UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO

EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME ELISEU RHODEN

DETECÇÃO DE INTRUSÕES EM BACKBONES DE REDES DE COMPUTADORES ATRAVÉS DA ANÁLISE DE COMPORTAMENTO COM SNMP

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Carlos Becker Westphall

Florianópolis, maio de 2002

DETECÇÃO DE INTRUSÕES EM BACKBONES DE REDES DE COMPUTADORES ATRAVÉS DA ANÁLISE DE COMPORTAMENTO COM SNMP

Guilherme Eliseu Rhoden

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação - Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Dr. Fernando A. O. Gauthier
Coordenador do Curso

Banca Examinadora

_____
Prof. Dr. Carlos Becker Westphall
Orientador

_____
Prof. Dr. Ricardo Felipe Custódio

_____
Prof. Dr. Carlos Alberto Maziero

_____
Prof. Dr^{a}. Carla Merkle Westphall

A meus pais Liseu e Carmelita,

Irmãos Cristiane, Sandro, Carla e

A minha namorada Fernanda

AGRADECIMENTOS

A Deus.

A minha família e namorada
pelo apoio e incentivo a continuar estudando.

Aos colegas do Núcleo de Processamento de Dados,
Edison Melo, Fernando Cerutti, Solange, Elvis, Augusto, Eduardo, Walter e Gerson
pelos momentos de trabalho passados juntos e por tudo o que aprendi de cada um.

Aos professores do curso de Pós-Graduação em Ciências da Computação
pelo apoio que vem sendo dado aos mestrandos, em especial ao Daniel Santana
por me acolher inicialmente como orientando.

Pelos amigos que aqui fiz e participaram destes meus dois anos.

A todas as outras pessoas não citadas que contribuíram direta ou
indiretamente para a realização deste trabalho.

Em especial ao professor Dr. Carlos Becker Westphall
e ao Msc. Edison Lopes Melo
pela orientação e co-orientação neste trabalho.

Sumário

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

bps – bits por segundo

ICMP – *Internet Control Message Protocol*

IEEE – *Institute of Electical and Electronics Engineers*

IP – *Internet Protocol*

ISO – *International Organization for Standardization*

DoS – *Denial of Service*

DDoS – *Distributed Denial of Service*

MIB – *Management Information Base*

MU - *Management Unit*

OID – *Object Identifier*

OSI – *Open Systems Interconection*

PC – *Personal Computer*

PDU – *Protocol Data Unit*

RCT – Rede Catarinense de Ciência e Tecnologia

RFC – *Request for Comments*

RNP – Rede Nacional de Pesquisa

SC – Santa Catarina

SNMP – *Simple Network Management Protocol*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

UFSC – Universidade Federal de Santa Catarina

LISTA DE TABELAS

LISTA DE QUADROS

RESUMO

Este trabalho apresenta um método de detecção de intrusão em redes de computadores, baseando-se no gerenciamento SNMP, onde são realizadas coletas de informações dos dispositivos de interconexão de redes, como o nível de utilização do processador, uso de memória e pacotes descartados, a fim de traçar um *baseline* (perfil de comportamento) dinâmico, de cada equipamento que está sendo monitorado. Com este método é possível detectar quando o padrão de comportamento afasta-se do esperado, gerando alertas para que a causa do problema possa ser descoberta e que o equipamento volte a trabalhar em suas condições normais.

A implementação do método é baseada em SNMP, Java, PHP e MySQL possibilitando a criação de uma interface WEB amigável para o usuário final. Futuras expansões do sistema podem ser integrados com um sistema de detecção de intrusos de rede para auxiliar no descobrimento automático da alteração do comportamento do equipamento.

ABSTRACT

This work shows a method of intrusion detection in computer networks. Having as a base the SNMP management, the information of the network interconnection devices are collected, such as the processor load, memory use and discarded packets. Its purpose is to draw a dynamic baseline, making possible to automatically acquire the operation of each equipment that is being monitored. Using this method, it becomes possible to find out that something is wrong with the equipment through the variation of its operation, sending signs so that the cause of the problem can be found and the equipment starts working properly again.

The implementation is based on SNMP, JAVA, PHP and MySQL what enables the creation of a good web interface to the final user and thinking about further expansions where the integration with a network intrusion detection system might help in the automatic discovery of the change in the equipment operation.

1 - INTRODUÇÃO

Atualmente a preocupação com a segurança em sistemas computacionais e, principalmente, em redes de computadores está sendo considerada um fator primordial para um melhor funcionamento de uma rede. Esta maior preocupação com segurança possibilitará uma melhor qualidade para seus usuários, visando garantir que a confiança na rede chegue aos patamares de 100% e os transtornos causados por atacantes sejam minimizados com a implantação de uma política de segurança adequada e com o auxílio de ferramentas de detecção de intrusões.

Líderes de nações e companhias ao redor do mundo, estão sendo forçados a aumentar sua atenção para necessidades de segurança em redes de computadores. Muitos concluíram que suas redes possuem carências básicas tratando-se de segurança e que não existem muitos profissionais, suficientemente treinados, para implantar segurança de redes (WENSTROM, 2001).

Existem medidas preventivas que já estão sendo empregadas para minimizar os riscos de estar conectado a uma rede de computadores tais como: políticas de segurança, melhor qualificação dos usuários aos perigos/problemas que ele pode encontrar, uso de *Firewalls*, detectores de intrusão baseados em *hosts*/rede, gerenciamento de segurança, analisadores de segurança/falhas e criptografia.

João Cabrera e sua equipe de pesquisadores (CABRERA et al, 2001) apresentaram o estudo realizado da detecção prematura de ataques de *Distributed Denial of Service* (DDoS), utilizando sistemas de gerência de redes para analisar as variáveis de tráfego da MIB (*Management Information Base*), onde analisaram 3 *scripts* de DDoS e fizeram a correlação de eventos de cada ataque gerado pelos *scripts* e com as variáveis MIB. Com esse estudo, a equipe conseguiu gerar "assinaturas"[1] específicas para cada um dos

[1] Assinaturas de ataques em redes de computadores são fragmentos ou traços que indicam algum tipo de ataque.

ataques, somente pela análise estatística destas variáveis, o que possibilitou a sua detecção com um reduzido número de falsos positivos.

1.1 MOTIVAÇÃO DO TRABALHO

A necessidade deste estudo surgiu devido a inúmeros ataques que vem ocorrendo na Internet além de algumas máquinas vulneráveis serem invadidas, as mesmas são utilizadas para procurar outras máquinas com vulnerabilidades conhecidas a fim de propagar o ataque. Normalmente quando uma falha grave em algum sistema operacional (SO) que permite a exploração remota é descoberta, são divulgados alertas de segurança para os administradores, através de listas e páginas *web* relacionadas com segurança. Normalmente o responsável pelo SO ou aplicação que está comprometida, disponibiliza uma versão atualizada do seu programa com a correção da vulnerabilidade presente na versão anterior, que são conhecidos como os famosos *patches* (remendos). Caso o fabricante ainda não tenha corrigido o problema, recomenda-se a desativação temporária do serviço até que a correção seja realizada.

Muitos acabam utilizando os avisos de falhas de segurança com o intuito de invadir estes sistemas vulneráveis, com *scripts* criados pelos invasores ou terceiros. Mas, parece ser mais grave quando um *worm* (verme) é criado para explorar tal vulnerabilidade, invadir a máquina e ficar procurando por outros *hosts* potencialmente vulneráveis, para auto-invadí-los. Dependendo do número de máquinas contaminadas em uma rede de tamanho grande, como uma classe B, somente as máquinas internas poderão acabar degradando todo o *backbone* com estas varreduras de máquinas vulneráveis, isso sem contar com os ataques oriundos da Internet originados pelo mesmo *worm*.

Como uma avaliação da potencialidade de propagação de um verme realizada por MOORE (2001), em 19 de julho de 2001, o verme Code-Red (CRv2) conseguiu infectar mais de 350 mil *hosts* em apenas 14 horas o que corresponde aproximadamente a 3,12% do número total de servidores Microsoft *Internet Information Server*-IIS em fevereiro de 2002 segundo NETCRAFT (2002). Isso causou uma lentidão na Internet e de até

muitas redes locais, como no ambiente em que foi realizado este trabalho, na rede da Universidade Federal de Santa Catarina, onde foram contaminadas aproximadamente 50 sistemas operacionais Windows NT 4/5 com servidores WEB IIS vulneráveis. Foi constatado que poucas máquinas infectadas eram suficientes para tornar o desempenho da rede inaceitável. Através de análises não automatizadas realizadas na época observou-se que o percentual de uso da CPU era o fator que mais se alterava nessas situações de ataques, possibilitando assim distinguir quando o equipamento estava ou não operando em suas características normais.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral dessa dissertação é analisar os equipamentos de roteamento gerenciáveis, que estão no núcleo e nas subredes de um grande *backbone*, a fim de traçar um *baseline*[2] (perfil) dinâmico para cada variável de um equipamento, auxiliando na detecção de ataques e problemas que degradam o desempenho do *backbone*.

1.2.2 Objetivos Específicos

No decorrer do trabalho será abordado o gerenciamento de redes como uma medida auxiliar na detecção de intrusão em grandes *backbones*, onde a análise de todo o tráfego torna-se atualmente inviável para os atuais sistemas de detecção de intrusão de redes.

Possibilitar o controle de roteadores de uma grande WAN de forma centralizada, com um baixo custo, visando traçar o comportamento de cada equipamento para determinar se o seu estado atual se enquadra ou não em seus padrões de comportamento analisados.

---

[2] *Baseline* é uma caracterização estatística do funcionamento normal de um segmento monitorado de uma rede (NETO, 98).

1.3 ESTRUTURA DO TRABALHO

O presente trabalho foi organizado em 6 capítulos:

O primeiro capítulo descreve uma introdução ao tema deste trabalho, quais são as motivações para sua realização, qual a importância do estudo de detecção de intrusões para o gerenciamento de redes de computadores e ainda descreve alguns trabalhos relacionados.

O segundo capítulo faz uma revisão bibliográfica sobre o gerenciamento de redes, para servir como base para o desenvolvimento do trabalho, onde serão aplicados conhecimentos práticos e teóricos sobre os protocolos de gerência de rede, mais especificamente SNMP, que é utilizado para buscar as informações dos roteadores analisados pela ferramenta que é apresentada no capítulo 4.

No terceiro capítulo são abordadas técnicas de detecção de intrusão, tanto para *Hosts* e redes, quais as arquiteturas de IDS (Sistemas de Detecção de Intrusão) existentes para redes de computadores, métodos de ataques a redes de computadores, buscando focar ataques que causam degradação do desempenho do *backbone*.

No quarto capítulo é mostrada a proposta de um modelo para o gerenciamento de roteadores que obtém o perfil de comportamento do equipamento através de algumas variáveis da MIB e indica índices que eventualmente fogem do seu padrão normal, adquirido automaticamente de acordo com seu histórico previamente analisado (*baseline*).

O quarto capítulo apresenta a implementação do protótipo, o ambiente onde ele é empregado, os resultados alcançados, além dos testes e validações finais.

E por fim no sexto capítulo são apresentadas as conclusões e sugeridos trabalhos futuros.

2 - FUNDAMENTAÇÃO TEÓRICA

Para haver uma melhor interação no decorrer do trabalho torna-se importante conhecer a tarefa de gerência de redes, os protocolos empregados, noções sobre segurança em redes de computadores e detecção de intrusos.

2.1 GERÊNCIA DE REDES

Por menor e mais simples que seja, uma rede de computadores precisa ser gerenciada, a fim de garantir aos seus usuários disponibilidade de serviços e um nível de desempenho aceitável.

À medida em que a rede cresce, aumenta a complexidade de seu gerenciamento, forçando a adoção de ferramentas automatizadas para a sua monitoração e controle.

A adoção de um *software* de gerenciamento, geralmente proprietário, na maioria das vezes não resolve todos os problemas da pessoa responsável pela Administração / Gerência da Rede. Geralmente o usuário de um software de gerenciamento espera muito dele e, conseqüentemente, frustra-se quanto aos resultados que obtém. Por outro lado, essas ferramentas proprietárias quase sempre são sub-utilizadas por possuírem inúmeras características inexploradas ou utilizadas de modo pouco eficiente. Para gerenciar um recurso, é necessário conhecê-lo muito bem e visualizar claramente o que este recurso representa no contexto da rede.

O investimento em ferramentas proprietárias de gerenciamento pode ser justificado pelos seguintes fatores:

- As redes e recursos de computação distribuídos estão se tornando vitais para a maioria das organizações, sejam elas privadas ou públicas. Sem um controle efetivo, os recursos não proporcionam o retorno que a corporação necessita;

- A expansão contínua da rede em termos de componentes, usuários, interfaces, protocolos e fornecedores ameaçam o gerenciamento com perda de controle sobre o que está conectado na rede e como os recursos estão sendo utilizados;

- Os usuários esperam uma melhoria dos serviços oferecidos (ou no mínimo, a mesma qualidade de serviços), quando novos recursos são adicionados ou quando são distribuídos;

- Os recursos computacionais e as informações da organização geram vários grupos de aplicações de usuários com diferentes necessidades de suporte nas áreas de desempenho, disponibilidade e segurança. O gerente ou o administrador da rede deve atribuir e controlar recursos para balancear estas várias necessidades;

- À medida que um recurso fica mais importante para a organização, maior fica a sua necessidade de disponibilidade. O sistema de gerenciamento deve garantir essa disponibilidade;

- A utilização dos recursos deve ser monitorada e controlada para garantir que as necessidades dos usuários sejam satisfeitas a um custo razoável.

Além desta visão qualitativa, uma separação funcional de necessidades no processo de gerenciamento foi apresentada pela ISO (*International Organization for Standardization*), como parte de sua especificação de Gerenciamento de Sistemas OSI (*Open System Interconnection*). Esta divisão funcional foi adotada pela maioria dos fornecedores de sistemas de gerenciamento de redes para descrever as necessidades de gerenciamento: falhas, desempenho, configuração, contabilização e segurança comumente abreviada por “FCAPS” (CERUTTI, 1999).

2.1.1 Gerenciamento de Falhas

A gerência de falhas consiste no processo de localizar problemas, ou falhas, em uma rede. Envolve as tarefas de descobrir o problema, isolá-lo e solucioná-lo quando possível. Uma falha é uma condição anormal cuja recuperação exige ação de

gerenciamento e normalmente é causada por operações incorretas ou um número excessivo de erros.

Para controlar o sistema de forma efetiva, cada componente essencial de uma rede, ou seja, roteadores, *switches, links,* etc... devem ser monitorados individualmente para garantir o seu perfeito funcionamento.

Outra forma de controle é a tentativa da minimização do impacto e duração da falha através do uso de componentes redundantes e rotas de comunicação alternativas para dar à rede um grau de tolerância à falhas.

2.1.2 Gerenciamento de Contabilização

Mesmo que nenhuma cobrança interna seja feita pela utilização dos recursos da rede, o Administrador / Gerente da rede deve estar habilitado para controlar o uso dos recursos por usuário ou grupo de usuários.

O Administrador / Gerente de Rede deve ser capaz de especificar os tipos de informações de contabilização que devem ser registrados em cada nodo, o intervalo de entrega de relatórios para nodos de gerenciamento de mais alto nível e os algoritmos usados no cálculo da utilização.

2.1.3 Gerenciamento de Configuração

Esse gerenciamento está relacionado com a inicialização da rede e com uma eventual desabilitação de parte ou de toda a rede. Também está relacionado com as tarefas de manutenção, adição e atualização de relacionamento entre os componentes e do status dos componentes durante a operação da rede.

O Administrador / Gerente de Rede deve ser capaz de, inicialmente, identificar os componentes da rede e definir a conectividade entre eles. Também deve ser capaz de modificar a configuração em resposta à avaliação de desempenho, recuperação de falhas, problemas de segurança, atualização da rede ou a fim atender as necessidades dos usuários.

Relatórios de configuração podem ser gerados periodicamente ou em resposta a requisições de usuários.

2.1.4 Gerenciamento de Desempenho

O gerenciamento do desempenho de uma rede consiste na monitoração das atividades da rede e no controle dos recursos através de ajuste e trocas. Algumas das questões relativas ao gerenciamento do desempenho, podem ser:

- Qual é o nível de capacidade de utilização?
- O tráfego é excessivo?
- A vazão foi reduzida para níveis aceitáveis?
- O tempo de resposta está aumentando?

Para tratar estas questões, o Administrador / Gerente deve focalizar um conjunto inicial de recursos a serem monitorados, a fim de estabelecer níveis de desempenho. Isto inclui associar métricas e valores apropriados aos recursos de rede que possam fornecer indicadores de diferentes níveis de desempenho. Muitos recursos devem ser monitorados para obter informações sobre o nível de operação da rede. Armazenando e analisando estas informações, os Administradores e Gerentes das redes podem ficar mais capacitados no reconhecimento de situações de degradação de desempenho.

2.1.5 Gerenciamento de Segurança

O gerenciamento da segurança provê facilidades para proteger recursos da rede e informações dos usuários. Estas facilidades devem estar disponíveis apenas para usuários autorizados. É necessário que a política de segurança seja robusta, efetiva e que o sistema de gerenciamento da segurança seja, ele próprio, seguro.

2.2 PROTOCOLOS DE GERENCIAMENTO

2.2.1 SNMP (*Simple Network Management Protocol*)

Com o crescimento dos equipamentos de interconexão de redes no modelo TCP/IP, o trabalho de gerenciamento de redes começou a tornar-se algo cada vez mais complexo e os métodos antes tidos para diagnosticar se um determinado equipamento estava operando adequadamente, começou a não suprir todas as necessidades desejadas. A partir destes fatos, tornou-se necessário a criação de um protocolo para auxiliar estas tarefas de gerenciamento. Este protocolo é denominado SNMP, que atualmente encontra-se em sua versão 3. Mas, a grande maioria dos equipamentos implementa somente a versão 1 do protocolo (não segura[3]) e nosso trabalho se baseará sobre esta primeira versão do protocolo.

O modelo atual para gerenciamento de redes baseadas em TCP/IP, está descrito nos seguintes documentos em forma de RFC (*Request For Comments)*:

- RFC 1155 (ROSE & MCCLOGHRIE, 1990)- Estrutura e identificação da Informação de Gerenciamento, que descreve como os objetos gerenciados contidos na MIB são definidos;

- RFC 1156 (MCCLOGHRIE e ROSE, 1990)- Base de Informação de Gerenciamento, que descreve os objetos gerenciados definidos na MIB;

- RFC 1157 (CASE et al, 1990) – *Simple Network Management Protocol* – SNMP (Protocolo Simples de Gerência de Redes), que define o protocolo usado para gerenciar estes objetos.

---

[3] A versão 1 do protocolo SNMP é considerado insegura por apresentar autenticação baseada em uma comunidade SNMP composta por caracteres que é enviada sem nenhuma forma de criptografia e todo tráfego entre a estação gerente e o agente SNMP ser realizada de forma não criptografada.

2.2.1.1 O Modelo SNMP

Segundo TANENBAUM (1997), podemos dividir o modelo de uma rede gerenciada em quatro componentes, que são os seguintes:

1. Nós gerenciados;
2. Estações de gerenciamento;
3. Informações de gerenciamento, e
4. Um protocolo de gerenciamento.

Os nós gerenciados podem ser *hosts*, roteadores, pontes, impressoras ou qualquer outro dispositivo capaz de comunicar informações sobre o seu status para o mundo externo.

O SNMP foi projetado para ser um protocolo da camada de aplicação da pilha TCP/IP. Ele trabalha sobre o protocolo UDP (*User Datagram Protocol*). A Figura 1 demonstra a típica configuração do protocolo SNMP. Para uma estação de gerenciamento *Stand-Alone*, um processo gerente controla o acesso para uma MIB central na estação de gerenciamento e fornece uma interface para o gerenciamento de redes. O processo gerente executa o gerenciamento de rede usando SNMP, que é implementado sobre o UDP, IP, e os protocolos de redes dependentes relevantes (por exemplo, Ethernet, FDDI e X.25) (*STALLINGS, 1997*).

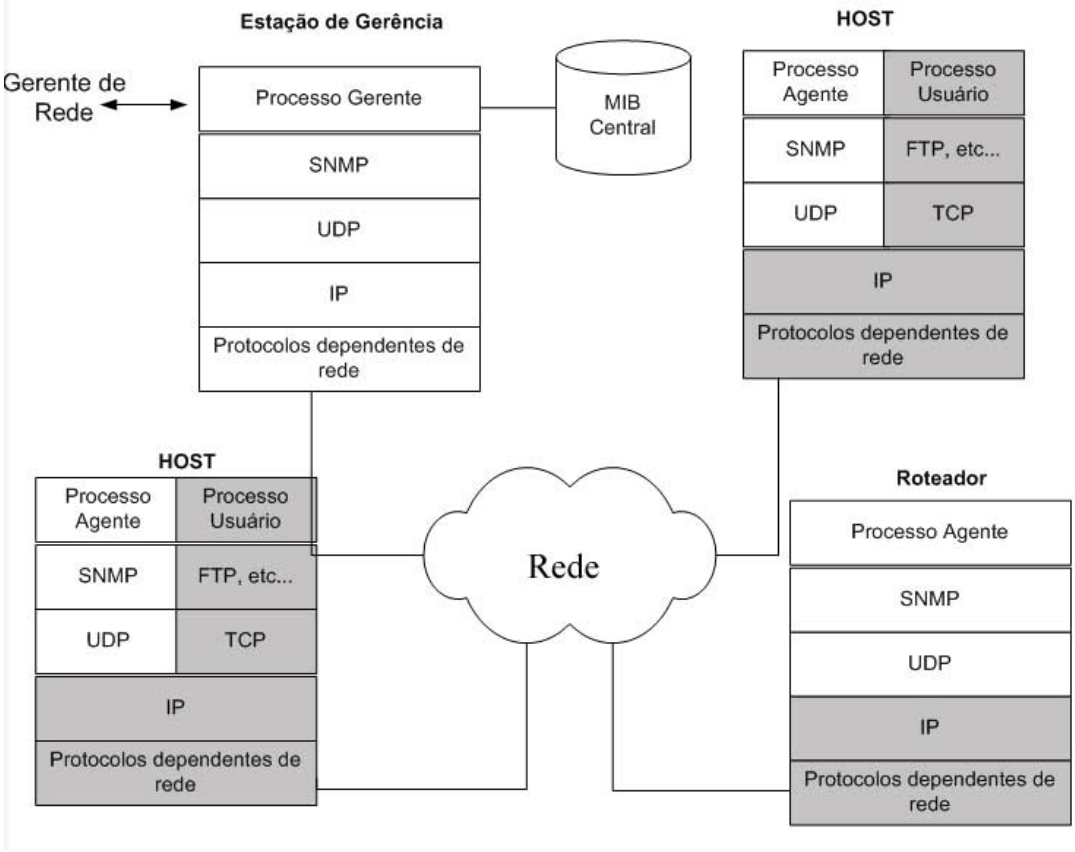

Figura 1 - Configuração do protocolo SNMP (STALLINGS, 1997)

Cada agente também deve implementar SNMP, UDP e IP. Adicionalmente, um processo agente intercepta as mensagens SNMP e controla o agente da MIB. Para um dispositivo agente que suporta outras aplicações, como FTP, ambos TCP e UDP são necessários.

Devido ao SNMP trabalhar sobre o UDP, que é um protocolo sem conexão, SNMP por ele mesmo é sem conexão. Nenhum diálogo de conexão é mantido entre a estação de gerência e seu agente. Em vez de disso, cada troca é uma transação separada entre a estação de gerência e um agente.

2.2.2 MIB (*Management Information Base*)

O conhecimento das MIBs (Base de Informação de Gerência), e principalmente, o conhecimento de como utilizar estas informações, são de fundamental importância na Gerência de Redes (VERONEZ, 2000).

A MIB apresenta uma estrutura hierárquica em forma de árvore, que identifica os objetos gerenciáveis (elementos da rede que podem ser gerenciados). Uma MIB também define de maneira não-ambígua a nomenclatura associada aos objetos gerenciados, representando esses objetos conforme apresentado na Figura 2 (CERUTTI, 1999). Ela pode conter as informações sobre, por exemplo, o percentual de utilização da CPU, número de pacotes que foram descartados em uma interface, etc...

Como definido em ROSE & MCCLOGHRIE (1990), a MIB e a MIB-II padrão para a Internet, contém 171 objetos. Estes objetos são agrupados por protocolos (incluindo TCP, IP, UDP, SNMP, e outros) e outras categorias, incluindo "sistemas" e "interfaces".

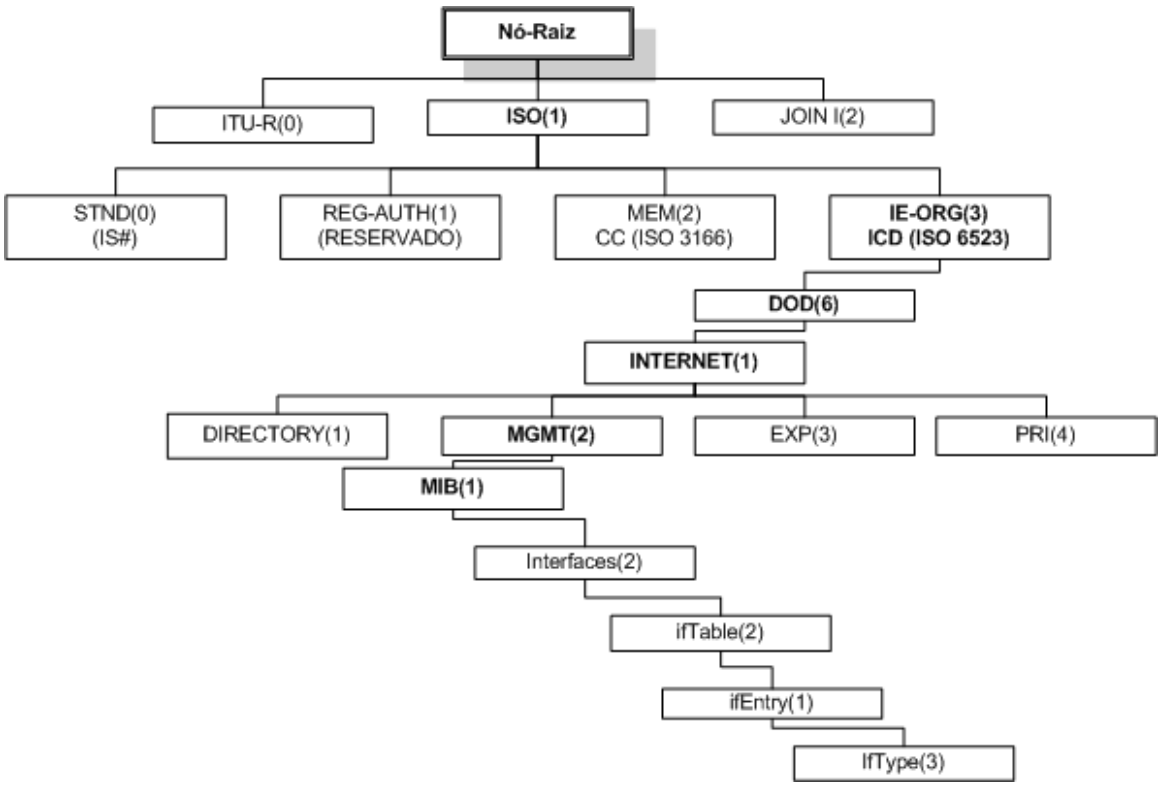

Figura 2 - estrutura em árvore das MIBs (CERUTTI, 1999)

As MIBs são associadas com o gerente ou com a estação de gerência (NMS) e com os sistemas gerenciados. Possuindo apenas uma base de dados numérica como estrutura para armazenar e recuperar dados, uma MIB possui uma organização bem definida. Essa organização lógica é denominada *Structure of Management Information* –SMI. A SMI é organizada em forma de “árvore”, começando pela “raiz” e com as ramificações

organizando os objetos gerenciáveis em categorias lógicas. As MIBs representam os objetos gerenciáveis como “folhas” nos “galhos” da “árvore” (CERUTTI, 1999), conforme é apresentado na Figura 2.

2.2.3 Operações com o SNMP

O SNMP possui como protocolo de troca de dados no envio e recebimento informações entre os gerentes e agentes as PDUs (*Protocol Data Unit*). O padrão existente no formato de uma PDU, pode executar as seguintes operações básicas no SNMPv1: (MAURO & SCHMIDT, 2001):

- *Get-request*: leitura do valor de uma variável;
- *Get-next-request*: leitura do valor da próxima variável;
- *Get-response*: resposta à operação de leitura (*get-request* ou *get-next-request*);
- *Set-request*: armazenamento do valor de uma variável, e
- *Trap*: notificação da ocorrência de um evento especifico.

Na operação *trap* os eventos que normalmente geram notificação são predefinidos e correspondem a erros, falhas ou operações anormais do sistema.

No próximo capítulo será brevemente abordada técnicas de detecção de intrusão, tanto para *Hosts* e redes, quais as arquiteturas de IDS (Sistemas de Detecção de Intrusão) existentes para redes de computadores, métodos de ataques a redes de computadores, buscando focar ataques que causam uma degradação do desempenho do *backbone*.

3 - INTRUSOS, ATAQUES E DETECÇÕES DE INTRUSÕES

Um das mais popularizadas ameaças para segurança são os intrusos, geralmente referenciados por *hacker*[4] ou *cracker*[5]. Em um importante estudo sobre intrusão, ANDERSON (1980) identificou três classes de intrusos:

- **Mascarado:** Um indivíduo não autorizado para usar um computador e que consegue burlar o controle de acesso de um sistema para explorar uma conta de usuário legítimo.

- **Malfeitor (*Misfeasor*):** Um usuário legítimo que acessa dados, programas ou outros recursos, onde o acesso não é permitido pelo sistema; ou é autorizado, mas faz uso impróprio de seus privilégios.

- **Usuário Clandestino:** Um indivíduo que se aproveita do controle de supervisor do sistema e usa este controle para escapar de auditorias e controles de acessos.

O mascarado é como se fosse externo ao sistema; O malfeitor geralmente é interno; e o usuário clandestino pode ser externo ou interno.

As ameaças de intrusos foram publicadas, particularmente devido ao famoso "*Hacker* Wily" incidente de 1986-1987 documentado por Cristofory Stoll (STOLL, 1988). Os

---

[4] *Hacker* – Estudioso das tecnologias, especialmente da informática, que utiliza boa parte de seu tempo na aplicação de técnicas sofisticadas para conhecer, utilizar, dominar ou modificar o funcionamento de programas e equipamentos.

[5] *Cracker* - Possui tanto conhecimento quanto os *hackers*, mas com a diferença de que, para eles, não basta entrar em sistemas, quebrar códigos, e descobrir falhas. Precisam deixar um aviso de que estiveram lá, geralmente com recados de baixo calão, algumas vezes destruindo partes do sistema, e até destruindo o que encontram. Também são atribuídos aos *crackers* programas que retiram os controles contra cópia em *software*.

ataques causados por *hackers* são freqüentes e se originaram com o nascimento das primeiras redes. Como exemplo, um grupo dos Laboratórios Bell (BELLOVIN, 1993) tem reportado freqüentes e persistentes ataques aos seus computadores através da Internet, em um período de tempo prolongado e vindo de várias fontes. Como resposta, o grupo Bell teve as seguintes experiências:

- O número de tentativas para copiar o arquivo de senhas do sistema (`/etc/passwd`), em percentual, excediam em relação aos dias anteriores.
- As requisições suspeitas de chamadas de procedimentos remotos (RPC – *Remote Procedure Calls*), excediam uma por semana.
- No período de duas semanas, era constatada uma nova tentativa de conexão para uma máquina não existente (que pode ser comparado nos dias atuais com os populares *scans*).

Intrusos bondosos podem ser tolerados, a não ser que consumam recursos a ponto de diminuir a performance de usuários legítimos. No entanto, não existe um método eficiente para conhecer se um intruso será bondoso ou maldoso, conseqüentemente, são classificados como iguais.

3.1 TÉCNICAS DE INTRUSÃO

Segundo STALLINGS (1998), o objetivo do intruso é ganhar acesso ao sistema ou aumentar o percentual de privilégios acessíveis neste. Geralmente, isto requer ao intruso adquirir informações que devem estar protegidas. Na maioria dos casos, esta informação está na forma de senha do usuário. Com o conhecimento de algumas outras senhas de usuários, um intruso pode acessar o sistema e exercer de todos os privilégios como o usuário legítimo teria.

Geralmente, um sistema deve manter um arquivo que associa a senha com cada usuário autorizado. Se este arquivo é guardado sem proteção, isso possibilitará um fácil acesso e aprendizado de senhas. O arquivo de senha pode ser protegido de uma ou duas maneiras:

- **Criptografia de sentido único** (*One-Way Encryption*): O sistema guarda somente em forma cifrada as senhas dos usuários. Quando o usuário fornece a senha, o sistema a cifra e compara-a com o valor guardado. Na prática, o sistema faz uma transformação em um caminho (não reversível) em que a senha é usada para gerar uma chave para a função de cifragem e como resultado desta função, uma saída de comprimento fixo é produzida.

- **Controle de Acesso:** Acesso ao arquivo de senhas é limitado para uma ou várias contas.

Se um ou ambos destes métodos são usados, algum esforço maior é necessário para um intruso potencial capturar as senhas.

Conforme SPARTA (1997), podemos dividir um ataque a máquinas UNIX em algumas fases distintas, onde cada uma possui suas próprias características:

1. Selecionar um alvo;

2. Colecionar informações sobre esse alvo. Existem diversos métodos para conseguir estas informações. Um ponto de partida seria a utilização de serviços como `finger`, `showmount` e `rpcinfo`. Todas as ferramentas empregadas por administradores para gerenciamento são também de grande valor para o atacante. DNS, `whois`, `sendmail`, `ftp` e `uucp` são outros serviços que fornecem ao atacante valiosas informações durante a fase de obtenção de dados;

3. Lançar um ataque sobre o alvo. Para tentar entrar em um sistema, o atacante utiliza vários métodos que exploram vulnerabilidades conhecidas de serviços, servidores e sistemas operacionais. Alternativamente, o atacante pode simplesmente tentar adivinhar senhas de usuários na máquina alvo;

4. Destruir evidências da invasão. Geralmente, as ações realizadas pelo atacante para conseguir entrar na máquina ficam registradas em *log*. É importante que eles sejam alterados de modo a encobrir estes passos e evitar sua análise pelo administrador do sistema;

5. Obter senhas de outras contas. Quanto mais contas forem comprometidas pelo atacante melhor. Na maioria das vezes, um simples ataque de dicionário[6] no arquivo `/ect/passwd` é suficiente para obtenção de um considerável número de senhas;

6. Obter acesso de administrador (root) na máquina invadida. Tendo acesso a uma conta de usuário, o atacante terá direitos de acesso limitado. Assim, é importante conseguir acesso root, pois este disporá de todo o sistema, empregando-o para realizar os passos seguintes e lançar futuros ataques. Para conseguir direitos de root, o atacante pode empregar diversos métodos, explorando geralmente *bugs* existentes em programas;

7. Instalar ferramentas para a captura de senhas. Para isto, atacantes podem instalar monitores de rede (*sniffers*) ou cavalos-de-tróia. Entre os programas candidatos a serem substituídos por cavalos-de-tróia estão `login`, `telnet` e `ftp`. Caso estes estejam protegidos e sua alteração seja impossível, atacantes podem, ainda, explorar erros tipográficos dos usuários. Para isto, basta colocar programas chamados telent (ou logn, fpt, etc.) em diretórios utilizados pelas vítimas;

8. Configurar caminhos secundários de entrada na máquina invadida, para o caso da rota principal ser descoberta e fechada. Dispondo de direitos root, fica extremamente fácil para um atacante construir rotas alternativas para entrar no sistema. Entre elas estão a inserção de um novo usuário ou a reconfiguração de alguns serviços; e

9. Encontrar outras máquinas que confiam na máquina invadida. Para isto, basta analisar arquivos como `.rhosts` e `hosts.equiv`.

---

[6] Ataque de dicionário, é um ataque que consiste na quebra de uma ou várias senhas através do auxílio de um dicionário de *strings* com possíveis senhas, nomes de cidades, lugares, frases, etc..., onde essas *strings* são testadas exaustivamente para verificar se conferem com uma senha do sistema.

Para que possamos analisar com mais clareza os métodos de detecção de intrusos existentes, torna-se necessário conhecer quais são as falhas normalmente encontradas em um sistema operacional, tanto ao nível de implementação de suas aplicações e protocolos vulneráveis . Na próxima sessão faremos um apanhado geral nos protocolos e sistemas operacionais vulneráveis, ressaltando algumas vulnerabilidades recentes.

3.2 VULNERABILIDADES DOS SO E PROTOCOLOS QUE COMPÕE A INTERNET

Muitos sistemas operacionais, devido a sua complexidade de implementação, acabam sendo susceptíveis a falhas. Estas falhas são geralmente causadas pela má implementação de algumas aplicações ou protocolos que compõe o sistema. Dependendo do tipo da falha, a segurança do sistema, poderá ser comprometida. Isso é agravado quando alguma aplicação ou serviço provê acesso a Internet e essa vulnerabilidade poderá ser explorada remotamente por um atacante para obter acesso não autorizado ao sistema.

3.3 PROBLEMAS DE SEGURANÇA RELACIONADOS COM REDES DE COMPUTADORES

Devido ao grande número de incidentes de segurança registrados nos últimos anos estão sendo despendidos grandes esforços para sua classificação, entendimento de como eles são realizados e quais são os métodos mais apropriados para se defender. Para ter uma visão geral dos principais problemas relacionados com segurança dando uma maior ênfase a *backbones* de Internet (Figura 3), podemos classifica-los como (WENSTROM, 2001):

- Reconhecimento do ambiente;
- Acesso não autorizado;
- Manipulação de dados, e
- Negação de Serviço.

As classes de problemas de segurança são geralmente conhecidas como vulnerabilidades – atributos de computadores ou redes que permitem alguém iniciar *exploits* contra uma rede. Um *exploit* é um método para tirar vantagem de uma vulnerabilidade através de um procedimento manual, um *script*, ou um programa executável. Seu propósito é coletar informações (realizar o reconhecimento do ambiente), negar serviços para impossibilitar seu uso para usuários legítimos, ganhando acesso não autorizado para sistemas ou dados, ou realizar a manipulação de dados.

Felizmente para a vantagem do administrador de segurança, a grande maioria das vulnerabilidades e *exploits* possuem um tipo de "assinatura"[7]. Os sistemas de detecção de intrusão trabalham com essas "assinaturas" para decifrar se está ocorrendo algum evento suspeito em sua rede.

A seguinte sessão é organizada conforme o diagrama apresentado na Figura 3.

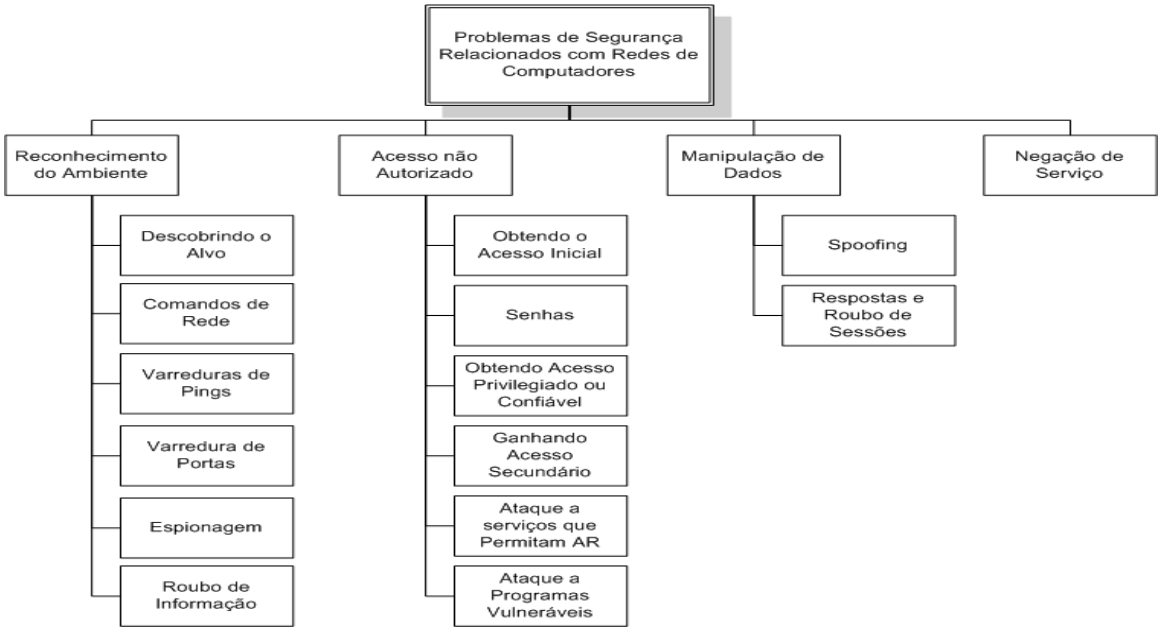

Figura 3 – Diagrama dos principais problemas segurança relacionados com redes de computadores

[7] Assinatura – define ou descreve um padrão tráfego de interesse. Uma assinatura existe no tráfego de rede e é utilizada para apreender e entender seus traços. (Northcutt e Novak, 2000). Após seus traços serem descobertos, pode ser utilizada para detectar certos tipos de ataques no momento que trafegam pela rede.

3.3.1 Reconhecimento do ambiente

Segundo WENSTROM (2001), o reconhecimento é realizado com descobrimentos não autorizados, mapeamentos, monitoramento de sistemas e serviços ou vulnerabilidades em uma rede de computadores. O reconhecimento também inclui monitoração de tráfego de rede, podendo ser classificado como ativo ou passivo. A informação reunida pode então ser utilizada para realizar outros ataques para redes ou para roubar dados vitais. A Figura 4 exemplifica algumas das opções que um atacante pode ter para realizar o levantamento da topologia de uma rede e os serviços que ela fornece.

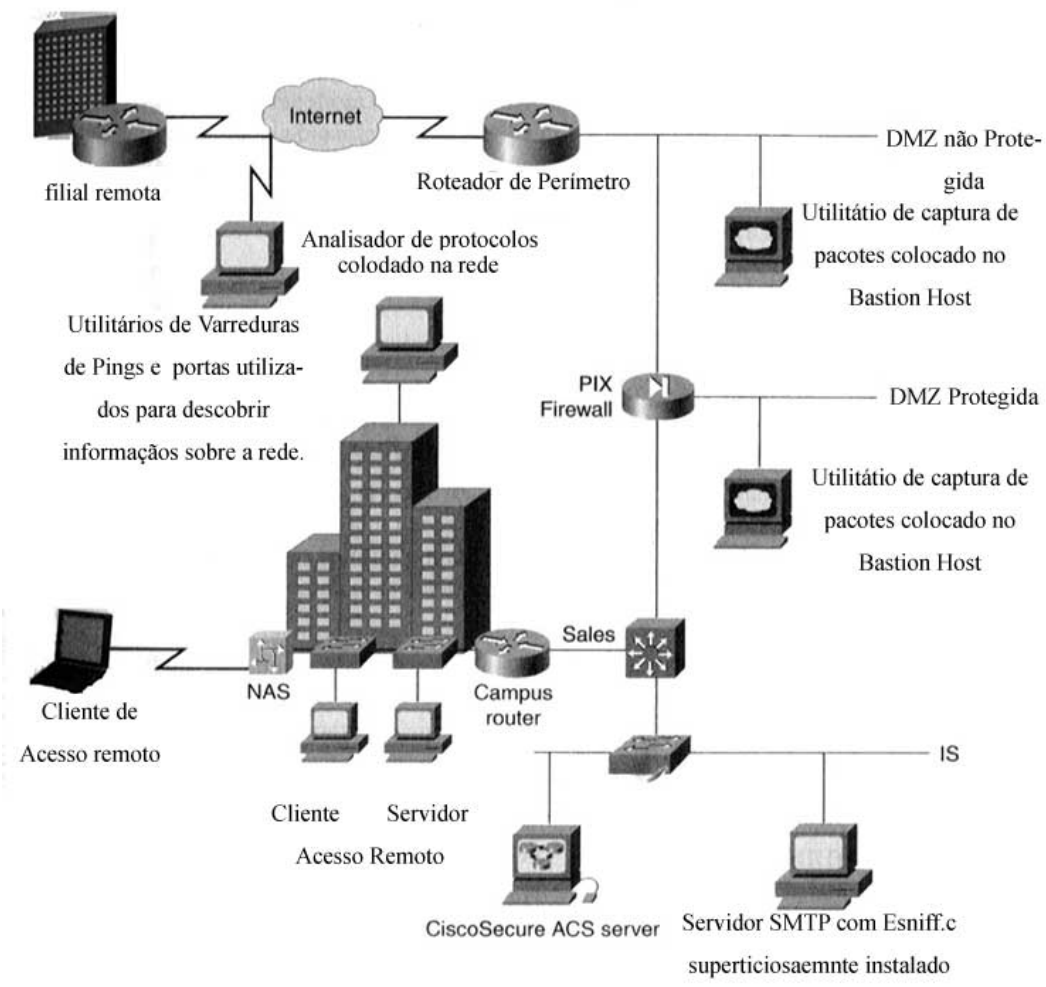

Figura 4 - Exemplos de locais onde pode ser realizado ataques de reconhecimento (WENSTRON 2001)

Ataques de reconhecimento podem ser realizados na forma de descobrimento do alvo, escutas (*eavesdropping*) e roubo de informações.

3.3.1.1 Descobrindo o Alvo

Este método inclui encontrar nomes de domínios e seus endereços IPs associados, aprendizado de faixas de endereçamentos IP da organização alvo ou descobrir o endereço IP especifico de um *host* alvo. Após o *hosts* alvo ser descoberto, pode-se verificar quais serviços estão disponíveis e/ou informações sobre o sistema que está rodando.

3.3.1.2 Comandos de Rede

O reconhecimento pode ser realizado através de comandos de rede disponíveis em sistemas UNIX, Linux e Windows: `ping`, `whois`, `finger`, `ruser`, `nslookup`, `rpcinfo`, `telnet`, `dig`, `Nmap`, `net view` ou através de outros comandos e utilitários que forneçam informações sobre um *host* especifico ou uma rede.

3.3.1.3 Varreduras de Pings

Ainda que comandos pings individuais possam ser realizados para ganhar informações sobre uma rede ou *host*, ferramentas de varreduras através de *pings* foram desenvolvidas para automatizar o descobrimento de uma rede ou subrede. O comando ping gera uma mensagem ICMP *Echo Request* tendo como destino o *host* alvo, o qual deve responder com uma mensagem ICMP *Echo Reply*. Alguns tipos de varreduras pings combinam vários tipos de mensagens ICMP *Echo Request*, com outras requisições ICMP como ICMP *TimeStamp*, ICMP *Address Mask*.

3.3.1.4 Varredura de Portas (*Port Scan)*

Quando um hacker descobre um *host* interessante, ele pode realizar um *port scan* sobre o alvo com o intuito de determinar quais portas TCP e/ou UDP estão aceitando conexões (ouvindo), verificando com isso quais serviços estão ativos como por ex. FTP (porta 21), Telnet (porta TCP 23), SNMP (UDP 161), Finger (TCP 79), etc... As

ferramentas de varredura de portas podem utilizar fragmentação de pacotes e combinações dos *bits* SYN e FIN do cabeçalho TCP para descobrir se existe alguma porta aceitando conexões.

Após uma porta específica ser encontrada, ele poderá conectar-se a ela e descobrir maiores informações a respeito do serviço que está rodando. Por exemplo, após a porta relativa ao serviço *sendmail* ser encontrada, o *hacker* poderá enviar comandos para obter maiores informações ou tentar ganhar acesso não autorizado ao sistema.

3.3.1.5 Espionagem (*Eavesdropping*)

Eavesdropping pode ser descrito como um método de observação passiva (não autorizada) do tráfego de rede através de um dispositivo ou ferramenta. O objetivo deste método é observar, capturar e analisar tráfegos de uma rede. *Networking snooping* e *packet sniffing* são sinônimos para *eavesdropping.* A informação adquirida por *eavesdropping* pode ser utilizada para lançar outros ataques ou para roubar informação.

Intrusos de rede podem utilizar *eavesdropping* para identificar usuários e senhas com intuito de ganhar acesso não autorizado a *hosts* ou equipamentos de rede, ou ainda para identificar informações sigilosas contidas nos pacotes, como números de cartões de crédito (que podem estar ou não cifrados) ou outras informações pessoais importantes.

Um exemplo de dado suscetível são as *strings* das comunidades SNMP versão 1 que são enviadas em texto aberto (não-crifrado). Um intruso pode interceptar essas mensagens e aprender informações valiosas sobre a configuração dos equipamentos de rede.

3.3.1.6 Roubo de Informação

Eavesdropping pode servir para o roubo de informações. Este roubo pode ocorrer com os dados que são transmitidos através das redes internas ou externas. O intruso também poderá roubar dados através do acesso não autorizado a um sistema.

Uma intrusão comum em redes é o quando um intruso copia arquivos ou utiliza recursos que não estão alocados para ele. Como exemplo, um intruso pode copiar o arquivo de senhas do sistema e o utilizá-lo em outro computador, onde tentará quebrar sua

criptografia, na maioria dos casos por força bruta, para conseguir acesso a outras contas de usuários.

3.3.2 Acesso não Autorizado

Um intruso de redes pode ganhar acesso não autorizado a um computador ou dispositivo de rede através de vários caminhos. O objetivo comum da maioria dos intrusos é obter acesso root (sistemas UNIX em geral) ou administrador (Windows NT) para um computador presente na rede, onde ele tem grande poder de controle sobre o sistema operacional alvo ou para acessar outros computadores. A Figura 5 ilustra alguns pontos de rede onde um intruso pode tentar conseguir acesso não autorizado.

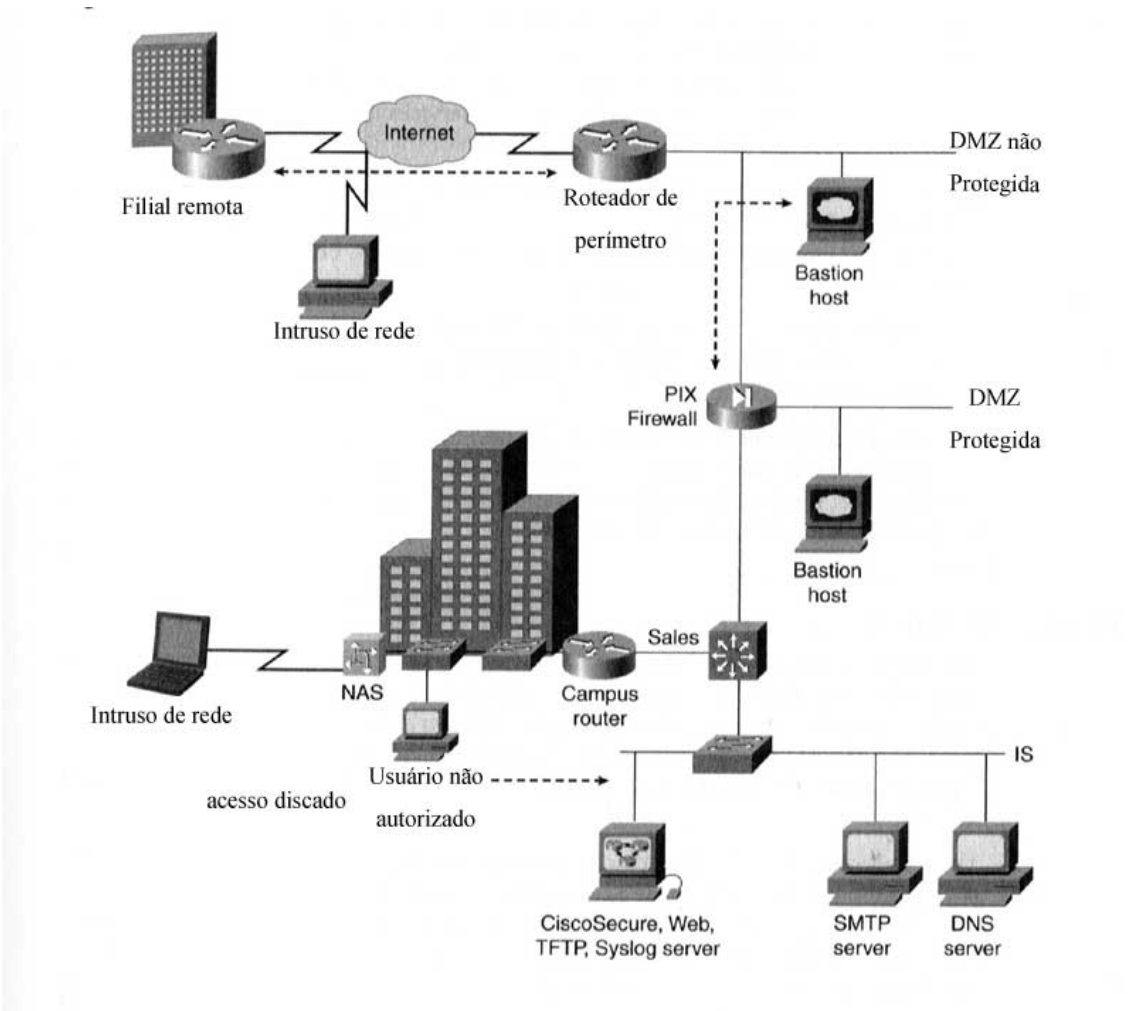

Figura 5 - Pontos de acesso não autorizados (WENSTROM, 2001)

3.3.2.1 Obtendo o Acesso Inicial

O intruso de rede geralmente tenta obter o maior número possível de informações sobre o *host*/rede alvo através de técnicas de reconhecimento. As mais corriqueiras foram descritas sucintamente na sessão anterior, e o intruso as usa para obter o acesso inicial. O intruso deve descobrir o endereço IP do *host* que ele pretende invadir, descobrir se o sistema alvo possui alguma vulnerabilidade que lhe permita a exploração remota ou simplesmente ele pode instalar um *sniffer* de pacotes para capturar usuários e senhas do *host*.

Mais facilmente, o intruso pode utilizar o ataque conhecido como engenharia social para ganhar um acesso inicial ou acesso privilegiado. Engenharia Social é o termo utilizado para a obtenção de informações importantes de uma empresa, através de seus usuários e colaboradores. Essas informações podem ser obtidas pela ingenuidade ou confiança. Os ataques desta natureza podem ser realizados através de telefonemas, envio de mensagens por correio eletrônico, salas de "bate-papo" e até mesmo pessoalmente (MAIA, 2001).

O atacante pode tentar acesso através do acesso discado utilizando um "*war dialer"*, que é um programa que procura por números telefônicos que estão conectados a máquinas que permitem acesso remoto por meio de modems. Usando essa ferramenta, o *hacker* pode vasculhar entidades comerciais por algumas horas, bem como identificar todos os *hosts* que estão aceitando acesso remoto. Muitas vezes o intruso pode trabalhar dentro da organização, sendo um usuário local.

3.3.2.2 Ataques Baseados em Senhas

Existem várias formas de ataques baseados em senhas que permitem um intruso de rede ganhar acesso remoto. Assim que o *hacker* escolhe um usuário, ele espera que o usuário tenha criado uma senha fraca[8] e tenta conseguir o acesso a conta do usuário

[8] Senha facilmente dedutível por meios de tentativa e erro, ataques de dicionário ou força bruta.

manualmente através do método de força bruta. O meio mais fácil de acessar um *host* "pela porta da frente" é entrando com a senha de acesso de um usuário autorizado.

O intruso também deve ser hábil para capturar um nome de usuário e sua respectiva senha através de um *sniffer* de pacotes. Ou ele pode obter um arquivo de senha do *host* alvo como no UNIX `/etc/passwd` e deve conseguir quebrá-las com o auxílio de força bruta sobre o arquivo de senhas.

É possível proteger-se contra ataques de força bruta criando e reforçando a política de segurança para dificultar ao máximo a quebra de senhas ou contas de acesso do tipo convidado, misturando caracteres não alfanuméricos com caracteres em maiúsculos alternados entre si, alem de estipular um número mínimo de caracteres e um tempo de vida para a senha. Também é necessário que o envio de senhas através de uma rede insegura seja cifrada e garanta a segurança do arquivo de senhas dos usuários.

3.3.2.3 Obtendo Acesso Privilegiado ou Confiável

Um computador confiável é um computador onde se possui controle administrativo ou um em que confiamos, para permitir acesso a uma rede/sistema.

Sabendo-se disso, um intruso tentará explora-lo para conseguir um acesso com mais privilégios ou para explorar as relações confiáveis entre máquinas visando acessar outros computadores. Espera-se com isso ganhar acessos privilegiados comparados com os níveis de privilégio de root ou administrador sobre um *host* sem possuir uma conta privilegiada no *host* alvo. O atacante pode forjar um usuário confiável usando seu nome de usuário e senha ou ele poderá se passar por um *host* confiável para ganhar acesso no outro *host*.

3.3.2.4 Ganhando Acesso Secundário

Depois de um intruso obter o acesso inicial ao sistema, ele pode deixar algo escondido no sistema para facilitar o seu retorno, mais conhecido como um *backdoor*, em seguida deve limpar os rastros de sua invasão e posteriormente retornar e utilizar o *host* invadido

como ponte para acessar mais alvos. O intruso pode tentar algumas das seguintes alternativas para tentar ocultar o acesso não autorizado:

- Limpar *logs* e remover traços do sistema;
- Mover arquivos de contabilização (*accounting*) para o diretório `/tmp`, esperando que eles sejam eventualmente apagados;
- Instalar um *sniffer* de pacotes para observar o tráfego;
- Instalar um ou vários *backdoors* com a criação de novos nomes de usuários e senhas ou instalando programas de cavalos de tróia como *rootkits*[9] (ALTUNERGIL, 2002) ou BackOrifice (CERT, 1998).

3.3.2.5 Ataque a Serviços que Permitam Acesso Remoto

Existem muitas aplicações IP e serviços que fazem *hosts* e dispositivos de rede vulneráveis a ataques remotos. Muitas aplicações foram desenvolvidas para facilitar, não para prevenir, seu acesso. Alguns serviços têm pouco ou nenhum método de autenticação implementado para assegurar que o usuário remoto possa ser validado, torna-se recomendado a desabilitação de todos os serviços que não estão sendo utilizados e principalmente aqueles que permitem o acesso remoto a um *host* ou equipamento de rede. Abaixo seguem algumas das principais vulnerabilidades de alguns dos muitos serviços de rede existentes:

- Comandos R do mundo UNIX: esses comandos conhecidos pela característica de começados pelo caractere **r** implementam a autenticação baseada no endereço fonte e são facilmente forjadas, resultando em acesso total a *hosts* executando os serviços remotos;

---

[9] Rootkits são programas que possuem como intuito ocultar as evidencias de que o sistema foi invadido e proporcionar ao atacante um conjunto de ferramentas como *sniffers*, substituição dos arquivos binários de diagnóstico como `ps`, `netstat`, `ifconfig`, `ls` dentre outros, que são instalados após uma invasão bem sucedida.

- FTP: é um protocolo utilizado para a transferência de arquivos que podem permitir a usuários anônimos ler e possivelmente escrever arquivos em um *host*;

- Finger: este serviço pode ser utilizado para descobrir informações sobre usuários visando descobrir os nomes de acesso, se o usuário já acessou o sistema pela primeira vez e qual foi seu ultimo acesso;

- NFS: Permitem acesso a arquivos em um *host* remoto. Possui sua autenticação baseada no endereço de origem, que pode ser facilmente forjado;

- Telnet: Permite um usuário acessar remotamente um *shell* sendo a comunicação não segura (em texto aberto). Seu acesso é controlado simplesmente pelo par usuário e senha que podem ser forjados ou descobertos;

- TFTP (*Trivial File Transference Protocol*): Possui a mesma finalidade do FTP mas não implementa nenhum tipo de autenticação;

- HTTP, Servidores WEB: As vulnerabilidades incluem *bugs* no software do servidor, má configuração e sistema operacional inseguro. Java, JavaScript e applets ActiveX podem carregar vírus ou cavalos de tróia.

3.3.2.6 Programas Vulneráveis que Permitem Acesso Remoto

Muitos programas utilizados para a comunicação na Internet foram e são escritos na linguagem de programação C/C++. Esses programas utilizam *buffers*, áreas de tamanho fixo na memória de trabalho, para as variáveis de dados. *Buffers overflows* ocorrem quando um intruso de rede com conhecimentos de programação em C deliberadamente tenta exceder o tamanho fixo de um *buffer* do programa para ganhar acesso não autorizado ao *host* alvo. *Buffers overflows* são erros comuns de programação.

3.3.3 Manipulação de Dados

Um intruso pode capturar, manipular e devolver dados enviados através de um canal de comunicação pelo uso da manipulação de dados. A manipulação de dados também é conhecida como personificação. Podendo ser divido em diversas formas como endereço IP *spoofing*, roubo ou manipulação de sessões, desvio de roteamento e repudio. A manipulação de dados torna-se viável pelas vulnerabilidades presentes no protocolo IP, serviços associados e suas aplicações. A Figura 6 exemplifica esta forma de ataque, onde um intruso intercepta algum tráfego, manipula-o e o devolve ao seu destino real.

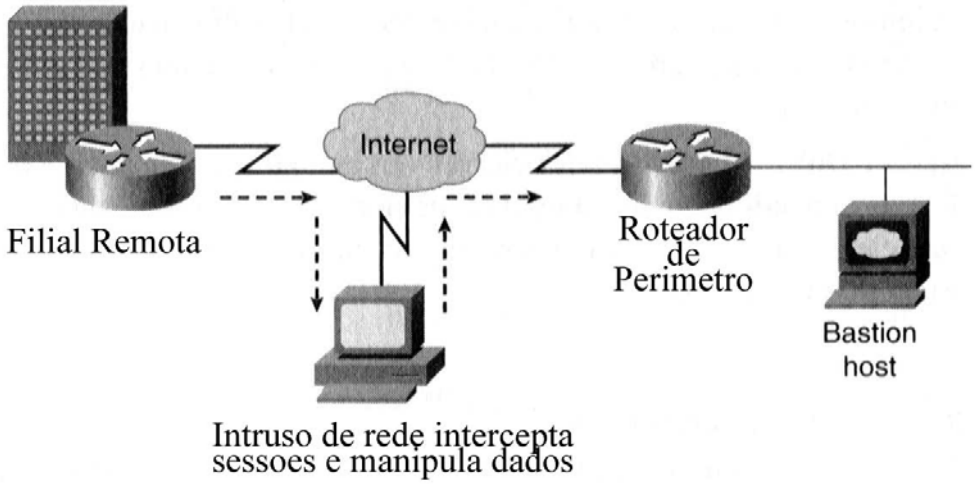

Figura 6 - Manipulação de Dados (WENSTROM, 2001)

3.3.3.1 IP Spoofing

IP *Spoofing* é uma técnica, na qual uma máquina é autenticada por outra, através da falsificação de pacotes IP, indicando proveniência de uma fonte segura (Ferguson e Senie, 1998).

A técnica de *spoofing*, quebra a segurança em um nível muito discreto, isto é, ele não necessariamente compromete o arquivo de senhas do equipamento, através de um ataque do tipo "força bruta", mas sim estabelece uma relação de confiança, entre o equipamento comprometido e o equipamento do atacante, passando pelas barreiras de segurança. Na Figura 7 é apresentado um demonstrativo do que consiste o ataque.

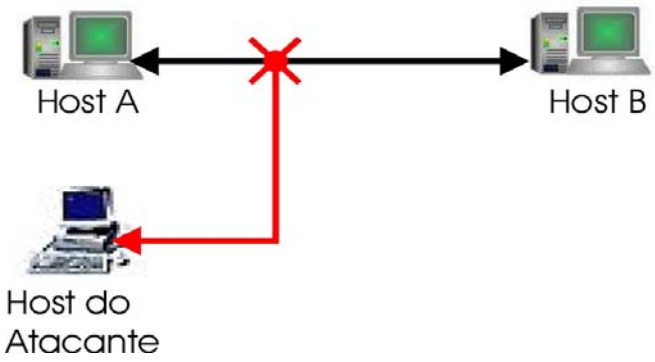

Figura 7 - IP Spoofing

Existem inúmeras maneiras de se utilizar à técnica conhecida como IP *spoofing*, podendo ser utilizada para os mais diversos fins, tanto para comprometer determinados equipamentos, obtendo informações não privilegiadas, como para comprometer uma rede inteira, enganando os usuários, para conseguir dados através da monitoração de seus pacotes.

Entre dois equipamentos, o mero fato da autenticação ser baseada em endereços IP de origem, não garante por si só um ataque de *spoofing*. Para que exista uma conexão entre dois equipamentos, é necessário mais do que o endereço IP de origem. É preciso que um diálogo completo e sustentado seja mantido entre as duas máquinas.

Em redes IP, este processo envolve procedimentos relacionados ao "*three way handshaking*" encontrado no protocolo TCP/IP (BRADER, 1994).

O protocolo IP é o responsável pelo transporte dos pacotes. Este transporte não é confiável, isto é, não há garantias de que o pacote irá chegar ao destino intacto (ele pode ser corrompido durante o transporte, pode ser perdido, etc). A idéia principal, é que o IP simplesmente leva os pacotes do ponto A ao ponto B, sem garantir a sua integridade. Assim, o primeiro passo para uma conexão além do mero transporte, é garantirmos um meio de validar se os pacotes chegaram inteiros ou não ao seu destino, e o mesmo esteja correto.

Uma vez que os pacotes tenham chegado ao seu destino, o TCP assume. Ele tem capacidades para conferir se um pacote foi desviado ou mesmo corrompido no

transporte. Como resultado, o TCP verifica os pacotes na sua recepção e envia mensagens de volta indicando que foram recebidos e entendidos.

Este processo ocorre seqüencialmente, isto é, caso três pacotes fossem enviados, (pacotes um, dois e três), cada um terá um número de seqüência associado e um índice. Ambos os equipamentos usam estes números para checagem e controle. Ao recebê-los no destino, o TCP irá reordená-los baseado nos seus índices e responderá à origem indicando a recepção dos mesmos.

O problema reside justamente no número de seqüência. O estabelecimento de uma sessão dá-se da seguinte forma:

1. A máquina de origem manda um pedido de conexão para a máquina alvo, contendo um número de seqüência;

2. A máquina alvo manda um pacote de resposta ao endereço IP de origem aceitando o pedido de conexão e enviando um número de seqüência aleatório, e uma confirmação: o número de seqüência recebido incrementado em uma unidade;

3. Neste momento, para consolidar a conexão, a máquina de origem deve também devolver um pacote de confirmação contendo o número de seqüência recebido do destino, incrementado mais uma unidade.

Assim, temos três pacotes (*three way handshaking*) para estabelecer uma conexão.

Para o hacker, a tarefa resume-se a dois passos: forjar o endereço IP de origem e manter um diálogo com o equipamento alvo. O segundo passo torna o IP *Spoofing* um ataque extremamente complexo, eis o porque:

O número de seqüência não é aleatório. O alvo estabelece o número de seqüência e o atacante deve presumir a resposta correta. Novamente, isto não é fácil, pois o atacante jamais recebe o pacote inicial de conexão do alvo, uma vez que o alvo está respondendo para o endereço IP forjado (que é de uma máquina confiável diferente da do atacante, obviamente).

3.3.3.2 Desvio ou captura de Sessões

Resposta à sessão é um ataque no qual um intruso de rede intercepta e captura uma seqüência de pacotes ou comandos de aplicações, manipulando-os (como por exemplo, alterar o valor de uma aplicação financeira) e finalmente os encaminha para seu destino real, o que causará uma ação não autorizada. Respostas a sessões exploram a fraqueza da autenticação do tráfego de dados.

3.3.4 Denial of Service

Os ataques de negação de serviço, mais popularmente conhecidos por DoS (*Denial of Service*), são utilizados em grande escala como uma tentativa para desabilitar ou corromper redes, sistemas ou serviços fazendo com que os usuários legítimos não consigam obter o acesso a estes recursos. Os intrusos que utilizam essa forma de ataque podem ser comparados a vândalos. O principal problema está no protocolo IP, mais especificamente na sua versão 4, que é altamente vulnerável a ataques DoS, além disso muitas ferramentas de ataques estão disponíveis para o acesso público e possuem seu uso relativamente fácil.

Segundo Toumas et. al. (2000) uma solução para este problema é realizar a autenticação do cliente antes que o servidor que ele deseja acessar tenha consumido qualquer recurso. A autenticação, no entanto, criaria novas oportunidades para ataques de DoS, devido o protocolo geralmente necessitar de um servidor para guardar os dados dos estados atuais de uma sessão, como por exemplo, computar operações que demandam bastantes recursos de processamento como operações de chave-pública.

Estes tipos de ataques podem ser lançados contra *perimeter routers*, ou *bastion hosts*, ou *firewalls*, conforme o ilustrado na Figura 8.

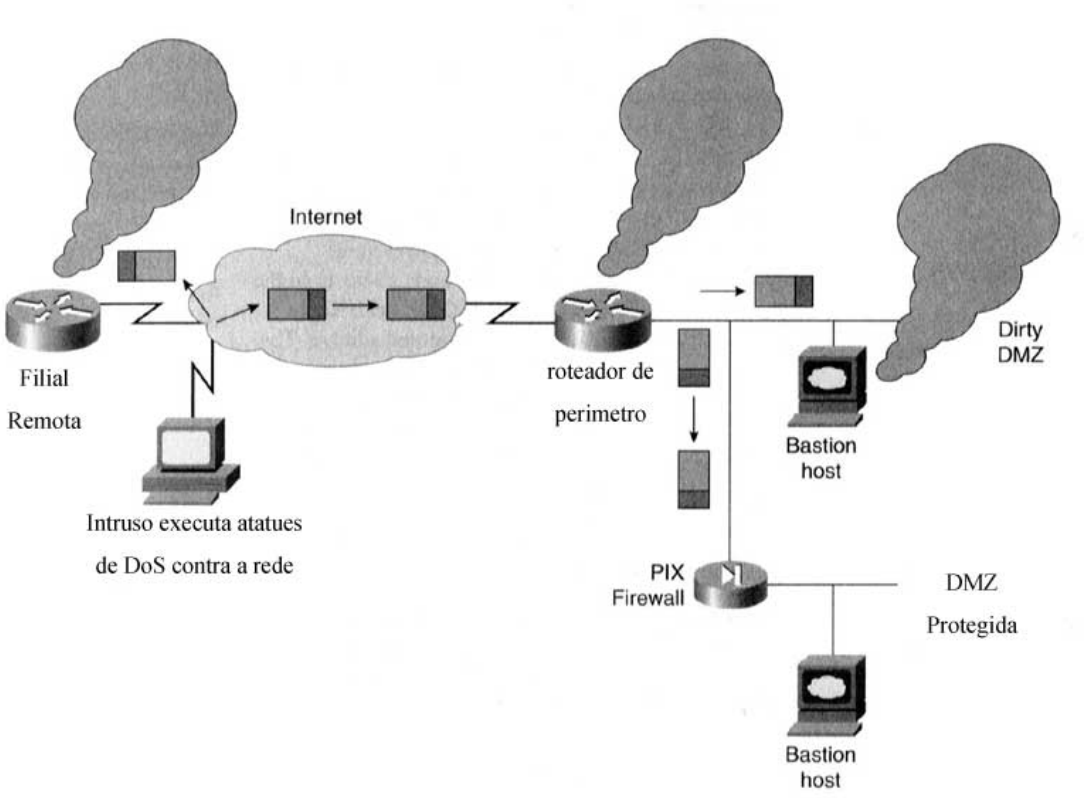

Figura 8 - Ataques de Negação de Serviço (WENSTROM, 2001)

Estes ataques acabam causando diversos problemas para uma rede ou serviços, onde o intruso tenta sobrecarregar os recursos alvos, incluindo a largura de banda de uma interface, o uso interno de memória, capacidade de processamento ou espaço em disco.

Este tipo de ataque foi aprimorado para ser executado de maneira distribuída, com isso surgio o DDoS.

3.3.5 Distributed Denial of Service

Segundo Dariva (2002), um ataque de *Distributed Denial of Service* (DDoS) consiste na criação uma rede de atacantes com tecnologia Cliente/Servidor. O atacante é capaz de multiplicar a efetividade do DoS significativamente, reunindo os recursos de múltiplos computadores cúmplices inconscientes que servem como plataforma de ataque.

Tipicamente um programa DDoS mestre é instalado em um computador, com uma senha roubada ou através da invasão da máquina. O programa mestre, na hora

designada, comunica-se com os programas agentes, instalados em quaisquer computadores na Internet. Os agentes, quando recebem o comando, iniciam o ataque.

O atacante é a pessoa que comanda os *handlers* ou *masters*, *softwares* que rodam em algum(s) computador(es), que por sua vez comunicam-se com os *daemons*, *softwares* que rodam no mesmo computador do *handler* ou em algum computador remoto e são os responsáveis por disparar o ataque contra a vítima, conforme apresentado na Figura 9.

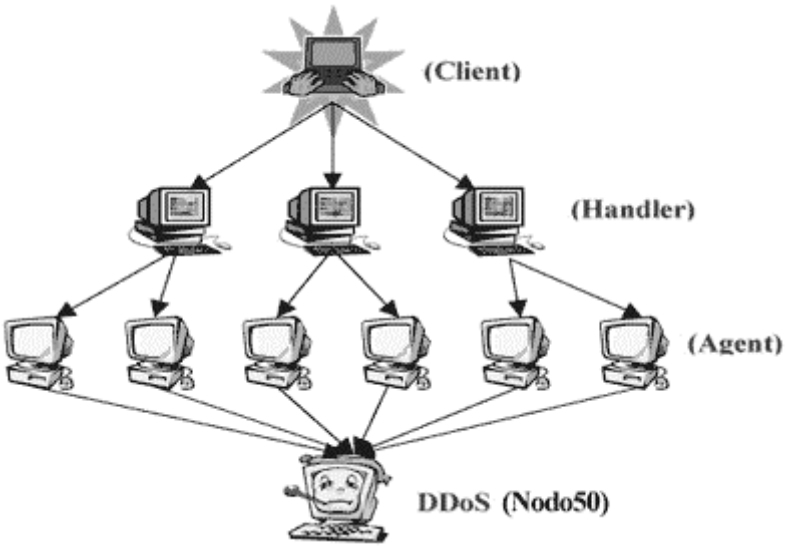

Figura 9 - Atacante envia comandos para master/handler que comanda daemons.

3.4 DETECÇÃO DE INTRUSÃO

A grande parte dos sistemas computacionais possuem algum tipo de vulnerabilidade originados por problemas de implementação, configuração ou projeto, que poderão ser futuramente explorados para os mais diversos fins, como meio de negação de serviço ou até mesmo permitir um acesso não autorizado ao sistema. Com o surgimento destes ataques, foram e continuam sendo desenvolvidos vários métodos/ferramentas a fim de detectar o mau uso a partir de análise do comportamento dos usuários/sistemas, análises de assinaturas de ataques dentre outras técnicas.

Uma segunda linha de defesa é a detecção de intrusão que está sendo constantemente estudada e utilizada nos dias atuais. A detecção de intrusão é motivada por diversos aspectos (STALLINGS, 1998):

- Se um intruso é detectado rapidamente, ele poderá ser identificado e expulso do sistema sem causar nenhum dano ou nenhum dado ser comprometido;
- Um efetivo sistema de detecção de intrusão pode servir como um impedimento, assim atuando na prevenção de intrusos, e
- Detecção de intrusão facilita a coleta de informações sobre técnicas de intrusão, que podem ser utilizadas para facilitar a prevenção de intrusão.

Detecção de intrusão é baseada na suposição de que o comportamento do intruso difere do usuário legítimo, podendo assim ser diferenciado. Não podemos esperar que o comportamento seja totalmente diferente, ou que exista distinção exata entre um ataque vindo de um intruso e o uso normal de recursos por um usuário autorizado. No entanto, devemos esperar que haja alguma sobreposição conforme apresentado na Figura 10.

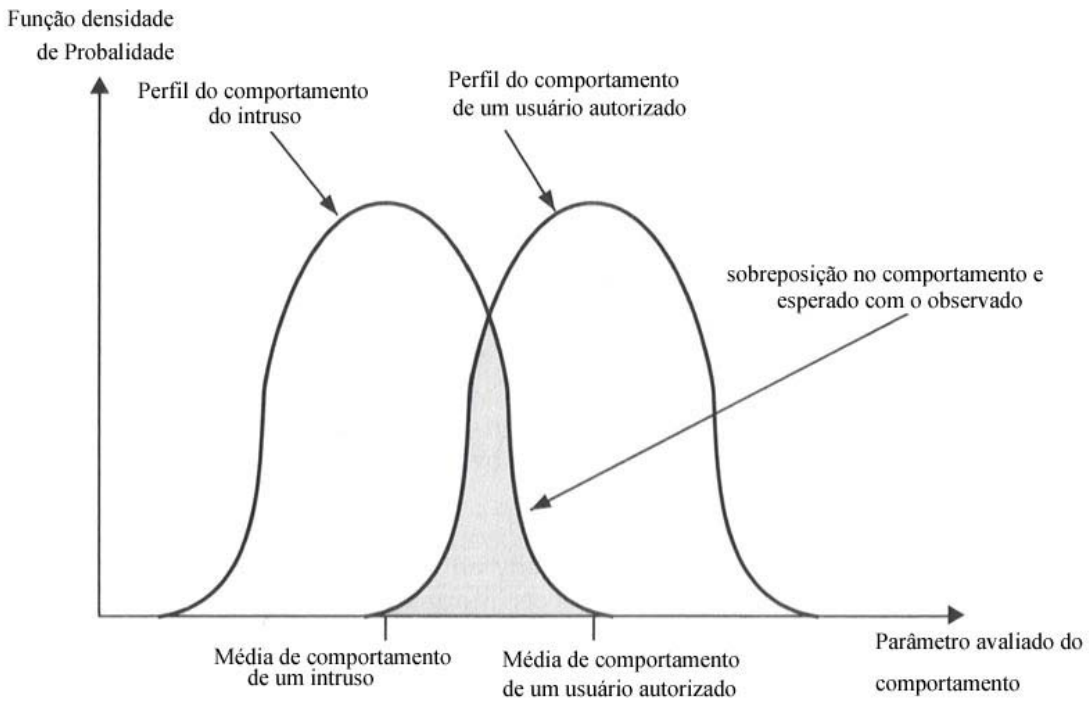

Figura 10 - Perfil do comportamento de Intrusos e Usuários Autorizados (STALLINGS, 1998)

A Figura 10 sugere, em termos muito abstratos, a natureza da tarefa que confronta o sistema de detecção de intrusão. O comportamento dos usuários legítimos e dos intrusos se sobrepõe e neste ponto a diferenciação entre eles torna-se muito difícil.

3.5 DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

Para detectar com uma maior eficácia as tentativas de intrusão são imprescindíveis conhecer bem o que se deseja defender, sendo a rede, sistema operacional ou algum serviço e, principalmente, os protocolos de comunicação utilizados para poder diferenciar a forma de como um serviço deve ou não se comportar.

Existem diversos métodos para detectarmos ataques que utilizam a infraestrutura de rede para atacar servidores interligados na Intranet/Internet. Um dos mais conhecidos e que tem ganhado um grande estudo e utilização são os NIDSs (*Network Intrusion Detection Systems*). Esses NIDSs trabalham de modo promíscuo; ficam escutando o que está trafegando pela rede, como um *sniffer*, e quando encontram "assinaturas" de ataques conhecidos geram alertas para os administradores.

Para uma melhor compreensão do funcionamento de um NIDS, analisaremos os *logs* gerados pelo sistema SNORT (2001). Quando uma nova forma de ataque ou ameaça que se propaga através das redes de computadores é descoberta, podemos analisar os pacotes que estão trafegando a fim de conseguir distingui-los dos demais. Para isso um bom *sniffer* irá ajudar nessa tarefa, como o TCPDUMP (2001) ou ETHEREAL (2001). Analisaremos o Verme Code-Red que explora a vulnerabilidade presente nos servidores *Internet Information Server* - IIS- Microsoft versão 4 e 5.

Segundo a análise realizada por MOORE (2001), em 19 de julho de 2001 mais de 350,000 computadores foram infectados com o verme Code-Red (CRv2) em menos de 14 horas. Em proporções podemos contar que 2000 novos *hosts* foram infectados a cada minuto.

Tabela 1 - Os 10 paises mais infectados em 14 horas (MOORE, 2001)

País	Hosts Contaminados	Percentual
Estados Unidos	157.694	43.91 %
República da Correa do Sul	37.948	10.57 %
China	18.141	5.05 %
Taiwan	15.124	4.21 %
Canada	12.469	3.47 %
Reino Unido	11.918	3.32 %
Alemanha	11.762	3.28 %
Austrália	8.587	2.39 %
Japão	8.282	2.31 %
Holanda	7.771	2.16 %

Para identificarmos este ataque, a seguinte "assinatura" (Quadro 1) foi inserida no detector de intrusos SNORT (2001):

```
alert tcp any any <> any 80 (msg: "CodeRed Defacement"; flags: A+;
content: "|FF8B8D64 FEFFFF0F BE1185D2 7402EBD3|"; depth:64;)
```

Quadro 1 - Regra para detecção do verme CodeRed no Snort

Esta regra indica que um alerta será gravado onde o protocolo utilizado seja **TCP**, tendo com origem qualquer máquina/porta fonte para qualquer destino na porta 80 (www). Com *flags A+* de ACK e + para qualquer outro e que contenha o código `(FF8B8D64 FEFFFF0F BE1185D2 7402EBD3)` em hexadecimal. E Finalmente o `depth:64` indica que a *string* deve ser procurada até o 64º byte do pacote a partir da carga útil.

O resultado dessa regra aplicada ao NIDS resultará gravação deste evento do arquivo de *logs* (Quadro 2) quando um pacote que contenha esta "assinatura" coincidir com a regra anteriormente vista no Quadro 1.

```
                [**] [1:0:0] CodeRed Defacement [**]
08/01-05:39:58.524827 62.242.150.120:40413 -> 200.237.249.253:80 TCP
TTL:107 TOS:0x0 ID:12067 IpLen:20 DgmLen:1155 DF
***AP*** Seq: 0xE1ED1B43 Ack: 0x1753144F Win: 0x4470 TcpLen: 20
```

Quadro 2 - Log gerado pelo Snort

Analisando o log gerado pelo Snort, podemos notar que a máquina infectada 62.242.150.120 está tentando atacar a máquina 200.237.249.253. Caso esta máquina alvo possua um servidor web vulnerável, o verme tentará explorar a vulnerabilidade e poderá conseguir infectar a máquina.

Após o administrador receber este alerta, deverá tomar as medidas adequadas para tentar controlar esse problema, como por exemplo atualizar os servidores Web, instalar filtros, remover o verme caso exista alguma máquina em sua rede. Esse ataque é de difícil controle, pois ele inicialmente verifica se existe algum servidor Web ativo, enviando inúmeros pedidos de requisição TCP/SYN na porta 80 para máquinas diferentes, inundando as redes (pois ele tenta atingir qualquer máquina ligada a Internet).

Em um caso real, a rede da Universidade Federal de Santa Catarina - UFSC, estava sendo inundada por milhares de requisições TCP/SYN/WWW oriundas da Internet, o que ocasionou em primeira análise, um decréscimo da performance na rede local, devido os roteadores não estarem agüentando o número de conexões. Para solucionar o problema foi implantado um filtro geral que nega todo o estabelecimento de conexão para servidores Web com o tráfego vindo de qualquer localidade para a rede interna. Isso resultou na não disponibilidade dos servidores Web internos para a Internet. Tornou-se necessário criar uma lista de servidores Web que podem ser acessados a partir da Internet, (antes de liberá-los foi verificado se aquele servidor não estava vulnerável). Foi realizada uma checagem em todas as máquinas da rede UFSC e quando uma máquina vulnerável for encontrada é enviado um alerta para o administrador do servidor. Esta lista em fevereiro de 2002 estava com aproximadamente 200 *hosts*.

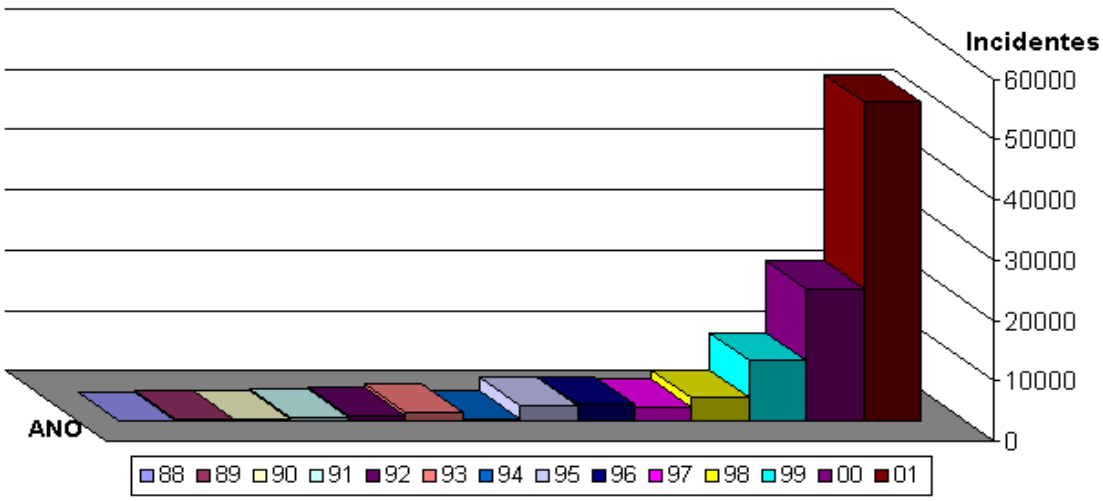

Figura 11 - Incidentes de segurança registrados pelo CERT (2002a)

Os ataques a sistemas computacionais registrados estão aumentando a cada ano em relação ao ano anterior. O CERT (2002a) registrou no período de 1998 a 2001 um total de 100,369 incidentes reportados, onde somente no ano de 2001 foram registrados 52,658 casos. A Figura 11 mostra a evolução dos ataques reportados ao Cert desde sua criação em 1998.

Este capítulo abordou sucintamente alguns ataques que são utilizados contra redes de computadores e sistemas computacionais, além das características de um intruso e um usuário autorizado, seus perfis de comportamento, os tipos de intrusos e alguns métodos de detecção de intrusão, dando um foco mais para a detecção de intrusos em redes de computadores.

No próximo capítulo são apresentadas as arquiteturas do sistema, as linguagens utilizadas no desenvolvimento e detalhes da implementação do protótipo.

4 - SISTEMA IMPLEMENTADO

O presente trabalho tem por objetivo principal desenvolver um método alternativo de detecção de intrusões em redes de computadores baseando-se na observação da alteração do comportamento característico dos equipamentos de rede gerenciáveis.

Suas principais dificuldades são detectar quando está sendo atacado ou servindo de ponte para um ataque a uma outra rede, principalmente em uma rede de grande porte, como nos *backbones* de grandes instituições.

Foram utilizadas diversas ferramentas, sendo JAVA a linguagem utilizada para realizar a análise das informações coletadas dos equipamentos através do protocolo de gerência de redes SNMP. Os relatórios serão apresentados ao analista através de uma interface web amigável com auxílio da linguagem PHP e do gerenciador de banco de dados MySQL.

4.1 MODELO LÓGICO DO PROTÓTIPO SRIDS (*SNMP ROUTER INTRUSION DETECTION SYSTEM*)

O presente trabalho visa monitorar um determinado número de roteadores, onde informações como percentual de uso da CPU, *queue drops*, *buffers* de filas, vazão de pacotes em uma determinada interface ou outra variável da MIB que possa ser monitorada através do protocolo de gerência SNMP e possibilite a criação de uma *baseline*. Estas variáveis são coletadas pelos gerentes e guardadas em uma base de dados para futuras análises, para que se possa modelar o comportamento de cada roteador, conforme ilustra a Figura 12.

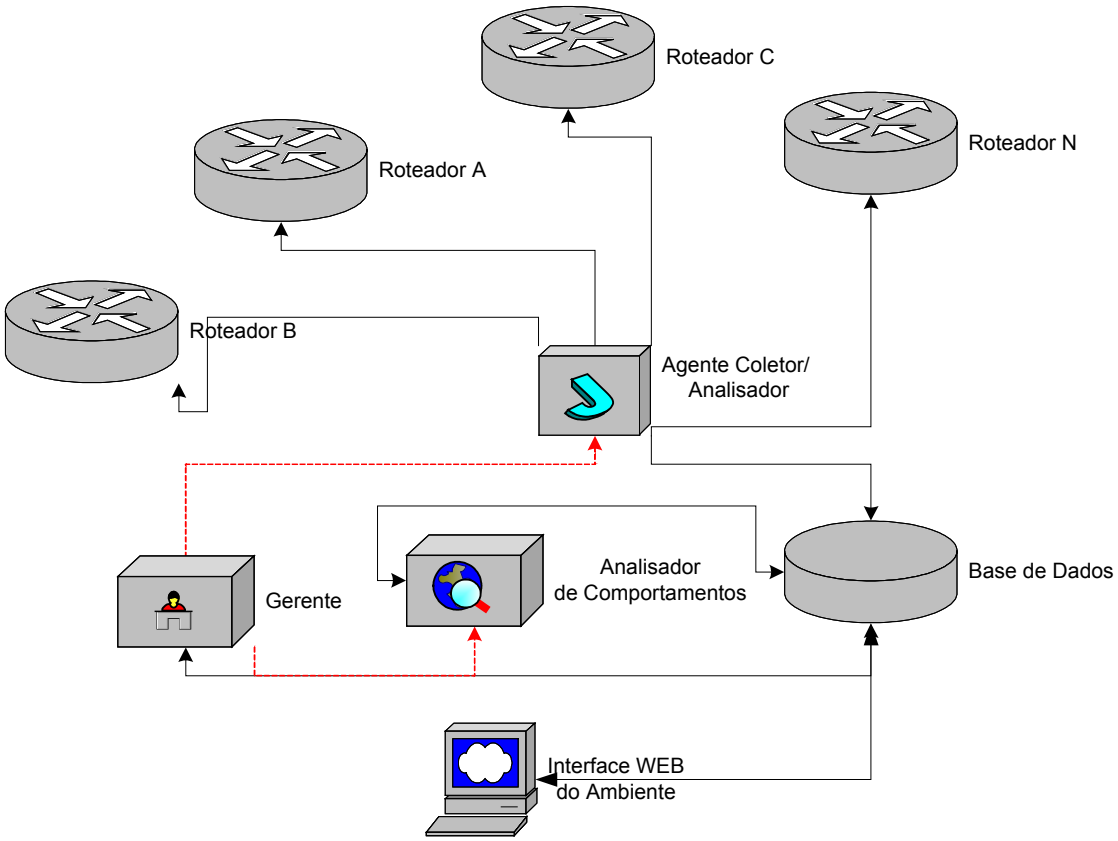

Figura 12 - Modelo Lógico do SRIDS

O analisador de comportamentos tem o papel de consultar a base de dados gerada pelo(s) agente(s) e a partir destas informações conseguir modelar o comportamento de um roteador específico para aquele período de tempo e comparando-o com os eventos anteriores de mesma ordem, isto é, nos dias anteriores no mesmo intervalo de tempo. Na grande maioria dos casos o uso da CPU comporta-se de maneira semelhante aos períodos de mesma ordem anteriores, onde será estipulado um percentual de tolerância para a geração de eventos quando o comportamento sair dos padrões analisados. Essa faixa de tolerância é tida pelo desvio padrão que é calculada automaticamente pela ferramenta e por um percentual de erro regulável pelo administrador do sistema.

O papel do gerente como seu próprio nome indica é ser o elemento principal que comandará as ações a serem executadas pelos outros elementos de menor ordem, isto é, configurar os agentes e o analista, alertar o administrador se algum componente do modelo está com mau funcionamento, dentre outras.

4.2 BASE DE DADOS UTILIZADA E SEUS RELACIONAMENTOS

No presente trabalho optou-se pelo uso do gerenciador de banco de dados MySQL (AXMARK, 2002) por ser *freeware,* dispor das funcionalidades mínimas para o desenvolvimento do trabalho, ser um gerenciador de banco de dados de alta performance possibilitando respostas rápidas às consultas realizadas, suportar *multi-threads* para acessos simultâneos. Um fator determinante para sua escolha foi a fácil integração com o JAVA e PHP, linguagens utilizadas na implementação.

A base de dados é composta por oito tabelas, que apresentam as seguintes funcionalidades:

Tabela 2 - Descrição das tabelas que compõe a base de dados do protótipo

Tabela	Descrição
Roteadores	Esta tabela possui o cadastro de todos os roteadores que estão sendo gerenciados, se ele está apto para o monitoramento e qual a comunidade SNMP de leitura para a coleta das variáveis.
Coletas	Apresenta os valores de todas as coletas referentes aos roteadores e suas variáveis da MIB e o tempo em que foi realizado a coleta.
Comportamento	O comportamento é utilizado como uma tabela *cache*, onde ficam armazenadas as análises de comportamentos de todos os equipamentos que estão sendo monitorados por 24hs.
Alarmes	Esta tabela descreve os tipos de alarmes que podem ocorrer de acordo com as características previamente estabelecidas.
MIB	Esta tabela contêm as variáveis da MIB que poderão ser gerenciadas. Torna-se possível à inclusão de novas variáveis pelo usuário final.
Estados	Esta tabela armazena os estados pré-estabelecidos que serão utilizados em outras tabelas, são eles: • 0: Não apto ou não configurado para o monitoramento; • 1: Apto para monitoramento; • 2: Equipamento em atividade suspeita mas ainda monitorando; • 3: Monitoramento desabilitado pelo software por atividade suspeita. • 4: Somente monitorando e não gerando alarmes.

Roteador_MIB	Esta tabela faz a junção entre os roteadores e as variáveis que estão sendo monitoradas para cada equipamento.
Alarmes_Histórico	Esta tabela guarda o histórico de todos os alarmes que ocorreram com cada roteador.
Administradores	A tabela de administradores é utilizada para armazenar para cada roteador quais são os endereços onde os alertas serão enviados, como e-mail e servidor de TRAPs.

Formalmente, a Figura 13 representa o diagrama das entidades e seus relacionamentos (ER) que é utilizada no presente trabalho.

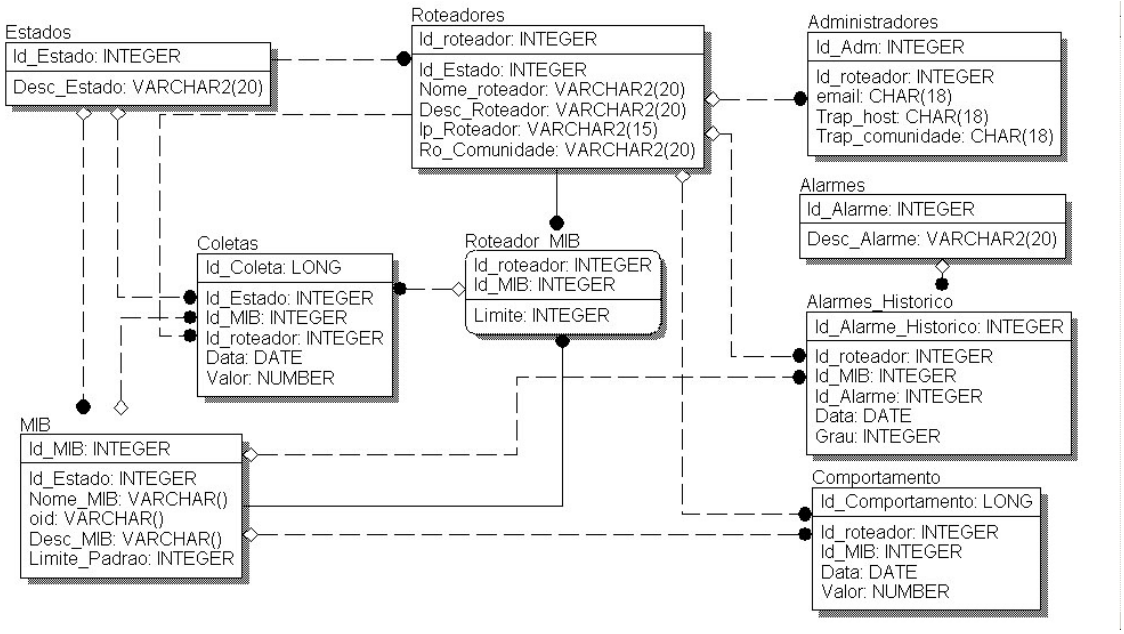

Figura 13 - Diagrama ER do protótipo

4.2.1 Implementação da Classe Banco

A classe Banco realiza a interface entre a aplicação e a base de dados através do *driver* de conexão JDBC MM.MySQL desenvolvido por Mark Matthews (MATTHEWS, 2002). Este *Driver* JDBC permite que desenvolvedores Java realizem conexões de usas aplicações e *applets* com servidores MySQL. O Quadro 3 apresenta uma parte da implementação desta classe.

Quadro 3- Implementação da Classe Banco

```
/ Banco.java
// Definição da Classe Banco

import java.util.*;
import java.sql.*;
import java.io.*;
/**
 *  Esta classe realiza vários funções no banco de dados MySQL
 *  Através do driver mm.mysql
 *  @autor Guilherme Eliseu Rhoden
*/
public class Banco extends Object {
  /**
  * Banco contrutor padrao irá criar as tabelas
  *
  */
  public Banco() {
   //inicializaTabelas();
  }

  public String execSQL(String host, String baseDados, String usuario,
String senha, String SQL){

    String erro="OK";
    try{ // TRY #1 - Carregar o driver MySQL - JAVA
        Class.forName("org.gjt.mm.mysql.Driver").newInstance();
        } catch (Exception E) {
           erro="Não foi possível carregar o driver mm.mysql !";
           E.printStackTrace();
       }// fim TRY #1

     try { //TRY #2 - Conecta na base de dados
          String end_conexao =
"jdbc:mysql://"+host+"/"+baseDados+"?user="+usuario+"&password="+senha
;
          Connection con = DriverManager.getConnection(end_conexao);
          Statement st = con.createStatement();
          st.executeQuery(SQL);
          st.close();
          con.close();
          } catch (SQLException E) {
             erro="Ocorreu um Erro na Execução do comando SQL!\n
SQLException: " + E.getMessage()+"\nSQLState: " +
E.getSQLState()+"\nVendorError: " + E.getErrorCode();
           }// FIM TRY #2
   return  erro;
      }// FIM public execSQL()
```

4.3 IMPLEMENTAÇÃO DO AGENTE/ANALISTA

Foi agregado ao agente uma parte da função do analista conforme o modelo mostrado na Figura 12, por motivos de praticidade e melhor precisão na realização dos alertas,

pois, o agente é encarregado de coletar as informações dos equipamentos gerenciados em intervalos de tempos pré-determinados e a melhor hora para verificar se os valores recolhidos estão de acordo com os padrões adquiridos.

4.3.1 Visão geral do Agente/Analista

O agente foi implementado na linguagem JAVA por motivos já apresentados e ainda por suportar o protocolo de gerenciamento de rede SNMP através da biblioteca AdventNET SNMP API, disponibilizada gratuitamente pela empresa AdventNET (ADVENTNET, 2001).

A estrutura geral do Agente/Analista é apresentada na Figura 14, que consiste inicialmente em recuperar da base de dados quais roteadores estão aptos a serem monitorados, destes roteadores são retiradas informações específicas, tais como, endereço IP do equipamento, comunidade SNMP para leitura das variáveis e os objetos da MIB que deverão monitorados, com seus respectivos OIDs e seu percentual de variação do comportamento.

Após estas variáveis serem retiradas da base de dados, o programa em Java inicializará *multithreads*, sendo atribuída uma *thread* para cada roteador, onde os seguintes procedimentos e testes são realizados para cada variável da MIB que está sendo monitorada pelo Agente/Analista, podendo ser acompanhado no fluxograma apresentado na Figura 14.

Inicialmente, é realizado a coleta do objeto gerenciável com o auxílio da API SNMP da AdventNET (ADVENTNET, 2001), onde o primeiro teste realizado é se o SNMP GET do OID do roteador retornou algum valor, caso não tenha retornado após o tempo limite máximo de espera, um aviso de *time out* é gerado no console e na tabela de alertas da base de dados e a *thread* é notificada para adormecer até o instante da próxima coleta, repetindo esta parte do ciclo. Caso tenha retornado algum valor, é realizado o teste se o valor extraio do agente SNMP da MIB está de acordo com o seu padrão de comportamento, que foi adquirido dinamicamente através de sua *baseline* de comportamento anteriormente gerada por outro processo no inicio de cada dia e tendo seus valores armazenados na base de dados. Caso o valor coletado estiver de acordo

com o perfil de comportamento do equipamento, o valor variável é armazenado na base de dados, onde contribuirá para o ajuste do *baseline* que será atualizada em no máximo 24h, em seguida o processo é adormecido até o tempo da nova coleta ser realizado.

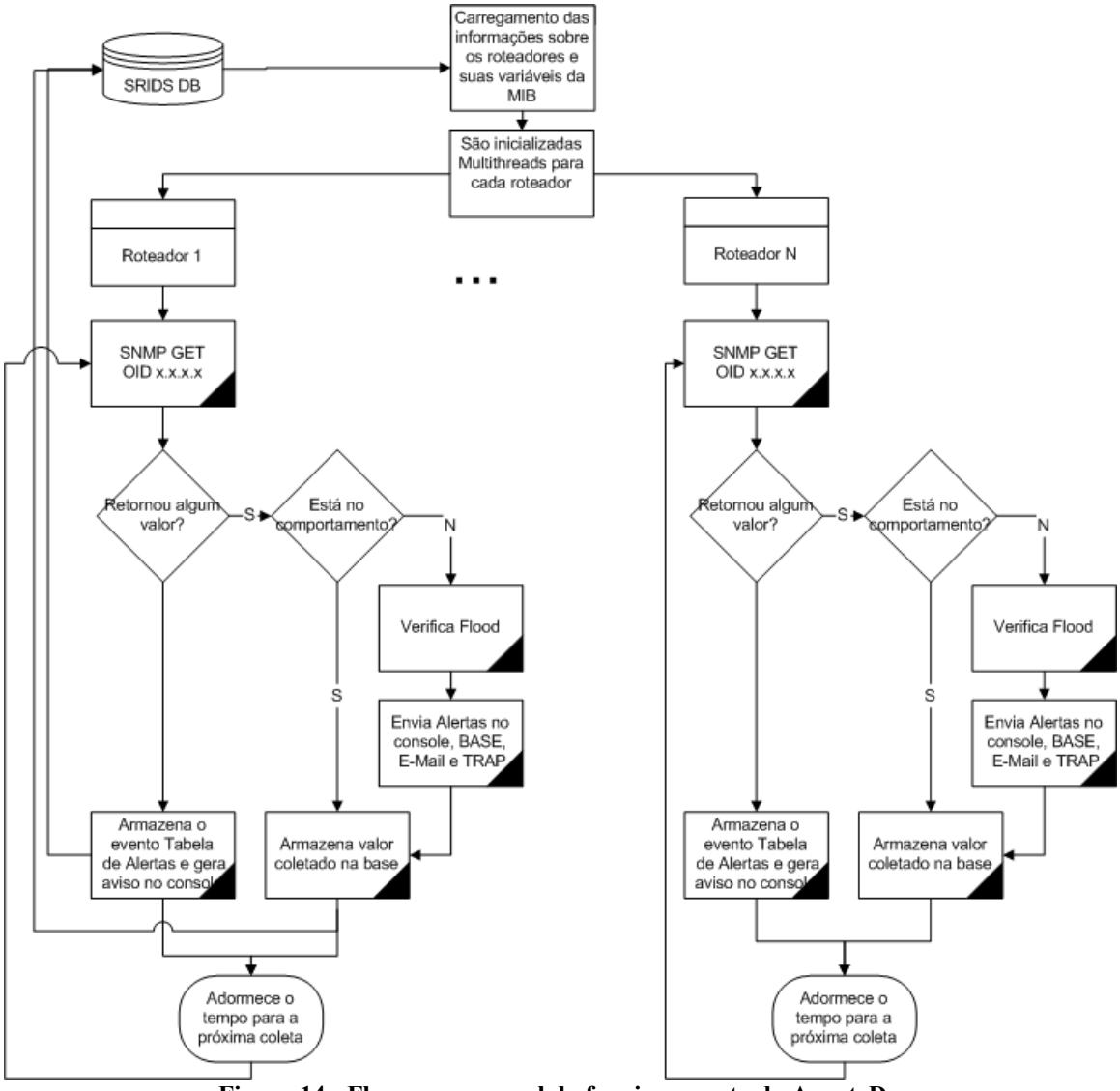

Figura 14 - Fluxograma geral do funcionamento do AgenteD

Se a variável coletada não está de acordo como o padrão de comportamento os últimos procedimentos a serem realizados, são:

- Verificar se o alerta já não foi enviado anteriormente, possibilitando assim não gerar *flood*, e
- Enviar o alerta.

No procedimento de verificação *anti-flood*, é verificado se o mesmo alerta já foi gerado anteriormente, em caso de positivo serão enviados outros alertas após um certo número de eventos ocorrerem, sendo apresentados que o último evento desta variável da MIB para este roteador ocorreu mais de 10 ou 100,ou 200, ou 500, ou 1000 vezes. Caso o evento que será gerado for o primeiro, alertas através de e-mail aos administradores são gerados informando-os do ocorrido, envio de TRAPs SNMP para a estação de gerência, alerta no console do sistema e por fim o armazenamento do alerta na base de dados.

4.3.2 Implementação da Classe AgenteD

A função do agente/analista é verificar quais roteadores estão aptos a serem monitorados com o uso da base de dados. Após haver a definição de quem deve ser monitorado, são inicializadas execuções simultâneas através do uso de *threads*, onde uma *thread* acaba sendo responsável pelo monitoramento de um roteador e suas variáveis. Conforme o Quadro 4.

Quadro 4 - Implementação do Agente/Analista – Classe AgenteD

```
// AgenteD.java
// Definição da Classe Agente

                                    ...

public class AgenteD{
  //Habilitar Debug
  private boolean debug = false;
  private String versao = "v0.3b", atualizado = "13/03/20002";

   /**
    * AgenteD contrutor padrao irá criar as tabelas
..  *
    */
   public AgenteD() {
      imprimeInfo();
      Inicializa();
  }

  public static void main (String args[]){
    AgenteD a = new AgenteD();
  }

}
```

O Quadro 5 consulta na base de dados quais roteadores estão aptos a serem monitorados e repassa suas identificações para o método `InicializaThreads` em forma de vetor e o número de roteadores que estão sendo monitorados.

Quadro 5 - Inicialização dos equipamentos que devem ser monitorados

```
public void Inicializa() {
  int total = 0;
  String tmp, SQL;
  String idRoteador[] = {"-1",...,"-1"};

  String ip_b, user_b, senha_b, base_b;
  //Carregar Configurações
   String configuracao[] = {"0"};
   try {
        Config c = new Config();
        configuracao = c.retornaConfig();
       }
        catch (IOException e)  {
            System.err.println("IO Error:" + e);
         }

     ip_b = configuracao[0];
     user_b = configuracao[1];
     senha_b = configuracao[2];
     base_b = configuracao[3];
    //retirar os valores referentes ao id do roteador do Bando de dado

     Banco b = new Banco();
     Snmp s = new Snmp();
     try {
       SQL = "SELECT Id_Roteador from Roteadores where Id_Estado > 0";
       ResultSet rs = b.retornaRS(ip_b,base_b,user_b,senha_b,SQL);

      //Retira todas os roteadores que estão aptos a serem monitorados
       while (rs.next()) {
          tmp = rs.getString("Id_Roteador");
          idRoteador[total] = tmp;
          total++;
         }
      } catch ( Exception e){
              e.printStackTrace();
              return;
     }//End Try

      InicializaThreads(idRoteador,total);
     }// FIM Inicializa()
```

A parte principal do programa está distribuída em forma de *threads*, onde uma *thread* é inicializada para cada roteador, sendo responsável pela coleta, armazenamento e

validação dos dados coletados das variáveis da MIB de cada equipamento. Podendo ser observado no Quadro 6.

Quadro 6 - Método InicializaThreads

```
private void    InicializaThreads(String idRoteador[], int total)  {
   int x;
   SNMPThread t1[] = {null,...,null};

   for (x=0;x<total;x++) {
      t1[x] = new SNMPThread(idRoteador[x]);
      if (idRoteador[x] != "-1") {
      t1[x].start();
    }
   }// FIM IncializaThreads
```

4.3.3 Implementação da Classe SNMPThread

Na classe SNMPThread são realizadas as coletas das variáveis e o seu armazenamento na base de dados. Em seguida são realizadas verificações conforme o comportamento adquirido e caso o comportamento não estiver adequado, alertas são inicializados. No Quadro 7 é apresentado o início da classe SNMPThread onde é repassado a identificação do roteador.

Quadro 7 - Classe SNMPThread

```
class SNMPThread extends Thread {
  private boolean debug = false;

  public SNMPThread(String Id){
     super( Id );
  }//FIM    public SNMPThread(String Id)

...
```

Para uma melhor compreensão do funcionamento de uma *thread*, será apresentada uma rápida visão do seu ciclo de vida e os estados que a compõe. Uma *thread* pode estar em um dos seguintes estados (DEITEL, 2001):

- Executando: A *thread* está rodando;
- Pronto: A *thread* está apta para rodar;

- Esperando: A atividade da *thread* está parada temporariamente, após receber uma notificação de *wait*;
- Adormecido: A *thread* está "dormindo" por um tempo estipulado;
- Bloqueado: A execução da *thread* está bloqueada à espera de um recurso de I/O, e
- Morto: A execução da *thread* foi encerrada, não podendo ser retomada.

A Figura 15 mostra graficamente os estados do ciclo de vida de uma *thread*:

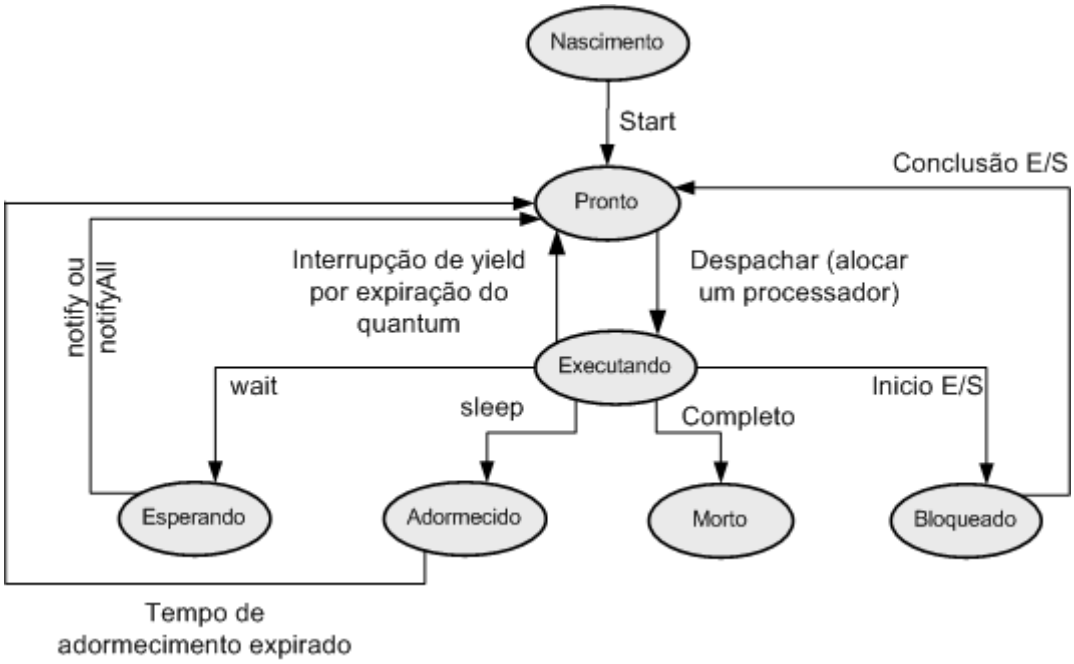

Figura 15 - Ciclo de vida de uma *thread* (DEITEL 2001)

No programa implementado, o principal estado da classe SNMPThread é o executando, pois é nele que as coletas são realizadas, armazenadas, realizada as verificações do perfil de comportamento e é onde são tomadas as ações iniciais de quando um alerta deve ser disparado.

Quadro 8 - Estado executando do SNMPThread

```
  public void run()     {
    int tempoColeta = 60, sleepTime;
    String ip, data, valor,sql,comunidade, h, m,ss, resposta, SQL,
ip_b, user_b, senha_b, base_b, smtp, email, msg;

    String mibOID_TIPO[] = {"1",..."1"}, mibOID[] = {"1",...,"15"},
mib_ID[] = {"1",...,"15"};

    long Valor_Anterior[] = {-1,...,-1}, comp1, comp2, tmp3, teste;
    int i = 0, numVar = 0 , estado = 1, desvio_padrao, numFlood =
0,numFlood2 = 0, tmp2;
    char tmp[] = new char[3];
    float testePercentual = 0;

//tempo entre coletas em milisegundos
    sleepTime = tempoColeta * 1000;

    //Alertas...
    Alerta a = new Alerta();
    String configuracao[] = {"0"};
    try {
       Config c = new Config();
       configuracao = c.retornaConfig();
     }
      catch (IOException e)  {
            System.err.println("IO Error:" + e);
       }

     ip_b = configuracao[0];
     user_b = configuracao[1];
     senha_b = configuracao[2];
     base_b = configuracao[3];
     smtp = configuracao[4];
     email = configuracao[5];

     //retirar os valores referentes ao id do roteador do Bando de
dado

    Banco b = new Banco();
    Snmp s = new Snmp();
    try {
      SQL = "SELECT r.Ip_Roteador, r.Ro_Comunidade, m.oid, m.Id_MIB,
m.Tipo  FROM Roteadores r, MIB m, Roteador_MIB rm where r.Id_Roteador
= '"+getName()+"' and rm.Id_Roteador = '"+getName()+"' and m.Id_MIB =
rm.Id_Mib";

      ResultSet rs = b.retornaRS(ip_b,base_b,user_b,senha_b,SQL);
      ip = rs.getString("Ip_Roteador");
      comunidade = rs.getString("Ro_Comunidade");

      //Retira todas as variáveis que deverão ser monitoradas
      while ( rs.next() ){
        mibOID[numVar] = rs.getString("oid");
        mib_ID[numVar] = rs.getString("Id_MIB");
        mibOID_TIPO[numVar] = rs.getString("Tipo");
        numVar++;
      }
```

```
      } catch ( Exception e){
              e.printStackTrace();
              return;
    }//End Try

     //Enviar aviso de inicio de monitoramento por e-mail
     a.SendMail(4, getName(), "0", 0, ip_b, base_b, user_b, senha_b,
smtp, email);

...
```

A parte principal do estado executando da *therad* é composta por um *loop* onde inicialmente é coletado o valor do objeto através da classe SNMP que é apresentado no Quadro 9. Após a coleta da variável, é recuperado da base de dados os valores referentes ao perfil de comportamento (valor da *baseline*) e o valor do desvio padrão para a variável da MIB que está sendo monitorada (Quadro 10).

Quadro 9 - procedimento principal da *thread* em *loop* e coleta do objeto na MIB

```
...
  //loop principal
  do{
      for (i=0;i<numVar;i++){
         resposta =  s.snmpGet(ip,comunidade,mibOID[i],10);
         java.sql.Date d = new java.sql.Date(
System.currentTimeMillis() );
         java.text.SimpleDateFormat format = new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
         if (resposta.equals("-1") || resposta== "" ){
              resposta = "-1";
              System.out.println(format.format(d)+" Thread: " +
getName() + " -> Tempo Limite ultrapassado!!!");
        }//end IF
...
```

Quadro 10 - retirada do valor atual da *baseline* e percentual de erro permitido para essa variável

```
...
   sql = "SELECT ROUND(Valor) FROM Comportamento where Id_Roteador =
'"+getName()+"' and ( Hora >= '"+h+":"+m+":"+ss+"' and Hora <
'"+h+":"+tmp2+":00') and Id_Mib="+mib_ID[i];
  ResultSet rs2 = b.retornaRS(ip_b,base_b,user_b,senha_b, sql);
  valor = rs2.getString("ROUND(Valor)");
  //retira o valor do desvio padrãp
  sql = "SELECT ROUND(STD(Valor))  FROM Coletas WHERE Id_Roteador =  "
+ getName()+" AND `Id_MIB` =  "+mib_ID[i];  rs2 =
b.retornaRS(ip_b,base_b,user_b,senha_b, sql);
  desvio_padrao = Integer.parseInt( rs2.getString(
"ROUND(STD(Valor))"));
...
```

Existem vários tipos de variáveis que podem ser implementadas em uma MIB SNMP, sendo os principais tipos implementados para o SNMP v1 apresentados na Tabela abaixo (PERKINS & MCGINNIS, 1997).

Tabela 3 - Tipos de variáveis do SNMP v1 (PERKINS& MCGINNIS, 1997)

Nome	Tamanho/Tipo	Descrição
INTEGER	-2147483648 (-2^{31}) até 2147483647 (2^{31}-1)	inteiro com sinal
Gauge	Inteiro sem sinal de 0 até 4294967295 (2^{32}-1)	Gauge
Counter	Inteiro sem sinal de 0 até 4294967295 (2^{32}-1)	contador
TimeTick	Inteiro sem sinal de 0 até 4294967295 (2^{32}-1)	tempo decorrido
OCTET STRING	Byte (2^{8}-1)	bytes
OBJECT IDENTIFIER	OBJECT IDENTIFIER	Identificação única de um objeto
IpAddress	OCTET STRING(SIZE (4)) sem sinal	Endereço IPv4
Opaque	OCTET STRING	Encapsulação de ASN.1 BER
BITS	OCTET STRING	Label de bits

O protótipo implementado está apto para monitorar e analisar o comportamento de qualquer variável do SNMP v1 do tipo inteira, ou seja, Integer, *gauge*, *counter*, etc... com tamanho até 64 *bits* (*long*) o que facilitará futuras adaptações do protótipo ao SNMP v2 e v3, pois a versão 1 do protocolo trabalha somente com inteiros de até 32 bits. No Quadro 11 são realizados os tratamentos necessários para a adequação destes tipos de variáveis numéricas ao programa.

Quadro 11 - Padronização dos tipos de variáveis e verificação do desvio padrão

```
...
comp1 = Long.parseLong(resposta);//valor atual

   //Testes para variáveis do tipo COUNTER de 32 bits
   //http://www.faqs.org/rfcs/rfc1155.html
   //Tamanho do counter 2^32-1 = 4294967295
   //
   if (mibOID_TIPO[i].equals("3")) {
     //caso tenha zerado o contador - OK
     if ((Valor_Anterior[i] >  comp1) && (Valor_Anterior[i] != -1 )) {
     tmp3 = (long) 536870912 * 8;
```

```
     comp1 = tmp3 -  Valor_Anterior[i] + comp1;
     Valor_Anterior[i] = comp1;
     comp1 = comp1 / tempoColeta;
     } else
     //se for a primeira variavel
     if (Valor_Anterior[i]== -1 ) {
        Valor_Anterior[i] = comp1;
        comp1 = 0;
       } else {
         tmp3 = comp1;
         comp1 = (comp1 - Valor_Anterior[i]) / tempoColeta;
         Valor_Anterior[i] = tmp3;
        }
     resposta = "" +comp1;
    }// Fim IF Para testes incrementais...

    //Realizar teste se a variável coletada desviou de seu padrão de
comportamento
    if ((valor != null) && (! valor.equals("0")) ) {
       comp2 = Integer.parseInt(valor);//valor atual
       testePercentual = comp2 - comp1;
...
```

Após haver a normalização das variáveis é realizado o teste se a variável coletada está fora do desvio padrão. Por exemplo, se o desvio padrão (DVP) para uma variável da MIB referente ao roteador é igual a 4, o seu padrão médio de comportamento (PC) é igual a 10 e o valor atual (VA) coletado da MIB é igual a 7, logo VA está dentro do padrão, pois o valor PC - VA está dentro do desvio padrão e todos os valores do intervalo de 4 a -4 inclusive estão de acordo com o comportamento. Caso o valor esteja fora do intervalo os seguintes alertas poderão ser gerados (Quadro 12):

- Envio de alertas no console;
- Envio de *e-mails* ao administrador;
- Envio de SNMP TRAPs a estação de gerência e
- Armazenamento do ocorrido na base de dados.

Quadro 12 - Geração de Alertas e armazenamento das variáveis

```
...
   if (testePercentual > desvio_padrao || testePercentual < -
desvio_padrao) {
    //Imprime Aviso no console do sistema

    System.out.println(format.format(d)+" -:ALERTA:- ROTEADOR:
"+getName()+ " resultado fora do padrao de comportamento em
"+testePercentual+ " pontos. Valor Atual " + resposta);

//envia e-mail e grava na base
    if (estado[i] == 1) {
       a.SendMail(estado[i], getName(), mib_ID[i], testePercentual,
ip_b, base_b, user_b, senha_b, smtp, email);
       //envia TRAP
       a.enviaTrap(estado[i], getName(), mib_ID[i], testePercentual,
ip_b, base_b, user_b, senha_b,valor);

       SQL = "INSERT INTO Alarmes_Historico (`Id_Alarme_Historico`,
`Id_Roteador`, `Id_MIB`, `Id_Alarme`, `Id_Estado`, `Data`, `Grau`)
VALUES ('','"+getName()+"','"+mib_ID[i]+"', '"+"', '"+estado[i]+"',
'"+format.format(d)+"', '"+testePercentual+"')";
       b.execSQL(ip_b,base_b,user_b,senha_b, SQL);
     }

    //Alertas sem fazer Flood!
    if (numFlood == 10 || numFlood == 100 || numFlood == 200 ||
numFlood == 500 || numFlood == 1000){
      a.SendMail(3, getName(), mib_ID[i], numFlood, ip_b, base_b,
user_b, senha_b, smtp, email);
       //envia TRAP
       a.enviaTrap(3, getName(), mib_ID[i], testePercentual, ip_b,
base_b, user_b, senha_b,numFlood);

      SQL = "INSERT INTO Alarmes_Historico (`Id_Alarme_Historico`,
`Id_Roteador`, `Id_MIB`, `Id_Alarme`, `Id_Estado`, `Data`, `Grau`)
VALUES ('','"+getName()+"','"+mib_ID[i]+"', '"+"', '4',
'"+format.format(d)+"', '"+testePercentual+"')";
      b.execSQL(ip_b,base_b,user_b,senha_b, SQL);
    }

     numFlood++;
     estado[i] = 2; //já foi enviado um alerta
    } else if (estado[i] == 2) {

       //Imprime Aviso no console do sistema
       System.out.println(format.format(d)+" -:OK:- ROTEADOR:
"+getName()+ " retornou ao seu perfil de comportamento = "+valor+ "
pontos. Valor Atual " + resposta);
       estado[i] = 1;
       numFlood = 0;
       numFlood2++;
       //Manda e-mail informando do retorno
       a.SendMail(2, getName(), mib_ID[i], testePercentual, ip_b,
base_b, user_b, senha_b, smtp, email);
       //envia TRAP
       a.enviaTrap(2, getName(), mib_ID[i], testePercentual, ip_b,
base_b, user_b, senha_b);
```

```
       //grava o registro na tabela de alertas...
       SQL = "INSERT INTO Alarmes_Historico (`Id_Alarme_Historico`,
`Id_Roteador`, `Id_MIB`, `Id_Alarme`, `Id_Estado`, `Data`, `Grau`)
VALUES ('','"+getName()+"','"+mib_ID[i]+"', '"+"', '"+estado[i]+"',
'"+format.format(d)+"', '"+testePercentual+"')";
       b.execSQL(ip_b,base_b,user_b,senha_b, SQL);
      }
     }//end IF if (valor!="")

   //Grava o Valor da coleta com o estado
   // 1 = Dentro do Comportamento
   // 2 = Fora do COmportamento
   SQL= "INSERT INTO `Coletas` (`Id_Coleta`, `Id_Roteador`, `Data`,
`Id_MIB`, `Valor`, `Id_Estado`) VALUES
('','"+getName()+"','"+data+"','"+mib_ID[i]+"','"+resposta+"','"+estad
o[i]+"')";
   b.execSQL(ip_b,base_b,user_b,senha_b, SQL);

...
   try {
      //adormece até a próxima coleta
      Thread.sleep( sleepTime ); }
...
```

4.4 ANÁLISE DAS VARIÁVEIS DA MIB

Este trabalho pode ser reutilizado para gerenciar qualquer equipamento que implemente a MIB SNMP, sendo ela privada ou pública, existindo uma gama muito diversificada de variáveis que poderão ter seu comportamento analisado a fim de detectar ataques ou desvio do comportamento normal para cada equipamento.

As variáveis aqui analisadas correspondem ao percentual de uso da CPU, vazão de pacotes em uma determinada interface, número de *buffers* utilizados e suas falhas e o percentual de uso da memória. A maioria destas variáveis não estão padronizadas pela MIB-II e por isso utilizamos as variáveis da MIB privadas dos equipamentos CISCO e IBM.

Em (Cisco, 1999) é apresentado um guia para monitoramento e correlação de eventos para seus equipamentos gerenciáveis (Cisco *switches* e routeadores), onde após diversos estudos e medições foram selecionadas as seguintes variáveis referentes as MIBs:

Tabela 4 - *OLD-CISCO-CPU-MIB* variáveis para o monitoramento da utilização de CPU

Objetos	Descrição	OID
BusyPer	Percentual de utilização da CPU nos últimos 5 segundos.	.1.3.6.1.4.1.9.2.1.56
AvgBusy1	Percentual médio de CPU ocupada no último minuto.	.1.3.6.1.4.1.9.2.1.57
AvgBusy5	Percentual médio de CPU ocupada nos últimos 5 minutos.	.1.3.6.1.4.1.9.2.1.58

Com relação aos testes realizados, o uso de cada variável dependerá do intervalo em que são realizadas as coletas e qual é o tempo em que se espera obter algum resultado. Por exemplo, se utilizarmos o objeto *BusyPer*, precisaremos realizar os SNMP *Pooling* a cada 5 segundos, o que nos trás um desvio de comportamento com uma maior rapidez mas tem seus inconvenientes, como sobrecarregarem mais os equipamentos e gerar mais falsos alarmes, pois alguns picos de uso da CPU em intervalos pequenos são freqüentes e dificilmente causam lentidão na rede. A variável que mais se adaptou em nosso ambiente de testes foi a *AvgBusy1*, que retorna a média de uso a cada 60 segundos, economizando assim o número de *Poolings* e gerando menos falsos possitivos.

Outra MIB analisada foi a *CISCO-MEMORY-POOL* (Tabela 5) e *OLD-CISCO-MEMORY* (), onde pretendemos monitorar a utilização de memória e *buffers*.

Tabela 5 - *CISCO-MEMORY-POOL-MIB*

Objetos	Descrição	OID
CiscoMemoryPoolName	Nome em formato texto assinalado ao pool de memória.	.1.3.6.1.4.1.9.9.48.1.1.1.2
CiscoMemoryPoolUsed	Número de *bytes* do pool de memória que estão em uso no momento.	.1.3.6.1.4.1.9.9.48.1.1.1.5
CiscoMemoryPoolFree	Número de *bytes* do pool de memória que não estão alocados no momento.	.1.3.6.1.4.1.9.9.48.1.1.1.6
CiscoMemoryPoolLargestFree	Maior número de *bytes* contíguos do pool de memória que não estão usados no momento.	.1.3.6.1.4.1.9.9.48.1.1.1.7

Tabela 6 – Objetos estudados da *OLD-CISCO-MEMORY-MIB*

Objetos	Descrição	OID
Elementos de Buffers		
BufferElFree	número de *buffers*	.1.3.6.1.4.1.9.2.1.9
BufferElMax	número máximo de *buffers*	.1.3.6.1.4.1.9.2.1.10
BufferElHit	número de *buffers* alocados com sucesso	.1.3.6.1.4.1.9.2.1.11
BufferElMiss	número de falhas na alocação dos *buffers*.	.1.3.6.1.4.1.9.2.1.12
BufferElCreate	Número de *buffers* criados.	.1.3.6.1.4.1.9.2.1.13
Falha de Buffer		
BufferFail	Número de falhas na alocação de *buffers*	.1.3.6.1.4.1.9.2.1.46
BufferNoMem	Número de falhas na criação de *buffers* devido a não haver memória disponível.	.1.3.6.1.4.1.9.2.1.47

Estas variáveis foram testadas nos roteadores Cisco, mas a que mais se adaptou a análise de comportamento foi a *AvgBusy1*. O outro conjunto de variáveis analisadas foi dos roteadores IBM-CPU-MIB (IBM, 1998).

Tabela 7 - IBM-Memory-MIB e IBMCPU

Objetos	Descrição	OID
IBM-MEMORY		
ibmappnMemoryUsed	número de bytes na memória compartilhada que são alocados atualmente para o processo APPN.	.1.3.6.1.4.1.2.6.2.13.1.7.2
IBMCPU		
IbmMainProcessorLoadTable	tabela contendo a média do percentual de uso do processador central para cada minuto nos últimos 60 minutos.	
IbmMainProcessorLoad	retorna o uso da CPU no momento que foi requisitado em percentual	
Nv6saComputerSystem	semelhante ao objeto a cima, foi implementado inicialmente para ser usado pela plataforma de gerência *NetView* da IBM. Esta MIB pode ser	.1.3.6.1.4.1.2.6.4.5.1

utilizada para monitorar o uso de CPU dos roteadores *Nways*. Utilizamos essa MIB nos MSS-8210 com sucesso.

4.5 ANÁLISE DO COMPORTAMENTO

Os roteadores de um *backbone* em seu funcionamento normal apresentam usos de recursos como CPU e Memória com características semelhantes aos mesmos períodos anteriores de tempo, onde torna-se possível traçar um perfil de comportamento de cada equipamento. O perfil de comportamento é basicamente o mesmo, podendo apresentar poucas variações.

Após a análise de comportamento ser atualizada a cada dia de acordo com os dias anteriores, é possível verificar se o equipamento está ou não trabalhando dentro de seus padrões normais. Observe o perfil de comportamento traçado em linha cheia da variável *ifOutUnicastPkts*, que corresponde ao número de pacotes *unicast* que estão saindo em uma interface e os valores atuais apresentados na linha grande, na Figura 16 correspondente a duas semanas de monitoramento com os valores coletados no dia Atual em linha cheia.

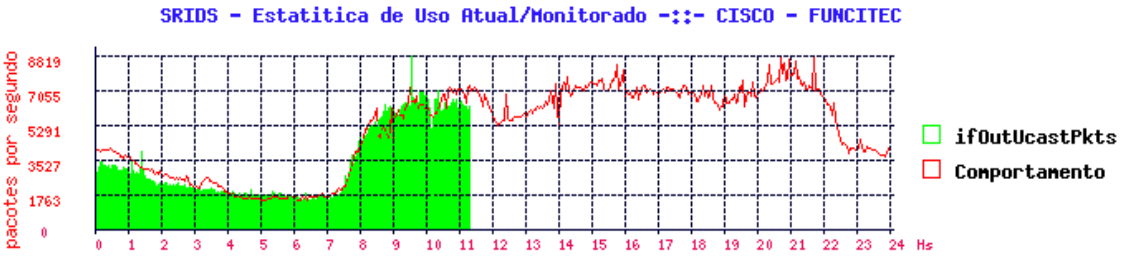

Figura 16 - Análise do comportamento de um roteador

4.6 GERAÇÃO E VISUALIZAÇÃO DE EVENTOS

Para facilitar a interação do usuário final com a ferramenta, optou-se pelo uso da linguagem PHP para criar páginas dinâmicas, que realiza a interface entre o usuário e a aplicação, através do uso da base de dados MySQL.

A configuração pode ser realizada através de uma aplicação JAVA ou WEB, inicialmente optou-se por realizar todo o trabalho em JAVA, pois além de sua portabilidade e suporte a SNMP e *Threads*, ser uma linguagem com grande reputação acadêmica. Com o desenvolvimento do trabalho, foi constatado que se a ferramenta fosse totalmente escrita em JAVA ela se tornaria muito pesada e assim deixando o usuário final descontente com sua performance acabaria não utilizando, observe a Figura 17.

Figura 17 - Aplicação de controle e visualização desenvolvida em JAVA

Para permitir seu gerenciamento em uma interface mais leve e que pudesse ser acessada de qualquer local, optou-se pela linguagem PHP que apresenta como características interessantes, possuir sintaxe semelhante à linguagem C, ser de fácil interação com banco de dados, rodar juntamente com servidor http, possibilitar a criação de páginas dinâmicas e a geração de gráficos.

Como o PHP ainda não suporta o uso de *threads*, o agente analisador é escrito em JAVA e possui como características principais verificar quais equipamentos estão aptos a serem monitorados através de uma base de dados, verificar qual é o intervalo de tempo que cada variável deve ser monitorada e em seguida iniciar a monitoração com o uso de threads. Quando uma variável específica for monitorada no roteador via SNMP, o valor recolhido é inserido na base de dados e analisado para verificar se está dentro dos padrões do equipamento. Caso não esteja, eventos serão gerados como a inserção de um aviso de alerta na base de dados para futuras verificações, envio de TRAPs SNMP para uma estação de gerência externa ao sistema e notificação através de e-mail informando sobre o ocorrido.

4.7 *BASELINE*

A consulta à MIB de um determinado equipamento gerenciável da rede nos informa uma instância da variável consultada. Porém, para definir quando um segmento monitorado está tendendo a um estado crítico, é preciso comparar a situação atual deste segmento com os dados estatísticos sobre o comportamento considerado normal para este segmento. Daí a necessidade da criação de *baselines* (VERONEZ, 2000).

Segundo Neto (NETO, 1998), a *baseline* é uma caracterização estatística do funcionamento normal de um segmento monitorado de uma rede.

Através da análise das variáveis da MIB de um equipamento gerenciável, torna-se possível adquirir o comportamento de cada variável, relacionando-a com um período de tempo. A partir desta análise, é criada uma *baseline* que representa a média do índice de uso desta variável para todos os períodos de tempo de um dia, possibilitando assim prever qual será o comportamento de uma determinada variável para o próximo dia. O estudo dessas variáveis é apresentado no capítulo 4.4.

4.7.1 Geração do Padrão de Comportamento (*baseline*)

Para se traçar um perfil de comportamento da rede, estudou-se o padrão de comportamento característico das variáveis, relacionando-o a um equipamento gerenciável, para distinguir se o mesmo está operando em suas condições normais. Desta forma, pode-se detectar ataques e/ou problemas antes do esgotamento de recursos relacionados com estes equipamentos, através do desvio de seu padrão característico de comportamento.

A maior dificuldade está na hora de traçar o comportamento de cada variável do equipamento. Esta tarefa é realizada uma vez por dia por motivos de tempo de processamento, onde são realizadas todas as médias de usos normais dos períodos anteriores, sendo dias e tempos de coletas equivalentes.

Onde a *baseline* de uma variável é obtida pela média aritmética das coletas realizadas em um mesmo intervalo de coleta, ou seja, supomos que o intervalo de coleta seja de

180 segundos e que a hora de coleta corresponde a 01:04:01hs, o comportamento é calculado pela média de todos os dias anteriores correspondentes ao intervalo de 01:03:00hs até 01:05:00hs. Esse processo é realizado para todos os períodos que constituirão o comportamento até o dia atual. Veja a linha do comportamento adquirida de um equipamento que opera em condições normais na Figura 18.

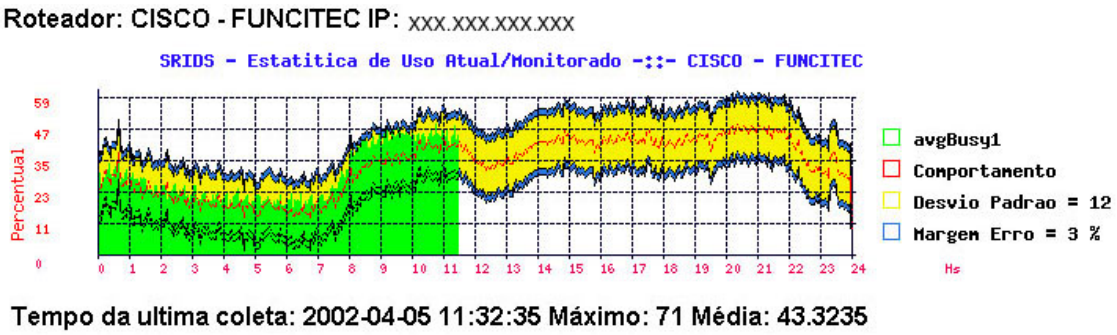

Figura 18 - Comportamento Atual, Adquirido, desvio padrão e margem de erro

Analisando o gráfico anterior retirado às 11:32hs, notamos que os valores da CPU de um roteador CISCO estão de acordo com suas condições até o presente momento. Continuando análise na linha do comportamento, que fica entre a faixa do desvio padrão e da margem de erro, espera-se que ao decorrer do dia a CPU do equipamento mantenha-se próxima da linha do comportamento. É considerado um comportamento normal quando a CPU possa chegar até os patamares do desvio padrão, podendo atingir até a margem de erro estipulada pelo administrador. Caso ocorra algum desvio superior ou inferior ao comportamento analisado, com uma margem de erro configurável, alertas serão gerados.

A visualização de eventos é apresentada de duas maneiras:

- De forma gráfica (Figura 18);
- Alertas no console (Quadro 13), e
- Também sendo possível acompanhar os alarmes na forma de tabelas para verificar o histórico de eventos que ocorreram com os equipamentos (Figura 19).

```
2002-04-16 19:15:54 -:ALERTA:- ROTEADOR: 2 resultado fora do padrão de
comportamento em -6.0 pontos. Valor Atual 11
 72002-04-16 19:16:56 -:ALERTA:- ROTEADOR: 2 resultado fora do padrão
de comportamento em -6.0 pontos. Valor Atual 11
 72002-04-16 19:17:56 -:ALERTA:- ROTEADOR: 2 resultado fora do padrão
de comportamento em -6.0 pontos. Valor Atual 11
 72002-04-16 19:18:57 -:ALERTA:- ROTEADOR: 2 resultado fora do padrão
de comportamento em -6.0 pontos. Valor Atual 11
 72002-04-16 19:19:57 -:OK:- ROTEADOR: 2 retornou ao seu perfil de
comportamento  = 5 pontos. Valor Atual 9
```

Quadro 13 - Alertas gerados no console da ferramenta

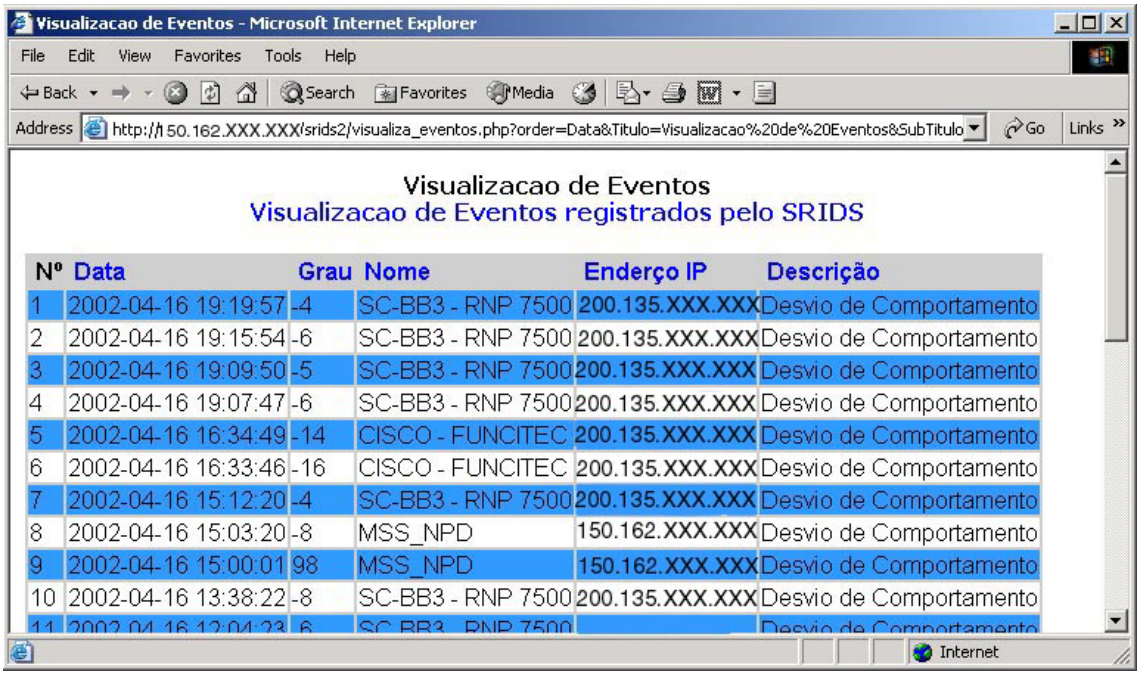

Figura 19 - Visualização de eventos pela interface WEB

O uso desta ferramenta é de grande valia para controlar o estado de um equipamento. Quando simplesmente monitorarmos suas variáveis não conseguimos obter mais do que gráficos com a plotagem dos valores de cada amostra, sem nenhum tratamento estatístico. Com isso, o administrador só poderia perceber alguma alteração visualmente e mesmo assim, se mantivesse os gráficos sob constante acompanhamento. Com o uso do SRIDS alertas são enviados e o responsável verificaria se tudo está correto.

No capítulo seguinte são apresentados os resultados, testes e validações do protótipo realizado em ambiente real e hipotético.

5 - TESTES, VALIDAÇÕES E RESULTADOS

5.1 AMBIENTE DE TESTES

O presente trabalho apresentou duas etapas de testes, sendo a primeira realizada em um ambiente de testes, onde buscou-se gerar ataques de *DoS* em uma rede experimental, a fim de detectar a presença do desvio de comportamento. A segunda parte dos testes foram realizados no *backbone* da rede da Universidade Federal de Santa Catarina, rede do PoP-SC e com equipamentos de núcleo da rede Catarinense de Ciência e Tecnologia – RCT.

5.1.1 Ambiente de Testes Simulado

Para o primeiro ambiente de testes, foi montada uma rede hipotética de um *backbone* típico, sem a presença de algum mecanismo de defesa. O ambiente de testes é mostrado na Figura 20.

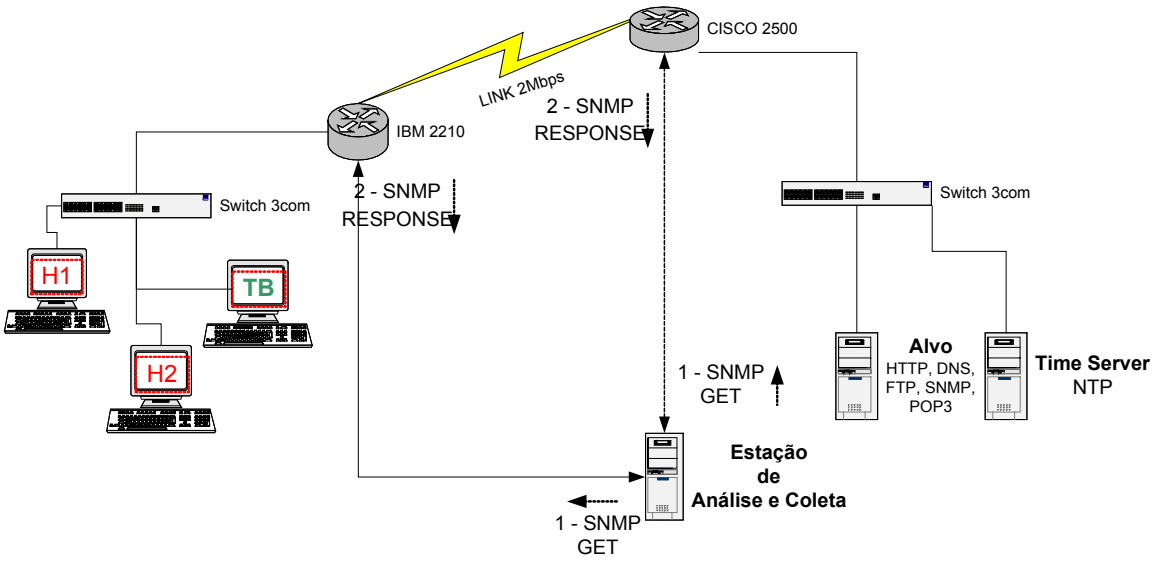

Figura 20 - Ambiente de Teste 1 (Hipotético)

O Ambiente acima é composto pelos seguintes equipamentos e sistemas operacionais:

- Sistema Operacional Linux RedHat 7.2 (H1, H2 e TB) na arquitetura IBM-PC;

- Estação de Análise, Coleta e Gerência Linux RedHat 7.1, com serviços de servidor http, suporte a linguagem PHP (V 4.0.4pl1), a máquina virtual JAVA (JDK 1.3) e como base de dados MySQL versão 3.23.36;

- Servidor FreeBSD 4.2 rodando sobre plataforma IBM-PC, com serviços habilitados de http, https, DNS, sendmail, POP3, MySQL. A configuração de *firewall* não foi habilitada;

- Servidor de *Network Time Protocol* (NTP) para realizar o sincronismo dos relógios das estações, servidores e roteadores, e

- Dois roteadores, sendo um IBM 2210 e um CISCO da série 2500, interligando as duas redes através de um link de 2Mbps.

Os servidores (NTP e o Alvo) representam serviços reais e estão dispostos no *backbone* da rede local. H1 e H2 são as estações que lançarão ataques de DoS contra o servidor alvo e os roteadores, estando em uma rede diferente representando a Internet. A estação de análise e coleta de dados ficará buscando as informações destes equipamentos de tempos em tempos e analisando seus valores de acordo com o padrão característico de cada equipamento. A estação de trabalho TB ficará encarregada de gerar tráfego e simular acessos aos serviços http, DNS e sendmail, simulando vários usuários normais.

Foram realizados 3 tipos de ataques utilizando uma série de *scripts*, que estão agrupados na ferramenta de DoS Toast 0.2 (BREEDER, 2001). Seu uso é bastante simples, e executa uma série de ataques, como: TCP, UDP e *Port floods* para diversos sistemas operacionais incluindo Linux, BSDs e Windows. As três séries de ataques envolveram as duas estações Linux tendo como alvo o servidor FreeBSD 4.4. No momento em que eram realizados esses ataques, foram coletadas informações da MIB-2 dos roteadores:

- *IfInOctets*: taxa de bytes recebidos em uma interface;

- *ifOutOctets*: taxa de bytes enviados em uma interface;

- *IpOutNoRoutes*: taxa de descartes ocorridos por falta de informação de roteamento;

- *IpOutRequests*: número de pacotes IP enviados;
- *IpInReceives*: numero de pacotes IP recebidos;
- *IpForwDatagramas*: taxa de datagramas repassados; e
- *IpInDelivers*: numero de pacotes IP de entrada entregues com sucesso.

Além dessas informações, foram analisadas as variáveis referentes ao uso percentual de processador e memória da MIB privada de cada equipamento. Enquanto os ataques eram realizados, um usuário normal tentava acessar os recursos do servidor, ou seja, http, DNS e Sendmail e foram obtidos os seguintes resultados:

1. Ataque 1, TCP SYN *flood*: esse ataque visa sobrecarregar o serviço da máquina alvo com inúmeras requisições de inicio de conexão SYN forjadas;
2. Ataque 2, UDP *Flood*: o objetivo desse ataque é inundar a máquina alvo com pacotes UDP; e
3. Ataque 3, lança uma série de ataques contra os sistemas BSDs, mais especificamente *teardrop, overdrop* , *land,* e *biffit*. Estes ataques são sucintamente explicados abaixo.

Tipos de ataques lançados contra o servidor BSD pelo script Toast (BREEDER, 2001):

1. *Teardrop* é um ataque onde um usuário remoto pode interromper o funcionamento de *host* vulnerável. Em algumas implementações do código TCP/IP na reconstrução da fragmentação IP não realizam corretamente a sobreposição de fragmentos IP, especialmente em números negativos. *Teardrop* é um desses ataque que explora esta vulnerabilidade (CERT, 1997), (NORTHCULTT & JUDY, 2000);
2. *Overdrop*: é uma modificação do *teardrop* que envia um número muito maior de pacotes;

3. *Land*: NORTHCUTT & JUDY (2000) consideram-no como sendo um ataque famoso por duas razões: por ele ser muito elegante pois pode derrubar uma máquina com um ou dois pacotes e é considerado como o "*hello word*" dos filtros de detecção de intrusão. O ataque consiste em enviar um pacote IP de início de conexão (SYN) para a máquina alvo, forjando o endereço e porta fonte para o mesmo endereço da máquina alvo (CERT, 1997). Como exemplo, pacote IP/TCP/SYN de origem 10.10.10.1 porta 80 para destino 10.10.10.1 porta 80, e

4. *Biffit*: este ataque visa travar máquinas BDSs vulneráveis através da inundação com pacotes UDPs (ANONYMOUS, 1999).

O percentual de uso da CPU nos três ataques analisados através do comando *report* em *talk 5* no IBM e *show process cpu* no CISCO são apresentados na Tabela 8.

Tabela 8 - Percentual máximo de uso da CPU referente aos 3 ataques

Roteador	Ataque 1	Ataque 2	Ataque 3
IBM	50%	13%	**100%**
CISCO	34%	20%	58%

5.1.1.1 Roteador Externo IBM

Os resultados obtidos após a realização dos ataques podem ser acompanhados graficamente na Figura 21, Figura 22 e Figura 23.

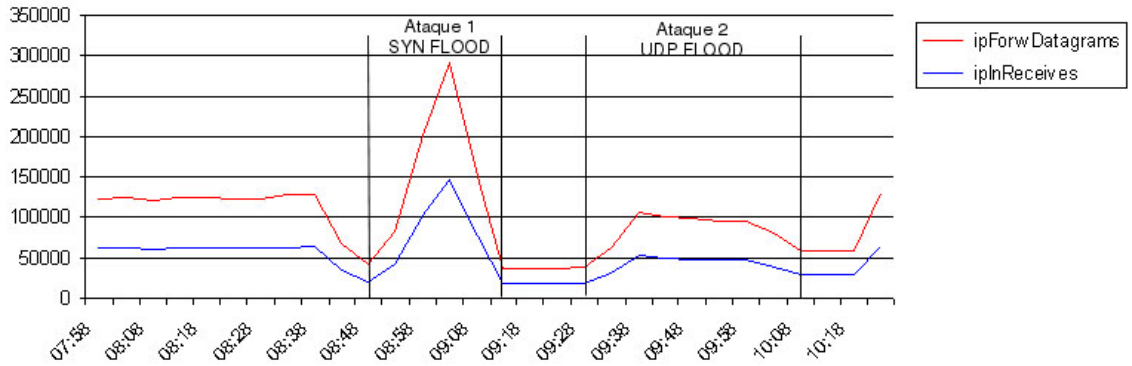

Figura 21 – Roteador IBM: *ipForwDatagrams* após os 2 primeiros ataques

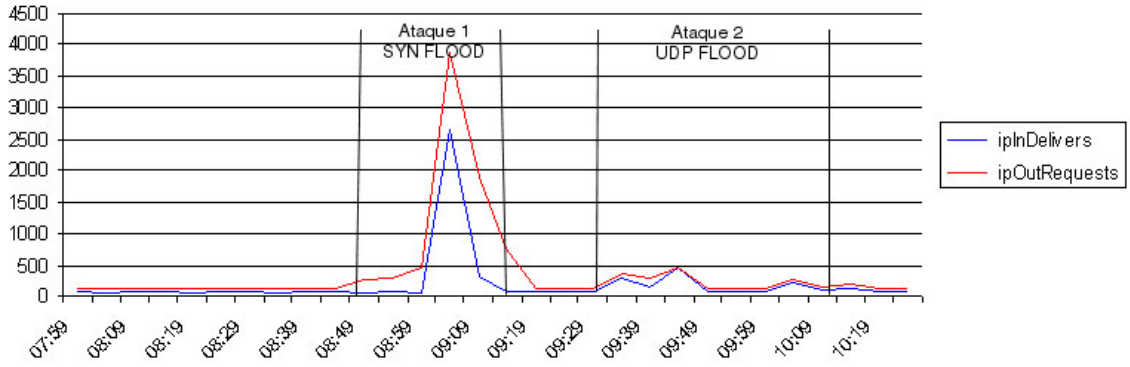

Figura 22 - Roteador IBM: *ipInDelivers* e *IpOutRequests* após os 2 primeiros ataques

No primeiro e no segundo ataque, o poder de processamento dos dois roteadores foi suficiente para garantir o funcionamento do *backbone*. O protótipo conseguiu detectar com êxito essa alteração no comportamento referente ao percentual de uso da CPU e alertas foram gerados. No 3º ataque destacado na Figura 23, foi possível detectar a alteração de comportamento, onde o uso da CPU chegou aos críticos 100%. Como o agente SNMP do roteador executa em baixa prioridade, as *query* não foram respondidas e seu resultado foi *time out*. Muitos *time outs* após a detecção de uma alteração de comportamento é sinal de que algo está errado e ações imediatas deverão ser tomadas para verificar se o problema foi causado por ataques ou por falha de algum dispositivo.

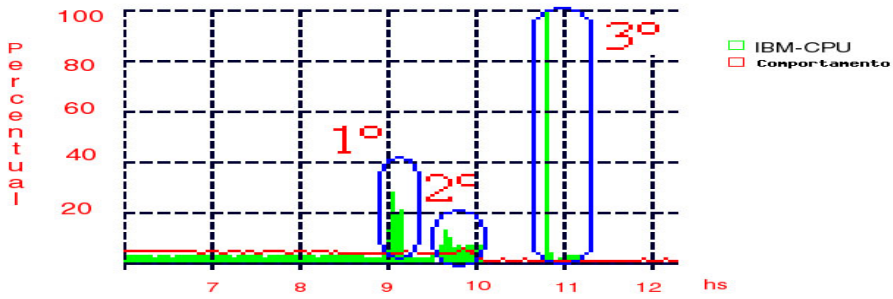

Figura 23 - Percentual de uso da CPU versus o comportamento nos 3 ataques realizados

5.1.1.2 Roteador Interno CISCO

O equipamento cisco 2500 estava no lado interno da rede e apresentou uma leve alteração no comportamento em relação ao uso da CPU e obteve basicamente os mesmos resultados do roteador IBM externo na relação aos objetos do grupo IP da MIB. Este fato pode ser notado na Figura 24 e Figura 25:

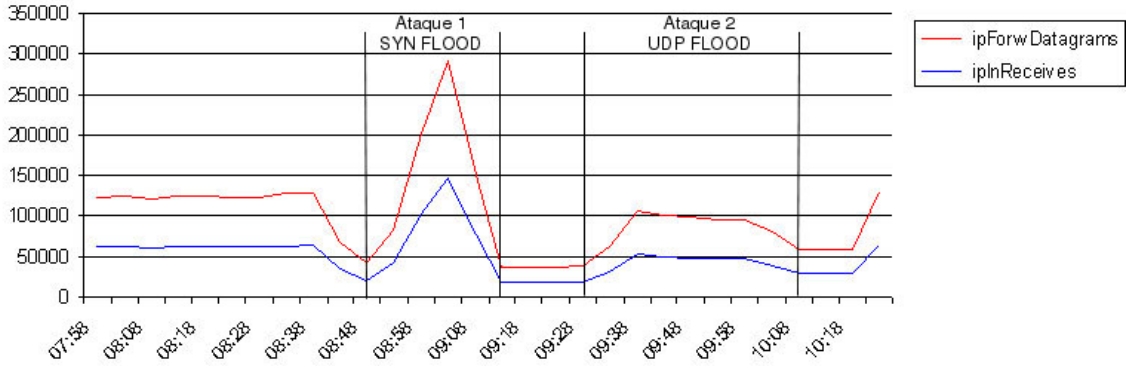

Figura 24 - CISCO: *ipForwDatagrams* e *ipInReceives* nos dois ataques.

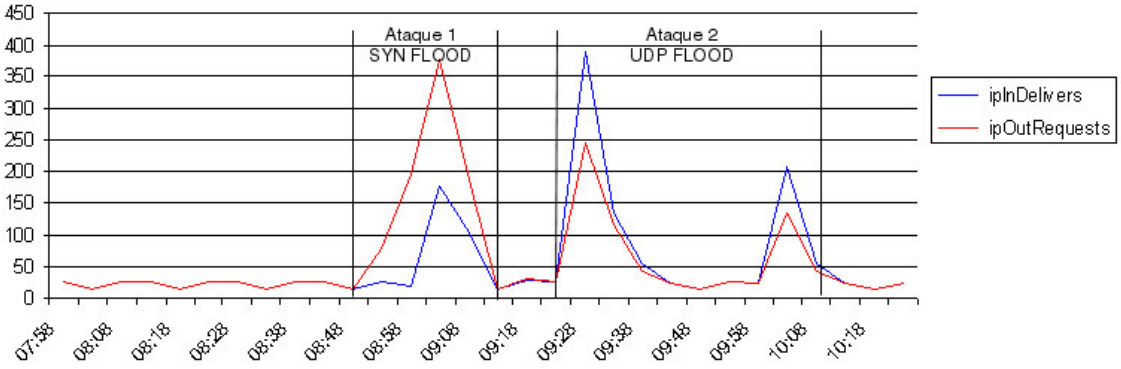

Figura 25 - CISCO: ipInDelivers e ipOutRequests após os 2 primeiros ataques

Com referência ao comportamento analisado da CPU, pode-se notar utilização acima do padrão normal nos dois primeiros ataques, o que era esperado, mas houve uma diminuição no terceiro. Essa diminuição de uso do padrão do comportamento é normalmente causada por problemas de rede e decorrente de ataques a equipamentos intermediários, onde esses acabam tendo seus recursos esgotados e não conseguem mais repassar todos os pacotes que lhe são entregues. Todos os desvios de comportamento foram detectados com sucesso pelo protótipo. Observe o gráfico na Figura 26.

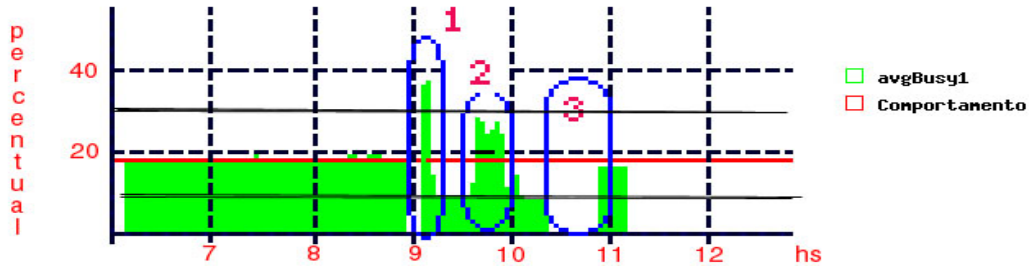

Figura 26 Utilização da CPU X Comportamento analisado do roteador CISCO

5.1.2 Aplicação do Protótipo em um Ambiente de Produção

Para obter resultados mais próximos da realidade de um ambiente de *backbone*, foi utilizado a infraestrutura da rede UFSC, PoP-SC e RCT. A Figura 27 representa a interligação lógica entre os roteadores. A rede UFSC tem o seu núcleo composto por roteadores IBM 8210 *Nways(r) Multiprotocol Switched Services Server* (MSS, 2002) que são roteadores ATM (*Asyncronus Transfer Mode*) com suporte a LANE (*Lan Emulation over ATM*). O PoP-SC possui 2 equipamentos principais, sendo um IBM 8210 e o CISCO 7513 que faz a interligação entre o PoP com a Internet.

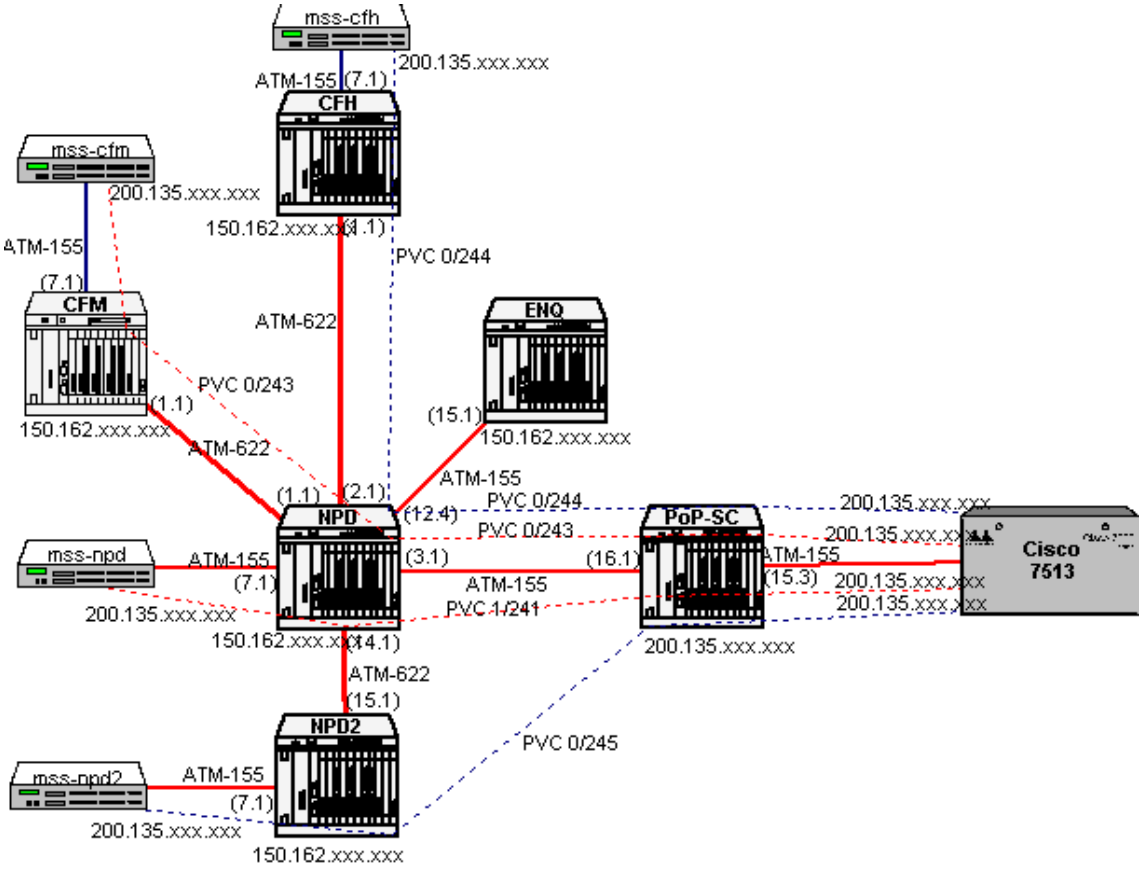

Figura 27 - Conexão redeUFSC, PoP-SC e Saída para Intenet

Neste ambiente, foi realizado o monitoramento das variáveis das MIBs proprietárias de cada equipamento correspondendo ao percentual de utilização da CPU. Nos roteadores IBM foi monitorada a variável correspondente a *nv6saComputerSystem* (OID: .1.3.6.1.4.1.2.6.4.5.1.0) que retorna o uso da CPU para os roteadores *Nways* (IBMCPU,

1998) e para o monitoramento dos roteadores CISCO *avgBusy1* (OID: .1.3.6.1.4.1.9.2.1.57) (CISCO CPU, 2002) que corresponde a média de uso do último minuto com intervalo entre coletas correspondente a 60 segundos.

No término do primeiro dia de monitoramento já é possível obter um comportamento característico do equipamento. Este comportamento é re-analisado no término de cada dia, assim quanto maior for o histórico de dados coletados de um equipamento, mais preciso será sua *baseline* e o resultado de um desvio de comportamento poderá ser detectado com mais facilidade e confiabilidade. As coletas onde for detectado um desvio de comportamento acima da margem de erro do desvio padrão de cada variável monitorada são indícios de um desvio de comportamento, provavelmente causado por ataques. Esta margem de erro além do desvio padrão deverá ser ajustada pelo administrador do sistema, tendo seu valor inicial igual a 0 %.

Logo nos primeiros quatro dias de monitoramento, a ferramenta detectou desvios muito elevados do comportamento, o que pode ser facilmente visto na Figura 28 onde alertas foram gerados através da base de dados, SNMP TRAPs e notificação por e-mail, informando aos administradores que algo estava errado com um dos equipamentos. Através do utilitário de visualização de eventos do roteador IBM MSS 8210 (*talk 2*[10]) foi possível constatar a origem do ataque e as medidas adequadas foram tomadas como filtragem da máquina que estava originando o ataque e notificação ao responsável pela faixa IP de onde originou o ataque.

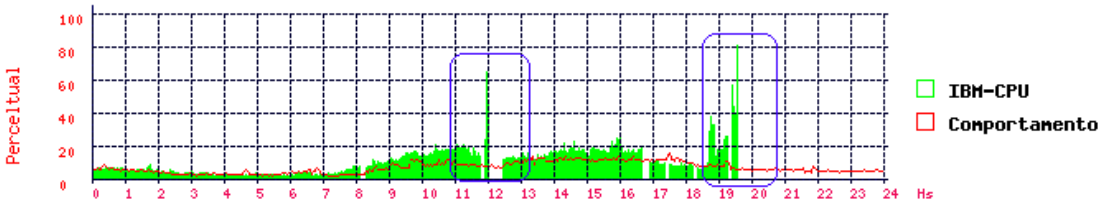

Figura 28 - MSS-NPD ataque ao *backbone* detectado pela análise do comportamento dos dias anteriores

[10] *Talk 2* – é o comando utilizado nos roteadores IBM para mudar para sistema de visualização de eventos de *logs*.

Na Figura 28, no intervalo próximo às 12 horas a utilização da CPU aumentou muito mais do que o desvio padrão analisado chegando próximo aos 70%. O desempenho do equipamento é prejudicado com uso superior a 40% e estouros nos *buffers* IP do equipamento tornam-se mais freqüentes, causando uma lentidão no *backbone* e sua identificação muitas vezes é tardia. O mesmo ocorreu no intervalo iniciando próximo as 18:40hs, diversos alertas foram enviados e a investigação do ataque só começou as 19:30hs que corresponde à última coleta deste gráfico.

Tabela 9 - Eventos gerados na base de dados referente ao MSS-NPD

Horário	Grau de Variação	Descrição do Alarme
2002-01-29 18:51:17	22	resultado fora do padrão de comportamento em 22 pontos
2002-01-29 18:48:16	25	resultado fora do padrão de comportamento em 25 pontos
2002-01-29 18:45:16	29	resultado fora do padrão de comportamento em 29 pontos
2002-01-29 18:42:16	20	resultado fora do padrão de comportamento em 20 pontos
2002-01-29 12:09:09	47	resultado fora do padrão de comportamento em 47 pontos
2002-01-29 12:03:17	34	resultado fora do padrão de comportamento em 34 pontos
2002-01-29 12:00:16	38	resultado fora do padrão de comportamento em 38 pontos
2002-01-29 11:54:11	16	resultado fora do padrão de comportamento em 16 pontos
2002-01-29 11:24:08	11	resultado fora do padrão de comportamento em 11 pontos

O uso deste protótipo auxiliou na detecção de vários problemas de *backbone* nas redes UFSC e PoP-SC que antes eram detectados manualmente, identificou-se diversos ataques de *DoS* e *flood*, contendo-os a tempo de causarem maiores danos. A utilização desta ferramenta para o auxílio da detecção de ataques é demonstrada com mais detalhes no Anexo 1, onde mostra um problema real ocorrido no ambiente da rede UFSC.

5.2 RESULTADOS ALCANÇADOS

Com o desenvolvimento e testes realizados neste trabalho, os seguintes resultados foram alcançados com o uso desta ferramenta:

- Monitoramento dos equipamentos de interconexão onde buscou-se estabelecer um modelo de comportamento para cada equipamento, integrando as peculiaridades de cada equipamento, através do protocolo de gerência de redes SNMP, Java, PHP e MySQL;

- Notificação de alterações do comportamento em forma de TRAPs SNMP, e-mail e alertas no console da ferramenta e na base de dados;

- Proporcionou mais uma maneira para auxiliar o trabalho dos analistas de seguranças e até mesmo dos gerentes de redes, onde através de uma interface WEB amigável podem visualizar os eventos com maior clareza e interação.

- É possível através de uma única estação gerenciar inúmeros equipamentos ao mesmo tempo, podendo estar dispostos em redes ou locais onde o uso de ferramentas de detecção de intrusão em redes de computadores não se torna viável atualmente devido basicamente a custos extras de *hardware* ou em *links* com largura de banda superior a 100 Mbps CAMPELO (2001), STALLINGS (1998)

- Para melhorar os resultados, é interessante integrar essa ferramenta com um NIDS, para prover uma rápida solução a um ataque que tenha como objetivo o *Denial of Service*.

- O uso do RMON (RFC2819, 2000) faz um papel semelhante com o envio de TRAPs SNMP, mas é necessário especificar qual será o valor limite para que o evento ocorra. O objetivo dessa ferramenta é analisar o comportamento e medir se algo está errado com o equipamento de acordo como histórico armazenando. Quanto mais tempo for analisado, o comportamento tende a ser o mais próximo

possível da realidade, levando em consideração que ele esteja inicialmente trabalhando em suas características normais.

6 - CONCLUSÕES E TRABALHOS FUTUROS

A área de detecção de intrusões, especialmente em redes de computadores, possui um campo bastante amplo e onde existem inúmeras alternativas, sendo elas comerciais ou de código aberto para facilitar a identificação de ataques e/ou ameaças a redes de computadores.

Este trabalho propôs a utilização de uma ferramenta para aumentar a visão do administrador de *backbones* o qual, ainda não é possível detectar/barrar todos os ataques que passam através dele que, principalmente, degradam o desempenho da rede .

A ferramenta proposta pode ser empregada como uma nova alternativa para o gerenciamento de redes e no auxílio na detecção de ataques em grandes *backbones* ou redes WAN. Todos os equipamentos gerenciáveis através de SNMP possuem em sua MIB proprietária ou nas MIBs padrões, variáveis que podem ser analisadas para esse fim. Neste estudo utilizaram-se roteadores IBM das séries (2210, 8210, 8371) e CISCO das séries (2500, 7000 e 7500) que estão presentes no ambiente da rede UFSC, PoP-SC e RCT e as coletas dos valores correspondem ao uso de CPU, onde notou-se uma grande modificação do seu comportamento quando ataques de DoS eram lançados contra eles. Tal alteração também foi observada nos equipamentos que serviam como rota para outras redes quando os ataques ocorreram. Outras variáveis que apresentaram e possibilitaram bons resultados na criação da *baseline* e estão padronizadas na MIB-II, são o número de pacotes *unicast* que entram e saem por uma interface (*ifInUcastPkts* e *ifOutUcastPkts*).

Através do estudo realizado notou-se que área de gerenciamento de segurança é uma área bastante promissora juntamente com a detecção de intrusão, pois mostrou-se efetiva na coleta dos dados necessários, e no envio de alertas. Por outro lado, quando o protocolo IPv6 chegar a ser utilizado em larga escala e a carga útil dos pacotes for cifrada, grande parte dos problemas de rede e segurança estarão resolvidos, principalmente o caso do *sniffing*. Essa técnica será menos utilizada, pois quando a criptografia estiver no nível de protocolo, esse problema de escuta será menos

vulnerável, e quando algo for capturado somente os criptoanalistas conseguirão decifrar qual é o conteúdo dos pacotes, através de força bruta ou criptoanálise, o que não é uma tarefa trivial.

Por outro lado, a maioria dos IDSs de rede acabarão não conseguindo detectar mais muitas "assinaturas" de ataques quando a criptografia passar a ser empregada pelo novo protocolo. Com isso abrirá um novo espaço a ferramentas e métodos de análise de comportamento baseado em informações estatísticas dos equipamentos gerenciáveis que compõe a infraestrutura das redes de computadores, conforme proposto neste estudo.

Este trabalho baseou-se na utilização do protocolo SNMPv1 devido a fatores de interoperabilidade com os equipamentos que foram testados, pois a maioria dos equipamentos acaba implementando somente esta versão do protocolo. Os problemas de segurança apresentados por essa versão do protocolo ficarão minimizados com a adoção do SNMPv3.

Como resultados, além de sua utilização para o monitoramento de segurança do *backbone* da redeUFSC e do PoP-SC, parte deste trabalho foi aceita para publicação em forma de artigo no Workshop em Segurança de Sistemas Computacionais - WSeg2002, (RHODEN 2002) que será realizado no será realizado durante o 20° Simpósio Brasileiro de Redes de Computadores – SBRC, em Búzios – RJ.

Este modelo de análise do comportamento, como os outros métodos de detecção de intrusão não resolvem o problema, mas contribuem para manter o *backbone* sempre monitorado e qualquer desvio de comportamento que afete o desempenho da rede possa ser detectado a tempo, sua causa encontrada e interrompida.

Com base nos resultados obtidos neste trabalho pode-se recomendar uma série de trabalhos futuros:

- Possibilitar suporte aos protocolos SNMPv2 e SNMPv3 visando aumentar a segurança na coleta das informações dos roteadores. As agregações dessas funcionalidades ao protótipo não foram implementadas devido principalmente pelo fato de que a maioria dos roteadores não os implementar. Essa funcionalidade pode ser facilmente agregada ao protótipo;

- A MIB SNMP possui uma grande gama de variáveis que podem ser utilizadas para a análise de seu comportamento. Pode-se assim analisá-las e incluí-las facilmente no protótipo em tempo de execução;

- O protocolo RMON pode ser explorado para monitorar passivamente algumas variáveis e enviar alertas a estação de gerência. Ele não foi empregado no trabalho por motivos anteriormente apresentados, mas pode ser empregado com o protótipo configurando dinamicamente os índices dos limites através da *baseline* adquirida pelo sistema. O uso de recursos do equipamento e de rede através do SNMP *get* terá significativas mudanças, pois deverá haver um constante monitoramento para manter a *baseline* atualizada base de dados e nos limites dos *thresholds* do RMON com SNMP *set*. O ponto chave do RMON é que ele notificará quase em tempo real o desvio de comportamento, possibilitando assim a verificação do problema no menor tempo possível;

- Realizar a integração da ferramenta com um sistema de detecção de intrusões de rede, possibilitando assim descobrir com maior agilidade a origem do problema e também automatizar a aplicação de filtros nos roteadores em tempo real para conter o ataque;

- Montar um ambiente de testes como o protocolo IPv6 e estudar quais são os impactos causados nos roteadores quando um ataque de DoS for executado e compará-los com os resultados obtidos no mesmo ambiente com o IPv4, e

- Estudar outras variáveis da MIB que possam ser utilizadas para o reconhecimento de certos tipos de ataques, a fim de construir um sistema especialista capaz de reconhecer mais do que desvios de comportamento, o qual é proposto aqui neste trabalho pelo modelo desta ferramenta, conseguiga identificar certos tipos de ataques, gerando assinaturas baseadas em informações estatísticas, conforme proposto por CABRERA et. al.(2001).

REFERÊNCIAS BIBLIOGRÁFICAS

ADVENTNET. **AdventNET SNMP API**. Disponível em: <http://www.adventnet.com/products/snmp/>. Acesso em: 15 de maio de 2001.

ALTUNERGIL, O. **Understanding Rootkits**. Disponível em:< http://linux.oreillynet.com/pub/a/linux/2001/12/14/rootkit.html>. Acesso em: 07 de fevereiro de 2002.

ANDERSON. P. James. **Computer Security Threat Monitoring and Surveillance.** Technical report, Fort Washington, Pennsylvania, April 1980.

ANONYMOUS. **Maximum Linux Security**. Sams Publishing - September 1999.

AXMARK, David; et al. **MySQL Reference Manual for version 4.0.2-alpha**. Disponível Em: <http://www.mysql.com/documentation/mysql/full/index.html >. Acesso em 22 de janeiro de 2002.

BELLOVIN, S. **Packets Found on an Internet**. Computer Communications, julho de 1993.

BIERMAN, A; IDDON R. **Remote Network Monitoring MIB Protocol Identifiers**. Network Working Group, Request for Comments: 2074. Jan 1997.

BRADEN, R. **TCP Extensions for Transactions Functional Specification**. RFC 1644, July 1994.

BREEDER. **Breeder Security Inc**. Disponível em: <http://breedersec.virtualave.net/code.html >. Acesso em 11 de dezembro de 2001.

CABRERA, J. et al. **Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables – A Feasibility Study**. *In Proceeding of The Seventh IFIP/IEEE International Symposium on Integrated Network Management*

(IM 2001), Seattle, WA, May 2001.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; and J. DAVIN. **Simple Network Management Protocol**. RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

CERT, **CERT/CC Statistics 1988-2001**. Disponível em <http://www.cert.org/stats/cert_stats.html>. Acesso em: 25 de março de 2002a.

CERT. **Back Orifice**. CERT Coordination Center - Vulnerability Note VN-98.07. Outubro de 1998. Disponível em: <http://www.cert.org/vul_notes/VN-98.07.backorifice.html>. Acessado em: 07 de fevereiro de 2002.

CERT. **IP Denial-of-Service Attacks**. CERT Advisory CA-1997-28. Disponível em: <http://www.cert.org/advisories/CA-1997-28.html>. Acesso em 21 de março de 2002.:

CERUTTI, A. Fernando. **Implementação e Utilização de um Ambiente de Gerência De Tráfego e Recursos em Redes ATM Utilizando a Tecnologia Web**. 1999. Dissertação (Mestrado em Ciência da Computação). Pós-Graduação em Ciência da Computação. Universidade Federal de Santa Catarina – UFSC. Florianópolis-SC

CISCO. **Baseline Process: Best Practices**. White Paper. Disponível em: <http://www.cisco.com/warp/public/126/HAS_baseline.html>.Acesso em 16 de abril de 2002.

CISCO. **Network Monitoring and Event Correlation Guidelines**. Cisco Systems, Inc.1999.

CISCO. **Remote Monitoring (RMON)**. Disponível Em: <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rmon.htm>. Acesso em: 22 de outubro de 2001a.

CISCO-CPU. **How to Collect CPU Utilization on Cisco IOS Devices Using SNMP**.

Disponível em:< www.cisco.com/warp/public/477/SNMP/collect_cpu_util_snmp.html>. Acesso em 29 de janeiro de 2002.

DARIVA, C. Roberto. **Distributed Denial of Service – um estudo de caso para plataformas Linux e Windows**. 2002. Dissertação (Mestrado em Ciência da Computação). Pós-Graduação em Ciência da Computação. Universidade Federal de Santa Catarina – UFSC. Florianópolis-SC.

DEITEL, M. Harvey; DEITEL, J. Paul. **Java, como programar**. Trad. Edson Furnankiewicz. – 3ª ed. – Porto Alegre: Bookman, 2001.

ETHEREAL, **Ethereal - Interactively browse network traffic**. Disponível em: <http://www.ethereal.com>. Acesso em: 30 de junho 2001.

ETHEREAL. **Ethereal - Interactively browse network traffic**. Disponível em: <http://www.ethereal.com>. Acesso em: 20 de junho de 2001.

FERGUSON, P.; SENIE, D. **Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing.** RFC 2267, Jan. 1998.

IBM. **Nways(r) Multiprotocol Switched Services Server (MSS)**. Produtos de rede. Disponível em: < www.networking.ibm.com/820/820prod.html>. Acesso em: 29 de janeiro de 2002.

IBMCPUMIB. **CPU Utilization**. Disponível em: < www.networking.ibm.com/support/code.nsf/mibscode/?OpenView>. Acesso em 29 de janeiro de 2002.

MATTHEWS, Mark. **MM.MySQL JDBC Driver**. Disponível em <http://mmmysql.sourceforge.net/> Acesso em 19 de março de 2002.

MAURO, Douglas; SCHMIDT, Kevin. **Essential SNMP**. 1º ed. O'Reilly Books.July 2001.

MCCLOGHRIE K.; ROSE, M.. **Management Information Base for Network Management of TCP/IP-based Internets**, RFC 1156, Hughes LAN Systems, Performance Systems International, May 1990.

MOORE, David. **The Spread of the Code-Red Worm**. Disponível em: <http://www.caida.org>. Acesso Em: 30 de junho de 2001.

NETCRAFT, **Netcraft Web Server Survey**. Disponível em: <http://www.netcraft.com/survey/>. Acesso em 15 de abril de 2002.

NETO, F. Watzko. **Aplicando a Técnica de Séries Temporais em Gerenciamento Pró-Ativo de Redes de Computadores**. Anais do Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro-RJ. Maio de 1998.

NORTHCULTT, Stephen; NOVAK, Judy. **Network Intrusion Detection – An Analyst´s Handbook**. 2. ed. Indianapolis, 2000 p. 246.

PERKINS, David; MACGINNIS, Evan. **Understanding SNMP MIBs**. 1º ed. New Jersey: Prentice Hall PTR, 1997. p. 38

RFC2021, S. WALDBUSSER. **Remote Network Monitoring Management Information Base Version 2**. Network Working Group, Request for Comments: 2021. January 1997

RFC2819. **Remote Network Monitoring Management Information Base. Network Working Group, Request for Comments: 2819**. S. Waldbusser of Lucent Technologies, May 2000.

RHODEN, E. Guilherme; MELO, L. Edison; WESTPHALL, B. Carlos. **Detecção de Intrusões em Backbones de Redes de Computadores Através da Análise de Comportamento com SNMP.** II Workshop em Segurança de Sistemas Computacionais – WSEG.

ROSE M.; MCCLOGHRIE K.. **Structure and Identification of Management Information for TCP/IP-based internets**. RFC 1155, Performance Systems

International, Hughes LAN Systems, May 1990.

SNORT, **The Open Source Network Intrusion Detection System**. Disponível Em: <http://www.snort.org/>. Acesso Em:18 de outubro de 2001.

SPARTA, Souza; MATOS, Rafael. **Um estudo de vulnerabilidade de serviços de Internet**. Trabalho de Diplomação, Bacharelado em Ciências da computação, Instituto de Informática, UFRGS, janeiro de 1997

STALLINGS, William. **Cryptography and network security: principles and practice**. 2º ed, 1998.

STALLINGS, William. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.** 3º ed. 1997.

STOLL, C. **Stalking The Wily Hacker**. Communication of the ACM, vol. 31. no. 5. pag 484-500, May of 1988.

TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Campus, tradução da 3. ed. original, p. 718-735, 1997.

TAROUCO, M.R. Liane**. Evolução do gerenciamento de Redes. In Sociedade Brasileira para Interconexão de Sistemas Abertos, Ed., Gerenciamento de Redes - Uma abordagem de sistemas abertos,** Makron Books do Brasil, São Paulo, p. 1-12, 1993.

TCPDUMP. **Tcpdump - Dump Traffic on a Network**. Disponível em: <http://www.tcpdump.org>. Acesso em: 20 de junho de 2001.

TUOMAS Aura, PEKKA Nikander, and JUSSIPEKKA Leiwo. DOS-resistant authentication with client puzzles. In: 8TH INTERNATIONAL WORKSHOP ON SECURITY PROTOCOLS, to appear in the lecture notes in computer science series, Cambridge, UK, April 2000. Springer.

VERONEZ, A. Cleverson. **Gerência De Desempenho Do Tráfego Em Redes Utilizando Baseline Bayesiana.** 2000. Dissertação (Mestrado em Ciência da Computação). Pós-Graduação em Ciência da Computação. Universidade Federal de Santa Catarina – UFSC. Florianópolis-SC.

WENSTROM, J. Michael. **Managing Cisco Network Security**. Cisco Systems, Inc. 2001.

ANEXO1 – Uso da Ferramenta no Auxílio da Detecção de um Ataque Ocorrido em Ambiente Real da redeUFSC.

Este anexo demonstra a detecção um incidente de segurança ocorrido na subrede do departamento de engenharia civil no dia 26 março de 2002, onde a ferramenta aqui demonstrada alertou sobre o desvio de comportamento, proporcionando que uma ação fosse tomada o mais rápido possível, quais eventos foram gerados pelo SRIDS e como foi descoberto o endereço do atacante.

Primeiramente a ferramenta detectou o desvio de comportamento as 7:23:54hs e disparou os seguintes alertas:

- Alerta no Console da Ferramenta;
- Alerta na Base de Eventos;
- E-mail para o administrador da rede;

Os itens eventos acima são vistos com maior clareza nas figuras 1, 2 e 3.

```
2002-04-26 07:23:54 -:ALERTA:- ROTEADOR: 2 -
150.162.252.252 resultado fora do padrão de
comportamento em 10.0 pontos. Valor Atual 13
```

Figura 1- Alerta no Console

•**Date:** Fri, 26 Apr 2002 07:24:37 -0300
•**From:** Guilherme Eliseu Rhoden <rhoden@inf.ufsc.br>
•**Subject:** SRIDS-> ALERTA 200.135.0.251
•----------
•Roteador (150.162.252.252)
•Comportamento fora do padrão em 10 pontos
•na variável: 2 (CPU).
•Msg gerada as 2002-04-26 07:23:54 •----------

•MSG Automática Gerada pelo SRIDS
•----------

Figura 2 - Alerta enviado ao Administrador

O administrador pode acessar os perfis de comportamento graficamente, onde sem possuir um grande conhecimento do perfil de uso dos equipamentos pode constatar visualmente

alterações no comportamento. No caso deste ataque, o perfil de comportamento do equipamento alterou-se bruscamente conforme mostrado na Figura 3.

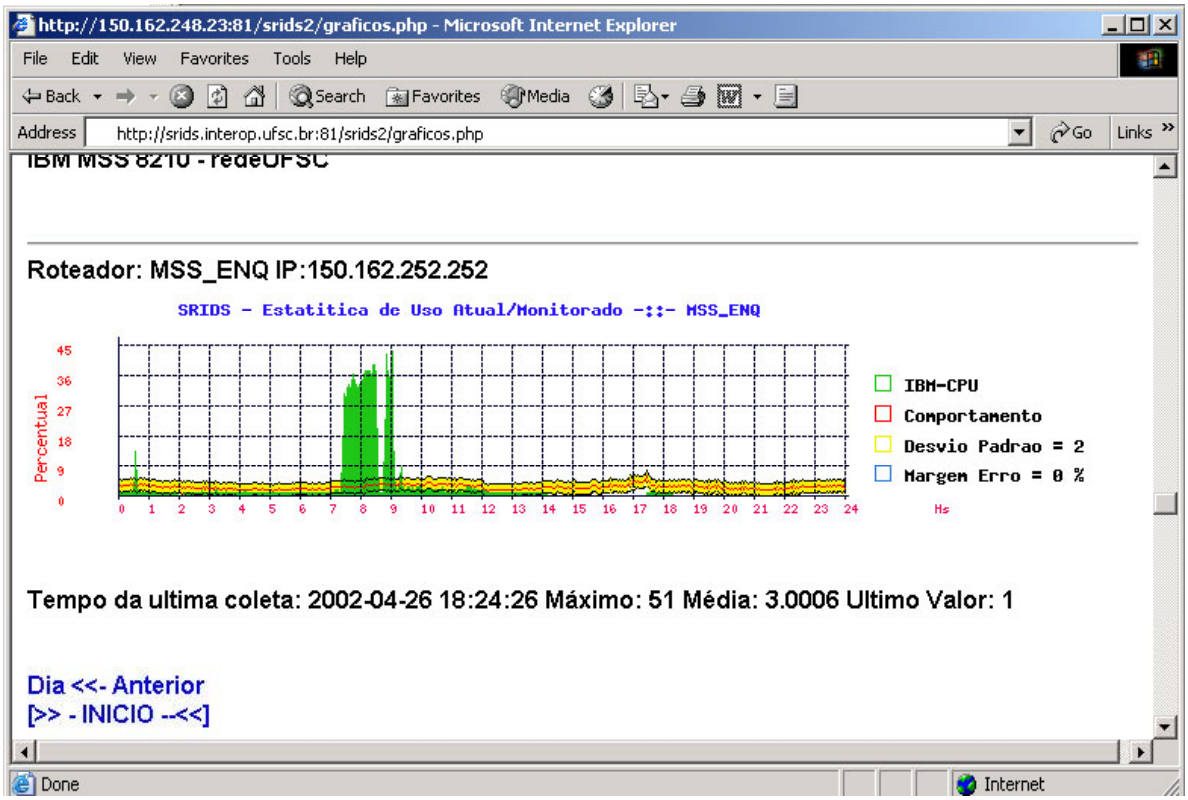

Figura 3 - Visualização do Perfil de Comportamento do Roteador Graficamente

Um dos métodos bastante utilizados para detectar certos tipos de ataques são a utilização das ferramentas de *debug* dos roteadores, tal como *ip-accounting* nos roteadores CISCO e *talk 2* nos roteadores IBM. Como neste caso estamos tratando de um roteador IBM, o *talk 2* do respectivo roteador é mostrado na Figura 4.

```
•214:47:05   IP.069: routing cache garbage collecting...
•214:47:05   IP.069: routing cache garbage collecting...
•214:47:05   IP.069: routing cache garbage collecting...
•214:47:10   IP.069: routing cache garbage collecting...
•214:47:10   IP.069: routing cache garbage collecting...
•214:47:10   IP.069: routing cache garbage collecting...
•214:47:16   IP.069: routing cache garbage collecting...
•214:47:16   IP.069: routing cache garbage collecting...
•214:47:16   IP.069: routing cache garbage collecting...
•214:47:22   IP.069: routing cache garbage collecting...
```

Figura 4 - Visualização de Eventos - Talk 2

Com a visualização de eventos mostrada na Figura 4 não foi possível obter maiores informações do ataque, então utilizou-se um *sniffer* (Ethereal) para poder observar com mais detalhes o tráfego que estava sendo gerado pela máquina atacante, conforme apresentado na Figura 5.

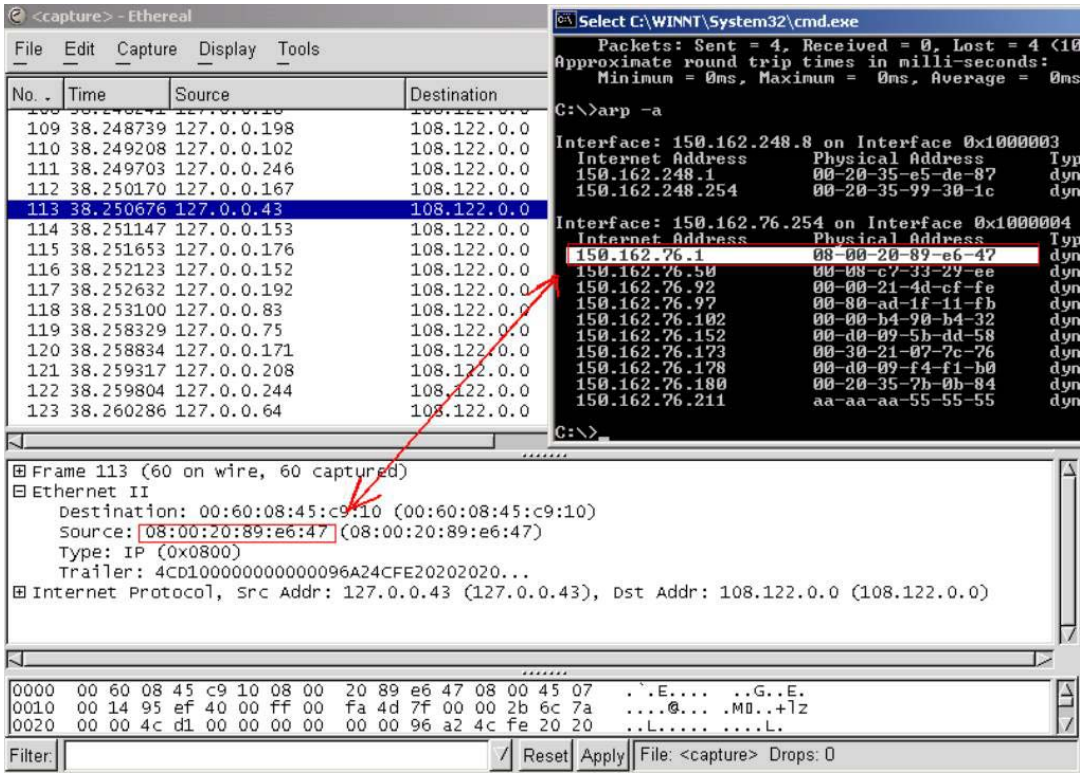

Figura 5 - Sniffer Posicionado como *Gateway* da Rede Suspeita

Observando a Figura 5 podemos observar diversos pacotes Ips de origem forjados com endereços correspondendo a (127.0.0.x/24), que são tipicamente endereços de *loolpback*, com destino ao endereço (108.122.0.0) que corresponde a um endereço de rede. O método utilizado para conheçer o real endereço do atacante foi a utilização do endereço MAC contido no frame Ethernet. Neste caso fazendo a associação de todos os endereços Ips da rede 150.162.76.0/24 concluímos que a máquina onde o ataque estava sendo originado corresponde ao *host* 150.162.76.1 com o respectivo endereço MAC 08:00:20:89:e6:47. O responsável pelo controle da máquina foi contatado e máquina foi retirada da rede, onde foi constatado que a máquina 150.162.76.1 havia sido invadida e estava atacando, o que conseqüentemente estava comprometendo o desempenho do backbone da redeUFSC.

O uso desta ferramenta auxilia no rápido diagnóstico do estado de um grande *backbone*, onde podemos detectar problemas que acabam causando degradação do desempenho do serviço

prestado, onde o principal problema é causado por ataques de negação de serviço que acabam consumindo boa parte dos recursos dos equipamentos de interconexão no backbone afetando assim as demais conexões.