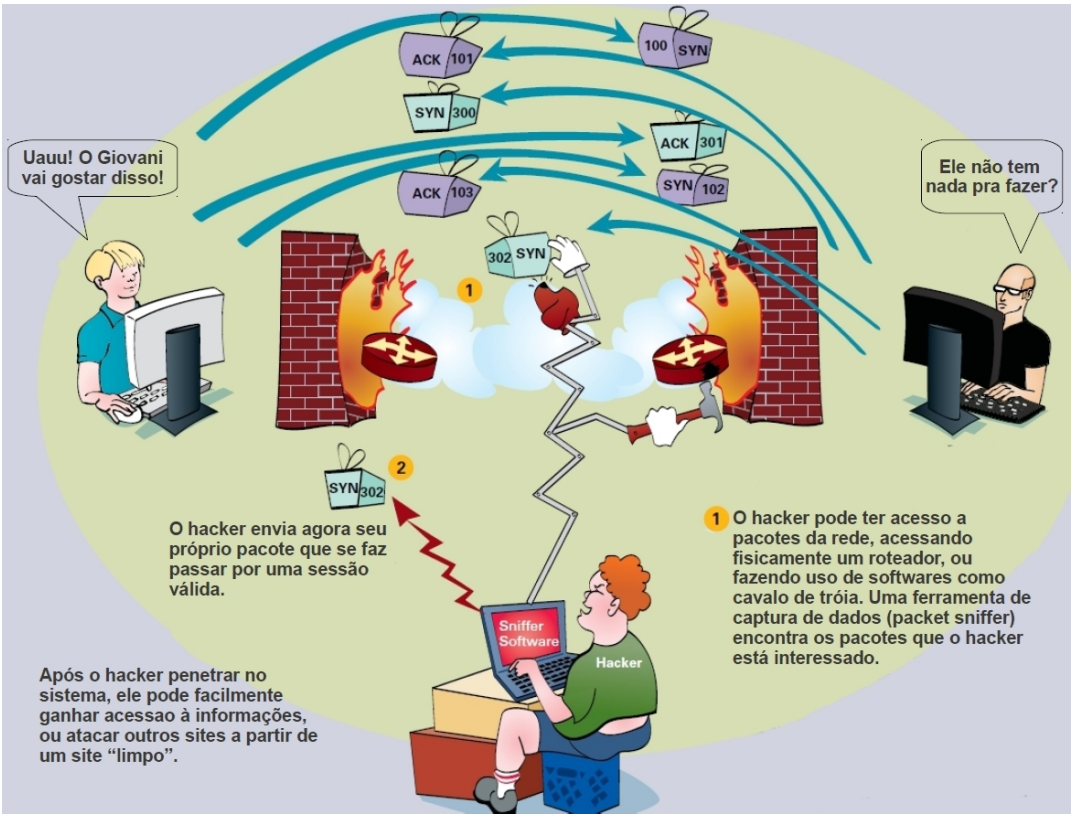

Figura 16 - Exemplo de um ataque do tipo *spoofing*
Fonte: Do autor com base em Doherty, Anderson e Maggiora (2008).

[5] *Unix* é um sistema operacional, portátil, multitarefa e multiusuário. Disponível em <http://unix.org>. Acesso em 20 jul. 2011.

3.3.5 Engenharia social

A engenharia social é o processo de obtenção de informações sobre algo ou alguém baseado na confiança, ou seja, o atacante busca por meio de diversos meios obter informações sigilosas de pessoas que as conhecem, fazendo a vítima confiar no engenheiro social. Este ataque também envolve uma interação entre o atacante e o atacado, seja por meio de telefone, um anexo de *e-mail* solicitando informações ou sites de bate papo nos quais a vítima, confiando no atacante, lhe concede informações (BHARDWAJ, 2007).

O ataque de engenharia social, segundo Bhardwaj (2007) e Smith e Marchesini (2007), é o mais difícil de ser bloqueado, sendo o único meio de mitigá-lo o treinamento e o convencimento dos usuários sobre a importância de manter os dados e as informações em sigilo.

Esta técnica, de acordo com Smith e Marchesini (2007), é umas das mais antigas, existindo casos de atacantes acionarem serviços de suporte fazendo-se passar por usuários legítimos. Tentativas utilizando mulheres teriam capacidade maior de coagir os técnicos, que geralmente são do sexo masculino e atacantes que vasculham o lixo de empresas ou de colaboradores buscando informações jogadas no lixo que podem ser úteis.

3.3.6 Buffer Overflow

Um ataque do tipo *Buffer Overflow* tira proveito de um erro de programação de um programa, aplicativo ou sistema. O *hacker* pode inserir seus próprios códigos em um programa e, a partir daí, assumir o controle de um sistema de destino. Por ser resultado de um erro de programação, é quase impossível para um engenheiro de rede detectar condições de *Buffer Overflow*. Eles geralmente são detectados por *hackers* ou pelo fornecedor do software (HARRINGTON, 2005).

Buffer Overflow, de acordo com Bhardwaj (2007) é uma condição no sistema que causa uma falha em sua segurança ou um uso de memória que cause falha no sistema. Um atacante pode lançar esse tipo de ataque escrevendo código malicioso especificamente destinado a usar toda a memória do sistema-alvo. Outro motivo para este tipo de ataque é a escolha de uma linguagem de programação que não seja capaz de proteger-se contra o estouro de memória.

Quando um programador escreve uma aplicação, esta deve criar *pools* de memória, referidas como os amortecedores, de modo a aceitar entrada de usuários ou outras aplicações. Por exemplo, uma aplicação de login deve alocar espaço de memória a fim de permitir que o

usuário insira um nome de logon e senha. A fim de alocar espaço de memória para esta informação, o programador deve fazer uma suposição sobre quantos dados serão recebidos para cada variável. Por exemplo, o programador pode decidir que os usuários não precisam digitar um nome de login maior do que 16 caracteres e que senhas nunca terão mais do que 10 caracteres. Um *Buffer Overflow* ocorre quando mais dados são recebidos por um processo do que o programador esperava, e não existe contingência para quando o processo tem de lidar com uma quantidade excessiva de dados (BRENTON; HUNT, 2001).

Segundo Malik (2002), são os ataques mais difundidos na *internet*, representando quase metade de todas as vulnerabilidades encontradas. Em Sistemas Operacionais, geralmente escritos na linguagem de programação C, coloca Malik (2002) ocorrem quando uma rotina escreve quantidade de dados muito maior do que a capacidade de dados. A Figura 17 ilustra um ataque de *Buffer Overflow*.

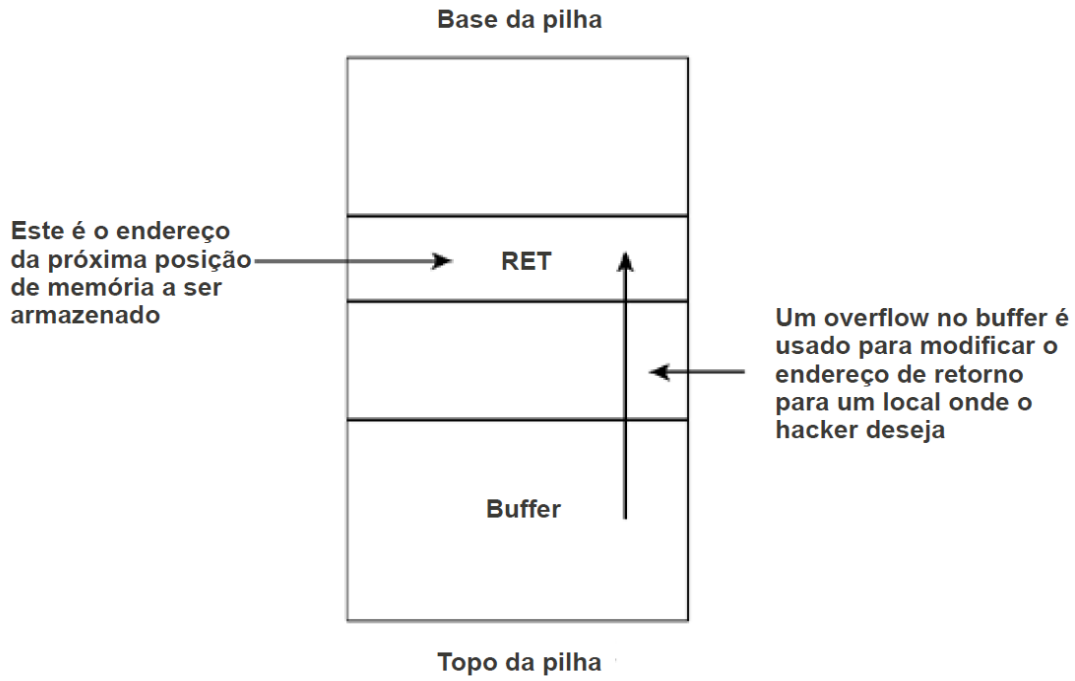

Figura 17 - Exemplo de um ataque do tipo *Buffer Overflow*
Fonte: Do autor com base em Malik (2002).

3.3.7 Repetição

De acordo com Ciampa (2009), um ataque de repetição é semelhante a um ataque passivo *man-in-the-middle*. Considerando que um ataque passivo envia a transmissão imediatamente, um ataque de repetição faz uma cópia da transmissão antes de enviá-lo para o destinatário.

Ainda, conforme Ciampa (2009) uma repetição simples *man-in-the-middle* implicaria na captura das credenciais de login entre o computador cliente e o servidor. Uma vez que a sessão tenha terminado, o *man-in-the-middle* tentaria repetitivamente as credenciais capturadas. Um ataque mais sofisticado aproveita as comunicações entre um dispositivo de rede e um servidor. Mensagens de administração que contêm solicitações de rede específicas são frequentemente enviadas entre um dispositivo de rede e um servidor. Quando o servidor recebe a mensagem, ele responde com outra mensagem administrativa para o remetente.

Cada uma dessas transmissões é criptografada para evitar que um atacante possa ver o conteúdo, e também contém um código que indica se ele foi adulterado. O servidor lê o código e, se reconhecer que a mensagem foi adulterada, não responde (VACCA, 2010).

Usando um ataque de repetição, um atacante pode capturar uma mensagem transmitida a partir do dispositivo de rede ao servidor. Mais tarde ele poderia enviar a mensagem original para o servidor e o servidor poderia responder pensando ter sido encaminhada pelo dispositivo válido. Agora, uma relação de confiança foi estabelecida entre o atacante e o servidor, pois o atacante sabe que vai receber uma resposta do servidor cada vez que ele envia uma mensagem válida. Sendo assim, ele pode usar esse conhecimento como uma ferramenta valiosa para alterar o conteúdo da mensagem captada. Se ele finalmente conseguir fazer a modificação, o servidor irá responder informando o atacante que ele teve sucesso na transferência da mensagem (TODOROV, 2007).

3.4 Dificuldades na defesa contra ataques

Segundo Ciampa (2009), existem várias dificuldades para conter um ataque, dentre elas citam-se:

- velocidade nos ataques: com várias ferramentas disponíveis, os atacantes podem rapidamente encontrar pontos fracos ou brechas no sistema pelos quais podem lançar ataques rápidos e sem precedentes;
- ataques mais sofisticados: devido à constante sofisticação dos ataques, eles estão se tornando cada vez mais complicados de serem interceptados;
- ataques distribuídos: para atacar um computador, os invasores podem fazer uso de várias fontes, o que tornaria impossível barrar o ataque por meio de identificação de um único *host;*

- confundir o usuário: nos atuais ataques tem-se pouca ou nenhuma informação para corrigir o problema, tornando assim complicado tomar uma decisão em contrapartida do ocorrido;
- simplicidade nas ferramentas de ataques: antigamente o atacante necessitava de um bom conhecimento técnico sobre a ferramenta que iria utilizar para efetuar o ataque. Hoje em dia essas ferramentas estão disponíveis gratuitamente na *internet* e não requerem nenhum conhecimento técnico para serem executadas.

A Figura 18 apresenta um comparativo dos últimos anos entre a sofisticação dos ataques *versus* a dificuldade de intrusão.

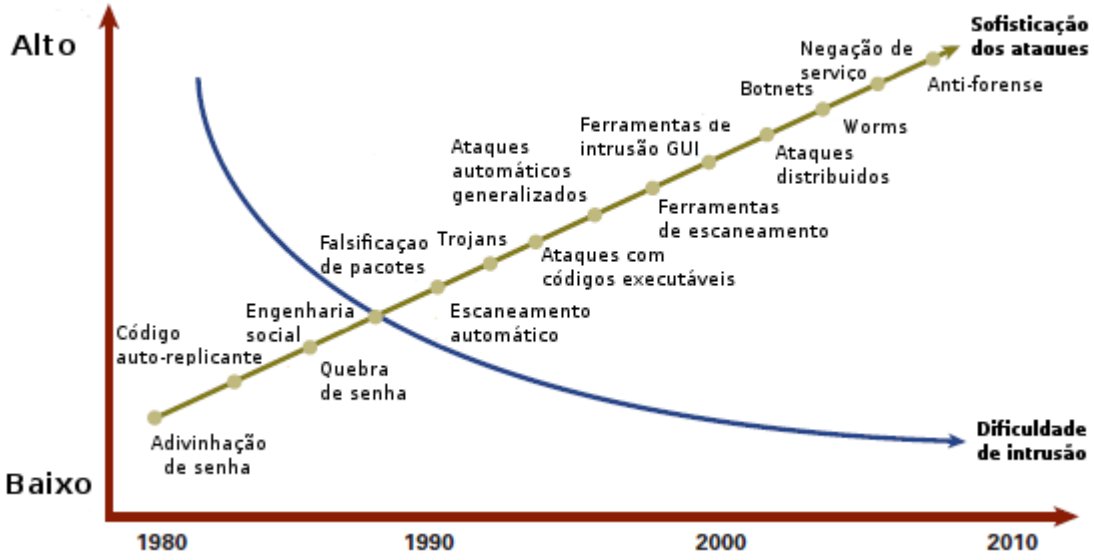

Figura 18 - Crescente sofisticação nas ferramentas de ataques
Fonte: Do autor com base em Doherty, Anderson e Maggiora (2008).

3.5 Quem são os atacantes

Com a crescente disponibilização de serviços via *internet*, as empresas ganharam em agilidade, porém abriram uma porta para acessos indevidos, que pode ser explorada por diversos grupos de atacantes (HARRINGTON, 2005).

Existem vários tipos de pessoas por trás dos ataques, podendo elas serem separadas em diferentes grupos. Estes incluem *hackers*, *crackers*, *script kiddies*, espiões, funcionários, criminosos e ciberterroristas, entre outros (CIAMPA, 2009).

3.5.1 *Hackers* e *Crackers*

Embora exista dificuldade em diferenciar os dois termos, que geralmente são utilizados para definir pessoas que invadem sistemas, o termo *hacker* conceitua pessoas que são bons programadores, que conhecem os sistemas, estudam com a finalidade do conhecimento e domínio da tecnologia, enquanto os *crackers* são realmente as pessoas que causam problemas de segurança por meio de atividades ilegais. O que mais desafia e motiva esses atacantes são declarações, geralmente de áreas comerciais, de que seus sistemas são 100% seguros (HARRINGTON, 2005).

Ciampa (2009) salienta que, por lei, todos os ataques são considerados ilegais, acreditando, alguns *hackers* que é ético invadir um *host*, desde que não afete sua integridade, sem cometer roubo, vandalismo, ou qualquer quebra de confidencialidade. Esses *hackers* afirmam que sua motivação é melhorar a segurança buscando brechas no sistema para que estas possam ser corrigidas.

3.5.2 White Hat Hackers

Este grupo considera-se do bem. Eles invadem um sistema, não para benefício pessoal, mas sim com a intenção de encontrar algum problema, seja uma rede vulnerável, um sistema mal configurado, um *bug* de software ou qualquer aspecto que possa prejudicar. Caso sejam encontradas falhas, eles relatam o ocorrido para o administrador da rede, ou para o responsável pelo software ou hardware (CIAMPA, 2009).

Os *White Hat Hackers* também são contratados por empresas para testar as defesas da rede. A principal característica deste grupo é estar bem informado sobre vulnerabilidades e consertos, assim como ter alta capacidade de programação. Eles geralmente escrevem suas próprias ferramentas de *cracking* (BRENTON; HUNT, 2001).

3.5.3 Espiões

Os espiões não atacam um *host* aleatoriamente. São indivíduos contratados para atacar um computador ou um sistema específico com o intuito de obter informações importantes ao seu contratante. Diferente dos *hackers* e dos *script kiddies*, os espiões procuram não deixar rastros e não chamar atenção as suas ações (HARRINGTON, 2005).

3.5.4 Empregados

Uma das maiores ameaças para as empresas quando o assunto é segurança da informação, podem originar-se de uma fonte muito improvável, o seu próprio funcionário, isto porque ele geralmente conhece mais da rede, dos sistemas e da segurança que os estranhos. Ele pode querer mostrar para a sociedade uma brecha de segurança, ou em outros casos, funcionários descontentes podem ter a intenção de retaliar a empresa. Em alguns casos, eles podem ser motivados por dinheiro, podendo um concorrente abordá-los e oferecer-lhe dinheiro em troca de informações (HARRINGTON, 2005). As principais ameaças vindas dos empregados são:

- empregados utilizando técnicas *hacker* para aumentar seu nível de acesso, permitindo o acesso e divulgação de segredos comerciais, roubar dinheiro, entre outros;
- empregados que divulgam os dados aos quais, legitimamente, têm acesso com o intuito de obter ganhos financeiros;
- familiares que, em momentos de visita, ganham acesso aos computadores e sistemas da empresa;
- pessoal que consegue acesso físico ao *datacenter*[6] e tem a possibilidade de causar algum dano;
- antigos funcionários que podem vingar-se em ataques físicos ou por meio de técnicas *hacking*.

Além das ameaças intencionais, Harrington (2005) cita as ameaças sem intenção que podem ser causadas pelos empregados, como:

- ser vítima de um ataque de engenharia social, ajudando um atacante a ganhar acesso não autorizado à rede ou aos sistemas;
- involuntariamente revelar dados confidenciais;
- fisicamente danificar ou causar falhas em equipamentos que resultem em perda de dados ou indisponibilidade de acesso;
- introduzir ou modificar dados com valores inconsistentes ou errados, ou, acidentalmente, apagar dados.

[6] Um *data center* é um mecanismo ou ambiente utilizado para abrigar os sistemas computacionais e componentes associados: telecomunicações e sistemas de armazenamento. Geralmente inclui fontes de alimentação redundantes ou de *backup*, comunicação redundante, ligações de dados, controles ambientais (ar-condicionado, combate a incêndios, etc.), e dispositivos de segurança especiais. (The Free Dictionary. Data center. ©2009. Disponível em: <http://encyclopedia.thefreedictionary.com/data+center>. Acesso em: 01 nov. 2011).

Essas ameaças somente podem ser tratadas e evitadas com educação aos usuários, já que muitos escrevem senhas em monitores, em cadernos, permitem o acesso ao computador da empresa, entre outros.

3.5.5 Ciberterroristas

Os ciberterroristas são *hackers* motivados por crenças políticas, religiosas ou filosóficas, que atacam posições opostas as suas ideologias e ensinamentos. São considerados os atacantes mais temidos, pois é quase impossível prever quando ou onde um ataque pode ocorrer. Seus objetivos podem incluir um pequeno grupo de computadores ou redes que podem afetar o maior número possível de usuários, tais como computadores que controlam uma rede de energia elétrica ou usinas nucleares de um país, provocando um colapso geral. Um ataque isolado pode causar um apagão afetando dezenas de milhões de pessoas (CIAMPA, 2009).

3.5.6 Black Hat Hackers

É considerado o grupo mais perigoso, já que são pessoas preparadas para agir, motivadas pela ganância ou por grande vontade de causar danos a terceiros. Criam suas próprias ferramentas e são muito cuidadosas para não deixar rastros (BRENTON; HUNT, 2001).

3.6 Etapas de um ataque

Atualmente existem vários tipos de ataques, estes sendo compostos por cinco etapas, conforme visto na Figura 19, e descrito a seguir (CIAMPA, 2009).

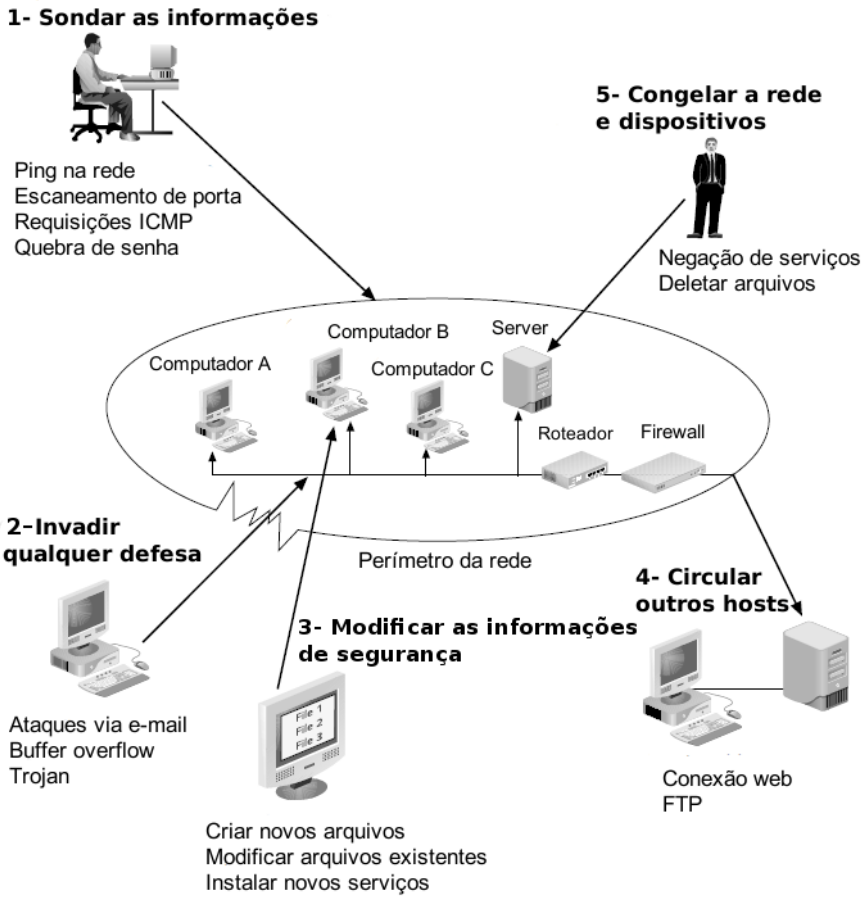

Figura 19 - Etapas de um ataque
Fonte: Do autor com base em Ciampa (2009).

- sondagem das informações: é a primeira etapa de um ataque, consiste em sondar informações que possam ser usadas nos próximos ataques, tais como: tipo de hardware utilizado, versão de software ou *firmware*[7] e até mesmo informações pessoais sobre os usuários. Esta sondagem inclui varreduras de *ping*[8] para determinar se um *host* está respondendo na rede, varreduras de portas que podem estar abertas e adivinhação de senha por força bruta;
- invasão de qualquer defesa: após a identificação de portas e o recolhimento de informações, o próximo passo é intensificar o ataque com a finalidade de quebrar a defesa, sendo este efetuado de várias formas, como, por exemplo, a manipulação ou a quebra de senhas;

[7] *Firmware* é um programa de computador que está incorporado em um dispositivo de hardware, por exemplo, um microcontrolador. Também pode ser fornecido em ROM flash ou como um arquivo de imagem binária que pode ser carregado para o hardware existente por um usuário. (The Free Dictionary. Data center. ©2009. Disponível em: <http://encyclopedia.thefreedictionary.com/firmware>. Acesso em: 01 nov. 2011).

[8] Ping (*packet inter-net groper*) – um programa usado para testar a conectividade da rede. O *ping* envia uma mensagem ICMP *Echo Request* para um destino e relata se ele recebe uma resposta ICMP *Reply* como esperado. COMER, Douglas E. **Redes de computadores e Internet.** 4 ed. São Paulo: Bookman. 2004.

- modificação das informações de segurança: logo após invadir um sistema, o próximo passo é modificar as informações de segurança, permitindo assim que o acesso seguinte seja feito com maior facilidade;
- apropriação de outros *hosts* ou sistemas: assim que o hospedeiro tenha sido comprometido, utiliza-se este para atacar outros computadores da rede;
- congelamento das redes e dispositivos: se o invasor quiser, ele pode trabalhar de forma maliciosa, infectando o computador, danificando dados, roubando arquivos valiosos ou realizando ataques de negação de serviço.

3.7 Defesas contra ataques

Um contexto ideal seria se qualquer indivíduo que utiliza um sistema de rede estar ciente das questões de segurança e ter a capacidade de tomar medidas defensivas, mas, na realidade, a tarefa de proteger redes e sistemas recai sobre os ombros dos administradores de sistema. Estes muitas vezes não são apenas responsáveis pela configuração e pelo acompanhamento das redes contra ataques, mas também ter um papel ativo na execução das políticas de segurança formais e informais e educar os usuários sobre possíveis vulnerabilidades (ALPCAN; BASAR, 2010).

Ainda, de acordo com esses autores um dos principais problemas relacionados com a segurança da rede do ponto de vista da defesa é a falta de investimentos que, em parte, decorre da dificuldade de quantificar o valor agregado pela segurança. Esta falta de quantificação, naturalmente afeta o processo decisório de investimentos, podendo levar a um “teatro de segurança”. No decorrer dos anos, esse impasse está mudando, as grandes organizações começaram a perceber a importância da segurança da rede de dados depois que elas se tornaram uma parte importante do negócio. Consequentemente, um grade número de empresas de segurança têm surgido, atuando na prestação de serviços de segurança, tanto para os indivíduos como para as organizações. Assim, a utilização de *firewalls*, atualização de sistemas operacionais, implementação de programas antivírus tornaram-se comuns.

Para Ciampa (2009), embora tenha aumentado a preocupação com a defesa da rede e o número de serviços prestados na área, as defesas devem ser baseadas em cinco princípios fundamentais de segurança: proteção por camadas, limitação, diversidade, obscuridade e simplicidade.

3.7.1 Camadas

Segundo Ciampa (2009), a abordagem em camadas tem a vantagem de criar uma barreira de defesas múltiplas que possam ser coordenadas para frustrar uma variedade de ataques. A segurança da informação também deve ser criada em camadas, pois um mecanismo de defesa simples pode ser relativamente fácil de ser atacado. Em vez disso, um sistema de segurança protegido por camadas torna improvável que um invasor tenha as ferramentas e habilidades para romper todas as camadas de defesas. Sendo assim, a abordagem em camadas pode ser útil para resistir a uma variedade de ataques, fornecendo uma proteção mais abrangente.

3.7.2 Limitação

Quando se limita o acesso à informação, consequentemente tem-se uma redução das ameaças sofridas. Limitar é garantir que somente pessoas autorizadas tenham acesso aos dados. Além disso, a quantidade de acesso concedido a alguém deve ser limitada ao que a pessoa precisa saber (CIAMPA, 2009).

Conforme Harrington (2005), algumas maneiras de limitar o acesso são baseadas em tecnologia (como a atribuição de permissões de arquivo para que um usuário só possa ler, e não o modificar), enquanto outros são processuais (proíbem um funcionário de remover um documento necessário para funcionamento do sistema).

3.7.3 Diversidade

A diversidade está estreitamente relacionada com a proteção em camadas. Assim, como é importante proteger os dados com camadas, também deve-se ter uma diversidade delas, de modo que os atacantes, ao penetrarem em uma camada, não possam usar as mesmas ferramentas para atingir as camadas seguintes. Sendo assim, ao ser violada uma camada, não se compromete todo o sistema (CIAMPA, 2009).

3.7.4 Obscuridade

Um exemplo de obscuridade seria não revelar o tipo do computador, sistema operacional, software e conexão de rede utilizados por um *host.* Com essas informações ocultas fica muito mais difícil a um atacante descobrir os pontos fracos do sistema. Inclusive,

em muitos casos, quando um atacante não consegue essas informações, ele parte para outro computador no qual a informação é facilmente encontrada (CIAMPA, 2009).

3.7.5 Simplicidade

Os ataques podem vir de várias fontes e de muitas maneiras. Na segurança da informação, por sua natureza complexa, onde algo mais complexo se torna mais difícil de entender. Além disso, os sistemas complexos permitem muitas oportunidades para que algo dê errado, por ser difícil compreendê-los, solucionar problemas e sentir-se seguro. Em suma, manter um sistema simples para o administrador e complexo para o atacante parece ser difícil, mas, se atingido, gera um grande benefício para a segurança da rede (CIAMPA, 2009).

3.8 Políticas de segurança

Uma política de segurança é um conjunto de regras, práticas e procedimentos ditando como as informações sensíveis serão geridas, protegidas e distribuídas. Na esfera de segurança de rede, as políticas são geralmente pontos específicos que abrangem uma única área. Uma política de segurança é um documento que expressa exatamente o que deve ser feito para estabelecer os níveis de segurança, quais serão os objetivos a serem realizados pelos mecanismos que controlam a segurança da rede de dados. Este documento é escrito pela gerência de segurança e destina-se a descrever os “quês” e “como” de segurança da informação, descrevendo procedimentos padrões, referências e diretrizes na implementação da política (BHAIJI, 2008).

Conforme Harrington (2005), é preciso uma quantidade significativa de esforço para preparar e manter útil uma política de segurança. Da mesma forma que um documento de requisitos de sistema, ela precisa ser revisada em intervalos regulares para determinar a necessidade de atualização.

A confiança é um dos principais temas em muitas políticas. Algumas organizações confiam demais em seus empregados, e acreditam que todos vão fazer a coisa certa. Mas isso nem sempre acontece. Sabe-se que a maioria das organizações precisa de políticas para assegurar que todos estão em conformidade, seguindo o mesmo conjunto de regras (BHAIJI, 2008). Ainda conforme Bhaiji (2008) em sua própria experiência, as políticas tendem a elevar a apreensão das pessoas, pois elas não querem se comprometer com regras e regulamentos. Em vez disso, as pessoas querem liberdade e falta de responsabilidade. A política deve definir o nível de controle de usuários observando e conciliando com as metas de produtividade. Uma

política de rigor excessivo será difícil implementar porque o cumprimento pode ser minimizado ou ignorado. Pelo contrário, uma política definida vagamente pode ser evadida e não garantir a responsabilização e responsabilidade. Uma boa política tem que ter o equilíbrio certo. A seguir são apresentados os principais itens que compõem uma política de segurança.

3.8.1 Objetivos

Vacca (2010) explica que os objetivos fundamentais da política de segurança são permitir acesso ininterrupto a recursos da rede para usuários autenticados e negar o acesso a usuários não autenticados. É claro que isso é sempre um ato de equilíbrio entre as demandas dos usuários e a natureza evolutiva da tecnologia da informação. A comunidade de usuários prefere abrir acessos, enquanto o administrador da rede insiste em acesso restrito e controlado. Assim que o *hacker* descobre uma possível falha de segurança por meio de acesso não autorizado, ele torna-se o árbitro da política de segurança, portanto, qualquer rede é segura até o último ataque que violou a sua segurança. Uma vez construída e protegida seria totalmente irrealista esperar uma rede segura em todos os momentos. Assim, o projeto de segurança de redes e sua implementação representa a derradeira batalha das mentes entre o chefe de segurança da informação e o *hacker*. Em resumo, a política de segurança de rede pode ser tão simples como permitir o acesso a recursos, ou conter centenas de páginas, detalhando os níveis de acesso e de punição se a violação for descoberta. As funções essenciais de uma boa política de segurança são:

- nomear um administrador de segurança que está familiarizado com as demandas dos usuários em uma base contínua que está preparada para acomodar as necessidades da comunidade de utilizadores;
- configurar uma política de segurança hierárquica para refletir na estrutura corporativa;
- definir os recursos de acesso éticos à *internet*;
- evoluir a política de acesso remoto;
- fornecer um conjunto de procedimentos de tratamento de incidentes.

3.8.2 A importância

A política de segurança é a base para todas as questões relacionadas à proteção da informação, desempenhando um papel importante em todas as organizações. A necessidade de estabelecer uma política de segurança é um fato realçado unanimemente em recomendações

provenientes tanto do meio militar como do meio técnico e, mais recentemente, do meio empresarial (NAKAMURA; GEUS, 2009).

Seu desenvolvimento é primeiro e o principal passo da estratégica de segurança das organizações. É por meio dessa política que todos os aspectos envolvidos na proteção de recursos existentes são definidos e, portanto, grande parte do trabalho é dedicado a sua elaboração e ao seu planejamento (BHAIJI, 2008).

Segundo Harrington (2005), uma política de segurança é um documento que define a filosofia e a estrutura de segurança da organização. Serve para vários propósitos:

- tornar mais fácil para a equipe de TI justificar gastos com a segurança;
- identificar o uso aceitável de recursos de computação em uma organização;
- identificar quem tem acesso, e o que este usuário pode acessar;
- funcionar como um contrato de segurança com os empregados. Eles devem aderir à filosofia e aos comportamentos incluídos na política de continuidade no emprego.

Além de seu papel primordial nas questões relacionadas com a segurança, a política de segurança, uma vez fazendo parte da cultura da empresa, tem uma importante função como facilitadora e simplificadora do gerenciamento de todos os seus recursos. De fato, o gerenciamento de segurança é a arte de criar e administrar a política de segurança, pois não é possível gerenciar o que não pode ser definido (NAKAMURA; GEUS, 2009).

3.8.3 O planejamento

Na fase inicial do planejamento da política de segurança, tem-se a necessidade de entender todos os riscos que podem ser encontrados e enfrentados. Uma abordagem reativa pode trazer problemas futuros para a organização. Sendo assim, a proatividade é, portanto, essencial e depende de uma política de segurança bem definida, com definições de responsabilidades individuais claras de modo que facilite o gerenciamento da segurança de rede (NAKAMURA; GEUS, 2009).

Ciampa (2009) observa o planejamento de uma política de segurança envolve definir o que é uma política de segurança, entender seu ciclo e conhecer as etapas do seu desenvolvimento. Existem vários termos para se definir uma política de segurança, mas um conjunto de requisitos específicos tem que ser seguido e cumprido. Como, por exemplo, uma norma pode descrever como um computador poderá conectar-se remotamente a uma máquina interna na organização. Para que isso seja possível, os usuários deverão seguir essas normas para conseguir o acesso remotamente.

Uma visão geral do planejamento pode ser observada na Figura 20, na qual a pirâmide mostra que a política fica no topo, acima das normas e dos procedimentos.

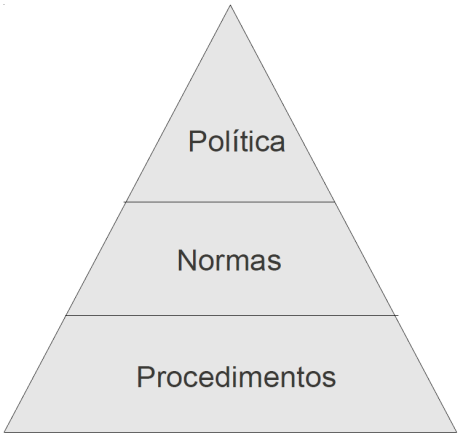

Figura 20 - O planejamento da política de segurança
Fonte: Do autor com base em Nakamura e Geus (2009).

A política é o elemento que orienta as ações e as implementações futuras, de uma maneira global, enquanto as normas abordam os detalhes, como os passos da implementação, os conceitos e os projetos de sistemas e controles. Os procedimentos são utilizados para que os usuários possam cumprir aquilo que foi definido na política e os administradores de sistemas possam configurar os sistemas de acordo com a necessidade da organização (NAKAMURA; GEUS, 2009). No próximo capítulo são apresentados os principais mecanismos de defesa na segurança de redes.

4 MECANISMOS DE DEFESA NA SEGURANÇA DE REDES

Neste capítulo são enfatizados os principais mecanismos de defesas na área lógica de segurança de redes, sendo abordados mecanismos de defesa como *firewall*, controle de acesso, antivírus, criptografia, entre outros.

4.1 Firewall

A necessidade de utilização cada vez maior da *internet* pelas organizações e a constituição de ambientes corporativos levam a uma crescente preocupação com a segurança. Como consequência, pode-se ver uma rápida evolução nessa área, principalmente com relação ao *firewall*, que é um dos principais, mais conhecidos e antigos componentes de um sistema de segurança. Para Nakamura e Geus (2009), um *firewall* é um ponto entre duas ou mais redes, que pode ser um componente ou um conjunto de componentes, pelo qual passa todo tráfego, permitindo o controle, a autenticação e os registros de todo o tráfego realizado.

Já Brenton e Hunt (2001) definem *firewall* como uma barreira de proteção, que controla o tráfego de dados entre um computador e a *internet*. Um *firewall* (ao contrário de um roteador simples, que apenas direciona o tráfego de rede) é um sistema que reforça uma política de controle de acesso. Depois de determinar os níveis de acesso que se deseja fornecer, é trabalho do *firewall* garantir que nenhum acesso além do determinado seja permitido. Cabe ao *firewall* garantir que a política de controle de acesso seja seguida por todos os usuários.

Conforme Ciampa (2009), um *firewall* é geralmente usado para filtragem de pacotes, projetado para impedir a entrada de dados maliciosos ou não permitidos. Pode ser baseado em software ou hardware, sendo o primeiro mais adequado para o uso doméstico, podendo ser executado localmente na máquina do usuário, e o segundo um dispositivo separado localizado em um ponto específico para proteger uma rede inteira. Os *firewalls* de hardware são geralmente localizados fora do perímetro da rede, atuando como a primeira linha de defesa. Um exemplo de localização é mostrado na Figura 21.

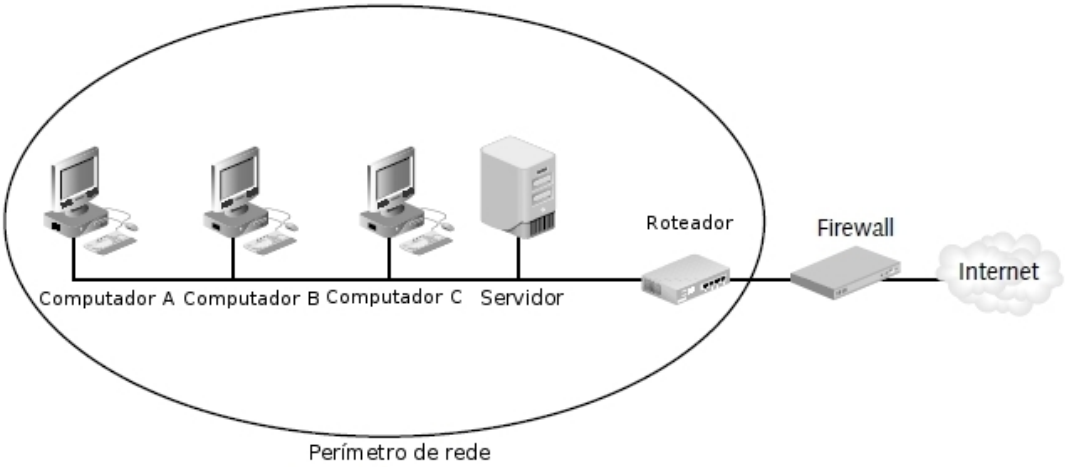

Figura 21 - *Firewall* atuando como primeira linha de defesa
Fonte: Do autor com base em Ciampa (2009).

Como uma primeira linha de defesa, o *firewall* tem como objetivo bloquear todos os tipos de acesso indevidos, que não estão de acordo com a política de segurança.

Sua fama, de certa forma, acaba contribuindo para uma falsa expectativa quanto à segurança total da organização. Assim é importante ter em mente que o *firewall* é apenas uma parte de um conjunto de componentes de um sistema de segurança necessário para a proteção das organizações. Sendo assim, para uma proteção ideal, é necessário fazer uso de outras ferramentas, entre elas IDS/IPS, antivírus, controle de acesso, entre outras, as quais são abordadas neste capítulo.

4.2 Controle de acesso

O controle de acesso refere-se ao método de conceder ou negar acesso a recursos de rede por meio de políticas de segurança. Em sua forma mais simples, controle de acesso é aplicada em arquivos e pastas, ou em outros recursos de rede compartilhados por meio de atribuição de permissões (BHARDWAJ, 2007). A seguir listam-se as principais categorias que envolvem o controle de acesso.

4.2.1 *Mandatory Access Control* (MAC)

Um mecanismo, normalmente codificado em um sistema operacional, tem a função de proteger computadores, dados e dispositivos do sistema de um uso ou acesso não autorizado. Pode também ser construído em um pedido para conceder ou negar permissões (BHARDWAJ, 2007).

4.2.2 *Discretionary Access Control* (DAC)

Geralmente é implementado no sistema operacional sob a forma de permissões e direitos de usuário. Um exemplo de DAC são as permissões NTFS *(New Technology File System)* usadas em computadores baseados no sistema operacional Windows (BHARDWAJ, 2007).

4.2.3 *Role Based Access Control* (RBAC)

Usado para implementar a segurança em objetos com base nas funções do trabalho executado por um usuário, ou grupos de usuários, o RBAC é altamente flexível e configurável, e fornece administração centralizada (BHARDWAJ, 2007).

4.3 Antivírus

Um antivírus é uma solução de segurança amplamente adotada. Utilizado em empresas de pequeno a grande porte, ele identifica e remove do computador vários tipos de *malware*. É uma das aplicações mais antigas em termos de segurança. Este software pode examinar infecções em um sistema operacional, bem como monitorar a atividade do computador verificando novos documentos, a fim de detectar algum vírus. Um exemplo disso é a varredura executada em anexos de um *e-mail* se um vírus for detectado, o software geralmente apresenta uma limpeza imediata, mandando o arquivo infectado para a quarentena, ou possibilita a exclusão do arquivo. Uma das desvantagens é que o software deve ser atualizado continuamente para o reconhecimento de novos vírus (CIAMPA, 2009).

4.4 Criptografia

A criptografia é uma ciência fundamental para a segurança da informação, ao servir de base para diversas tecnologias e protocolos, tais como a infraestrutura de chaves públicas, o IP*Security* (IPSec) e o *Wired Equivalent Privacy* (WEP). Suas prioridades incluem sigilo, integridade, autenticação e o não repúdio, e garantem o armazenamento, as comunicações e as transações seguras essenciais no mundo atual (NAKAMURA; GEUS, 2009).

A criptografia tem função e importância cada vez mais fundamentais para a segurança das organizações. É a ciência que visa manter as mensagens seguras. A cifragem *(encryption)* compreende o processo de disfarçar a mensagem original, o texto claro, de tal modo que sua substância é escondida em uma mensagem com texto cifrado, enquanto a decifragem consiste

no processo de transformar o texto cifrado de volta em texto claro original (TODOROV, 2007).

Os processos de cifragem e decifragem são realizados via uso de algoritmos com funções matemáticas que transformam os textos claros, que podem ser lidos, em textos cifrados, que são inteligíveis (NAKAMURA; GEUS, 2009).

O capítulo seguinte abrange o tema mecanismos de segurança, pois apresenta ferramentas do tipo IDS/IPS, de suma importância, pois atuam quando um, ou mais de um, dispositivo mencionado anteriormente falhar. Por se tratar do foco principal do trabalho proposto, o tema é apresentado como um novo capítulo.

5 SISTEMA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO

Os conceitos de prevenção e detecção de intrusão, segundo Baker, Caswell e Beale (2007), são antigos, advindo da proteção à propriedade física, como veículos, edificações, entre outros bens que podem ser subtraídos, porém, devido ao valor das informações e dos dados armazenados em sistemas computacionais, é fundamental a aplicação de mecanismos de detecção e prevenção de intrusão digital.

Tratam-se, em separado, os temas detecção e prevenção, pois mesmo complementares, o entendimento individual é importante.

5.1 Sistema de detecção de intrusão

Do inglês *Intrusion Detection System* (IDS), refere-se, de acordo com Scott, Hayes e Wolfe (2004), ao sistema capaz de detectar ataques e atividades maliciosas em rede de dados ou sistema computacional. Na definição de Rehman (2003), um IDS é software, hardware ou a combinação de ambos para detectar intrusões.

Para Cox e Gerg (2004), o sistema deve ser capaz de detectar ameaças antes de o dano ser causado ao sistema computacional que necessita de proteção. No mesmo sentido, Rehman, (2003) afirma que o IDS consiste na utilização de técnicas e de métodos que buscam identificar, tanto em rede quanto em *host*, atividades anormais consideradas suspeitas, baseadas, portanto, em assinaturas de tráfego ou detecção de anomalias.

A detecção baseada em assinaturas funciona da seguinte maneira: todo o dado que trafega na rede é composto, entre outras informações, por dados que o identificam ou identificam assinaturas, que podem ser detectadas com a utilização de softwares específicos. Então, o sistema de detecção verifica todo o tráfego em busca de pacotes com assinaturas conhecidas. Detectar atividades suspeitas fundamentadas em anomalias tende a ser um processo mais simples, visto que se baseia em dados presentes no cabeçalho dos pacotes de dados, e utiliza os dados históricos para identificar comportamento considerado anormal.

Não existe consenso entre autores, mas, de acordo com Scott, Hayes e Wolfe (2004), os sistemas podem ser divididos em duas grandes categorias: os *Network Intrusion Detection System* (NIDS) e os *Host Intrusion Detection System (*HIDS), detalhados na próxima seção.

5.1.1 *Network Intrusion Detection System* (NIDS)

Baker, Caswell e Beale (2007) definem NIDS como uma derivação de IDS que tem seu funcionamento baseado no monitoramento de toda a rede a partir da perspectiva do local de sua implementação, geralmente um segmento de rede.

Os sensores de um sistema de NIDS geralmente são placas de rede trabalhando em modo promíscuo, ou seja, com a capacidade de capturar todo o tráfego de rede que passar pelo segmento em questão. Essa técnica pode ser implementada utilizando equipamentos do tipo *HUB* ou pela utilização de espelhamento de portas em *switches*.

A Figura 22 mostra uma rede com três NIDS estrategicamente posicionados, protegendo os servidores que possuem uma visão para a rede externa e também as estações de trabalho conectadas ao segmento de rede interno. Segundo Baker, Caswell e Beale (2007), a utilização de múltiplos NIDS dentro de uma mesma rede é exemplo do que se chama de arquitetura de defesa em profundidade.

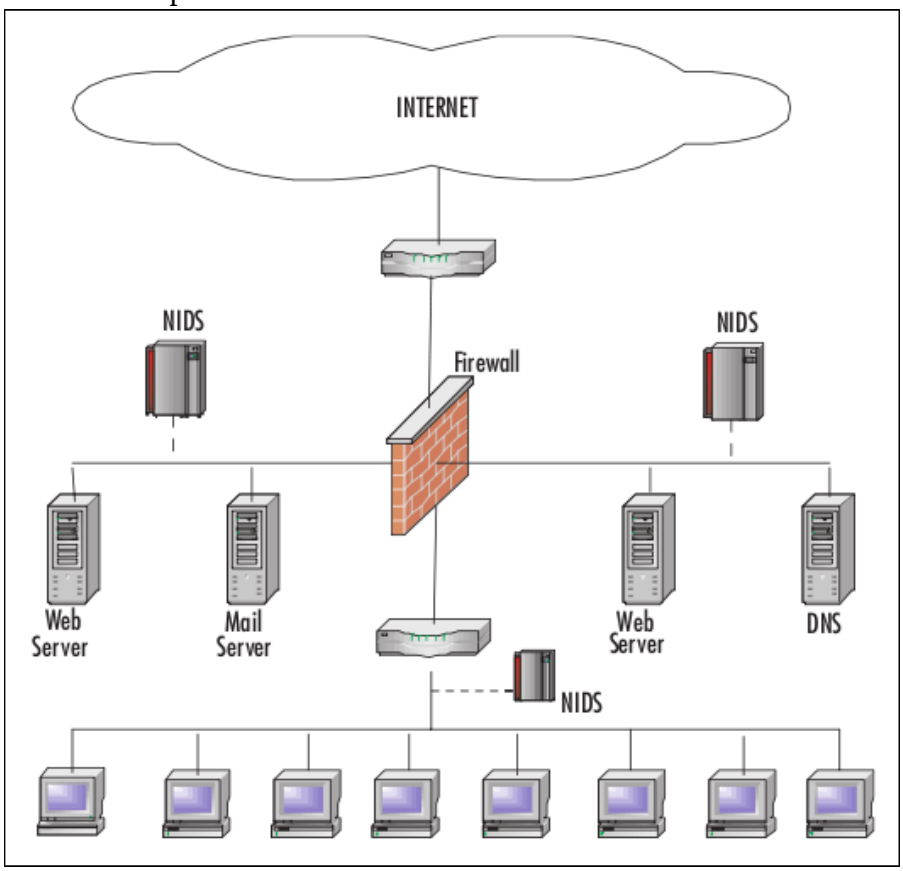

Figura 22 - Posicionamento de NIDS em rede
Fonte: Baker, Caswell e Beale (2007, p. 12).

Uma das vantagens da implementação de NIDS, segundo Beale (2004), é o fato de ele não ter qualquer impacto sobre a rede ou sistemas que esteja monitorando, já que não agrega

carga aos *hosts* da rede. Outro benefício é que o sistema baseado em NIDS é totalmente transparente aos atacantes.

Como desvantagem, tem-se o fato de que segmentos de rede com muito tráfego podem ocasionar a saturação da capacidade de servidores ou de ativos de rede, necessitando de estudo específico para a segmentação do tráfego em sistemas NIDS complementares.

5.1.2 *Host Intrusion Detection System* (HIDS)

Basicamente, de acordo com Scott, Wolfe e Hayes (2004), um HIDS é um IDS que monitora somente o sistema no qual está instalado, monitorando o tráfego de rede do *host*, os registros de eventos, e a integridade dos arquivos do sistema.

Complementar a esta definição Baker, Caswell e Beale (2007) salientam que um HIDS, por estar em execução em um *host*, conhece melhor as informações locais no que tange à segurança, como chamadas e registro de sistemas.

Outra vantagem dos HIDS é o fato de as regras serem específicas, ou seja, apenas serão verificados os serviços do sistema operacional do *host* e os serviços que realmente estão em execução, sem necessidade de verificações genéricas sobre como os NIDS devem operar.

Segundo o conceito de segurança em profundidade, defendido por Baker, Caswell e Beale (2007), o HIDS pode ser a última instância de segurança após o ataque ter obtido êxito ao passar pelos sistemas de *firewall* e NIDS existentes na rede.

Há algumas desvantagens quanto à utilização de HIDS, tendo em vista que ele deve ser específico para o sistema operacional em questão, o que pode ser um fator complicante em redes heterogêneas, já que fornecedores podem não disponibilizar seu software ou hardware para os mais diversos sistemas operacionais encontrados no mercado.

Outra questão importante é a carga que o HIDS agrega ao *host* no qual está em execução demanda extra que deve ser avaliada com critério antes da implementação. Outro fator de cuidado é o entendimento quanto ao funcionamento do HIDS, tendo em vista que algumas regras podem conflitar com aplicativos instalados.

A manutenção de rede com um grande número de HIDS é apontada como uma desvantagem e um desafio, assinala Beale (2004), pois se o sistema HIDS não tiver uma gestão centralizada, o trabalho para administrar sistemas individuais pode inviabilizar a adoção de tais ferramentas, já que a quantidade de alertas gerados pode ser intensa o suficiente e demandar um tempo muito grande para a análise.

A Figura 23 representa uma rede em que alguns *hosts* implementam um sistema de HIDS. Como é possível observar, nem todos os *hosts* possuem a ferramenta e deve-se supor que em cada um deles estejam configuradas regras específicas.

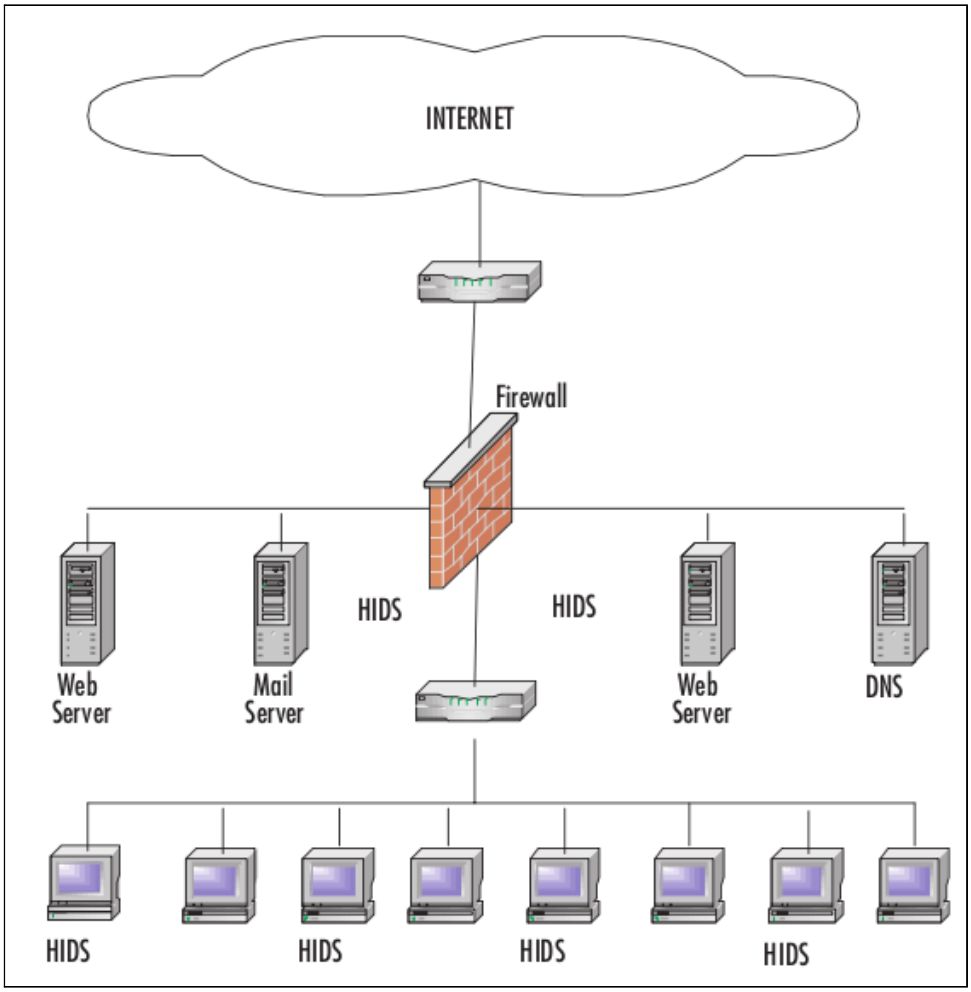

Figura 23 - HIDS em *hosts* específicos
Fonte: Beale, Baker e Caswell (2007, p. 14).

5.2 Sistema de prevenção de intrusão

A prevenção de intrusão é uma tecnologia promissora na área de segurança de redes. Os sistemas de prevenção de intrusão surgiram como uma evolução dos IDS e combinam a capacidade de profunda inspeção de pacotes com características de filtragem naturais dos *firewalls*, porém de forma transparente (TIPTON; KRAUSE, 2007).

O conceito de IPS é semelhante ao IDS, ambos requerem assinaturas de ataques conhecidos para detectar ameaças na rede de dados. Enquanto um IDS normalmente detecta tentativas de intrusão e efetua o registro e/ou manda alertas para o administrador da rede, o IPS inclui outras medidas para deter a intrusão em tempo real. Desenvolvido como medida de prevenção, atua bloqueando automaticamente possíveis ataques antes que eles tenham

sucesso. Sendo assim, o IPS é um dispositivo que irá bloquear um ataque antes que ele chegue a seu alvo. Embora pareça que esta tecnologia possa substituir um *firewall*, é necessário ter em mente que ela oferece uma camada a mais de proteção e, por ser uma evolução do IDS, considera-se uma das primeiras linhas de defesa da segurança da informação. Em resumo, o IDS é como se fosse um alarme de um carro que soa somente quando alguém abre a porta, e o IPS dispara o alarme e trava as portas com o intuito de que o invasor não leve o carro (DOHERTY; ANDERSON; MAGGIORA, 2008).

Um sistema de IPS pode tomar várias ações, dentre elas: bloqueios de portas no *switch*, interação com políticas de *firewall* externos, regras dos roteadores, ou, ainda, geração de tráfego na camada de transporte (MAIA; REHEM, 2005).

Segundo Endorf, Schultz e Mellander (2004), um IPS tipicamente consiste em quatro componentes principais mostrados na Figura 24:

- normalizador de tráfego: irá interpretar o tráfego de rede e fazer análise, reestruturar os pacotes e executar as funções básicas de bloqueio, além de fornecer alimentação para *scanner* de serviço e máquina de detecção;
- *scanner* de serviço: cria uma tabela de referência, que classifica as informações e ajuda o modelador de tráfego a gerenciar o fluxo das informações;
- máquina de detecção: realiza uma comparação entre a tabela de referência e determina a resposta adequada;
- modelador de tráfego: controla o fluxo das informações.

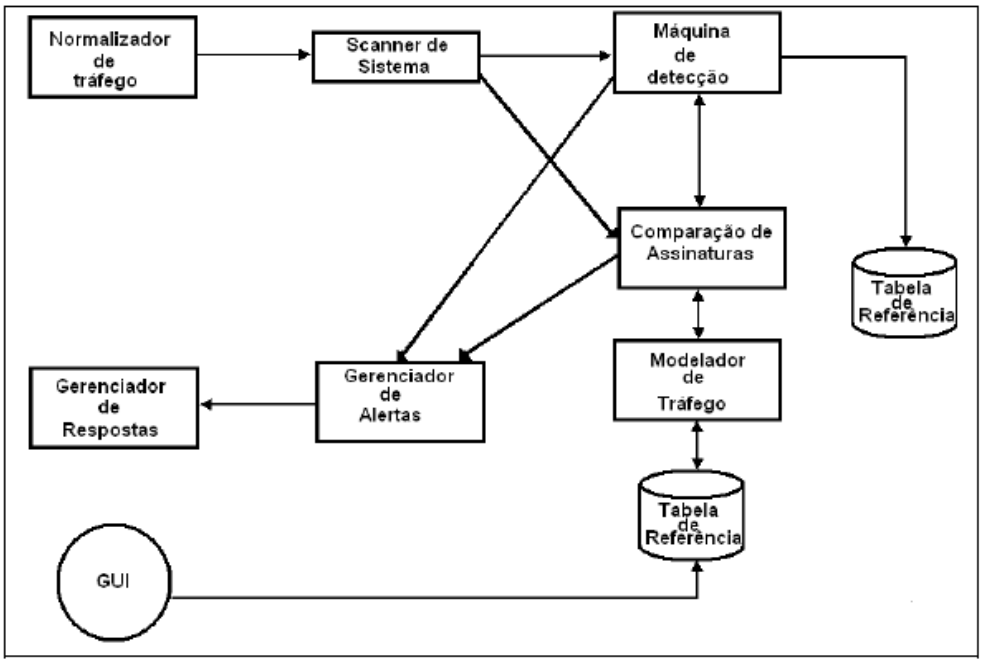

Figura 24 - Padrão do sistema IPS
Fonte: Do autor com base em Endorf, Schultz e Mellander (2004).

Existem dois tipos de sistema de prevenção de intrusão: IPS baseado em rede (NIPS) e IPS baseado em *host* (HIPS), os quais são vistos nas seções a seguir.

5.2.1 *Network Intrusion Prevention System* (NIPS)

Uma rede baseada em IDS é projetada para monitorar o tráfego de forma passiva, e levantar alarmes quando um tráfego suspeito for detectado, enquanto um sistema baseado em NIPS é projetado para ir um passo além e realmente tentar impedir o ataque (VACCA, 2010).

Um IPS baseado em rede é exatamente um sistema de prevenção de intrusão instalado no *gateway*, de forma que o tráfego passe por ele como se fosse mais um elemento, ou seja *inline*, assim posicionado para que ele possa evitar tentativas de ataques maliciosos como cavalos de troia, *backdoors*, *rootkits*, vírus, *worms*, entre outras ameaças. A Figura 25 exibe o NIPS em uma estrutura de rede.

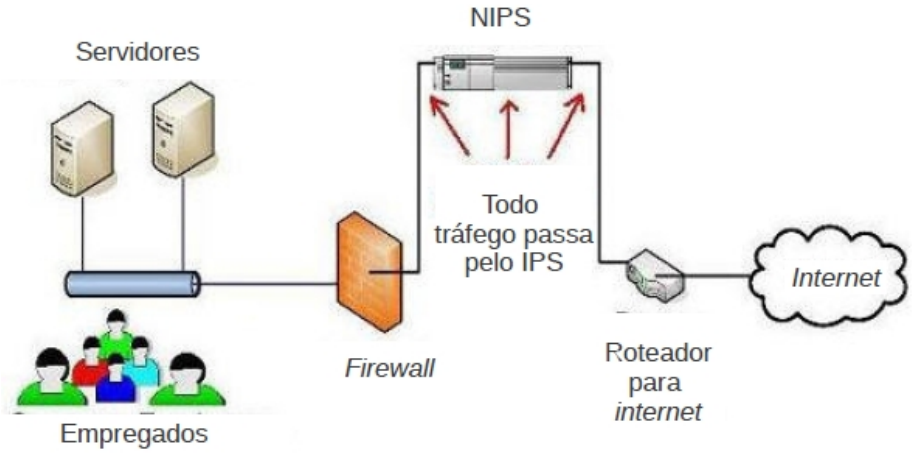

Figura 25 - Localização do NIPS em uma rede de computadores
Fonte: Do autor com base em Carter e Hogue (2006).

Tecnicamente, um NIPS realiza dois tipos de funções: filtragem de pacotes e detecção de intrusão, podendo prover contra-medidas nas seguintes camadas:

- enlace: administrativamente *shutdown* na porta do *switch* que está saindo o ataque tal atitude só é praticável para ataques que são gerados de um sistema local;
- rede: interage com o *firewall* externo ou roteador, para adicionar uma regra geral para bloquear todas as comunicações de endereço de IP individual ou a rede inteira;
- transporte: gera pacotes TCP-RST *(transmission control protocol)-(reset)* para derrubar conexões TCP maliciosas ou emitir algum pacote de erro ICMP *(internet control message protocol)* dos diversos disponíveis para responder um tráfego UDP *(user datagram protocol)* malicioso;

- aplicação: alertas de dados maliciosos na camada de aplicação não podem causar danos antes de alcançar o sistema alvo. Esta contra medida requer que o IPS esteja *inline* no caminho da comunicação.

5.2.2 *Host Intrusion Prevention System* (HIPS)

Um IPS baseado em *host* também atua como a última linha de defesa. O software é instalado em cada máquina que precisa ser protegida. Um HIPS geralmente consiste em um servidor de gerenciamento e um agente que está em execução entre a aplicação e o *kernel* do sistema operacional. Incorporado a um módulo carregável do *kernel*, se o *host* for *Unix*, ou a um *driver* de sistema, se o *host* for *Windows*, oferece proteção robusta. Dessa forma, o agente pode interceptar chamadas do sistema para o *kernel*, compará-las às listas de controle de acesso ou regras de comportamento definidas no HIPS e decidir em permitir ou bloquear o acesso a recursos específicos, tais como pedidos de leitura e escrita em disco, solicitações de conexão de rede, modificações de registro, ou escrever para a memória (TIPTON; KRAUSE, 2007).

Segundo Vacca (2010), um dos benefícios do HIPS é que o tráfego de rede criptografado pode ser analisado após o processo de descriptografia que ocorre no sistema protegido, proporcionando assim uma oportunidade para detectar um ataque que teria sido escondido em um NIPS ou NIDS. O próximo capítulo apresenta a implementação do IDS/IPS em uma rede de computadores.

6 IMPLEMENTAÇÃO DO SISTEMA DE DETECÇÃO E PREVENÇÃO DE INTRUSÃO

A implementação de uma ferramenta de IDS/IPS, descrita neste capítulo, tem como base o estudo abordado até o momento, levando em consideração o funcionamento básico desse tipo de ferramenta, mostrando como o IDS e o IPS foram dispostos na rede de computadores e como as anomalias foram detectadas.

O projeto foi desenvolvido exclusivamente em software livre, sem a necessidade de aquisição de licenças para a sua prática. Tendo como objetivo principal a identificação e automatização do bloqueio de ameaças utilizando ferramenta de IDS/IPS, fazendo uma análise qualitativa dos bloqueios gerados, a fim de viabilizar a implementação.

Este capítulo inicia com a apresentação do modelo base. Logo após, expõe a efetuação da identificação dos pontos passíveis de falhas e implementação do sistema de IDS/IPS. Em seguida é apresentado o modelo ideal e dá-se prosseguimento com o detalhamento do funcionamento e dos softwares necessários para a implementação.

6.1 Modelo de ambiente base encontrado

Esta seção visa a apresentar o ambiente básico inicial ao qual aplicar-se o estudo. Trata-se da estrutura física e lógica na qual há a necessidade de inclusão de meios para aumentar a segurança da rede.

A Figura 26 ilustra o ambiente encontrado, sendo o nível físico e lógico característico de empresas de grande porte. Neste caso, para a implementação da ferramenta é apresentada e utilizada a estrutura de rede de dados do Centro Universitário UNIVATES. Optou-se por esta, pela alta tecnologia, grande variedade de equipamentos e inúmera heterogeneidade de perfis de usuários que utilizam a rede de dados, a qual é utilizada diariamente por cerca de 600 a 1000 usuários simultaneamente em períodos de picos. Tais afirmações têm base no monitoramento efetuado pelo Núcleo de Tecnologia de Informação da Univates com o auxílio do software Zabbix[9].

[9]http://www.zabbix.com

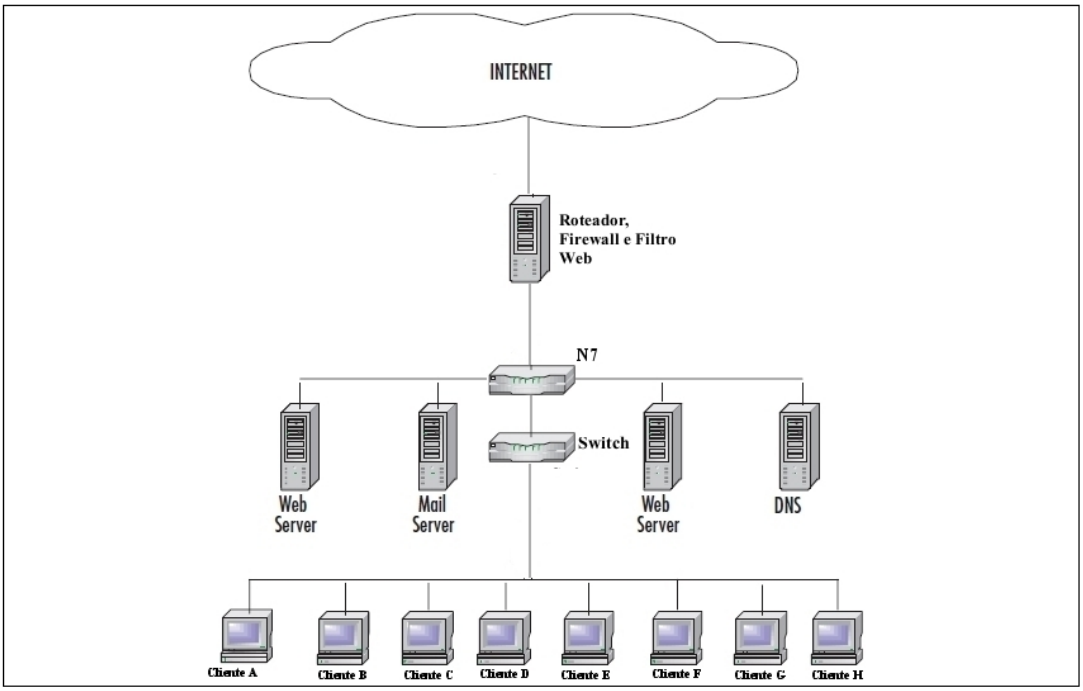

Figura 26 - Modelo do ambiente encontrado
Fonte: Do autor com base em Baker, Caswell e Beale (2007).

A estrutura da rede é caracterizada por aplicação do tipo cliente/servidor, sendo o servidor a peça fundamental para o modelo, pois é ele que disponibiliza os serviços requeridos pelos clientes, sendo *firewall*, roteador e filtro de acessos da rede de dados. Uma breve descrição dos componentes principais é apresentada abaixo:

- *internet*: acesso a informações e todo tipo de transferência de dados;
- servidores: computadores com arquitetura x86;
- core *switch* (n7): *firewall* da rede interna,
- *switch de borda*: meio de interligação entre clientes e servidores;
- clientes: computadores clientes conectados à rede de dados interna e externa.

Além do *firewall* e do filtro *web*, é possível perceber que a atual rede não apresenta nenhuma ferramenta que monitore e analise o tráfego em tempo real. As únicas ferramentas implantadas não são capazes de desempenhar este propósito. Elas simplesmente autorizam ou não o envio de pacotes para os destinos predefinidos pela segurança da rede. Sendo assim, o ambiente está passível de sofrer diversas ameaças, interferindo na sua integridade e na de seus usuários.

Com a apresentação do modelo de ambiente base, podem-se identificar alguns pontos de vulnerabilidades, os quais são abordados na próxima seção.

6.2 Determinação do problema

Conforme descrito na seção anterior, a rede de dados em que se efetuou a implementação é utilizada por diferentes níveis de usuários, desde os que possuem conhecimento avançado na área de informática até os que pouco conhecem um sistema operacional. Sendo assim, todos os usuários correm o risco de serem infectados na rede ou acomodarem em seus computadores vírus, *malwares* e *spywares*, os quais podem interferir na segurança local ou divulgar informações sigilosas e não autorizadas para o mundo exterior, ou, até mesmo, a ameaça pode vir dos próprios usuários tentarem contra a segurança da rede local. Tendo em vista esses e outros problemas que envolvem a segurança, efetua-se uma análise no ambiente apresentado na Figura 26. Nela, podem-se observar várias vulnerabilidades (pontos de falhas de segurança).

A Figura 27 ilustra os pontos de falhas mais críticos encontrados no modelo do ambiente:

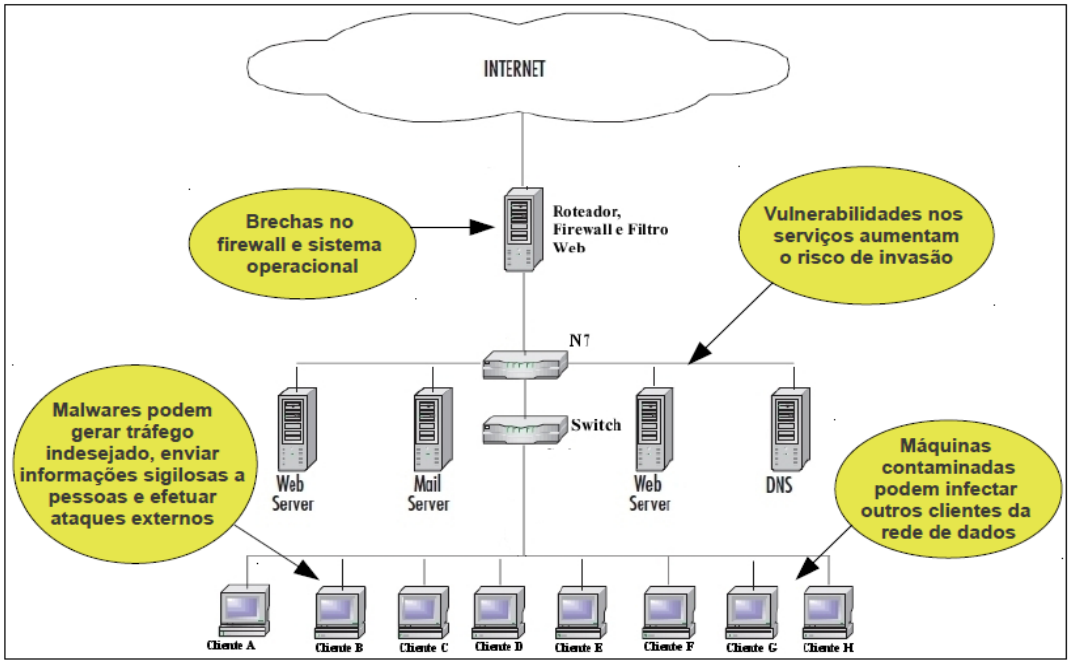

Figura 27 - Pontos de falha de segurança encontrada modelo do ambiente
Fonte: Do autor com base em Baker, Caswell e Beale (2007).

Ao analisar-se a Figura 27, pode-se notar diversos pontos de falha quanto à segurança, sendo estes originados das redes interna ou externa. O modelo descrito, ao qual foi aplicada a implementação do trabalho proposto, é típico de uma rede de grande porte, no caso, é a rede wireless do Centro Universitário UNIVATES, a qual é utilizada por professores, alunos e visitantes. Por se tratar de uma rede com amplos acessos, abrangendo um público de inúmeras características, ela sofre constantes ameaças, sendo que não é possível efetuar um controle específico do software que cada usuário está executando na rede, pois este utiliza seu

computador pessoal com permissões administrativas no sistema operacional, podendo instalar e executar qualquer programa que esteja a seu alcance.

Para uma melhor compreensão, apresentam-se os principais pontos considerados críticos na estrutura apresentada:

- erro de configuração no *firewall*: atacantes em potencial podem fazer uso de uma má configuração do *firewall* que protege a rede de dados;
- vulnerabilidades: *bugs* em sistemas operacionais, ou softwares utilizados em servidores, podem gerar uma vulnerabilidade;
- máquinas internas contaminadas: máquinas contaminadas podem dissipar vírus por toda a rede, contaminando outros computadores;
- *malwares*: podem gerar tráfego indesejado, enviar informações sigilosas a atacantes, e até mesmo gerar ataques externos estes normalmente são obtidos a partir de ataques de engenharia social ou acessos a sites infectados.

Para tentar minimizar os problemas apresentados, a seção seguinte apresenta o que seria um ambiente ideal de utilização, corrigindo os diversos pontos de falhas observados.

6.3 Pontos ideais para aplicação da ferramenta

Como é possível observar, a segurança da rede de dados pode ser afetada de diversas formas. No entanto, não há uma forma automática de detectar uma ameaça ou intrusão na rede, assim como de mensurar a quantidade de ameaças sofridas. Diante disso, surge como solução a implementação de um sistema de IDS/IPS, o qual incorpora as funcionalidades estudadas até o momento, e traz como resultado final um aumento no nível da segurança da rede de dados com a automatização de bloqueios de ameaças após sua implantação.

Para tal implementação, deve-se analisar o ambiente base encontrado, verificar se a estrutura de rede do local apresenta os requisitos necessários para utilização da ferramenta, assim como definir os pontos vulneráveis a ataques entre outras ameaças. Com uma análise do ambiente, é possível extrair informações que ajudam a definir os pontos ideais para a aplicação da ferramenta IDS/IPS.

Com a obtenção das informações necessárias é possível implementar a ferramenta nos principais pontos de vulnerabilidades. A Figura 28 ilustra o modelo da solução proposta que deve atender os requisitos de segurança de redes elencados neste trabalho.

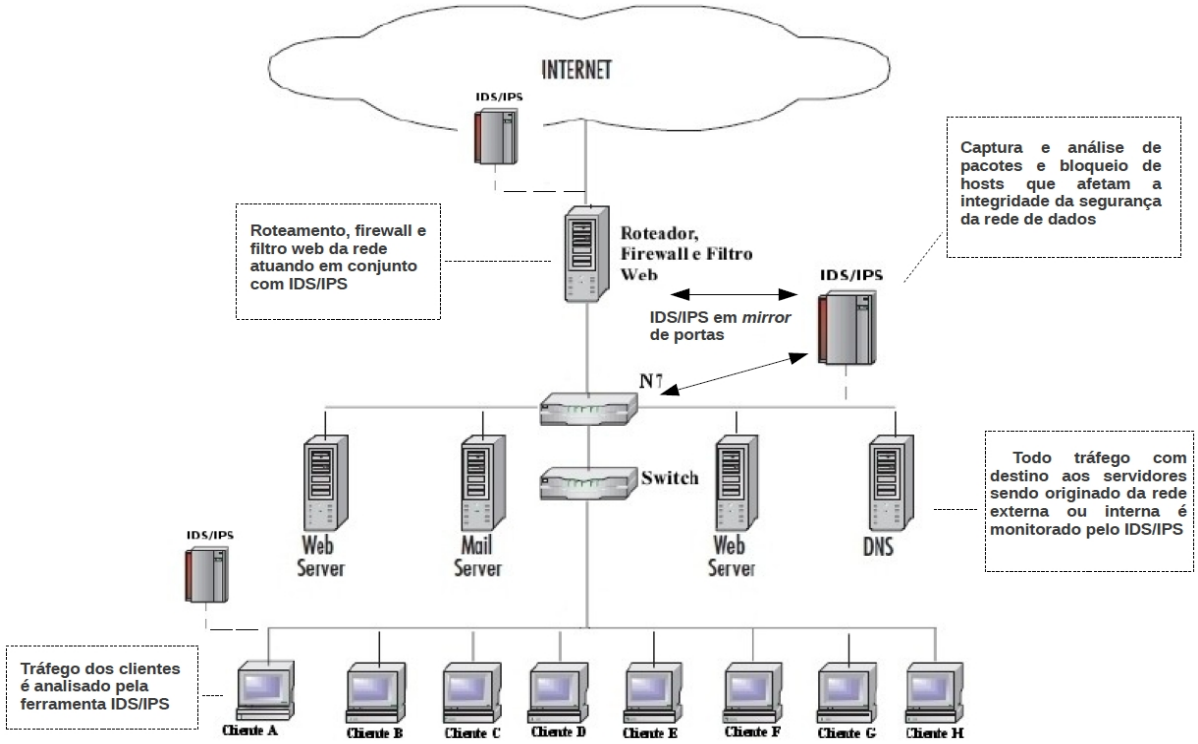

Figura 28 - IDS/IPS proposto como solução de segurança da rede de dados
Fonte: Do autor com base em Baker, Caswell e Beale (2007).

Analisando a Figura 28, verifica-se o modelo ideal após a implementação de um sistema de IDS/IPS e pode-se notar que os pontos de aplicação foram estrategicamente escolhidos, por serem os mais propícios à vulnerabilidade na rede de dados, requerendo um monitoramento minucioso com todo o tráfego gerado pelos *hosts*.

A funcionalidade primordial da estrutura de IDS/IPS destina-se a analisar o tráfego da rede de dados e a homologar possíveis ameaças ou ataques. Sendo confirmado um desses casos, com a automação do sistema, este atuará de forma automática bloqueando a origem ou o destino que originou o problema.

No decorrer deste capítulo são apresentados as ferramentas e os equipamentos necessários para a implementação do projeto, assim como para o funcionamento deste.

6.4 Arquitetura de *software*

Esta seção tem por objetivo apresentar a arquitetura de software existente na implementação prática da ferramenta de IDS/IPS. A subseção seguinte descreve os softwares utilizados e seu funcionamento.

6.4.1 O IDS *Snort*

Segundo Beale (2004), *Snort* é um aplicativo de segurança moderno com três funções principais: ele pode atuar com um *sniffer* de pacotes, *logger* de pacotes ou sistema de detecção de intrusão. Foi desenvolvido pela primeira vez em novembro de 1998. Originalmente destinado para funcionar com um *sniffer* de pacotes, possuía cerca de 1600 linhas de códigos e tinha um total de dois arquivos. No final de janeiro de 1999 implementou o recurso baseado em análise de assinaturas, podendo ele ser utilizado como um sistema de detecção de intrusão, na verdade, um dos mais poderosos sistemas de detecção de intrusão de rede.

Ele tem como diferencial a capacidade de inspecionar o *payload* do pacote área que contém os seus dados, fazendo o registros dos pacotes, além de detectar as invasões. O *Snort* é um sistema avançado capaz de detectar quando um ataque está sendo realizado e, baseado nas características do ataque, alterar ou remodelar sua configuração de acordo com as necessidades, e até avisar o administrador do ambiente sobre o ataque (CASWELL; BEALE; BAKER, 2007).

Seu mecanismo de detecção depende de uma linguagem rudimentar, por meio de assinaturas que servem como base para o reconhecimento de padrões de entrada que caracterizam ataques. O *Snort* implanta regras para decodificação dos dados dentro dos pacotes, em seus cabeçalhos, ajudando-o a reconhecer ataques como *buffer overflow* e ataques *web* indevidamente codificados (ENDORF; SCHULTZ; MELLANDER, 2004).

Um característica marcante do *Snort* é a facilidade de criação de assinaturas de ataques. Profissionais da área de segurança da informação sabem da velocidade que os eventos ocorrem no mundo virtual. A prontidão, exatidão e rapidez são assuntos essenciais em se tratando de segurança. Como o *Snort* é um sistema de código aberto e apresenta linguagem fácil de criação de assinaturas, permite a pronta criação e adição de uma nova assinatura de ataque por parte do administrador (COX; GERG, 2004).

Assim que o *Snort* captura os pacotes, ele os analisa segundo assinaturas de reconhecimento de ataques, que nada mais são do que padrões de conteúdos dos pacotes que são comparados aos dos pacotes reais. Os conjuntos de pacotes que forem reconhecidos como maliciosos fazem o *Snort* logar uma tentativa de ataque em seu *log* de alertas. A Figura 29 exibe o funcionamento interno do *Snort*:

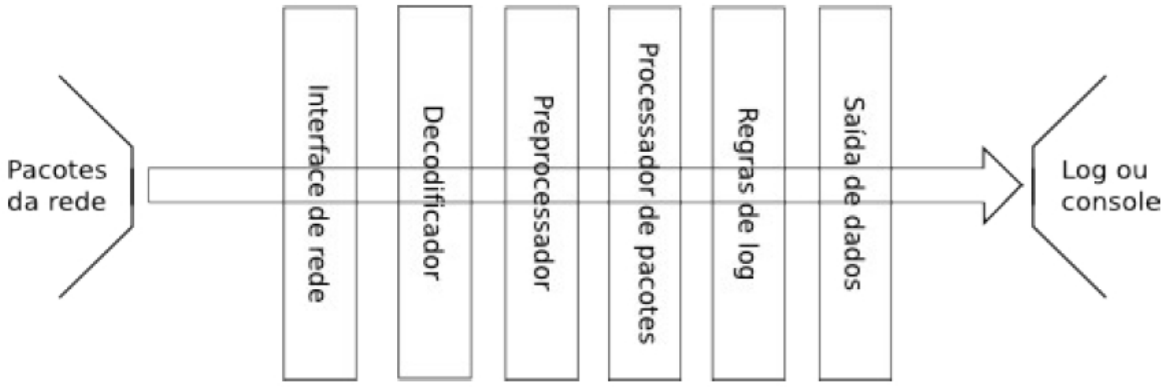

Figura 29 - funcionamento interno *Snort*
Fonte:Do autor com base em Cox e Gerg (2001).

Analisando a Figura 29, percebe-se que o *Snort* faz a captura dos pacotes que trafegam por uma dada interface de rede. Para descobrir se está ou não havendo uma tentativa de intrusão, o S*nort* atravessa os dados dos pacotes por uma série de módulos de análise e de diagnóstico. O decodificador recebe os pacotes e os analisa no nível dos protocolos e dos cabeçalhos, descobrindo os IP envolvidos, as portas, o tamanho do pacote e sua integridade, tudo antes mesmo de eles serem remontados para uma aplicação.

O pré-processador identifica comportamentos suspeitos de pacotes e os encaminha para uma análise de conteúdo para o módulo seguinte. Já o processador de pacotes compara o conteúdo e a natureza do pacote com assinaturas de ataques conhecidos, as quais estão armazenadas em um diretório e indicadas pelas regras carregadas pelo *Snort.* As mesmas regras que enviam a informação sobre os pacotes para serem processadas encaminham o diagnóstico para ser logado e/ou impresso no console, podendo assim ser visualizadas pelo administrador da rede até este momento somente foi detectada e informada uma ameaça (BEALE, 2004).

6.4.1.1 Componentes do *Snort*

O *Snort* é composto por componentes internos, sendo eles responsáveis pela análise dos dados que trafegam pela rede. Entre os principais componentes podem-se citar o decodificador de pacotes, pré-processadores, mecanismo de detecção e *plugins* de saída, apresentados na Figura 30:

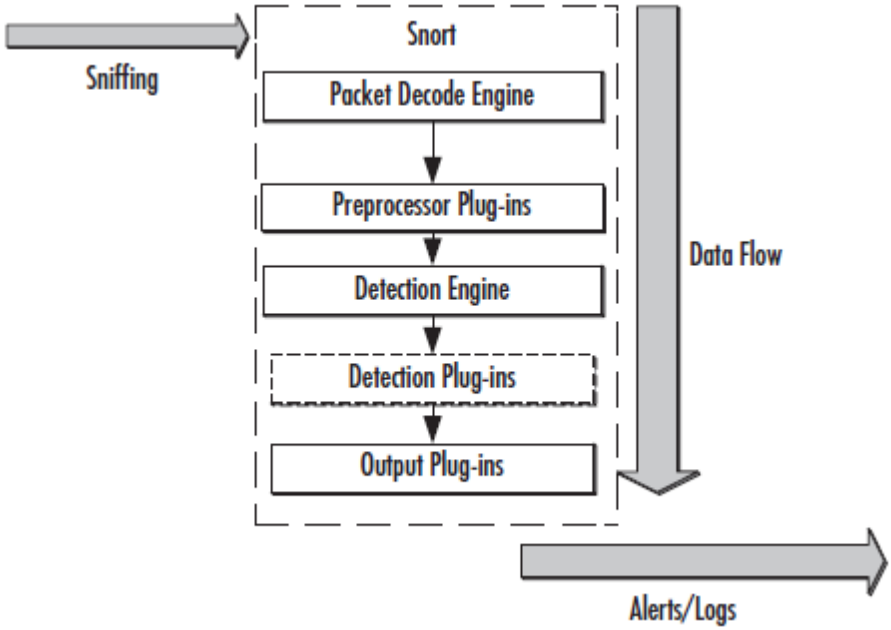

Figura 30 - Componentes internos *Snort*
Fonte: Beale e Foster (2003, p. 95).

Para melhor entendimento, apresentam-se os quatro principais componentes do *Snort* e a função básica de cada componente indicados por Beale e Foster (2003):

1. mecanismo de captura/decodificador de pacotes: inicialmente o tráfego é adquirido a partir do *link* de rede por meio da biblioteca *libcap*. Os pacotes passam pelo mecanismo de detecção, que monta a estrutura dos pacotes para os protocolos de enlace, os quais são ainda mais decodificados para os protocolos de nível mais alto, como as portas TCP e UDP;
2. *plugins* pré-processadores: pacotes são enviados por meio de um conjunto de pré-processadores. Estes são examinados e manipulados antes de serem entregues ao mecanismo de detecção. Cada pré-processador verifica se cada pacote possui algo que deve ser examinado, alertado a respeito ou modificado.
3. mecanismo de detecção: os pacotes são enviados para o mecanismo de detecção. O mecanismo de detecção verifica cada pacote em relação a várias opções listadas nos arquivos de regras do *Snort*. Cada uma das opções da regra é vinculada a um *plugin* de detecção que pode realizar testes adicionais;
4. *plugins* de saída: em seguida, o *Snort* gera os alertas a partir do mecanismo de detecção, pré-processadores ou do mecanismo de decodificação.

Após a passagem dos pacotes por todos os componentes internos, um alerta em modo texto é gerado para o administrador do sistema, podendo este ser gravado em banco por meio

da ferramenta chamada *Barnyard* e exibido em tela com o auxílio do *front-end* desenvolvido para o *Snort* chamado *Snorby*.

6.4.2 O IPS *SnortSam*

O *SnortSam* é uma ferramenta de resposta ativa que interage com dispositivos de *firewalls* de código aberto e comerciais, bloqueando endereços de IPs. Esses bloqueios podem ser estipulados em minutos, horas, dias, semanas ou até mesmo anos. Sendo assim, o *SnortSam* desempenha o papel de IPS, trabalhando em conjunto com o IDS (BEALE, 2004).

Essa ferramenta consiste em duas partes, conforme Beale (2004):

1. um *plugin* de saída do *Snort* está instalado no sensor da rede;
2. um agente do *SnortSam* está instalado no *firewall*.

O sensor *Snort* está configurado com o endereço do agente, e as regras que devem solicitar uma ação de bloqueio são estendidas com parâmetros específicos de bloqueio. Quando uma regra for alcançada, o sensor *Snort* envia um pacote TCP encriptado a um ou mais agentes *SnortSam* que estão em execução no *firewall*. Em seguida, ele descriptografa a solicitação, se esta for bem sucedida, ou seja, se as senhas ou chaves do sensor *Snort* e o agente *SnortSam* coincidir, o agente vai aceitá-lo como um pedido válido. O agente executa algumas verificações, descobre o endereço IP do *host* que violou a regra do *Snort*, verifica o período que o *host* deverá ficar bloqueado e, se permitido, irá solicitar o *firewall* para bloquear o endereço IP relatado. Todos esses processos acontecem em tempo real, fazendo com que automaticamente os *hosts* sejam barrados antes que qualquer ameaça afete a integridade da rede de dados (REHMAN, 2003).

6.4.3 Barnyard

O *Barnyard* foi desenvolvido para separar as tarefas comuns de processamento de saída das tarefas mais críticas de monitoramento de tráfego de rede. Nesse sentido, o *Barnyard* pode ser visto como um evento de processamento assíncrono e uma ferramenta de envio projetada para ser usada com o *Snort*. Em seu modo normal de operações, *Barnyard* aguarda o *Snort* gerar um evento ou alertas de segurança e, em seguida, o despacha por meio de um ou mais *plug-ins* de saída, gravando os alertas em um banco de dados (CASWELL; BEALE; BAKER, 2007).

É essencial a utilização do *Barnyard* quando se está monitorando uma rede de alta velocidade, isto porque o *Snort* não consegue gerenciar todos os alertas e gravá-los em banco

se ele estiver exercendo ao mesmo tempo a tarefa de monitoramento da rede, ficando assim sobrecarregado e podendo vir a perder alguns eventos. Em suma, o *Barnyard* é uma ferramenta que faz a leitura dos alertas gerados pelo *Snort* e os grava em um banco de dados para que eles possam ser lidos e apresentados ao administrador da rede (CASWELL; BEALE; BAKER, 2007).

6.4.4 Snorby

Snorby é um *front-end* para o IDS *Snort*. Os conceitos básicos fundamentais por trás do *Snorby* são a simplicidade e o poder, por ser uma ferramenta de fácil manipulação e visualização de alertas. O objetivo do projeto é criar uma fonte livre, aberta e com aplicação altamente competitiva para monitoramento de rede, para uso privado e corporativo. Sendo assim, o *Snorby* é uma ferramenta que faz a leitura dos alertas em banco de dados e os apresenta de forma gráfica e intuitiva para serem analisados.

6.5 Validação

A fim de validar a viabilização da implementação da ferramenta de IDS/IPS na rede *wireless* do Centro Universitário UNIVATES, submeteu-se ela, a um período de avaliações e homologação no monitoramento de tráfego da rede a fim de detectar e impedir ameaças automaticamente. Os resultados obtidos com as avaliações devem viabilizar ou não a implementação da ferramenta automatizada descrita neste trabalho.

6.5.1 Cenário

O cenário para a validação é idêntico ao apresentado na Seção 6.3, o qual, depois analisadas as vulnerabilidades, identificados os pontos de falha e definidos os locais de monitoração, foi considerado como o ambiente ideal para a implementação do trabalho. A composição deste é feita com diversos dispositivos de rede e servidores físicos, os quais são:

1. Servidor Dell modelo PowerEdge 2850;
2. Enterasys Matrix N7 Core Switch;
3. Enterasys RoamAbout Wireless Switch 8500;
4. Enterasys RoamAbout Wireless Switch 8200;
5. Enterasys C2-H124-48;
6. Enterasys RBT-1002 RoamAbout AP 1002;
7. Máquinas clientes diversas.

A Figura 31 exibe o modelo físico do cenário utilizado para a implementação do trabalho proposto. Nela não foi possível adicionar todos os servidores da rede, não sendo alguns adicionados na ilustração, mas todos os segmentos de rede e dispositivos foram monitorados pela ferramenta IDS/IPS.

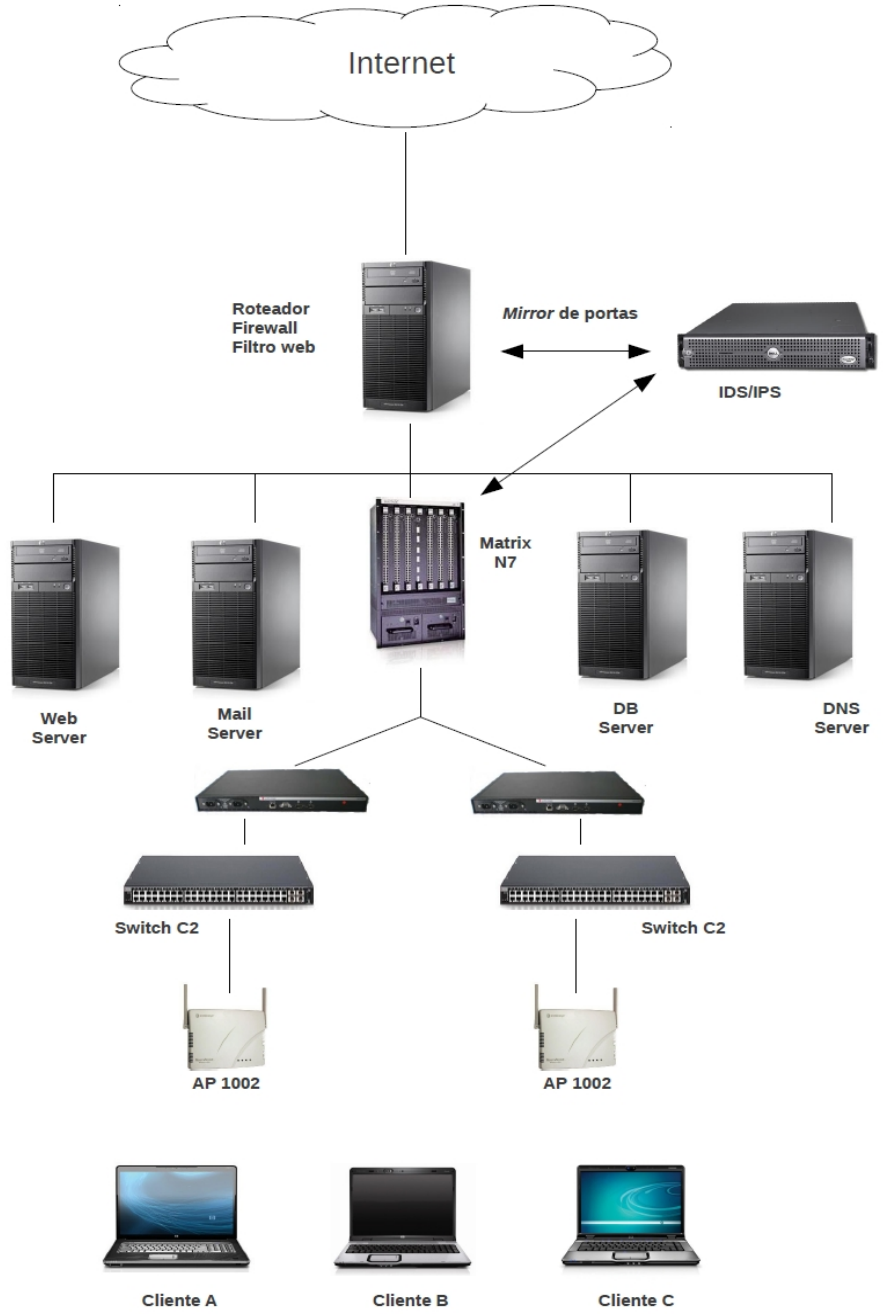

Figura 31 - Modelo físico
Fonte: Do autor.

A ferramenta de IDS/IPS foi posicionada em local estratégico devido o projeto da rede, sendo assim capaz de analisar todo o tráfego externo, por estar em *mirror* de porta com o roteador, e interno, por estar em *mirror* com a Matrix N7, a qual é o ponto de interligação de todos os s*witches* da rede.

Com os softwares descritos nas subseções 6.4.1, 6.4.2, 6.4.3 e 6.4.4 devidamente instalados, iniciou-se o processo de validação. Vale ressaltar que o processo de instalação dos softwares necessários não são descritos, mas os arquivos de configurações são apresentados nos Apêndices A, B, C e D.

6.5.2 Descrição das experimentações

Com as ferramentas em funcionamento, iniciou-se o processo de prova no período de 1º a 28 de outubro de 2011. Ao longo destes dias, o sistema de IDS/IPS efetuou o monitoramento em tempo real de todo o tráfego da rede *wireless* do Centro Universitário UNIVATES, identificando e impedindo fraudes na rede de dados de forma automática. Durante estes período o número de usuários que fez uso da rede da instituição é significante, gerando milhares de alertas. As experimentações foram separadas por subseções, apresentando cada uma delas as ameaças e a interação exercida pelas ferramentas implementadas.

6.5.2.1 Ameaças oriundas da rede externa

Ao possibilitar aos usuários acesso ao mundo exterior por meio de uma rede interconectada, várias portas são abertas, podendo estas serem atingidas por inúmeros *hosts* em qualquer lugar do mundo. O administrador de rede pode estar ciente de que o *firewall* é sua proteção máxima, e que com ele nada poderá afetar a integridade de seus dados, mas o *firewall* mesmo sendo uma ferramenta de segurança trabalha de forma que determinadas portas sejam abertas para possibilitar o acesso a um determinado serviço. Com isso, o ambiente sempre está sujeito a correr riscos de segurança. A seguir são apresentados dois casos de ameaça à segurança da rede de dados, que somente o sistema de detecção e prevenção de intrusos implementado foi capaz de identificar o ocorrido.

1. Ataque do tipo DoS:

O *Slowloris* é uma ferramenta de ataque HTTP DoS. Ela cria *sockets* e envia requisições HTTPs válidas para a vítima continuamente em intervalos regulares. Dessa forma, ele tenta enganar o *webserver* evitando que essas conexões sejam fechadas. O *Slowloris* aguarda a conexão bem sucedida de todos os *sockets* para iniciar o ataque. Essa técnica permite indisponibilizar o alvo aos poucos, dificultando a detecção do ataque. A ferramenta concorre com os acessos válidos, normalmente ganhando essa concorrência, fazendo com que o serviço *web* fique indisponível para quem queira acessá-lo. A seguir é efetuado um ataque

do tipo DoS contra a rede de dados do Centro Universitário UNIVATES a fim de comprovar a eficácia do sistema implementado.

A Figura 32 exibe o momento em que o *Slowloris* foi executado a partir de uma fonte externa contra a rede de dados.

```
./slowloris.pl -dns 200.132.250.252
Welcome to Slowloris - the low bandwidth, yet greedy and poisonous HTTP client

Defaulting to port 80.
Defaulting to a 5 second tcp connection timeout.
Defaulting to a 100 second re-try timeout.
Defaulting to 1000 connections.
Multithreading enabled.
Connecting to 200.132.250.252:80 every 100 seconds with 1000 sockets:
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Sending data.
Current stats:  Slowloris has now sent 2453 packets successfully.
This thread now sleeping for 100 seconds...

                Sending data.
                Sending data.
Current stats:  Slowloris has now sent 2718 packets successfully.
This thread now sleeping for 100 seconds...

Current stats:  Slowloris has now sent 2769 packets successfully.
This thread now sleeping for 100 seconds...

                Sending data.
Current stats:  Slowloris has now sent 2931 packets successfully.
This thread now sleeping for 100 seconds...

                Sending data.
Current stats:  Slowloris has now sent 3173 packets successfully.
This thread now sleeping for 100 seconds...
```

Listagem 1- Momento que é executado o Slowloris

Fonte: Do autor.

Conforme pode ser visto, no momento em que o software é executado, ele abre várias conexões com o endereço de destino. Com o passar do tempo, o número de conexões tende a aumentar até que o servidor não consiga mais responder a todos os pedidos, fazendo com que o serviço fique indisponível. Sendo assim, nesse momento o *Snort* que está monitorando o tráfego de rede deve identificar e atuar sobre o ocorrido, sendo que a ferramenta em conjunto

com o IPS *Snortsam* atue em tempo real, realizando automaticamente o bloqueio do *host* de origem.

A Figura 33 apresenta os alertas enviados para o administrador do sistema:

Sev.	Sensor	Source IP	Destination IP	Event Signature	
1	Snorby:eth1	177.44.240.253	200.132.250.252	SPECIFIC-THREATS Slowloris http DoS tool	11:27 PM
1	Snorby:eth1	177.44.240.253	200.132.250.252	SPECIFIC-THREATS Slowloris http DoS tool	11:27 PM
1	Snorby:eth1	177.44.240.253	200.132.250.252	SPECIFIC-THREATS Slowloris http DoS tool	11:27 PM
1	Snorby:eth1	177.44.240.253	200.132.250.252	SPECIFIC-THREATS Slowloris http DoS tool	11:27 PM

Figura 32 - Alertas sobre o ataque ocorrido
Fonte: Do autor.

Com os alertas apresentados, pode-se verificar quem originou o ataque, qual foi seu destino e que tipo de evento foi originado, mas ficaria inviável o acompanhamento em tempo real de todos os *logs*. Para isso, o administrador do sistema teria que estar sempre acompanhando o monitoramento da rede. Contudo, mesmo que isso fosse possível, nenhuma pessoa conseguiria analisar em tempo hábil milhares de rajadas de alertas simultaneamente. Para isso, o sistema automático de IPS deve atuar bloqueando a ameaça assim que for detectada. A Figura 34 exibe o IPS atuando no sistema.

```
2011/10/25, 23:27:13, 10.100.0.44, 2, snortsam, Blocking host 177.44.240.253 completely for 86400 seconds (Sig_ID: 2009413).

2011/10/25, 23:27:34, 10.100.0.44, 2, snortsam, Extending block for host 177.44.240.253 completely for 86400 seconds (Sig_ID: 2009413).
```

Listagem 2 - Bloqueio do host que originou o ataque
Fonte: Do autor.

Imediatamente o comando de bloqueio foi enviado para o *firewall*, informando-lhe o endereço do *host* a ser bloqueado, tempo de bloqueio e assinatura responsável por identificar o tipo de tráfego gerado. Nota-se que o alerta e o bloqueio aconteceram no mesmo instante, o que é muito importante, pois faz com que o ataque seja interrompido antes de causar qualquer interferência nos serviços oferecidos pelo servidor. Neste caso, mesmo que o tráfego de rede não esteja sendo acompanhado pelo administrador, o IDS/IPS interage de forma automática garantindo a segurança.

2. Escaneamento de portas:

Escanear as portas é uma forma alternativa de detectar serviços abertos em uma máquina, mesmo que todas as portas TCP estejam fechadas no *firewall*. Com a técnica de escaneamento de portas, podem-se obter várias informações dos serviços disponíveis em um determinado servidor, ou até mesmo em uma máquina interna da rede. Muitos atacantes utilizam essa técnica para sondar informações quanto a um *host*, pois com o escaneamento pode-se determinar quais serviços podem ser inspecionados e possivelmente atingidos.

Existem diversas ferramentas para efetuar o escaneamento de portas, sendo para prova de testes utilizada a ferramenta chamada *Nmap.*

O *Nmap* é uma ferramenta software livre que realiza *port scan.* É muito utilizado para avaliar a segurança dos computadores e para descobrir serviços ou servidores em uma rede de computadores, neste caso, ela é executada contra o servidor posto em prova, a fim de descobrir serviços ativos. A Figura 35 exibe o momento em que o *Nmap* foi executado.

```
solis@router:~$ nmap 200.132.250.252

Starting Nmap 5.00 ( http://nmap.org ) at 2011-10-26 10:52 BRST
Interesting ports on 200.132.250.252:
Not shown: 992 closed ports
PORT     STATE    SERVICE
22/tcp   filtered ssh
25/tcp   filtered smtp
80/tcp   filtered http
139/tcp  filtered netbios-ssn
898/tcp  open     sun-manageconsole
3000/tcp filtered ppp
3128/tcp filtered squid-http
8080/tcp filtered http-proxy
```

Listagem 3 - *Nmap* em execução
Fonte: Do autor.

Conforme o funcionamento da ferramenta, ela faz uma varredura nos serviços e respectivas portas disponíveis. Com isso um atacante pode adquirir as informações iniciais para definir os pontos em que ele poderá atuar. Ao mesmo tempo em que o servidor está sendo atingido por um *port scan*, o tráfego é monitorados e são exibidos os alertas observados na Figura 36.

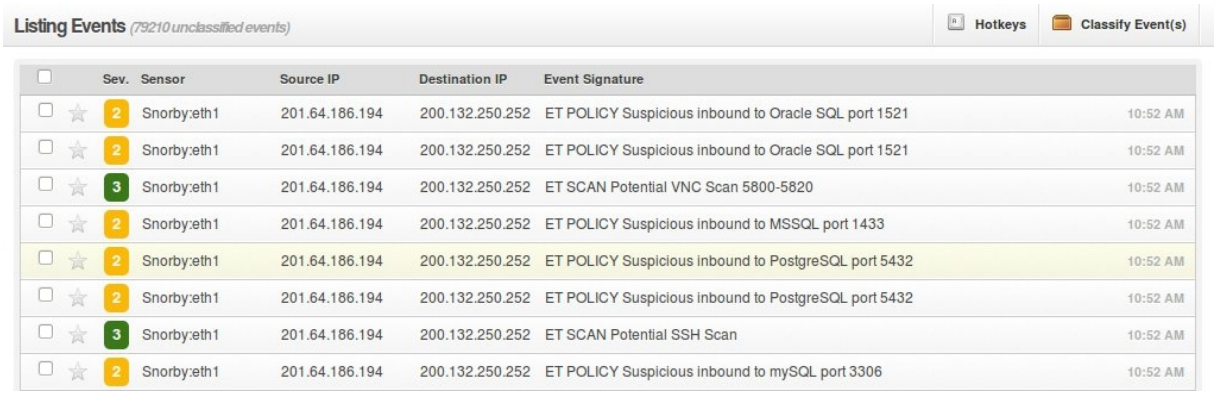

Figura 33 - Alertas gerados durante o escaneamento de portas
Fonte: Do autor.

Na Figura 36 é possível verificar as informações necessárias contidas no *port scan*, hora que foi executado, quem o originou e quais serviços foram inspecionados. Tudo acontece ao mesmo tempo, o escaneamento, o alerta e o bloqueio. Então, quando é detectado o tráfego indesejado, o IDS/IPS atuará a fim de impedir que o *host* que executou o comando não consiga atingir o seu destino por um tempo determinado, o qual foi definido na assinatura responsável pela detecção. É efetuado um bloqueio geral e o IP que originou o escaneamento

não conseguirá mais interagir com seu destino. A Figura 37 apresenta o bloqueio automático sendo efetuado pela ferramenta implementada.

```
2011/10/26, 10:52:17, 10.100.0.44, 2, snortsam, Blocking host 201.64.186.194 completely for 86400 seconds (Sig_ID: 2001219).

2011/10/26, 11:00:59, 10.100.0.44, 2, snortsam, Extending block for host 201.64.186.194 completely for 86400 seconds (Sig_ID: 2001219).
```

Listagem 4 - Momento em que o bloqueio é enviado para o f*irewall*
Fonte: Do autor.

A interação foi feita com o f*irewall* da rede, que por sinal emitido por mensagem recebida pelo IPS insere regras de *iptables* bloqueando qualquer interação do atacante. Isso só é possível pois existe um agente do *SnortSam* sendo executado no *firewall*, conforme descrito na subseção 6.4.2. Assim se finaliza todo o processo, desde a descoberta de uma ameaça até o bloqueio de quem a originou.

6.5.2.2 Ameaças oriundas da rede interna

Em um rede de computadores interconectada, as ameaças podem ser originadas tanto externa como internamente, e muitas vezes o perigo pode estar dentro da própria instituição. No caso da rede *wireless* do Centro Universitário UNIVATES isso não é diferente, pois diariamente centenas de usuários fazem uso da rede para troca de *e-mails*, acessos a arquivos e navegação na *web*. Muitas vezes sem saber, esses usuários podem estar gerando ameaças e interferindo no funcionamento de todo o sistema. Por estar concentrada em diversos pontos da rede de dados, a ferramenta implementada também faz o monitoramento de *hosts* internos, garantindo a segurança com bloqueios automáticos assim que anomalias forem detectadas. Nesse contexto, são apresentados alguns casos de ameaças dos inúmeros já identificados durante o período de validação.

1. Máquina cliente infectada por vírus:

Um usuário interno pode estar com seu computador infectado sem mesmo saber do ocorrido. Para garantir que isso não ocorra, é necessário que o IDS/IPS consiga identificar quem está infectado e o bloqueie para que não infecte os demais clientes da rede. Neste caso foi identificado um usuário que está infectado pelo w*orm* identificado como *Conficker*, que tem como objetivo afetar computadores dotados do sistema operacional Microsoft Windows, Ele é um *worm* que pode infectar o computador e se espalhar para outros computadores de uma rede automaticamente, sem interação humana, propagando-se por meio de três vetores principais:

a) explorando uma vulnerabilidade no serviço “Servidor” do Windows (SVCHOST.EXE – TCP/445);

b) executando ataques de força bruta contra redes compartilhadas, tentando adivinhar a senha de acesso do administrador;
c) infectando dispositivos removíveis normalmente utilizados em diversos computadores (*pendrives*, cartões de memória USB).

A Figura 38 exibe o funcionamento do *worm Conficker:*

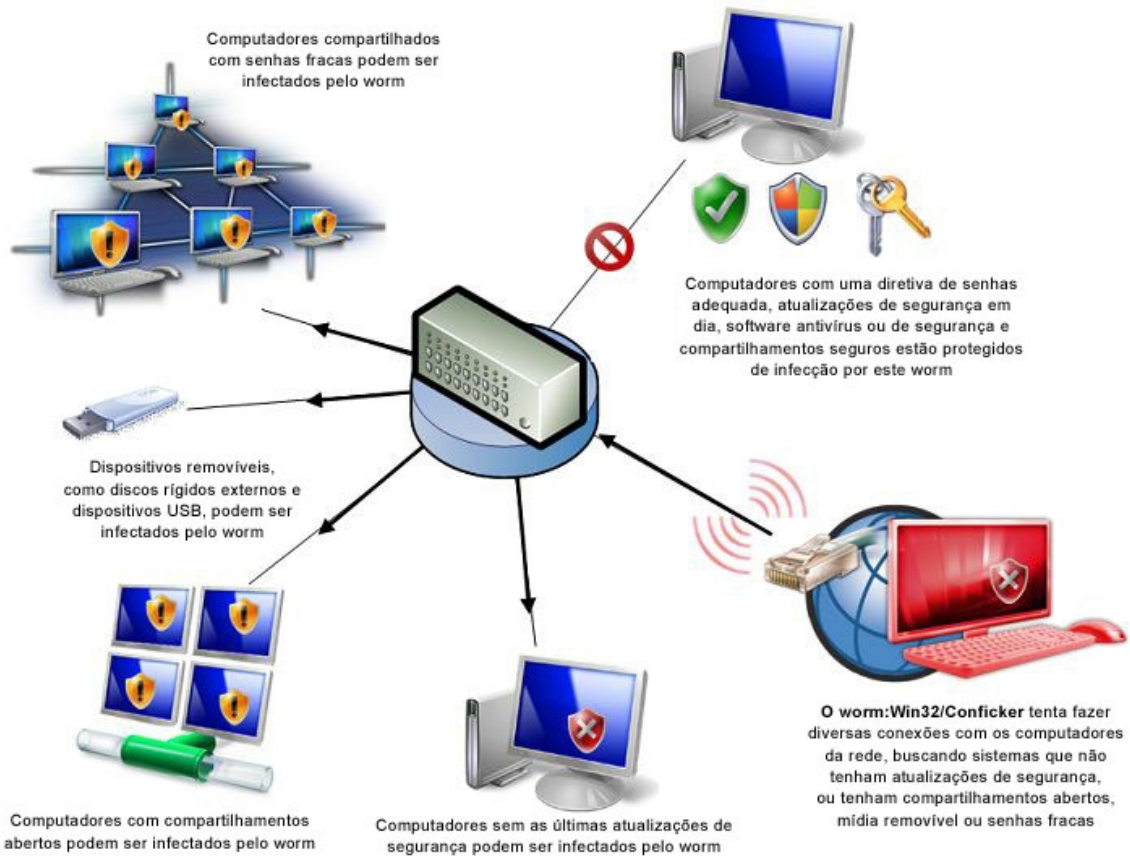

Figura 34 - Funcionamento do *worm Conficker*
Fonte: Do autor.

O *worm* descrito é uma praga de fácil disseminação. Conforme seu funcionamento, podem ser facilmente infectados computadores sem atualização, compartilhamentos de arquivos abertos ou com senhas fracas e dispositivos removíveis. Por esses motivos, é necessário que sejam efetuados a identificação e o bloqueio de todos os usuários que estejam infectados, para que o vírus não se propague a outras máquinas da rede. A Figura 39 exibe os alertas gerados quando uma praga desse tipo é identificada pelo *Snort.*

	Sev.	Sensor	Source IP	Destination IP	Event Signature	
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011
	2	Snorby:eth1	10.8.2.199	221.8.69.25	ET TROJAN Downadup/Conficker A or B Worm reporting	10/27/2011

Figura 35 - Alertas do *worm Conficker*
Fonte: Do autor.

Neste caso, a máquina interna de IP 10.8.2.199 está infectada e comunicando-se com o IP de destino 221.8.69.25, que provavelmente deve ser a fonte de origem da infecção ou a máquina zumbi que recebe as informações adquiridas pelo *worm*. Então, se esse cliente não for bloqueado, a chance de propagação dos vírus é grande. Para que isso não ocorra, após a detecção, imediatamente o IPS entra em ação enviando um comando de bloqueio para o *firewall* da rede isto fará com que o IP do cliente seja banido da rede até que a infecção seja removida do computador. A Figura 40 exibe o momento exato em que o cliente está sendo banido da rede.

```
2011/10/27, 12:16:39, 10.100.0.44, 2, snortsam, Blocking host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:18:51, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:19:35, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:19:57, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:20:44, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:21:07, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:21:29, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:21:52, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:22:15, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:23:46, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:24:09, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:24:32, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:24:55, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:25:18, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
2011/10/27, 12:25:40, 10.100.0.44, 2, snortsam, Extending block for host 10.8.2.199 completely for 43200 seconds (Sig_ID: 2009024).
```

Listagem 5 - Cliente da rede interna sendo bloqueado pelo IPS
Fonte: Do autor.

Nota-se que tudo acontece em tempo real. O momento em que foram gerados a ameaça, a sua detecção e seu bloqueio, e a cada tentativa de acesso ou disseminação do *worm* o bloqueio é estendido. Se isso não acontecesse de forma automática, provavelmente o administrador da rede não conseguiria atuar de forma imediata, a fim de impedir a infecção de outras máquinas internas.

Assim que um cliente interno é bloqueado, este não possui informações do ocorrido. Ele apenas percebe que não usufrui de acesso à rede. Então, para que fique sabendo do ocorrido, foi desenvolvida uma interface que trabalha em conjunto com o filtro *web* da rede para informar ao usuário que seu endereço de IP foi bloqueado. Essa interface foi criada por meio de código HTML (*Hyper Text Markup Language)* e *scripts* em linguagem *bash.* Utilizaram-se estas duas linguagens para que seja possível efetuar a leitura do *host* bloqueado pela ferramenta implementada, e posteriormente houve a apresentação da mensagem de

bloqueio correspondente ao IP barrado na *web*. Os códigos de desenvolvimento da interface *web* e *scripts* podem ser verificados nos Apêndices E e F. Mesmo não sendo um objetivo do trabalho proposto, houve a necessidade do desenvolvimento da mensagem de retorno para o usuário engrandecendo a evolução da ferramenta implementada com ela obteve-se uma interação com o cliente da rede, podendo dar um *feedback* do ocorrido.

A Figura 41 exibe a mensagem recebida pelo cliente quando este tiver seu computador bloqueado.

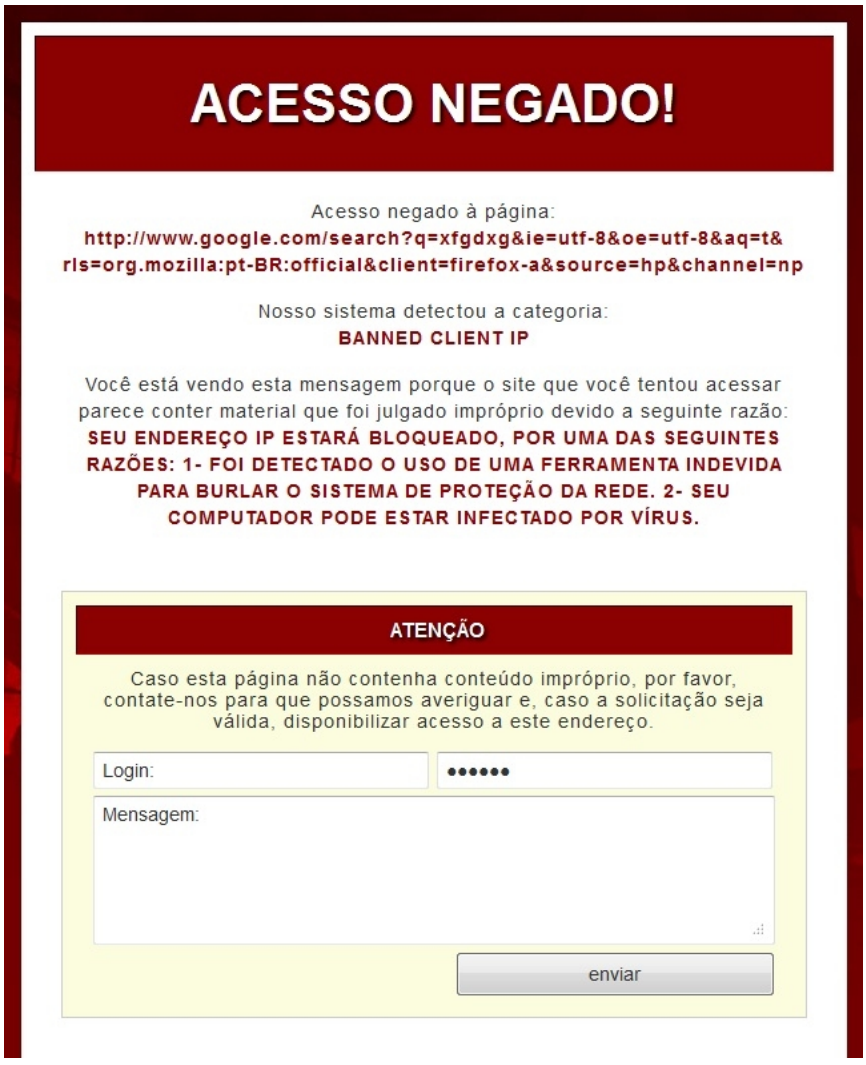

Figura 36 - Mensagem recebida pelo cliente assim que o endereço de IP é bloqueado
Fonte: Do autor.

A Figura 41 é apresentada indiferente do destino que o cliente queira acessar. Ela sempre será exibida enquanto o *host* estiver bloqueado. Nela podem-se obter informações básicas do seu bloqueio, assim como enviar um e-mail ao administrador da rede para esclarecer outras dúvidas que venham a surgir. A liberação do cliente somente é efetuada quando o incidente não for mais detectado na rede, sendo feita automaticamente pelo IDS/IPS assim que decorrer o tempo estimado de bloqueio.

2. Tunelamento do protocolo SSH:

Em informática, *tunnelling* refere-se à capacidade de criar túneis entre duas máquinas por onde certas informações passam. E, em se tratando do protocolo TCP/IP, o SSH pode criar uma conexão entre dois computadores, intermediada por um servidor remoto, fornecendo a capacidade de redirecionar pacotes de dados. No caso da rede *wireless* do Centro Universitário UNIVATES, a conexão à *internet* é protegida por um *firewall* que bloqueia determinadas portas de conexão, indisponibilizando alguns serviços externos, além de ser uma ferramenta de filtro *web* que efetua o bloqueio de páginas indevidas com base na política de acesso da instituição. Contudo, isso compromete a dinamicidade de aplicações na *internet.* Para quebrar essa imposição rígida, porém necessária, o SSH oferece o recurso do túnel, fazendo os usuários que utilizam a rede de dados da instituição uso desse para conseguir burlar os bloqueios impostos. Com essa técnica, não é possível verificar o conteúdo trafegado pelo usuário, pois ele está sendo encapsulado pelo SSH, e nos *logs* do *firewall* é possível observar somente uma requisição sendo direcionada sempre para o mesmo endereço de IP. Mesmo que o protocolo esteja bloqueado no *firewall*, o usuário ativa o SSH no servidor remoto trocando sua porta de destino para uma que esteja liberada na rede, como é o caso das portas HTTP (80) e HTTPS (443). Com tais alterações, assim que for executado o tunelamento do protocolo, todo o tráfego gerado consegue alcançar seu destino, pois está sendo direcionado para uma porta aberta no *firewall*, e também não é detectado pela ferramenta de IDS, pois esta não possui uma assinatura compatível com o padrão de tráfego identificado.

A Figura 42 apresenta o comando executado para efetuar o tunelamento:

```
ezequiel@ezequiel-desktop:~$ sudo ssh  -L 3128:127.0.0.1:3128 -N -p 443 nti@187.91.198.180
The authenticity of host '[187.91.198.180]:443 ([187.91.198.180]:443)' can't be established.
ECDSA key fingerprint is 74:b0:fb:57:e2:15:c1:8f:bd:30:1a:c4:15:63:e4:dc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[187.91.198.180]:443' (ECDSA) to the list of known hosts.
nti@187.91.198.180's password:
```

Listagem 6 - Execução do túnel SSH
Fonte: Do autor.

A interpretação do comando pode ser feita da seguinte forma, primeiramente é necessário definir a porta local que receberá o tráfego do servidor remoto, neste caso foi definida a porta 3128. O endereço de IP 127.0.0.1 corresponde ao *host* interno da rede que será acessado para disponibilizar o serviço solicitado. E, por fim, é efetuado o acesso à máquina de borda com IP 187.91.198.180, a qual fará o tunelamento com o *host* interno da rede. Neste exemplo a máquina interna é o próprio *host*, mas poderia ser qualquer outra pertencente à rede, por isso o uso do endereço 127.0.0.1, a qual disponibiliza um serviço de

proxy na porta 3128 por meio de um *proxy* túnel. Um *proxy* túnel é uma conexão SSH criada por meio de um proxy até um *host* com um servidor SSH. Por meio desse túnel, podem ser trafegadas quaisquer informações sem possibilidade de bloqueio por filtro de conteúdo *web*, uma vez que todos os dados estão criptografados. Sendo assim, houve a necessidade do desenvolvimento de uma assinatura que suprisse a atual demanda, pois as existentes não desempenham essa função no cenário atual.

Para a criação da regra, num primeiro momento é necessário identificar o tráfego gerado pelo túnel SSH. Para isso, foi utilizada uma ferramenta chamada *Wireshark*[10], que, em suma, é um programa que analisa o tráfego de rede e o organiza por protocolos. Foram efetuadas diversas capturas para se estabelecer um padrão de tráfego, pois só assim é possível escrever uma assinatura. Depois de diversas execuções do túnel, foi detectado no *payload* do pacote o conteúdo apresentado na Figura 43.

Figura 37 - Captura de pacotes
Fonte: Do autor.

Com a identificação do conteúdo, é possível adquirir as informações iniciais para o desenvolvimento da regra, que se observado a Figura 43, um pacote foi enviado à *internet* gerando o tráfego desejado para poder ser analisado. A formação de um pacote desse gênero, é composto pelo *header* e o *payload*. Onde o *header* carrega as informações de onde o pacote vem e para onde o pacote vai e o p*ayload* carrega o conteúdo do pacote principal, neste caso identificado como sendo protocolo SSH.

Para o desenvolvimento da regra do *Snort*, obedeceu-se ao seguinte modelo:

<tipo_de_alerta><protocolo><rede_origem><porta_origem>-><rede_destino> <porta_destino> (cabeçalho da regra; opções; sid:X;);

Então, a partir das informações obtidas com a análise de tráfego, e um estudo dos parâmetros das assinaturas existentes, deu-se início ao trabalho de desenvolvimento chegando a obtenção da seguinte assinatura:

alert tcp any any -> any !$SSH_PORTS (msg:"Detectado tentativa de conexão via SSH"; flow: from_client,established; content:"SSH-"; offset: 0; depth: 4; classtype:policy-violation; sid:3000000; rev:1; fwsam: src, 12 hours;);

Como pode-se notar na regra acima, o modelo é obedecido, nela tem-se:

- *alert*: mensagem é do tipo alerta;
- *tcp*: protocolo de comunicação;

[10]www.wireshark.org

- *any*: rede de origem;
- *any*: porta de origem;
- - >: fuído dos pacotes;
- *any*: rede de destino;
- !SSH_PORTS: porta de destino;
- *msg*:imprime uma mensagem em alerta e registra o pacote;
- *flow*: permite a definição da direção do pacote em relação aos fluxos de comunicação cliente-servidor.;
- *content:* procura por um molde no conteúdo do pacote ;
- *offset:* especifica quantos *bytes* há no *payload* para começar a análise do conteúdo;
- *depth*: oposto da opção *offset* que pará no número de *bytes* especificado;
- *classtype:*conjunto de classificação conforme a prioridade;
- *sid:* identificador exclusivo da regra;
- *rev:* identifica revisões nas regras*;*
- *fwsam: plugins* de bloqueio;
- *src:* informa que o bloqueio é efetuado na origem*;*
- *hours:* período em que o *host* ficará bloqueado.

Com o tráfego identificado e o desenvolvimento da assinatura, foi possível colocá-la em prática para que ela identifique possíveis *proxy* túneis por meio da utilização do protocolo SSH.

A Figura 44 exibe o alerta gerado pelo ferramenta no momento em que um tráfego de padrão SSH é identificado.

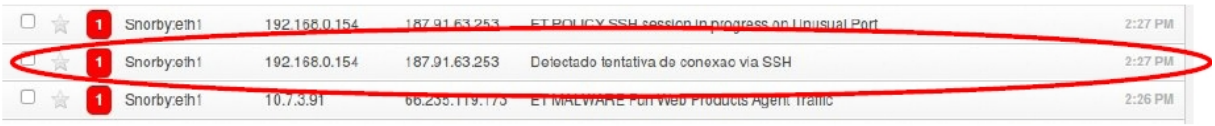

Figura 38 - Alerta de detecção de túnel SSH
Fonte: Do autor.

Quando executado o comando da Figura 42, automaticamente será detectado o tráfego indevido, fazendo com que o alerta seja exibido e, consequentemente, o *host* que originou a ameaça seja bloqueado. Conforme a Figura 44, a origem do problema é uma máquina interna da rede de IP 192.168.0.154, com destino ao *host* de IP 187.91.63.253 isto tudo é apresentado no *log* de assinatura com a mensagem do problema. Contudo, o *Snortsam* encaminhará um pedido de bloqueio ao *firewall* para barrar imediatamente o túnel.

A Figura 45 apresenta o momento em que o *host* é bloqueado.

```
2011/11/10, 14:25:22, 10.100.0.44, 2, snortsam, Blocking host 192.168.0.154 completely for 43200 seconds (Sig_ID: 3000000).
2011/11/10, 14:26:09, 10.100.0.44, 2, snortsam, Extending block for host 192.168.0.154 completely for 43200 seconds (Sig ID: 3000000).
2011/11/10, 14:34:25, 10.100.0.44, 2, snortsam, Extending block for host 192.168.0.154 completely for 43200 seconds (Sig_ID: 3000000)
```

Listagem 7 - Momento em que o *host* é bloqueado
Fonte: Do autor.

Sendo assim, todo o processo está concluído, e, enquanto o *proxy* túnel continuar a ser executado, o bloqueio é estendido, permanecendo o tempo limite da sua última intervenção. Com o desenvolvimento e a homologação da assinatura, foi possível conter eventuais acessos indevidos com o fim de burlar o sistema de segurança da rede, pois não é possível determinar quantas outras ameaças originam quando um túnel for executado na rede.

6.5.3 Análise qualitativa dos bloqueios executados

Com o intuito de mensurar a quantidade de alertas gerados durante o período de validação da implementação, constatou-se que a rede *wireless* do Centro Universitário UNIVATES foi utilizada por cerca de 13.125 usuários (fonte dos dados pode ser obtida com a equipe do Núcleo de Tecnologia da Informação em http://manet.univates.br/zabbix), os quais geraram 72.861 alertas que foram gravados em banco de dados, envolvendo desde ameaças graves até eventos de tráfego considerado normal em uma rede de computadores.

A Figura 46 exibe o número de alertas coletados:

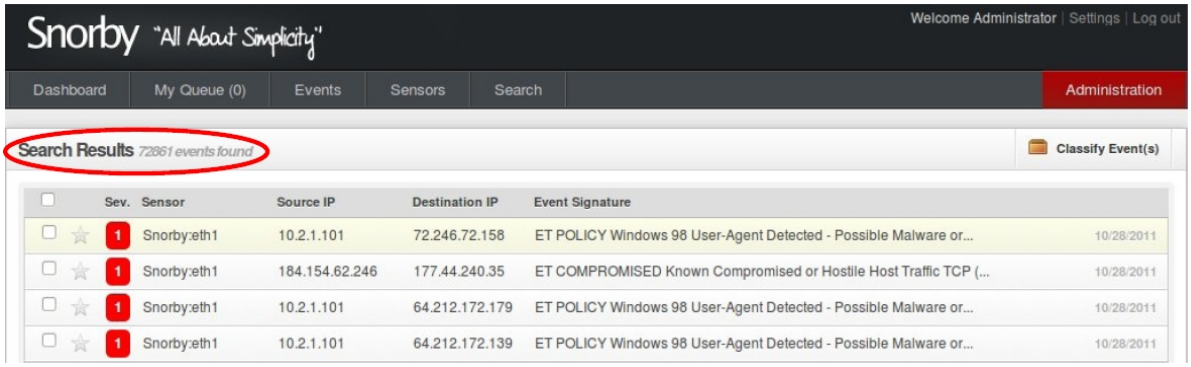

Figura 39 - Lista de eventos ocorridos durante o período de prova
Fonte: Do autor.

Foi efetuada uma análise qualitativa, pois ela estabelece padrões de comportamento verificados por meio de fatos observáveis, sendo neste caso todos os bloqueios efetuados durante o período de prova. É possível quantificar os bloqueios por meio das informações recebidas pelo software implementado, podendo assim gerar dados e apresentar os resultados.

Definiu-se uma análise qualitativa, pois não é possível mensurar quantos problemas de segurança pode-se ter após a identificação de uma ameaça. Sendo assim, obtêm-se os resultados por meio dos fatos observados, não se conseguindo prever quantos mais deles

podem acontecer. Na verdade, não se consegue determinar quantos problemas posteriores poderão ser ocasionados caso as ameaças ocorram com sucesso.

Tabela 4 - Dados semanais obtidos durante a validação do trabalho

Período	Semana 1	Semana 2	Semana 3	Semana 4
Acessos	3.685	2.748	3.140	3.552
Alertas	26.417	14.988	11.978	19.478
Bloqueios	1.486	478	364	563

Fonte: Do autor.

Os dados obtidos foram separados por semana para uma melhor visualização, sedo efetuado o somatório dos dias correspondentes a cada semana. Os dados apresentados na tabela acima, os quais apresentam o número de acessos, foram obtidos por meio do monitoramento efetuado pelo software *Zabbix* administrado pela equipe do Núcleo de Tecnologia da Informação e disponível em: http://manet.univates.br/zabbix. Quanto à parte de alertas, foi adquirida a partir da ferramenta de *front-end* do IDS chamada *Snorby* e os bloqueios foram obtidos por meio do arquivo de *log* da ferramenta *Snortsam*.

A Tabela 5 apresenta o somatório do número de usuários que fizeram uso da rede, quantidade de alertas e bloqueios gerados pelo *Snort* e pelo *Snortsam:*

Tabela 5 - Somatório dos dados obtidos no período de homologação do trabalho

	Somatório dos dados coletados
Acessos	13.125
Alertas	72.861
Bloqueios	2.891

Fonte: Do autor.

Uma observação a ser feita é quanto ao valor 13.125 correspondente ao número de acessos: este foi obtido a partir do somatório de *logins* diários não repetitivos. Foram descartados todos os acessos repetidos diariamente, obtendo-se assim somente logins únicos que utilizaram a rede, podendo-se distinguir o usuário que logou somente uma vez daqueles que efetuaram vários *logins* durante o dia. Quanto aos valores dos alertas e bloqueios, foi efetuado um somatório dos dados coletados durante as quatro semanas de validação. Com os dados obtidos, pôde-se efetuar uma análise de quanto eficaz foram os bloqueios automatizados gerados pela ferramenta. A fim de efetuar uma análise mais clara, os resultados foram repassados para um gráfico, podendo-se verificar o quanto foi eficiente a ferramenta implementada.

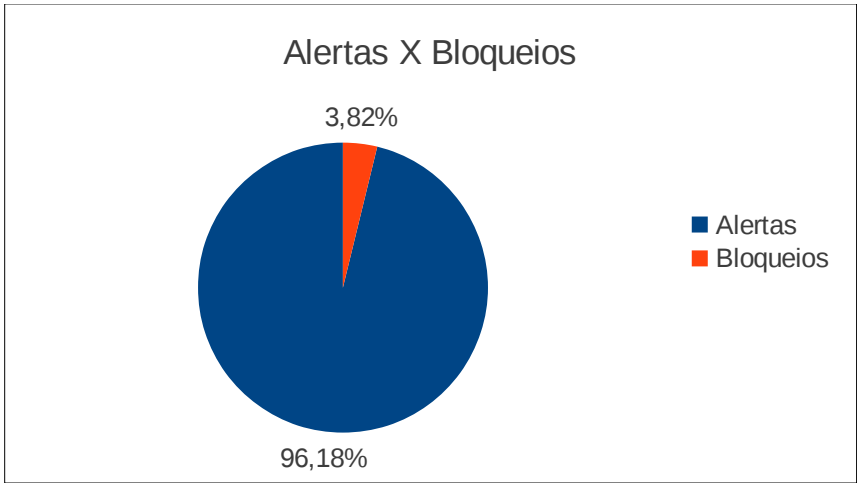

Gráfico 1 - Porcentagem de alertas x bloqueios
Fonte: Do autor.

Conforme pode ser observado no Gráfico 1, foi realizado o cruzamento dos valores obtidos na Tabela 5 a fim de mensurar a porcentagem de bloqueios efetuados durante o período de validação. Dos 72.861 alertas gerados, foram identificados 2.891 como assinaturas de tráfego indesejado, correspondendo a 3,82% do total. Em um ambiente em que não é efetuada essa ação automaticamente, seria humanamente impossível o administrador da rede conseguir identificar e bloquear todos os *hosts* aqui detectados, isto porque não se consegue identificar ou mensurar quantas ameças poderiam surgir após o primeiro incidente, ou seja, uma ameaça não detectada pode gerar outras.

A Tabela 6 apresenta o número de bloqueios por assinaturas:

Tabela 6 - Quantidade de bloqueios por assinaturas

Assinaturas	TeamViewer	Spywares	Conficker	Tunel SSH	Logmein	UltraSurf	Slowloris	Total
Número de bloqueios por assinaturas	1.864	737	217	34	27	6	6	2.891

Fonte: Do autor.

Durante o período de validação foram atingidos 2.891 *hosts* bloqueados. Vale ressaltar que muitas outras assinaturas poderiam ser identificadas e barradas pela ferramenta, mas nenhum tráfego correspondente a elas foi gerado pelos usuários durante a homologação. O Gráfico 2 apresenta o percentual de cada assinatura no total de bloqueio gerado.

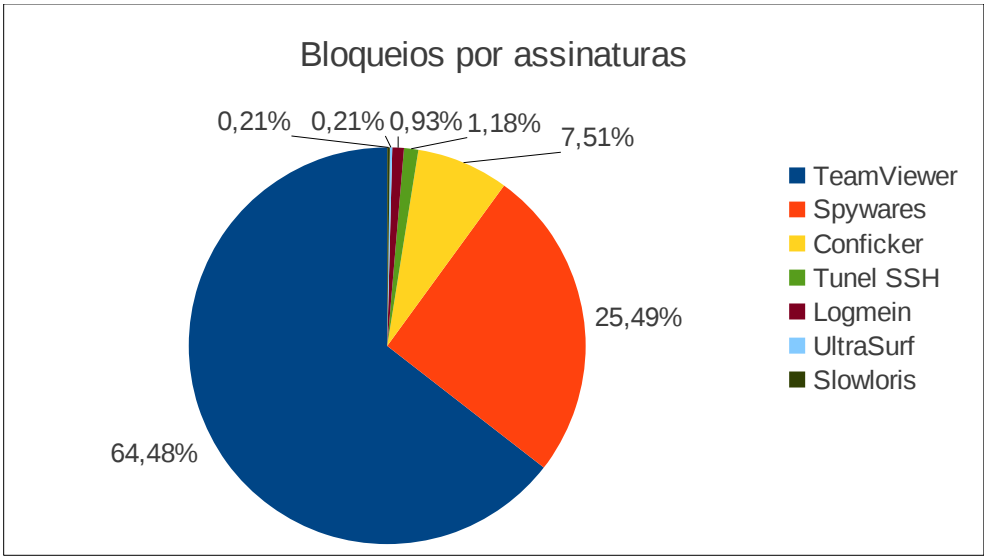

Gráfico 2 - Percentual por assinatura bloqueada
Fonte: Do autor.

Conforme pode-se observar no Gráfico 2, houve maior incidência do uso de ferramenta de acesso remoto pela *internet*. Com o software chamado *TeamViewer,* o usuário pode acessar qualquer outro *host* que esteja interconectado na rede e fazer uso dos acessos por ele disponibilizados. O segundo maior incidente, correspondendo a 25,49% dos bloqueios, foram os s*pywares, que* consistem em um programa automático de computador que recolhe informações sobre o usuário e transmite essa informação a uma entidade externa, sem o conhecimento nem o consentimento do usuário. A assinatura desenvolvida para identificar tunelamento SSH corresponde a 1,18% dos bloqueios, tornando-se mais atuante que assinaturas como *UltraSurf, Logmein* e *Slowloris,* atingindo assim o objetivo do seu desenvolvimento.

7 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um estudo da implementação de ferramenta IDS/IPS aplicada especificamente para automatizar os processos de contenção de ameaças por meio de assinaturas com padrões de tráfego.

Os resultados obtidos durante o período de avaliação demonstraram que 3,82% dos 72.861 alertas gerados foram devidamente contidos, equivalendo a um montante de 2.891 *hosts*, o que seria humanamente impossível de ser atingido sem o auxílio de uma ferramenta automatizada. Caso tal tráfego não fosse contido, a disseminação das ameaças poderiam gerar um número de incidentes significativamente maior, já que o objetivo principal da maioria dos *malwares* é a auto replicação. Com a integração do IDS/IPS com outras ferramentas de segurança, foi possível obter um relacionamento com o usuário, podendo reportar a ele o problema ocorrido, diferencial que as atuais ferramentas disponíveis no mercado não possuem.

A possibilidade de cadastrar e definir novas assinaturas de tráfego indesejado habilita a aplicação das técnicas aqui utilizadas em qualquer outra rede, além de se manter estável para qualquer peculiaridade oriunda desta.

A aplicação do IDS/IPS pode ser feita em qualquer rede de dados, desde que obedecidos os padrões de funcionamento da ferramenta, onde o ponto chave de sua utilização é o conhecimento do tráfego da rede, e a aplicação das assinaturas já existentes.

A utilização do *Snort* e *Snortsam* em modo de espelhamento, torna a implementação do monitoramento mais dinâmico, porém a contenção muito mais complexa, já que esta é realizada por outros dispositivos que devem ser interligados. Por outro lado, as falhas nos dispositivos de monitoramento e/ou contenção não interferem no funcionamento da rede.

A arquitetura de software utilizada, por ser totalmente de código aberto, possibilita que a investigação criteriosa de seus aplicativos possa resultar num ganho ainda maior, principalmente para os agentes de detecção e bloqueios os quais podem ter seus recursos melhorados.

Fica como sugestão de estudo posterior a incrementação da assinatura desenvolvida, assim como o desenvolvimento de outras assinaturas que identifiquem determinados padrões de tráfegos. Também a aplicação do IDS/IPS *inline* na rede, que requer outros padrões de implementação que não foram abordados no presente trabalho. Além disso, realizar a comparação do modelo aqui aplicado com outras ferramentas comerciais existentes no mercado, mesmo que elas apresentem recursos diferentes dos utilizados neste estudo.

REFERÊNCIAS

ALPCAN, Tansu; BASAR Tamer . **Network security:** decision and game theoretic approach. 1ª edição. Illinois: Cambridge University Press, 2010.

BEALE, Jay. **Snort 2.1 intrusion detection.** 2ª edição. Rockland :Syngress, 2004.

BEALE, Jay; FOSTER James C. **Snort 2.0 Intrusion Detection.** 1ª edição. Rockland :Syngress, 2003.

BHAIJI, Yusuf. **CCIE professional development series network security technologies and solutions.** 1ª edição. Indianapolis :Cisco Press, 2008.

BHARDWAJ, Pawan K. **A+, Network+, Security+ Exams.** 1ª edição. Sebastopol: O'Reilly, 2007.

BRENTON, Chris; HUNT, Cameron. **Active defense.** A comprehensive guide to network security. Alameda: SYBEX Inc., 2001.

CARNE, E. Bryan. **A professional's guide to data communication in a TCP/IP world.** London: Artech House Inc. 2004.

CARTER Earl; HOGUE, Jonathan. **Intrusion prevention fundamentals.** 1ª edição. Indianapolis :Cisco Press, 2006.

CASTELLI, Matthew J. **LAN Switching first-step.** Indianapolis: Cisco Press, 2004.

CASWELL, Brian; BEALE, Jay; BAKER, Andrew. **Snort IDS and IPS Toolkit.** Jay Beale's Open Source Security. Rockland:Syngress, 2007.

CIAMPA, Mark, **Security + Guide to network security fundamentals.** 3ª edição. Boston:Course Technology, 2009.

COX, Kerry J; GERG, Christopher. **Managing security with snort and IDS tools.** Sebastopol :O'Reilly, 2004.

COX, Kerry J; GERG, Christopher. **Snort and IDS Tools.** Sebastopol:O'Reilly, 2004.

DOHERTY, Jim; ANDERSON, Neil; MAGGIORA, Paul Della. **Cisco neworking simplified.** CISCO. 2 ª edição. Indianapolis: Cisco Press, 2007.

DOSTÁLEK, Libor; KABELOVÁ, Alena. **Understanding TCP/IP:** a clear and comprehensive guide to TCP/IP protocols. Birmingham: Packt Publishing Ltd, 2006.

ENDORF, Carl; SCHULTZ, Eugene; MELLANDER, Jim. **Intrusion Detection e Prevention.** 1ª edição. Chicago:McGraw-Hill Osborne Media, 2004.

HALL, Eric. **Internet core protocols**: The Definitive Guide. Sebastopol: O'Reilly, 2000.

HARRINGTON, Jan L. N**etwork security:** a practical approach. San Francisco: Elsevier Inc. 2005.

HELD, Gilbert. **Ethernet networks:** Design, Implementation, Operation, Management. 4. ed. Chichester: John Wiley & Sons Ltd, 2003.

HUNT, Craig. **TCP/IP network administration.** 3ª edição. Sebastopol: O'Reilly, 2002.

LOCKHART, Andrew. **Network security hacks.** Sebastopol:O'Reilly, 2006.

MAIA, Igor da Silva Neiva; REHEM, Sandro Herman Pereira **Sistemas de prevenção de intrusão baseado em software livre:** debian e snort. Projeto final para obtenção do grau de especialista em Rede de Computadores. Brasília, 2005 103 p. Disponível em <http://www.scribd.com/doc/13224983/Sistemas-de-Prevencao-de-Intrusao-baseado-em-Software-Livre-Igor-Neiva-e-Sandro-Herman-UCB> Acessado em: 16 jun. 2011.

MALIK, Saadat. **Network security principles and practices**. Indianapolis: Cisco Press, 2002.

MCQUERRY, Steve. **Interconnecting cisco network devices**, Part 1 (ICND1). 2ª edição. Indianapolis : Cisco Press, 2008.

NAKAMURA, Emilio Tissato; GEUS Paulo Licio de. **Segurança de redes em ambientes cooperativos.** 1 ª edição. São Paulo: Novatec, 2007.

NOONAN, Wes; DUBRAWSKY, Ido. **Firewall fundamentals.** Indianapolis: Cisco Press, 2006.

PALMAS, Luciano; PRATES, Rubens. **TCP/IP: g**uia de consulta rápida. São Paulo:Novatec, 2000.

REHMAN, Rafeeq. **Intrusion Detection with SNORT:** Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID. 1ª edição. New Jersey : Prentice Hall, 2003.

SCOTT Charlie; WOLFE, Paul; HAYES, Bert. **Snort for Dummies.** Hoboken :For Dummies, 2004.

SMITH, Sean; MARCHESINI John. **The craft of system security** 1ª edição. Boston:Addison-Wesley Professional, 2008.

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sergio. **Redes de computadores**: das LANs, MANs e WANs as Redes ATM. 6 ªedição. Rio de Janeiro: Campus, 1995.

SPORTACK, Mark A. **TCP/IP first-step.** Indianapolis: Cisco Press. 2004.

TANENBAUM, Andrew S. **Redes de computadores.** 4 ª edição. Rio de Janeiro: Campus, 1997.

TANENBAUM, Andrew S. **Computer networks**. 4 ª edição. New Jersey: Prentice Hall, 2003.

TEXEIRA, José Helvécio Júnior; et al. **Redes de computadores:** serviços, administração e segurança. São Paulo: Makron Books, 1999.

TIPTON, Harold F; KRAUSE, Micki. **Information security management handbook.** 6ª edição. Boca Raton: Auerbach Publications, 2007.

TODOROV, Dobromir. **Mechanics of user identificatlion and authentication**: fundamentals of identity management. New York: Auerbach Publications, 2007.

TORRES, Gabriel. **Redes de computadores.** 1ª edição. Rio de Janeiro:Novaterra, 2009.

VACCA, John R. **Managing information security.** 1ª edição. Burlington :Syngress, 2010.

VACCA, John R. **Network and system security.** 1ª edição. Burlington :Syngress, 2010.

VYNCKE, Eric; PAGGEN, Christopher. **LAN switch security**. What hackers know about your switches. Indianapolis: Cisco Press, 2007.

APÊNDICES

APÊNDICE A - ARQUIVO DE CONFIGURAÇÃO DO SOFTWARE SNORT

Abaixo é apresentado o arquivo de configuração utilizado para operação do software *Snort* de acordo com a estrutura montada.

Arquivo localizado em: */etc/snort/snort.conf*

```
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom configuration:
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################
# Setup the network addresses you are protecting
var HOME_NET
[200.132.250.252/32,177.44.240.29/32,177.44.240.30/32,177.44.240.34/32,177.44.240.35/32,
177.44.240.36/32,10.3.0.0/16,172.16.0.0/22,10.9.0.0/22,10.1.0.0/24,10.6.0.0/22,10.2.1.0/24,1
0.7.0.0/22,10.8.0.0/22,192.168.0.0/22,10.100.0.0/24]
# Set up the external network addresses. Leave as "any" in most situations
var EXTERNAL_NET !$HOME_NET
# List of DNS servers on your network
var DNS_SERVERS $HOME_NET
# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET
# List of web servers on your network
var HTTP_SERVERS $HOME_NET
# List of sql servers on your network
```

```
var SQL_SERVERS $HOME_NET
# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET
# List of ssh servers on your network
var SSH_SERVERS $HOME_NET
# List of ftp servers on your network
var FTP_SERVERS $HOME_NET
# List of ports you run web servers on
portvar HTTP_PORTS
[80,311,591,593,901,1220,1414,1830,2301,2381,2809,3128,3702,5250,7001,7777,7779,8000
,8008,8028,8080,8088,8118,8123,8180,8181,8243,8280,8888,9090,9091,9443,9999,11371]
# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80
# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:
# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22
# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]
# other variables, these should not be modified
var AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,2
05.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.24
8.0/24]
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################
# Stop generic decode events:
```

```
config disable_decode_alerts
# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts
# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts
# Stop Alerts on T/TCP alerts
config disable_tcpopt_ttcp_alerts
# Stop Alerts on all other TCPOption type events:
config disable_tcpopt_alerts
# Stop Alerts on invalid ip options
config disable_ipopt_alerts
# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet
# config enable_decode_oversized_alerts
# Same as above, but drop packet if in Inline mode (requires
enable_decode_oversized_alerts)
# config enable_decode_oversized_drops
# Configure IP / TCP checksum mode
config checksum_mode: all
# Configure maximum number of flowbit references.  For more information, see
README.flowbits
# config flowbits_size: 64
# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53
# Configure active response for non inline operation. For more information, see
REAMDE.active
# config response: eth0 attempts 2
######################################################
# Step #3: Configure the base detection engine.  For more information, see
README.decode
######################################################
# Configure PCRE match limitations
config pcre_match_limit: 3500
config pcre_match_limit_recursion: 1500
```

```
# Configure the detection engine  See the Snort Manual, Configuring Snort - Includes - Config
config detection: search-method ac-split search-optimize max-pattern-len 20
# Configure the event queue.  For more information, see README.event_queue
config event_queue: max_queue 8 log 3 order_events content_length
##################################################
# Per packet and rule latency enforcement
# For more information see README.ppm
##################################################
# Per Packet latency configuration
#config ppm: max-pkt-time 250, \
#   fastpath-expensive-packets, \
#   pkt-log
# Per Rule latency configuration
#config ppm: max-rule-time 200, \
#   threshold 3, \
#   suspend-expensive-rules, \
#   suspend-timeout 20, \
#   rule-log alert
##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################
#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks
##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################
# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
# path to dynamic rules libraries
```

```
dynamicdetection directory /usr/local/lib/snort_dynamicrules
##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
##################################################
# Inline packet normalization. For more information, see README.normalize
# Does nothing in IDS mode
#preprocessor normalize_ip4
#preprocessor normalize_tcp: ips ecn stream
#preprocessor normalize_icmp4
#preprocessor normalize_ip6
#preprocessor normalize_icmp6
# Target-based IP defragmentation.  For more inforation, see README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10
min_fragment_length 100 timeout 180
# Target-Based stateful inspection/stream reassembly.  For more inforation, see
README.stream5
preprocessor stream5_global: max_tcp 8192, track_tcp yes, track_udp yes, track_icmp
no max_active_responses 2 min_response_seconds 5
preprocessor stream5_tcp: policy windows, \
#detect_anomalies, require_3whs 180, \
#   overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
   ports client 21 22 23 25 42 53 79 109 110 111 113 119 135 136 137 139 143 \
      161 445 513 514 587 593 691 1433 1521 2100 3306 6070 6665 6666 6667 6668
6669 \
      7000 8181 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \
   ports both 80 311 443 465 563 591 593 636 901 989 992 993 994 995 1220 1414
1830 2301 2381 2809 3128 3702 5250 7907 7001 7802 7777 7779 \
      7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913
7914 7915 7916 \
      7917 7918 7919 7920 8000 8008 8028 8080 8088 8118 8123 8180 8243 8280
8888 9090 9091 9443 9999 11371
preprocessor stream5_udp: timeout 180
```

```
# performance statistics.  For more information, see the Snort Manual, Configuring Snort - Preprocessors - Performance Monitor
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000
# HTTP normalization and anomaly detection.  For more information, see README.http_inspect
preprocessor http_inspect: global iis_unicode_map unicode.map 1252
preprocessor http_inspect_server: server default \
   chunk_length 500000 \
   server_flow_depth 0 \
   client_flow_depth 0 \
   post_depth 65495 \
#        oversize_dir_length 500 \
#    max_header_length 750 \
#    max_headers 100 \
   ports { 80 311 591 593 901 1220 1414 1830 2301 2381 2809 3128 3702 5250 7001 7777 7779 8000 8008 8028 8080 8088 8118 8123 8180 8181 8243 8280 8888 9090 9091 9443 9999 11371 } \
#   non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
  enable_cookie \
#   extended_response_inspection \
  normalize_utf \
   apache_whitespace no \
   ascii no \
   bare_byte no \
   base36 no \
        directory no \
        double_decode no \
        iis_backslash no \
        iis_delimiter no \
        iis_unicode no \
        multi_slash no \
 #      utf_8 no \
        u_encode yes \
        webroot no
```

```
# ONC-RPC normalization and anomaly detection.  For more information, see the Snort Manual, Configuring Snort - Preprocessors - RPC Decode
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no_alert_multiple_requests no_alert_large_fragments no_alert_incomplete
# Back Orifice detection.
preprocessor bo
# FTP / Telnet normalization and anomaly detection.  For more information, see README.ftptelnet
preprocessor ftp_telnet: global inspection_type stateful encrypted_traffic no
preprocessor ftp_telnet_protocol: telnet \
  ayt_attack_thresh 20 \
  normalize ports { 23 } \
  detect_anomalies
preprocessor ftp_telnet_protocol: ftp server default \
  def_max_param_len 100 \
  ports { 21 2100 3535 } \
  telnet_cmds yes \
  ignore_telnet_erase_cmds yes \
  ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
  ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
  ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
  ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
  ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
  ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
  ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
  ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
  ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
  ftp_cmds { XSEN XSHA1 XSHA256 } \
  alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP PASV PWD QUIT REIN STOU SYST XCUP XPWD } \
  alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR STOU XMKD } \
  alt_max_param_len 256 { CWD RNTO } \
  alt_max_param_len 400 { PORT } \
```

```
        alt_max_param_len 512 { SIZE } \
        chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
        chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
        chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
        chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \
        chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \
        chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \
        chk_str_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \
        chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \
        cmd_validity ALLO < int [ char R int ] > \
        cmd_validity EPSV < [ { char 12 | char A char L char L } ] > \
        cmd_validity MACB < string > \
        cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
        cmd_validity MODE < char ASBCZ > \
        cmd_validity PORT < host_port > \
        cmd_validity PROT < char CSEP > \
        cmd_validity STRU < char FRPO [ string ] > \
        cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [ number ] } >
    preprocessor ftp_telnet_protocol: ftp client default \
        max_resp_len 256 \
        bounce yes \
        ignore_telnet_erase_cmds yes \
        telnet_cmds yes
    # SMTP normalization and anomaly detection.  For more information, see
README.SMTP
    preprocessor smtp: ports { 25 465 587 691 } \
        inspection_type stateful \
        enable_mime_decoding \
        max_mime_depth 20480 \
        normalize cmds \
        normalize_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO
EMAL ESAM ESND ESOM ETRN EVFY } \
        normalize_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT
RCPT RSET SAML SEND SOML } \
```

```
        normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-
ADAT X-DRCP X-ERCP X-EXCH50 } \
        normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50
XGEN XLICENSE XQUE XSTA XTRN XUSR } \
        max_command_line_len 512 \
        max_header_line_len 1000 \
        max_response_line_len 512 \
        alt_max_command_line_len 260 { MAIL } \
        alt_max_command_line_len 300 { RCPT } \
        alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
        alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG
EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \
        alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN ETRN
DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB X-EXPS X-
LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN
XUSR } \
        valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL
ESAM ESND ESOM ETRN EVFY } \
        valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT
RCPT RSET SAML SEND SOML } \
        valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT
X-DRCP X-ERCP X-EXCH50 } \
        valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN
XLICENSE XQUE XSTA XTRN XUSR } \
        xlink2state { enabled }
    # Portscan detection.  For more information, see README.sfportscan
    #preprocessor sfportscan: proto  { all } memcap { 10000000 } sense_level { low }
    # ARP spoof detection.  For more information, see the Snort Manual - Configuring
Snort - Preprocessors - ARP Spoof Preprocessor
    # preprocessor arpspoof
    # preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
    # SSH anomaly detection.  For more information, see README.ssh
    preprocessor ssh: server_ports { 22 } \
            autodetect \
```

```
                max_client_bytes 19600 \
                max_encrypted_packets 20 \
                max_server_version_len 100 \
                enable_respoverflow enable_ssh1crc32 \
                enable_srvoverflow enable_protomismatch
        # SMB / DCE-RPC normalization and anomaly detection.  For more information, see
README.dcerpc2
        preprocessor dcerpc2: memcap 102400, events [co ]
        preprocessor dcerpc2_server: default, policy WinXP, \
           detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \
           autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \
           smb_max_chain 3, smb_invalid_shares ["C$", "D$", "ADMIN$"]
        # DNS anomaly detection.  For more information, see README.dns
        preprocessor dns: ports { 53 } enable_rdata_overflow
        # SSL anomaly detection and traffic bypass.  For more information, see README.ssl
        preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 7801 7802 7900 7901
7902 7903 7904 7905 7906 7907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917
7918 7919 7920 }, trustservers, noinspect_encrypted
        # SDF sensitive data preprocessor.  For more information see
README.sensitive_data
        preprocessor sensitive_data: alert_threshold 25
        ##################################################
        # Step #6: Configure output plugins
        # For more information, see Snort Manual, Configuring Snort - Output Modules
        ##################################################
        # unified2
        # Recommended for most installs
        # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
        output unified2: filename snort.out, limit 128
        # Additional configuration for specific types of installs
        # output alert_unified2: filename snort.alert, limit 128, nostamp
        # output log_unified2: filename snort.log, limit 128, nostamp
        # syslog
```

```
# output alert_syslog: LOG_AUTH LOG_ALERT
# pcap
# output log_tcpdump: tcpdump.log
# database
# output database: alert, <db_type>, user=<username> password=<password> test dbname=<name> host=<hostname>
#output database: log, mysql, user=root password=teste123 dbname=snorby host=localhost
# prelude
# output alert_prelude
########################################################
# snortsam
########################################################
# In order to cause Snort to send a blocking request to the SnortSam agent,
# that agent has to be listed, including the port it listens on,
# and the encryption key it is using. The statement for that is:
# output alert_fwsam: {SnortSam Station}:{port}/{password}
#  {SnortSam Station}: IP address or host name of the host where SnortSam is running.
#  {port}:            The port the remote SnortSam agent listens on.
#  {password}:        The password, or key, used for encryption of the
#                     communication to the remote agent.
# At the very least, the IP address or host name of the host running SnortSam
# needs to be specified. If the port is omitted, it defaults to TCP port 898.
# If the password is omitted, it defaults to a preset password.
# (In which case it needs to be omitted on the SnortSam agent as well)
# More than one host can be specified, but has to be done on the same line.
# Just separate them with one or more spaces.
# Examples:
# output alert_fwsam: firewall/idspassword
# output alert_fwsam: fw1.domain.tld:898/mykey
output alert_fwsam: localhost:898
# metadata reference data.  do not modify these lines
include classification.config
```

```
include reference.config
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
# NOTE: All categories are enabled in this conf file
##################################################
# site specific rules
#include $RULE_PATH/local.rules
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/backdoor.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
#include $RULE_PATH/chat.rules
#include $RULE_PATH/content-replace.rules
include $RULE_PATH/ddos.rules
#include $RULE_PATH/dns.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/exploit.rules
#include $RULE_PATH/finger.rules
#include $RULE_PATH/ftp.rules
#include $RULE_PATH/icmp.rules
#include $RULE_PATH/icmp-info.rules
#include $RULE_PATH/imap.rules
#include $RULE_PATH/info.rules
#include $RULE_PATH/misc.rules
#include $RULE_PATH/multimedia.rules
#include $RULE_PATH/mysql.rules
#include $RULE_PATH/netbios.rules
#include $RULE_PATH/nntp.rules
#include $RULE_PATH/oracle.rules
#include $RULE_PATH/other-ids.rules
#include $RULE_PATH/p2p.rules
#include $RULE_PATH/phishing-spam.rules
```

```
#include $RULE_PATH/policy.rules
#include $RULE_PATH/pop2.rules
#include $RULE_PATH/pop3.rules
#include $RULE_PATH/rpc.rules
#include $RULE_PATH/rservices.rules
#include $RULE_PATH/scada.rules
include $RULE_PATH/scan.rules
#include $RULE_PATH/shellcode.rules
#include $RULE_PATH/smtp.rules
#include $RULE_PATH/snmp.rules
include $RULE_PATH/specific-threats.rules
include $RULE_PATH/spyware-put.rules
#include $RULE_PATH/sql.rules
include $RULE_PATH/telnet.rules
#include $RULE_PATH/tftp.rules
include $RULE_PATH/virus.rules
#include $RULE_PATH/voip.rules
#include $RULE_PATH/web-activex.rules
include $RULE_PATH/web-attacks.rules
#include $RULE_PATH/web-cgi.rules
#include $RULE_PATH/web-client.rules
#include $RULE_PATH/web-coldfusion.rules
#include $RULE_PATH/web-frontpage.rules
#include $RULE_PATH/web-iis.rules
#include $RULE_PATH/web-misc.rules
#include $RULE_PATH/web-php.rules
#include $RULE_PATH/x11.rules
include $RULE_PATH/emerging.conf
##################################################
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
##################################################
# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
```

```
# include $PREPROC_RULE_PATH/decoder.rules
# include $PREPROC_RULE_PATH/sensitive-data.rules
##################################################
# Step #9: Customize your Shared Object Snort Rules
# For more information, see http://vrt-sourcefire.blogspot.com/2009/01/using-vrt-certified-shared-object-rules.html
##################################################
# dynamic library rules
#include $SO_RULE_PATH/bad-traffic.rules
include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/icmp.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nntp.rules
# include $SO_RULE_PATH/p2p.rules
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/sql.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules
# Event thresholding or suppression commands. See threshold.conf
include threshold.conf
```

APÊNDICE B - ARQUIVO DE CONFIGURAÇÃO DO SOFTWARE SNORTSAM

Arquivo: */usr/local/etc/snortsam/snortsam.conf*

```
# snortsam.conf.sample - Frank Knobbe <frank@knobbe.us>
# This is a sample configuration file (derived from README.conf).
# On Windows systems it is called snortsam.cfg by default and is located in
# the same directory where SnortSam.exe resides. On Unix systems, the default
# is /etc/snortsam.conf. Please remove the # to enable an option.
# The config file is a text file containing one or more of the following lines.
# SnortSam specific options:
# a remark
#   Lines starting with # or ; are remarks. All text after # (or ;) is
#   truncated which means that you can list an # after a valid option as well.
#   If you intend to use a # (or ;) as part of an option, you have to escape it
#   with a back-slash, for example:
#     <option> This is a \# valid char  # But this is a comment
#   This would translate after parsing to:
#     <option> This is a # valid char
#defaultkey teste123
#   Set's the default key for ALL allowed hosts to <key>.
#   The default key is used when no other key is specified in an ACCEPT option.
#   You have to use the same key in the snort.conf file in the
#   "output alert_fwsam line". If the keys, or passwords if you will, don't
#   match, SnortSam can not decrypt the request from Snort and ignore it.
#   Example:   defaultkey mydefaultpassword
#   If omitted, SnortSam will use a default key (in which case it would have to
#   omitted in snort.conf as well).
port 898
#   This set's the listening port to <port>.
#   Example:   port 666
#   It defaults to 898 if this line is omitted.
accept 10.100.0.44
accept localhost
```

```
accept 127.0.0.1
#   This option lists Snort sensors that SnortSam is accepting packets from.
#   You can specify hostname, IP address, IP address and network mask, and
#   optionally an encryption key used configured for that host or network.
#   Examples:  accept 10.10.0.0/16, officepassword
#              accept snort1, hostpassword
#              accept 192.168.1.1
#   If the password is omitted, the default key specified with DEFAULTKEY will
#   be used. You can only specify one host per line, but you can supply
#   unlimited lines.
# keyinterval <time>
#   This causes the agent to request/create a new encryption key every <time>.
#   If this line is omitted, the key lifetime defaults to 4 hours. You can use
#   'hours', 'days', 'months', 'weeks', 'years' in the duration.
#   Example:   keyinterval 30 minutes
dontblock 10.1.0.0/24
dontblock 172.18.7.0/24
dontblock 10.100.0.0/24
dontblock 200.132.250.252
#   This adds the host or network to the white-list of hosts/networks that will
#   never be blocked or unblocked. Blocking or unblocking request for hosts
#   on this list are ignored.
#   Examples:  dontblock a.root-servers.net
#              dontblock 192.168.10.0/24
#   Only one host/network per line can be specified, but you can list unlimited
#   lines
# onlyblock <host>/<mask>
# onlyunblock <host>/<mask>
#   If this is specified, Snortsam will only block IP address that match this
#   list of IP's or networks. All other block requests are ignored. The same
#   applies to unblocks if the "onlyunblock" keyword is specified. Uses for
#   the latter might be limited, but it's available. DONTBLOCK still applies
#   within this list.
#   Examples:  onlyblock 10.0.0.0/8
```

```
#   Only one host/network per line can be specified, but you can list unlimited
#   lines
# override <host>/<mask>,<time>
#override 10.0.0.0/8, 5 min
#   Each Snort rule requests its own time interval for the blocking request.
#   Here on the agent, you can override the duration with a specified value.
#   This is good for proxy servers, or other situations, where an attacker
#   'shares' an IP address with other hosts/users that you don't want to
#   block for long. (You don't want to block ALL of AOL for a week... :)
#   Examples:  override proxy.aol.com, 5 min
#              override 192.168.1.0/24, 10 sec
# upperlimit <host>/<mask>,<time>
# limit <host>/<mask>,<time>
#   This statement allows you to set a maximum time duration for all SID/blocks
#   dependent on the reporting sensor. It acts like 'override', but instead of
#   setting a new duration, this statement limits the duration to the defined
#   maximum blocktime. Note that the host/network refers to a Snort sensor
#   (or a forwarding Snortsam station) and does not refer to the IP address to
#   be blocked like 'override' does.
#   Example:   limit 192.168.1.0/24, 2 weeks
# lowerlimit <host>/<mask>,<time>
# atleast <host>/<mask>,<time>
#   This statement allows you to set a minimum time duration for all SID/blocks
#   dependent on the reporting sensor. It is the complement to 'limit', but
#   instead of reducing the duration to a maximum limit, it bumps up any
#   duration that is lower to this minimum duration.
#   Example:   limit 192.168.1.0/24, 1 day
# denysidfrom <host>/<mask>: <sid>,<sid>,...
#denysidfrom 10.7.0.0/22: 2008739, 2008738, 2008737, 2009201, 2009200, 2009114,
2009024
#   This statement causes Snortsam to ignore blocking requests for particular
#   SID based on a given sensor. Either a single SID or multiple SIDs can be
#   listed. When listing multiple SIDs, make sure you separate them with commas
#   and not just spaces.
```

```
#   Examples:  denysidfrom 192.168.1.0/24: 1345
#              denysidfrom othersnortsam.someone.net: 1411, 1422, 0, 2002123
#   (Note the use of SID 0 which is typically used for manual blocks/unblocks
#   when no SID is specified, for example, on the command line when using the
#   samtool.)
# allowsidfrom <host>/<mask>: <sid>,<sid>,...
#allowsidfrom 10.8.0.0/22: 5998
#   This statement is the invert of 'denysidfrom'. It will cause Snortsam to
#   only accept the SIDs listed and by default ignore all other SIDs.
#   If a conflict exist by the same SID appearing in an 'allowsidfrom' line and
#   a 'denysidfrom' line, the deny takes priority.
#   Example:   allowsidfrom 10.0.0.0/8: 3200, 3201, 3203, 4332, 4333, 4334
# rollbackhosts <amount>
#   This tells SnortSam to keep a record of <amount> last blocks for each
#   Snort sensor. These blocks will be rolled back, meaning the hosts
#   unblocked, in the event that the blocking threshold is exceeded.
#   Example:   rollbackhosts 50
#   If omitted, SnortSam will not keep a record of the IP addresses that have
#   been blocked for rollback purposes.
# rollbackthreshold <amount> / <time>
#   This specifies the blocking threshold. If the threshold is exceeded (more
#   than <amount> blocking requests in <time>), SnortSam will unblock the last
#   <x> hosts specified by the ROLLBACKHOSTS statement.
#   Example:   rollbackthreshold 20 / 30 secs
# rollbacksleeptime <time>
#   When the rollback threshold has been exceeded, SnortSam will ignore
#   blocking requests until the level drops back below the threshold. Using
#   this option you can specify an additional time period that SnortSam will
#   wait until it starts acting on blocking requests again.
#   Example:   rollbacksleeptime 1 minute
#   If omitted, and the rollback mechanism is used, it defaults to 15 minutes.
# skipinterval <time>
#   SnortSam skips repetitive, identical blocking requests (for performance
#   reasons). Here you specify the time interval for which blocks are
```

```
#   considered repetitive.
#   Example:   skipinterval 30 secs
#   If omitted, SnortSam will use a default time period of 10 seconds in which
#   it considers requests to be repetitive.
# skiphosts <amount>
#   Tells SnortSam how many hosts it should remember for repetitive block
#   checks.
#   Example:   skiphosts 10
#   If omitted, SnortSam will remember a default of 10 hosts.
logfile /var/log/snortsam.log
#   SnortSam will use this file to log certain events such as program start,
#   block/unblock actions performed and error events. If only a file name is
#   specified (without a path), the file will be created a) on Windows systems
#   in the same directory where SnortSam.exe resides, and b) on Unix systems
#   in /var/log.
#   Example:   logfile snortsam.log
#   No logging occurs if this line is omitted.
#loglevel 3
#   The file logging level can be set to 0, 1, 2, or 3:
#     0: Quiet - No logging occurs.
#     1: Sparse - Only errors are logged.
#     2: Normal - Errors and blocks are logged.
#     3: Verbose - Additional information (such as connections/disconnections)
#       are logged as well.
#   Example:   loglevel 2
#   If omitted, a level of 2 (normal logging) is assumed.
# screenlevel <level>
#   The logging level, just like loglevel, but for screen output.
#    (See above for values)
#   Example:   screenlevel 3
#   If omitted, a level of 2 (normal logging) is assumed.
# include <file>
include /usr/local/etc/snortsam/rootservers.cfg
#   This statement includes another configuration file. Only one level of
```

```
#   inclusion is supported.
#   Example:   include dontblocklist.cfg
# statefile <filename>
statefile /var/db/snortsam.state
#   SnortSam will use this file name for the state file instead of the default.
#   This avoids conflicts on hosts with mutliple Snortsam instances.
#   The default of /var/db/snortsam.state (or snortsam.sta on Windows) is used
#   if this line is omitted.
#   Example:   statefile /var/db/2nd-snortsam.state
# avoidstatefile
#   Starting with version 2.8, SnortSam will always keep a state file so the
#   additions to dontblock-list can be checked against current blocks (and
#   unblocked automatically if a host is on the DONTBLOCK list, but had been
#   blocked before). If you are using SnortSam only to block on Checkpoint
#   firewalls, you could avoid the state file since FW-1 will time-out blocks
#   by itself. To do that, just use this statement in the config file.
#   Example:   avoidstatefile
#   Note that if you load a plugin that requires SnortSam to unblock the
#   blocks, and thus requires the state file, it will be created regardless if
#   this option is present.
# disablereverselookups
#   This option turns off reverse name resolution in logging plugins, currently
#   only used by the email plugin.
#   Example:   disablereverselookups
# disablepersistentconnections
# disablepersistenttcp
#   This option turns off persistent TCP connections for the FORWARD plugin as
#   introduced with version 2.51. It also does not use persistent connections
#   for connecting hosts like the Snort plugin, a remote forwarder, or the
#   samtool. In essence, Snortsam will behave like pre-2.51 versions.
#   The default is now to leave persistent-TCP disabled. See also below.
#   Example:   disablepersistentconnections
# enablepersistentconnections
# enablepersistenttcp
```

This option turns on persistent TCP connections for the FORWARD plugin as
introduced with version 2.51. It also accepts persistent connections from
connecting hosts like the Snort plugin, a remote forwarder, or the samtool.
In essence, Snortsam will behave like pre-2.51 versions.
By default, persistens TCP connections are disabled now, and you need this
option to forcefully enable it. Beware, persistent TCP connections are
still experimental and may cause problems.
Example: enablepersistentconnections
disableseqnocheck
This option turns off sequence number checking in SnortSam. SeqNo
violations are currently not punished (by banning the offending Snort
sensor), but it was planned to do so in the future to increase security.
Use this option to turn packet sequence number checking off.
Example: disableseqnocheck
holdsnort
This option requires version 1.13 or higher of the Snort plugin. It places
Snort 'on hold' during processing of the blocking request, and resumes
Snort once the block is completed.
Example: holdsnort
THIS WILL SLOW SNORT DOWN! USE ONLY FOR TESTING OR IN CONJUNCTION WITH BARNYARD!
nothreads
This option disables the multi-threading capability and causes SnortSam
not to use thread functions at all. Instead, all plugins are executed
sequentially within the main process. This makes SnortSam behave like the
old, single-threaded version 1. It is useful for testing, or if you
encounter problems with plugins that have problems with POSIX threads.
Example: nothreads
forcethreads
This option forces use of multi-threading capability on systems that have
it disabled by default, which currently is all Linux versions.
Example: forcethreads
daemon
This option causes Snortsam to turn into a daemon upon startup. It is

```
#   similar to the -D option of many other tools and services.
#   Example:   daemon
#bindip 10.100.0.44
#bindip 127.0.0.1
#   This option causes Snortsam to bind only to one IP address (or interface)
#   instead of listening on all interfaces/addresses.
#   Example:   bindip 192.168.0.1
# Firewall specific options:
# fwexec <path/fw.exe>
#   If specified, SnortSam will call the fw.exe executable to create the blocks
#   on Firewall-1. Normally you would use either 'fwsam' or 'opsec' (see
#   below). This line is useful if there are problems with OPSEC or you don't
#   want to send packets to the firewall. SnortSam will have to run on the
#   FW-1 host of course.
#   Example:   fwexec c:\winnt\fw\bin\fw.exe
#fwsam 127.0.0.1
#   This statement tells SnortSam to use the self-assembled OPSEC packet to
#   initiate blocks. You have to specify the name or IP address of the
#   firewall to which to send the block. You can only list one IP address per
#   line, but supply unlimited lines (one for each firewall you have).
#   Examples:  fwsam 127.0.0.1
#              fwsam wanfw.corp.com
# fwsamipflip
#   The fwsam method should block the correct IP address if SnortSam is run on
#   the firewall host itself. However, if SnortSam runs on a small-endian box,
#   and FW-1 runs on a big-endian box, it may block the reversed IP address.
#   Use this option to flip it back to normal.
#   Example:   fwsamipflip
# opsec <file>
#  This statement tells SnortSam to use the OPSEC API functions of the OPSEC
#  plug-in, configured through the <file> config file (opsec.conf is available
#  as an example. Also see README.opsec). Use this instead of fwsam for use of
#  the official OPSEC API. You can add more than one config to allow more than
#  one firewall to execute the block (each firewall would need its own conf
```

```
# file). Currently, only clear-text is supported, but you may have luck with
# auth_port or SSL. If so, please let me know.
# Examples:  opsec opsec.conf
#            opsec wan_firewall.conf
# In opsec.conf, or whatever your file is named, change the IP of the server
# to reflect your firewalls IP (or leave at 127.0.0.1 if you run SnortSam on
# the firewall itself).
# NOTE TO ABOVE METHODS:
# If you are blocking on Checkpoint Firewall-1, use ONE OF THE THREE METHODS
# listed above. You don't have to specify them all. It is your choice which
# method to use (although I personally recommend fwsam).
# pix <ip> <telnetpw> <enablepw>
# pix <ip> <username>/<password> <enablepw>
#   This statement tells SnortSam to use the PIX plugin. SnortSam will telnet
#   into the PIX at address <ip>, log in with the given telnet and enable
#   password, and use the SHUN command to block IP addresses. If the second
#   parameter contains a /, SnortSam will use the word before the / as the
#   username and the remainder as the user password. This is useful when a PIX
#   has been configured to use RADIUS or TACACS for login authentication.
#   Examples:  pix 1.2.3.4 letmein enableme
#   If the enable password is omitted, the telnet password will be used at the
#   telnet and enable prompt.
# ciscoacl <ip> <telnetpw> <enablepw> <acl_filename>
# ciscoacl <ip> <username>/<password> <enablepw> <acl_filename>
#   This statement tells SnortSam to use the Cisco ACL plugin to block IP's on
#   a Cisco router. SnortSam will telnet into router at address <ip>, log in
#   with the <telnetpw> as the password at the telnet prompt, or use
#   <username> and <password> if TACACS is used for authentication, and modify
#   the Access Control List. You need to supply your baseline configuration
#   file <acl_filename> in the configuration line. SnortSam will insert ACL
#   statements so that access from and to the intruding IP address is denied,
#   and upload the config to the router.
#   Example:   ciscoacl 1.2.3.4 telnetpw enablepw myconfig
```

```
#   If the router is configured to authenticate access with TACACS, you would
#   use:
#           ciscoacl 1.2.3.4 user/password enablepw myconfig
# cisconullroute <ip> <telnetpw> <enablepw>
# cisconullroute <ip> <username>/<password> <enablepw>
#   This statement tells SnortSam to use the Cisco Null-Route plugin to block
#   IP's on a Cisco router. SnortSam will telnet into router at address <ip>,
#   log in with the <telnetpw> as the password at the telnet prompt, or use
#   <username> and <password> if TACACS is used for authentication, and issue
#   a route command that will "null-route" the IP to be blocked. It will then
#   save the configuration to memory. Once the block has expired, Snortsam
#   again log in and remove the added route, saving the config to memory.
#   Example:   cisconullroute 1.2.3.4 telnetpw enablepw
#   If the router is configured to authenticate access with TACACS, you would
#   use:
#           cisconullroute 1.2.3.4 user/password enablepw
# cisconullroute2 r=<ip> p=telnetpw e=enablepw
# cisconullroute2 r=<ip> p=telnetpw e=enablepw t=<tag>
# cisconullroute2 r=<ip> u=username p=password e=enablepw t=<tag> a=y
#   This statement tells SnortSam to use the Cisco Null-Route2 plugin to block
#   IP's on a Cisco router. This is a more flexible version of the
#   Cisco Null-Route plugin (see above) with a few more options.
#   You can specify a 'route tag' to mark the route on the router.
#   Eg. t=667, would result in 'ip route x.x.x.x 255.255.255.255 Null0 tag 667'.
#   You can also set the auto-enable option to y (a=y), if SnortSam should NOT
#   run the enable command because it enters directly in eg. priv-level 15
#   Parameters:
#   r=<router ip> (required)
#   u=<username> (optional)
#   p=<password> (required)
#   e=<enable password> (optional)
#   t=<route tag> (optional [1-4294967295])
#   a=<auto-enable> (optional, [yn])
#   Example:
```

```
#   cisconullroute2 r=1.2.3.4 u=username p=password e=enablepw t=666 a=y
#   cisconullroute2 r=1.2.3.4 p=telnetpw e=enablepw t=666
#   cisconullroute2 r=1.2.3.4 p=telnetpw e=enablepw
# email <smtpserver>:<port> <recipient> <sender>
#   This statement sends an email for every block and unblock event. You
#   specify your SMTP server by name or IP address, and the email address you
#   want to send the notification to. Only one recipient per line is supported,
#   more than one line be be specified. By default, SnortSam will send the
#   email from SnortSam@<hostname>, but you can override the sender by adding
#   a specific sender after the recipient. Also, you can optionally specify
#   a custom port in case you run SMTP on a different port. (Default is 25)
#   Example:   email mailserver.mydom.com root@mydom.com
#              email 127.0.0.1 admin@mydom.com SnortSam@mydom.com
#              email localhost:10025 ops@mydom.com
# email-blocks-only <smtpserver>:<port> <recipient> <sender>
#email-blocks-only mercurio.univates.br laschneiders@univates.br snort@univates.br
#   This statement is the same as "email" except that it only sends emails for
#   block events, not unblock events. This was easier to implement as a plugin
#   since "email" requires the creation of a state file while
#   "email-blocks-only" does not (see also "avoidstatefile").
#   Example:   email mailserver.mydom.com root@mydom.com
# netscreen <ip> <login id> <login password> <optional groupname> <opt zone
name>
#   This statement will activate the Netscreen plugin. It is similar to the PIX
#   plugin in that is telnets into the firewall, but instead of issuing a shun
#   command (which the Netscreen doesn't have), it adds the IP to be blocked to
#   a group which you can use for a global 'deny' rule. For more info, please
#   see README.netscreen.
#   Example:   netscreen 10.0.0.1 admin mypassword MyBlockGroup MyZone
#   If the group name is omitted, SnortSam will add/remove IP's to/from the
#   default group called 'SnortSam'.
#   Also, one can override the default zone name with a custom zone name. The
#   MyZone parameter is optional. If used, a block group name must also be
#   specified.
```

```
# ipf <adapter> <loglevel>
#   This plugin will execute the command ipf locally and block the host by
#   adding a rule to the ipf policy. You have to specify the adapter to block
#   on (for example, fxp0) and you can optionally add a logging facility
#   (default is local7.info).
#   Example:   ipf ep0 local7.info
# pf <adapter> <logoption>
#   This plugin will use an ioctl syscall to control the pf device in order to
#   block the host by adding a rule to the active rule set of pf. You have to
#   specify the adapter to block on (for example, fxp0) and you can optionally
#   add a log option (log, logall).
#   Example:   pf dc0 log
# pf2 <anchor> <table> <kill>
#   This plugin will use an ioctl syscall to control the pf device in order to
#   block the host by adding the IP into a pf table. Additional active pf
#   states to/from the host will be killed.
#   Example: pf2 anchor=snortsam table=block kill=all
# ipchains <adapter> <logoption>
#   This plugin will use an setsockopt call to control the ipchains options
#   in order to block the host by adding a rule to the active rule set.
#   You have to specify the adapter to block on (for example, eth0) and you can
#   optionally add a log option (log, logall).
#   Example:   ipchains eth0 log
iptables eth0 syslog.info
#   This plugin will call the iptables executable in order to block the host by
#   adding a rule to the active rule set. You have to specify the adapter to
#   block on (for example, eth0) and you can optionally add a log option.
#   Example:   iptables eth0 syslog.info
# ebtables <adapter> <logoption>
#   This plugin will call the ebtables executable in order to block the host by
#   adding a rule to the active rule set. You have to specify the adapter to
#   block on (for example, eth0) and you can optionally add a log option.
#   Example:   ebtables eth0 syslog.info
# watchguard <path/to/fbidsmate> <ip-of-firebox> <configpassword>
```

```
# watchguard <path/to/fbidsmate> <ip-of-firebox> file <configpassfile>
#   This plugin will call the fbidsmate program to block the host on Watchguard
#   firewalls. You have to specify the path to the fbidsmate program, the
#   IP address of the firewall, and either a clear-text password, or the name
#   of a file containing the encrypted password. For more information, please
#   see the README.wgrd file.
#   Examples: watchguard /bin/fbidsmate 10.1.0.1 mySecretPass
#             watchguard /bin/fbidsmate 10.1.0.1 file /etc/fbidsmate.passphrase
# 8signs <path/dfw.exe> <tarpit>
#   SnortSam will call the specified dfw.exe executable to create the block
#   on the 8signs firewall. Snortsam will always block IPs without expiration
#   (-expiry n) because the 8signs firewall can only block for a day, a week,
#   or permanently. Snortsam blocks permanently and then times-out the blocks
#   itself, issuing an unban of the IP to 8signs, so that normal time
#   intervals are possible (for example, 10 minute blocks).
#   Optionally, the word "tarpit" can be appended to cause 8signs to ban and
#   tarpit the IP address.
#   Examples:  8signs c:\progra~1\8signs~1\dfw.exe
#              8signs c:\progra~1\8signs~1\dfw.exe tarpit
# isa <log>
#   SnortSam will use the API in msfpccom.dll to control the Microsoft ISA
#   Server interface in order to add blocking rules to the ISA firewall
#   rules.
#   Optionally, the word "log" can be appended to cause ISA Server to log
#   connection attempts from the blocked IP address.
#   Examples:  isa
#              isa log
# chx-i <path/fltcon.exe> <log>
#   SnortSam will call the specified fltcon.exe executable (or just fltcon if
#   none is specified, in which case fltcon would need to be in the PATH) to
#   create the block on the CHX-I packet filter. Snortsam can only block, and
#   can not forcefully unblock IP addresses. In order to frocefully remove
#   a blocked host, just restart the CHX-I service and all blocked IP addresses
#   are released.
```

```
# Optionally, the word "log" can be appended to cause CHX-I to log blocked
# packets.
# Examples:  chx-i c:\somewhere\fltcon.exe
#            chx-i fltcon.exe log
# ipfw2 <adapter> <inbound-table> <outbound-table>
# This plugin will add/remove IP addresses to be blocked/unblocked into the
# corresponding table(s). Tables are a new feature of ipfw2. You have to set
# up these tables manually before starting Snortsam exactly like this:
#      deny ip from any to table(<inbound-table>) via <adapter>
# and:  deny ip from table(<outbound-table>) to any via <adapter>
# If these tables are not present in your ipfw2 rule set, Snortsam will not
# start and report an error. With the tables present, configure Snortsam
# accordingly.
# Example:   ipfw2 ep0 1 2
# With tables rules like:
#            00010 deny ip from any to table(1) via ep0
#            00011 deny ip from table(2) to any via ep0
# forward <snortsam-ip>:<port>/<password>
forward 10.100.0.5:898
# This plugin will forward a block/unblock request to another Snortsam agent
# running on this or another host. This allows you link Snortsams in a chain,
# providing for a completely distributed blocking infrastructure. You can
# configure two Snortsam agents to forward to each other. The loop is avoided
# by the repetitive block prevention. IF YOU DISABLE REPETITIVE BLOCK
# SETTINGS, YOU WILL CREATE AN ENDLESS LOOP CAUSING RESOURCE
EXHAUSTION OR
# STARVATION OR A DENIAL-OF-SERVICE CONDITION!
# Take note that any white-list or override lists are processed before the
# request is forwarded. The planned "passthrough" plugin will avoid this
# limitation in the future. It is recommended to create separate Snortsam
# instances for "distribution hubs" which don't have white-list or override
# restrictions.
# Example:   forward secondsnortsam.domain.net
#            forward 127.0.0.1:898
```

APÊNDICE C - ARQUIVO DE CONFIGURAÇÃO DO SOFTWARE SNORBY

Abaixo é apresentado o arquivo de configuração utilizado para operação do software *Snorby*. Este divido em dois arquivos, os quais são responsáveis pela leitura dos *logs* em um banco de dados para depois serem apresentados em tela, assim como a geração de gráficos.

Arquivo: */var/www/snorby/config/snorby_config.yml*

```
development:
domain: localhost:3000
wkhtmltopdf: /usr/local/bin/wkhtmltopdf
test:
domain: localhost:3000
wkhtmltopdf: /usr/local/bin/wkhtmltopdf
production:
domain: localhost:3000
wkhtmltopdf: /usr/local/bin/wkhtmltopdf
```

Arquivo: */var/www/snorby/config/database.yml*

```
snorby: &snorby
adapter: mysql
username: root
password: teste123
host: localhost
development:
database: snorby
test:
database: snorby
production:
database: snorby
```

APÊNDICE D - ARQUIVO DE CONFIGURAÇÃO DO SOFTWARE BARNYARD

Arquivo: */etc/snort/barnyard2.conf*

```
# This file contains a sample barnyard2 configuration.
# You can take the following steps to create your own custom configuration:
#   1) Configure the variable declarations
#   2) Setup the input plugins
#   3) Setup the output plugins
# Step 1: configure the variable declarations
# in order to keep from having a commandline that uses every letter in the
# alphabet most configuration options are set here.
# use UTC for timestamps
#config utc
# set the appropriate paths to the file(s) your Snort process is using.
config gen_file:           /etc/snort/rules/gen-msg.map
config sid_file:            /etc/snort/rules/sid-msg.map
config gen_file:           /etc/snort/gen-msg.map
config sid_file:            /etc/snort/sid-msg.map
config reference_file:     /etc/snort/reference.config
config classification_file: /etc/snort/classification.config
# define dedicated references similar to that of snort.
#config reference: mybugs http://www.mybugs.com/?s=
# define explicit classifications similar to that of snort.
#config classification: shortname, short description, priority
# set the directory for any output logging
config logdir: /var/log/barnyard2/
# to ensure that any plugins requiring some level of uniqueness in their output
# the alert_with_interface_name, interface and hostname directives are provided.
# An example of usage would be to configure them to the values of the associated
# snort process whose unified files you are reading.
# Example:
#   For a snort process as follows:
#     snort -i eth0 -c /etc/snort.conf
#   Typical options would be:
```

```
#     config hostname: thor
#     config interface: eth0
#     config alert_with_interface_name
config hostname:     Snorby
config interface:    eth1
# enable printing of the interface name when alerting.
#config alert_with_interface_name
# at times snort will alert on a packet within a stream and dump that stream to
# the unified output. barnyard2 can generate output on each packet of that
# stream or the first packet only.
#config alert_on_each_packet_in_stream
# enable daemon mode
config daemon
# make barnyard2 process chroot to directory after initialisation.
#config chroot: /var/spool/barnyard2
# specifiy the group or GID for barnyard2 to run as after initialisation.
#config set_gid: 999
# specifiy the user or UID for barnyard2 to run as after initialisation.
#config set_uid: 999
# specify the directory for the barnyard2 PID file.
#config pidpath: /var/run/by2.pid
# enable decoding of the data link (or second level headers).
#config decode_data_link
# dump the application data
#config dump_payload
# dump the application data as chars only
#config dump_chars_only
# enable verbose dumping of payload information in log style output plugins.
#config dump_payload_verbose
# enable obfuscation of logged IP addresses.
#config obfuscate
# enable the year being shown in timestamps
#config show_year
# set the umask for all files created by the barnyard2 process (eg. log files).
```

```
#config umask: 066
# enable verbose logging
#config verbose
# quiet down some of the output
#config quiet
# define the full waldo filepath.
config waldo_file: /var/barnyard2/waldo
# specificy the maximum length of the MPLS label chain
#config max_mpls_labelchain_len: 64
# specify the protocol (ie ipv4, ipv6, ethernet) that is encapsulated by MPLS.
#config mpls_payload_type: ipv4
# set the reference network or homenet which is predominantly used by the
#log_ascii plugin.
#config reference_net: 192.168.0.0/24
# CONTINOUS MODE
# set the archive directory for use with continous mode
#config archivedir: /tmp
# when in operating in continous mode, only process new records and ignore any
# existing unified files
#config process_new_records_only
# Step 2: setup the input plugins
# this is not hard, only unified2 is supported ;)
input unified2
# Step 3: setup the output plugins
# alert_cef
# Purpose:
#  This output module provides the abilty to output alert information to a
# remote network host as well as the local host using the open standard
# Common Event Format (CEF).
# Arguments: host=hostname[:port], severity facility
#           arguments should be comma delimited.
#   host            - specify a remote hostname or IP with optional port number
#                 this is only specific to WIN32 (and is not yet fully supported)
#         severity          - as defined in RFC 3164 (eg. LOG_WARN, LOG_INFO)
```

```
#         facility - as defined in RFC 3164 (eg. LOG_AUTH, LOG_LOCAL0)
# Examples:
#         output alert_cef
#         output alert_cef: host=192.168.10.1
#         output alert_cef: host=sysserver.com:1001
#         output alert_cef: LOG_AUTH LOG_INFO
# alert_bro
# Purpose: Send alerts to a Bro-IDS instance.
# Arguments: hostname:port
# Examples:
#         output alert_bro: 127.0.0.1:47757
# alert_fast
# Purpose: Converts data to an approximation of Snort's "fast alert" mode.
# Arguments: file <file>, stdout
#           arguments should be comma delimited.
#   file - specifiy alert file
#   stdout - no alert file, just print to screen
# Examples:
#   output alert_fast
#   output alert_fast: stdout
output alert_fast: stdout
# prelude: log to the Prelude Hybrid IDS system
# Purpose:
#  This output module provides logging to the Prelude Hybrid IDS system
# Arguments: profile=snort-profile
#   snort-profile          - name of the Prelude profile to use (default is snort).
# Snort priority to IDMEF severity mappings:
# high < medium < low < info
# These are the default mapped from classification.config:
# info   = 4
# low    = 3
# medium = 2
# high   = anything below medium
# Examples:
```

```
#   output alert_prelude
#   output alert_prelude: profile=snort-profile-name
# alert_syslog
# Purpose:
#  This output module provides the abilty to output alert information to a
# remote network host as well as the local host.
# Arguments: host=hostname[:port], severity facility
#           arguments should be comma delimited.
#   host            - specify a remote hostname or IP with optional port number
#                 this is only specific to WIN32 (and is not yet fully supported)
#         severity          - as defined in RFC 3164 (eg. LOG_WARN, LOG_INFO)
#         facility - as defined in RFC 3164 (eg. LOG_AUTH, LOG_LOCAL0)
# Examples:
#         output alert_syslog
#         output alert_syslog: host=192.168.10.1
#         output alert_syslog: host=sysserver.com:1001
#         output alert_syslog: LOG_AUTH LOG_INFO
# log_ascii
# Purpose: This output module provides the default packet logging funtionality
# Arguments: None.
# Examples:
#   output log_ascii
# log_tcpdump
# Purpose
#  This output module logs packets in binary tcpdump format
# Arguments:
#   The only argument is the output file name.
# Examples:
#   output log_tcpdump: tcpdump.log
# sguil
# Purpose: This output module provides logging ability for the sguil interface
# See doc/README.sguil
# Arguments: agent_port <port>, sensor_name <name>
#           arguments should be comma delimited.
```

agent_port - explicitly set the sguil agent listening port

(default: 7736)

sensor_name - explicitly set the sensor name

(default: machine hostname)

Examples:

output sguil

output sguil: agent_port=7000

output sguil: sensor_name=argyle

output sguil: agent_port=7000, sensor_name=argyle

database: log to a variety of databases

Purpose: This output module provides logging ability to a variety of databases

See doc/README.database for additional information.

Examples:

output database: log, mysql, user=root password=teste123 dbname=snorby host=localhost

output database: alert, postgresql, user=snort dbname=snort

output database: log, odbc, user=snort dbname=snort

output database: log, mssql, dbname=snort user=snort password=test

output database: log, oracle, dbname=snort user=snort password=test

APÊNDICE E – CÓDIGO DESENVOLVIDO PARA APRESENTAR O BLOQUEIO EM INTERFACE WEB

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br"
dir="ltr">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title>Acesso Negado - Univates</title>
<style>
/* * GERAL * */
* {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 14px;
        color: #333;
        list-style: none;
        margin: 0;
        padding: 0;
}
body {
        background: #760000
url(http://www.univates.br/files/files/univates/bloqueio/img/fundo.jpg) no-repeat top left;
}
h1{
        background: #8b0000;
        border-bottom: #b20000 solid 1px;
        border-left: #3b0000 solid 1px;
        border-right: #b20000 solid 1px;
        border-top: #3b0000 solid 1px;
        color: #FFF;
        display:block;
        font-size: 40px;
        line-height: 100px;
```

```
        margin-bottom: 10px;
        text-align: center;
        text-shadow: 2px 2px 2px #000;
}
h2{
        background: #8b0000;
        border-bottom: #b20000 solid 1px;
        border-left: #3b0000 solid 1px;
        border-right: #b20000 solid 1px;
        border-top: #3b0000 solid 1px;
        color: #FFF;
        display:block;
        font-size: 15px;
        margin-bottom: 10px;
        line-height: 35px;
        padding-left: 5px;
        text-align: center;
        text-shadow: 2px 2px 2px #000;
}
p{
        clear: both;
        display: block;
        letter-spacing: 1px;
        line-height: 20px;
        padding: 10px 20px;
        text-align: center;
}
strong{
        color: #8b0000;
        display: block;
        margin-bottom: 15px;
}
table{
        border-spacing: 5px;
```

```
        padding: 0;
        width: 500px;
}
textarea{
        height: 100px;
        padding: 5px;
        width: 500px;
}
input{
        padding: 5px;
        width: 240px;
}
/* * LAYERS * */
#alerta{
        bottom: 20px;
        right: 20px;
        position: absolute;
        z-index: 0;
}
#centro {
        background-color:#fff;
        margin-left: 50%;
        left: -300px;
        padding: 10px;
        position: absolute;
        top: 30px;
        width: 600px;
        z-index: 1;
}
#help{
        background: #fcfdde;
        border: #CCC solid 1px;
        clear: both;
        margin: 20px;
```

```
        padding: 10px;
        text-align: center;
    }
    #help p{
        clear: both;
        line-height: normal;
        padding: 0 10px 10px 10px;
        text-align: center;
    }
    /* * STYLES * */
    .categoria{
        text-transform: uppercase;
    }
    /* * L I N K S * */
    a, a:link, a:active, a:visited {
        color: #004b97;
        font-size: 14px;
        font-weight: bold;
        text-decoration: none;
    }
    a:hover {
        color: #004b97;
        font-size: 14px;
        text-decoration: underline;
    }
    </style>
    </head>
    <body>
        <div id="centro">
      <h1>ACESSO NEGADO!</h1>
        <p>
          Acesso negado à página:<br />
              <strong>-URL-</strong> Nosso sistema detectou a
categoria:<br />
```

-CATEGORIES-

Você está vendo esta mensagem porque o site que você tentou acessar parece conter material que foi julgado impróprio devido a seguinte razão:

-REASONGIVEN-

<h2>ATENÇÃO</h2>

Caso esta página não contenha conteúdo impróprio, por favor, contate-nos para que possamos averiguar e, caso a solicitação seja válida, disponibilizar acesso a este endereço.

<form method="post" action="http://www.univates.br/bloqueio.php">

<input type="hidden" name="ip" value="-USER-" />

<input type="hidden" name="url" value="-URL-" />

<input type="hidden" name="categoria" value="-CATEGORIES-" />

<input type="hidden" name="razao" value="-REASONGIVEN-" />

<input type="hidden" name="src" value="wireless" />

<table>

<tr>

<td><input id="user" name="user" type="text" value="Login:" onfocus="if (this.value=='Login:') this.value=";" /></td>

<td><input id="senha" name="senha" type="password" value="Senha:" onfocus="if (this.value=='Senha:') this.value=";" /></td>

</tr>

<tr>

<td colspan="2"><textarea name="mensagem" onfocus="if (this.value=='Mensagem:') this.value=";">Mensagem:</textarea></td>

</tr>

<tr>

<td colspan="2" align="right"><input type="submit" value="enviar" /></td>

</tr>

</table>

```
</form>
        </div>
    </div>
    </div>
    <div id="alerta">
          <img src="http://www.univates.br/files/files/univates/bloqueio/img/alert.gif" />
   </div>
</body>
</html>
```

APÊNDICE F – SCRIPT DESENVOLVIDO PARA COLETAR E ENVIAR INFORMAÇÕES AO FILTRO WEB

Script 01: */usr/local/bin/geralista.sh*

```
#!/bin/bash
#script para gerar listas ao squid.
# As seguintes listas são geradas: bloqueios_novos, bloqueios_removidos e bloqueios_atuais
# Copia a ultima lista
cp /tmp/bloqueios_atuais /tmp/bloqueados_antigo
# Consulta IPs atualmente bloqueados
#/usr/bin/consulta.pl | grep DMZ | cut -d" " -f 3 > /tmp/bloqueados
/sbin/iptables -L |cut -d " " -f12 |grep 10. |sort -u > /tmp/bloqueados
# Gera a lista de IPs atualmente bloqueados
cp /tmp/bloqueados /tmp/bloqueios_atuais
scp /tmp/bloqueios_atuais root@10.100.0.5:/root
# Gera a lista de novos IPs a serem bloqueados
diff --left-column /tmp/bloqueados_antigo /tmp/bloqueios_atuais | grep ">" | cut -d " " -f2 >/tmp/bloqueios_novos
scp /tmp/bloqueios_novos root@10.100.0.5:/root
# Gera a lista de IPs a serem desbloqueados
diff --left-column /tmp/bloqueados_antigo /tmp/bloqueios_atuais | grep "<" | cut -d " " -f2 >/tmp/bloqueios_removidos
scp /tmp/bloqueios_removidos root@10.100.0.5:/root
```

Script 02:*/usr/local/bin/script.sh*

```
#!/bin/sh
IPTABLES=/sbin/iptables
diff -q  /root/bloqueios /root/bloqueios_atuais
if [ ! $? -eq "0" ]; then
      rm /root/aplicar*
       touch /root/aplicar
       diff --left-column bloqueios_atuais bloqueios | grep "<" | cut -d " " -f2 > aplicar
       cp /root/bloqueios_atuais /root/bloqueios
```

```
        cp /root/bloqueios /root/manter
       cat aplicar | while read LINE; do
        $IPTABLES -I INPUT -p tcp --dport 8080 -s $LINE -j ACCEPT
        $IPTABLES -I FORWARD -p udp --dport 53 -s $LINE -j ACCEPT
        done
       cp /root/bloqueios_atuais /etc/dansguardian/lists/bannediplist
        /bin/sh /etc/init.d/dansguardian reload
fi
cat /root/bloqueios_removidos | while read IPS; do
$IPTABLES -D INPUT -p tcp --dport 8080 -s $IPS -j ACCEPT
$IPTABLES -D FORWARD -p udp --dport 53 -s $IPS -j ACCEPT
$IPTABLES -D INPUT -s $IPS -i eth0 -j DROP
$IPTABLES -D INPUT -s $IPS -i eth1 -j DROP
$IPTABLES -D INPUT -s $IPS -i eth2 -j DROP
$IPTABLES -D INPUT -s $IPS -i eth3 -j DROP
$IPTABLES -D INPUT -s $IPS -i eth4 -j DROP
$IPTABLES -D INPUT -d $IPS -i eth0 -j DROP
$IPTABLES -D INPUT -d $IPS -i eth1 -j DROP
$IPTABLES -D INPUT -d $IPS -i eth2 -j DROP
$IPTABLES -D INPUT -d $IPS -i eth3 -j DROP
$IPTABLES -D INPUT -d $IPS -i eth4 -j DROP
$IPTABLES -D FORWARD -s $IPS -i eth0 -j DROP
$IPTABLES -D FORWARD -s $IPS -i eth1 -j DROP
$IPTABLES -D FORWARD -s $IPS -i eth2 -j DROP
$IPTABLES -D FORWARD -s $IPS -i eth3 -j DROP
$IPTABLES -D FORWARD -s $IPS -i eth4 -j DROP
$IPTABLES -D FORWARD -d $IPS -i eth0 -j DROP
$IPTABLES -D FORWARD -d $IPS -i eth1 -j DROP
$IPTABLES -D FORWARD -d $IPS -i eth2 -j DROP
$IPTABLES -D FORWARD -d $IPS -i eth3 -j DROP
$IPTABLES -D FORWARD -d $IPS -i eth4 -j DROP
done
```