UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ricardo de Oliveira Portes

ABORDAGEM DE AGENTES ESTÁTICOS PARA DETECÇÃO DE INTRUSOS BASEADO EM HOST

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

Prof. Luis Fernando Friedrich, Dr.

Florianópolis, maio/2003

ABORDAGEM DE AGENTES ESTÁTICOS PARA DETECÇÃO DE INTRUSOS BASEADO EM HOST

Ricardo de Oliveira Portes

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 forma final pelo programa de Pós–Graduação em Ciência da Computação.

_____
Prof. Fernando Alvaro Oustine Gauthier, Dr.

Banca Examinadora

_____
Prof. Luiz Fernando Friedrich, Dr.
(orientador)

_____
Prof. Carlos Montez, Dr.

_____
Prof. Luiz Carlos Zancanella, Dr.

Ao meu pai, Carlos de Oliveira Portes, *in me-*
morian, à minha mãe, Elisabeth Margarida
de Oliveira Portes, aos meus avós.

AGRADECIMENTOS

Agradecimentos especiais ao Prof. Luis Fernando Friedrich, pela sua orientação e valoroso apoio.

Aos professores Marcos Francisco Ferreira de Macedo e Gerson Battisti, pela confiança depositada na forma de cartas de recomendação.

Aos amigos da pensão da Dona Dárcia (Rogério Xavier de Azambuja, Cristhian Flamarion Gomes de Carvalho, Carlos Pantaleão, Humberto, Marionei Zerbielli, Fernando Luiz e Roger Imich), pela amizade e incentivo durante a realização deste trabalho.

À amiga Elisa Feitosa, pelo seu apoio e conselhos que foram muito válidos no decorrer do desenvolvimento do curso e deste trabalho.

À amiga Luciana Bitencourt, pelo apoio e insentivo muito importantes para a conclusão deste trabalho.

Ao amigo Paulo João Martins, que muito ajudou, lendo e revisando este trabalho, fornecendo contribuições valiosas para sua conclusão.

Ao amigo Edison Tadeu Lopes Melo, pelo incentivo e apoio no desenvolvimento deste trabalho.

Às funcionárias da secretaria do CPGCC, Verinha e Valdete, que sempre estiveram prontas à nos atender. Serão sempre lembradas.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACL – Access Control List

CVE – Common Vulnerabilities Consortium

DOS – Denial of Service

DDOS – Distributed Denial of Service

HIDS – Host–based Intrusion Detection System

HTTP – HyperText Transfer Protocol

IDS – Intrusion Detection System

IMAP – Interim Mail Access Protocol

IP – Internet Protocol

LIDS – Linux Intrusion Detection System

MIB – Management Information Base

NIDS – Network–based Intrusion Detection System

POP3 – Post Office Protocol

SNMP – Simple Network Management Protocol

SSH – Secure Shell

TCP – Transfer Control Protocol

UDP – User Datagram Protocol

RESUMO

O trabalho, em questão, tem como finalidade propor uma abordagem para detecção de intrusos através do uso de agentes estáticos. O princípio adotado é a utilização de uma lista de controle de acesso, na qual fica explícito o que o usuário não pode realizar no sistema, quais diretórios e arquivos não deve acessar. Dessa forma, o agente pode responder rapidamente caso uma dessas regras seja quebrada. O objetivo é minimizar o uso de uma lista com assinaturas baseadas em comportamentos de usuários, em prol de detectar ações, definidas em uma lista de controle de acesso, que denotem um mau uso do sistema, por parte do usuário. Outro ponto, é possibilitar a interação, por parte do administrador de sistemas, através de uma console central de gerenciamento que recebe as mensagens do referido agente, uma vez que este detecte alguma ação que possa ser considerada suspeita. Assim, o administrador pode participar mais efetivamente nas tomadas de decisão sobre uma possível resposta do HIDS (Host–based Intrusion Detection System) proposto.

ABSTRACT

The objective of the current study is the proposal of an approach to detect intruders by the use of static agents. The principle which is applied is the use of a control list of access, in which is explicit the things that the user cannot do in the system, which are the files and directories that must not be accessed. This is the way that the agent can quickly respond if one of these rules is broken. The objective is to minimize the use of a list of signatures based on users' behavior, in order to detect actions which are defined in a list of access control,, that shows a misuse of the system by the user. Besides that, this study wants to allow an interaction of the system administrator, through a central console management that receives messages from the agent, once the system detects any action that might be considered suspicious. Thus, the system administrator may more effectively participate in the decisions about a possible response from the proposed HIDS (Host–based Intrusion Detection System).

1 – INTRODUÇÃO

No decorrer dos últimos anos, impulsionado pela expansão da Internet, o comércio eletrônico tem apresentando um crescimento surpreendente, dominando o panorama econômico e criando uma nova forma de comércio e negócios para bens e serviços. A interconectividade entre as grandes e pequenas empresas virtuais tem se tornado o principal motivo para o seu sucesso. (PROCTOR, 2001)

Dessa forma, as redes de computadores, principalmente no uso para o comércio eletrônico, fizeram que muitos computadores conectados à Internet se tornassem alvos em potencial de inúmeras tentativas de invasão aos seus sistemas, objetivando nada mais que o acesso e cópia de dados que possam ser considerados importantes, os quais podem comprometer a produtividade da empresa, assim como alguma vantagem competitiva no mercado, além de outros possíveis crimes.(GARFINKEL & SPAFFORD, 1996)

A necessidade de manter a integridade das informações dessas organizações, bem como a essa interconectividade mencionada, tem ocasionado um crescimento no grau de confiança atribuída aos controles de segurança dos computadores. Tal confiança pode aumentar, uma vez que exista uma forma de verificar periodicamente estes controles. Por essa razão, o uso de um sistema de detecção de intrusão vem se tornando fundamental para efetuar essas verificações.

Os Sistemas de Detecção de Intrusão (IDS – Intrusion Detection System) apresentam–se, cada vez mais, como ferramentas imprescindíveis na realização desse tipo de monitoramento, pois uma invasão pode comprometer a segurança de um sistema. Esse tipo de sistema vem se tornando mais necessário, conforme cresce o número de computadores que estão conectados na rede.

1.1 – Justificativa

A crescente necessidade em proteger a integração entre os sistemas, bem como sua interoperabilidade, contra aqueles que tem como finalidade causar danos, tanto para alimentar uma satisfação pessoal como uma tentativa de obter lucros provenientes da comercialização de informações roubadas desses sistemas, apresenta–se como motivo, cada vez maior, para implantação de tecnologias para garantir o máximo possível de segurança as informações desses sistemas.

Existe um grande número de estudos e pesquisas que vêm sendo realizado, na busca de soluções que possam amenizar essa crescente onda de invasões, que têm vitimado vários sistemas conectados à Internet. Através deste trabalho, buscamos uma contribuição na busca de soluções para implantação de segurança em sistemas.

1.2 – Objetivos

Propor a construção de um protótipo para uma ferramenta que utiliza agentes estáticos distribuídos, os quais efetuam monitoramento constante na ocorrência de eventos, com análise baseada em uma ACL (Access Control List), para mau uso do sistema, e numa base de assinaturas.

A meta é proporcionar que o referido protótipo possa realizar uma análise e possível reação imediata por parte dos agentes, assim como a informar à console central, quando o mesmo detectar um alarme baseado na ACL, bem como reportar informações à console central, a qual poderá gerar alarmes, caso essas informações constem em sua base de assinaturas. Além disso, possibilitar à console central o armazenamento das informações recebidas dos agentes, gerando assim uma base de dados para futuras consultas.

1.3 – Organização do Trabalho

O presente trabalho organiza–se em 07 capítulos, os quais estão distribuídos da seguinte forma:

- Capítulo 1 – Introdução ao tema, fornecendo uma visão geral do trabalho, justificativa de escolha e objetivos.

- Capítulo 2 – Apresenta uma bordagem conceitual sobre Segurança de Sistemas.

- Capítulo 3 – Definição e conceito sobre Detecção de Intrusos e apresentação de uma breve descrição dos métodos mais utilizados para efetuar a Detecção de Intrusos.

- Capítulo 4 – Descreve alguns projetos desenvolvidos na área, bem como algumas soluções comerciais.

- Capítulo 5 – Descrição da Abordagem Proposta no trabalho, definindo os módulos componentes do sistema proposto e o resultado obtido no experimento.

- Capítulo 6 – Apresenta alguns aspectos sobre a conclusão deste trabalho e também algumas indicações sobre possibilidades de trabalhos futuros relacionados ao tema.

- Capítulo 7 – Referências Bibliográficas.

2 – SEGURANÇA

2.1 – Conceito

Um sistema é considerado seguro quando é possível confiar nele, assim como nos programas que compõem esse sistema e que apresentam um comportamento de acordo com as expectativas do usuário. (GARFINKEL & SPAFFORD, 1996)

Pode–se definir, também, um sistema seguro como aquele que possui a habilidade de guardar um segredo. Mas, devemos lembrar, que tão ou mais importante que manter uma informação em segredo, é mantê–la em segredo enquanto esta é transmitida através de um meio físico entre dois *hosts*[1], como uma rede de computadores. (BRUCE & DEMPSEY, 1997)

Dessa forma, podemos definir segurança como a implantação de um conjunto de regras, cuja finalidade é restringir o acesso a determinados recursos de um computador ou de uma rede. Essas regras podem ser referenciadas como políticas de segurança que definem quais usuários podem acessar o sistema, à quais informações ou recursos poderão ter acesso e o que poderão fazer uma vez obtido esse acesso.

Essa é uma característica que podemos encontrar nos sistemas operacionais que têm como enfoque principal os ambientes de redes de computadores, pois apresentam um conceito multi–usuário, o que permite o emprego das políticas mencionadas, aos usuários cadastrados no sistema.

O acesso físico aos servidores é um outro fator que devemos levar em consideração. Mesmo que exista uma boa política de segurança, e que essa seja

1 Host – Denominação dada à um computador conectado em uma rede.

empregada, deve haver uma restrição de acesso, apenas aos administradores de sistemas, ao local onde se encontram os servidores, afinal, qualquer política de segurança é inútil se houver acesso de pessoas estranhas aos servidores.

2.1.1 – Sistemas

Um sistema, segundo (SCHNEIER, 2001), apresenta–se constituído por um conjunto de componentes ou programas, que interagem entre si de tal forma que possam oferecer uma solução para um problema complexo.

Quanto maior o número de componentes interagindo em um sistema, maior a complexidade deste. Pode–se citar a Internet, como um exemplo, onde existem milhões de computadores que estão conectados em uma rede física extremamente complexa. Cada computador possui programas que interagem uns com os outros, bem como com programas em outros computadores. (SCHNEIER, 2001)

Os sistemas possuem algumas propriedades que deve ser consideradas:

– São complexos, diferentemente de um objeto qualquer, um sistema apresenta–se de forma mais complicada, pois possui componentes, infra–estrutura, bem como a integração entre vários objetos, entre outros.

– Interagem uns com os outros, de tal forma que pode vir a originar um sistema maior. Isso pode ocorrer propositalmente, em uma implementação orientada à objetos para divisão de um sistema grande em sistemas menores, ou naturalmente, como a necessidade ocasionada por um outro.

– Possuem propriedades emergentes, pois efetuam ações que não podem ser antecipadas pelos usuários e projetistas, as quais mudam o comportamento

das pessoas. Um exemplo são os editores de texto, que mudaram o modo com as pessoas escrevem, ou o telefone, que mudou e tornou mais rápida a comunicação.

– Possuem bugs, responsáveis por comportamentos estranhos no sistema.

2.1.2 – Segurança de Sistemas

Como já percebido, os sistemas apresentam uma grande complexidade, de tal forma que essa complexidade torna difícil a tarefa de proteger os mesmos e os dados nele contidos.(SCHNEIER, 2001)

Para que o trabalho de proteção ao sistema possa ser executado de maneira satisfatória, é imprescindível ao administrador de sistemas a posse das ferramentas e da documentação necessárias que o ajude a desempenhar o seu trabalho com precisão. Mas, vale lembrar que tais ferramentas de nada servem, se não houver uma eficiente política de segurança implantada.

2.2 – Políticas de Segurança

Primeiramente, antes de implantar algum tipo de segurança em um sistema, é necessário que exista um conhecimento sobre o que deve ser protegido e, principalmente, de quem se proteger. Para tanto, é preciso criar e adotar um conjunto de regras que serão utilizadas na implantação das políticas de segurança. Tais regras definem um conjunto de ações que podemos considerar como permitidas e outro, cujas ações podem ser tratadas como não permitidas, que serão tomados como base nas decisões referentes à questão da segurança do sistema. São as políticas de segurança que responderão mediante qualquer tipo de violação de segurança que possa vir a ocorrer.

Segundo (MOURANI, 2000), existem algumas indagações que devem ser levadas em consideração, para só então, iniciar a implantação das políticas de segurança. Eis alguns exemplos:

- Como as informações confidenciais são arquivadas?
- O sistema contém informações confidenciais?
- Contra quem exatamente você pretende se proteger?
- Usuários realmente precisam acessar o seu sistema remotamente?
- Senhas e criptografia fornecem proteção suficiente?
- Você precisa de acesso à Internet?
- Qual tipo de acesso você deseja permitir ao seu sistema a partir da Internet?
- Que ação tomar quando descoberta uma brecha em sua segurança?

As políticas de segurança devem apresentar um certo equilíbrio entre as permissões necessárias para que os usuários possam acessar as informações e a devida restrição de acesso aos dados que não são de importância para os usuários.

3 – DETECÇÃO DE INTRUSOS

3.1 – Conceito

Os IDS podem ser definidos como uma ferramenta com a finalidade de identificar e responder aos sinais gerados pelos eventos que ocorrem em uma rede ou um *host* e que, depois de analisados, possam revelar indivíduos que estejam utilizando um sistema de computador sem autorização, o que pode comprometer a confiabilidade, integridade e disponibilidade dos serviços fornecidos pelo ambiente em questão. Essas invasões, ou tentativas, podem ter sua origem proveniente da Internet, bem como serem originadas da rede interna, por meio de usuários legítimos ao sistema que pode, intencionalmente ou não, causar algum dano aos sistemas da organização em questão.

O objetivo de um IDS não está focado em prevenir qualquer tipo de intrusão que seja, mas sim em detectar essa intrusão ou indícios que evidenciem uma tentativa de invasão. Uma vez que o IDS obtenha algum resultado, o administrador da rede é notificado sobre o ocorrido, para que as ações necessárias possam ser tomadas.

Técnicas diferentes usadas na detecção acarretam em diferentes tipos de benefícios, aplicáveis em uma grande variedade de ambientes sendo, portanto, necessário selecionar aquele tipo que mais possa se adequar às necessidades do ambiente a ser protegido. A forma mais correta de definir o tipo a ser utilizado consiste em um conhecimento pleno de quais são às requisições específicas do sistema para, então, executar a implantação de um IDS. (PROCTOR, 2001)

Os IDS estão, atualmente, divididos em dois modelos, os quais têm sido utilizados por administradores de sistemas e de redes em geral, de acor-

do com as necessidades definidas pelas políticas de segurança adotada: Sistemas baseados em *host* (HIDS – Host–based Intrusion Detection System), operando em *hosts* específicos, como servidores e estações de trabalho, e sistemas baseados em rede (NIDS – Network–based Intrusion Detection System), colocados em determinados pontos da infraestrutura de redes. (BACE & MELL, 2000)

Ambos os sistemas, NIDS e HIDS podem efetuar suas análises a partir de um conjunto de regras pré–definidos, armazenados em uma base de dados, que são como uma espécie de impressão digital das ações originadas de vários tipos de ataques conhecidos, conhecidos como *assinaturas*. Desta forma, uma simples particularidade presente em um determinado tipo de ataque pode servir para que seja identificado pelo IDS.

Quando ocorre uma detecção, o IDS gera uma resposta. Essa resposta pode requerer uma reação manual por parte do administrador ou uma reação automática previamente definida, que pode ir desde desconectar o usuário suspeito até a reconfiguração das regras do *firewall*. Se a reação a ser tomada for por parte do administrador de sistemas, este irá tomar as medidas cabíveis para resolver o problema, antes que ocorra algum dano grave ao sistema. As reações automáticas, configuradas no IDS, devem ser um procedimento cuidadosamente estudado, pois seu uso pode ser perigoso para os serviços oferecidos, uma vez que mediante a presença de um falso–positivo, na detecção, pode parar um processo essencial ao sistema.

3.2 – Formas de Execução

A forma como o IDS executa suas análises pode variar conforme a solução pretendida. Mais comumente usados pelos atuais IDS para análise dos dados coletados são os modelos de tempo–real e o periódico.

3.2.1 – Tempo–Real

Este modelo é citado por (PROCTOR, 2001) como um dos "*mitos*" existentes entre os IDS´s. As empresas, normalmente, acham que a detecção e reação em tempo–real consiste em um requisito incondicional para justificar todo e qualquer investimento em um sistema de detecção de intrusos, ou seja, se um IDS não pode barrar os invasores em questão de segundos o investimento não compensa.

Esse modelo é muito utilizado por soluções NIDS, principalmente nas comerciais, pois os dados analisados são provenientes do tráfego constante de pacotes existente na rede, exigindo assim, uma análise imediata desses dados. Apenas alguns segundos de análise, feita por um NIDS, já são o suficiente para que algum tipo de invasão seja detectado.

As soluções HIDS também podem efetuar suas análises de eventos em tempo–real, possibilitando detectar intrusões com relativa rapidez, o que acaba por degradar a performance do *host*, principalmente pelo alto uso da CPU do equipamento, prejudicando os demais serviços que este estiver provendo. (BROOK,2002)

3.2.2 – Periódico

Através deste, os dados coletados pelos registradores de eventos do sistema, são analisados em períodos pré–determinados. Isto faz com que o uso da CPU seja reduzido e melhor gerenciado, evitando assim, uma degradação no desempenho do restante do sistema. O grande problema desse modelo é que um possível intruso pode ser detectado quando já houver feito a invasão, ou seja, tarde demais. Isso se o mesmo já não ter finalizado o IDS. (ILGUN, 1992)

3.3 – Firewall x IDS

Devemos saber diferenciar as funções propostas entre dois sistemas utilizados na implementação da segurança em ambiente de rede: o *Firewall* e o IDS. O *firewall*, normalmente está localizado entre a rede interna e a Internet e implementa uma política de filtro de pacotes, bloqueio de portas e restrições de acesso ao sistema. Isto faz do *firewall* um alvo constante de ataques, afinal é a primeira linha de defesa da rede de qualquer organização. Esses ataques podem ser de diversas formas, porém os mais comuns ocorrem através dos próprios protocolos de rede e da exploração de possíveis falhas e brechas das aplicações.

Os ataques efetuados com o uso dos protocolos de rede tiram proveito da característica básica do *firewall*, o filtro de pacotes, cujo foco está nas informações contidas no cabeçalho dos pacotes, ignorando o conteúdo desses pacotes. Dessa forma, o código malicioso pode ser encapsulado dentro do pacote, o que vem a mascarar seu conteúdo, possibilitando assim, sua passagem através do *firewall*.

Já os ataques baseados em aplicações exploram as vulnerabilidades existentes em determinadas aplicações, através do envio de pacotes diretamente para a aplicação em questão. Um exemplo clássico é o *nuke*[2], que ao enviar uma string para a porta 139[3] em sistemas Windows 95, 98 e NT, ocasionava o travamento do sistema, de tal maneira que o usuário via-se obrigado a reiniciar o computador. Outro exemplo, o envio de comandos ao servidor HTTP (HyperText Transfer Protocol), que pode ocasionar um *buffer overflow*[4] na aplicação web. A chance de ocorrência de uma situação dessas é maior, uma vez que o *firewall* normalmente é configurado de forma a permitir o tráfego de

2 Nuke – Software utilizado para causar negação de serviço em alguns sistemas Windows mais antigos.

3 Porta 139 – Serviço de Netbios do sistema.

4 Buffer overflow – consiste em exceder a capacidade de dados suportada pelo buffer de memória.

pacotes HTTP, mesmo aqueles que possam conter seqüências de comandos, os quais podem passar desapercebidos pelos filtros do *firewall*.

3.4 – NIDS – Detecção de Intrusos baseado em Rede

A função de um IDS baseado em redes é verificar o conteúdo dos pacotes que trafegam pela rede, buscando padrões de atividades suspeitas, como *IP Spoofing*[5], por exemplo. É bem utilizado na diferenciação entre ataques que envolvam uma manipulação de baixo nível da rede e ataques disparados contra múltiplos *hosts* na rede.

A análise dos dados obtidos na verificação dos pacotes vem tirar proveito do fato de que os *hosts* fazem parte de uma conexão comum a todos os demais *hosts* da rede, dentre eles, aquele onde está instalado o NIDS. (SYMANTEC,2001)

Para (LAING, 2000), o NIDS efetua a inspeção de conteúdo não só dos cabeçalhos dos pacotes em busca de sinais que denunciem atividades suspeitas, mas tambem do pacote propriamente dito. Dessa forma, uma suspeita de ataque pode ser detectada quando estiver ocorrendo, em tempo-real, proporcionando uma resposta rápida e um alerta sobre esse ataque.

O NIDS caracteriza-se por trabalhar com alguns cenários de ataques já conhecidos, como acesso não autorizado, roubo de dados e negação de serviço, por exemplo. A maioria desses ataques tem como alvo principal as vulnerabilidades existentes nos sistemas operacionais.

---

5 IP Spoofing – Forma de ataque onde o atacante se aproveita da confiabilidade entre computadores, na Internet, e se disfarça como um desses, de forma que o computador alvo possa aceitar seus comandos tranqüilamente.

3.4.1 – Acesso não autorizado

Podemos definir, como todo aquele acesso de *logon*[6] , ao sistema, feito por um usuário externo, através da rede, sem a devida permissão. Uma vez autenticado, esse acesso pode ser monitorado com mais eficiência com um HIDS, mas o ideal seria que a detecção ocorresse antes que o meliante obtivesse sucesso, ou seja durante a tentativa de acesso.

Login não autorizado – normalmente não deveria ser permitido, mas, devido aos programas utilizados para compartilhar informação e recursos entre computadores, existe uma grande quantidade de vulnerabilidades que podem ser exploradas, visando conseguir acesso.

Ponto de partida para outros ataques – os "atacantes" raramente atingem seus alvos a partir do computador de sua casa. Normalmente utilizam-se de computadores previamente "hackeados", os quais podem conter informações que possibilitem o acesso a outros computadores da empresa, os quais podem ser utilizados como ponto de partida para ataques futuros. Esse tipo de ataque pode ser identificado por padrões de tráfego na rede. Porquê o servidor de arquivos faria acesso à um determinado site, uma Instituição Financeira, por exemplo?

3.4.2 – Roubo de dados

Não é apenas uma brincadeira da comunidade *hacker*[7]. Muitos governos treinam e fazem uso de cyberespiões, cujas habilidades são utilizadas para efetuar espionagem industrial em outros países. Além disso, existem

6 Logon – Ato de autenticação de usuário para acessar o sistema operacional.

7 Hacker – Usuário especializado cujo objetivo nas invasões à sistemas é apenas o conhecimento e o desafio, sem intenção destrutiva.

aqueles *crackers*[8] que efetuam invasões com o intuito de roubar informações que possam vender para as empresas concorrentes.

Download de senhas – é um dos mais tradicionais tipos de roubo de arquivos que o NIDS detecta. O eventual download não autorizado dos arquivos que armazenam a base de usuários, contendo dados pessoais e senhas criptografadas, por exemplo, pode até vir a comprometer outros sistemas existentes no ambiente.

3.4.3 – Negação de Serviço – DoS

Esse nome deve–se ao efeito causado, que resulta em indisponibilizar os serviços prestados pelos site–alvos. Normalmente, esse ataque é efetuado através da utilização de um site intermediário, bem como seus *hosts*, para amplificar esse ataque. Pode ocorrer de várias formas e com diferentes níveis de gravidade. O mais famoso exemplo de ataque DOS (Denial of Service) ocorrido foi quando o site da livraria virtual Amazon.com (CARR, 2002) ficou indisponível devido à um forma avançada do ataque de DOS, o DDOS (Distributed Denial of Service) que, diferentemente do DOS, utiliza–se muitos *hosts* de redes diferentes para atacar o site–alvo. Os três exemplos que seguem referem–se às três técnicas de ocasionar um DOS. (procurar site que informa esse problema)

Pacotes mal–formados – são pacotes que apresentam variações em seu formato e tamanho, ocasionando assim, um estouro na pilha do protocolo, uma vez que este não consegue efetuar a junção dos mesmos, devido à sua diferença.

Inundação de pacotes – esta técnica que consiste no envio de uma grande quantidade de pacotes à um determinado *host*, de tal forma que esse

8 Cracker – Hackers mau intencionados, que invadem sistemas com intenções inescrupulosas e objetivando danos.

não consiga responder a todos e, mediante ao acúmulo de requisições, venha a ocasionar uma degradação no tráfego da conexão.

DoS distribuído – é semelhante ao modelo anterior, porém, este procede com o uso de vários computadores para enviar a inundação de pacotes ao *host* alvo. A associação desta com a técnica de *IP Spoofing* torna, virtualmente, impossível identificar a origem do ataque.

3.4.4 – Arquitetura

Um NIDS consiste em sensores espalhados pela rede que respondem a uma console central. Esses sensores normalmente contêm um mecanismo cuja finalidade é receber todos os pacotes que trafegam pela rede, sendo a ele direcionados ou não, para então verificar seu conteúdo e reportar a notificação de um eventual alarme à console central. Existem dois modelos de arquiteturas, *network–node* e *sensor–bases*.

No modelo *network–node*, cada um dos computadores da rede possui um agente instalado, cuja finalidade é analisar os pacotes que lhe são destinados. Esse modelo caracteriza–se como amplamente distribuído, abrangendo todos os alvos de missão crítica.

No modelo *sensor–bases* um computador específico é utilizado para monitorar todo um segmento de rede. Não pode ser considerado exatamente um modelo amplamente distribuído, pois o número de segmentos de redes a serem monitorados é menor que o número de computadores conectados à rede. Para efetuar esse trabalho, a placa Ethernet do computador em questão é chaveada para modo *promíscuo*, ocultando o endereço MAC, o que lhe proporciona receber absolutamente todos os pacotes que trafegam pelo segmento de rede. Esta técnica é popularmente conhecida como *sniffer*[9].

9 Sniffer – Processo de coletar todos os pacotes que trafegam na rede para análise.

O modelo de uma solução NIDS mais comumente implementado, demonstrado na figura 1, consiste em posicionar o *firewall* entre dois *hubs*[10], na sua conexão entre a rede local e a Internet. Essa arquitetura permite que tráfego de ambos os sentidos passe, também, pelo NIDS, onde será possível verificar e analisar tanto os pacotes oriundos da Internet quanto da rede interna, podendo detectar ações suspeitas em ambos os lados. Nesse modelo, o NIDS fornece as informações ao gerente de segurança, para que este possa tomar as devidas providências, ou, caso esteja configurado para isso, pode atualizar as configurações, correspondente ao evento detecado, nas regras do *firewall*.

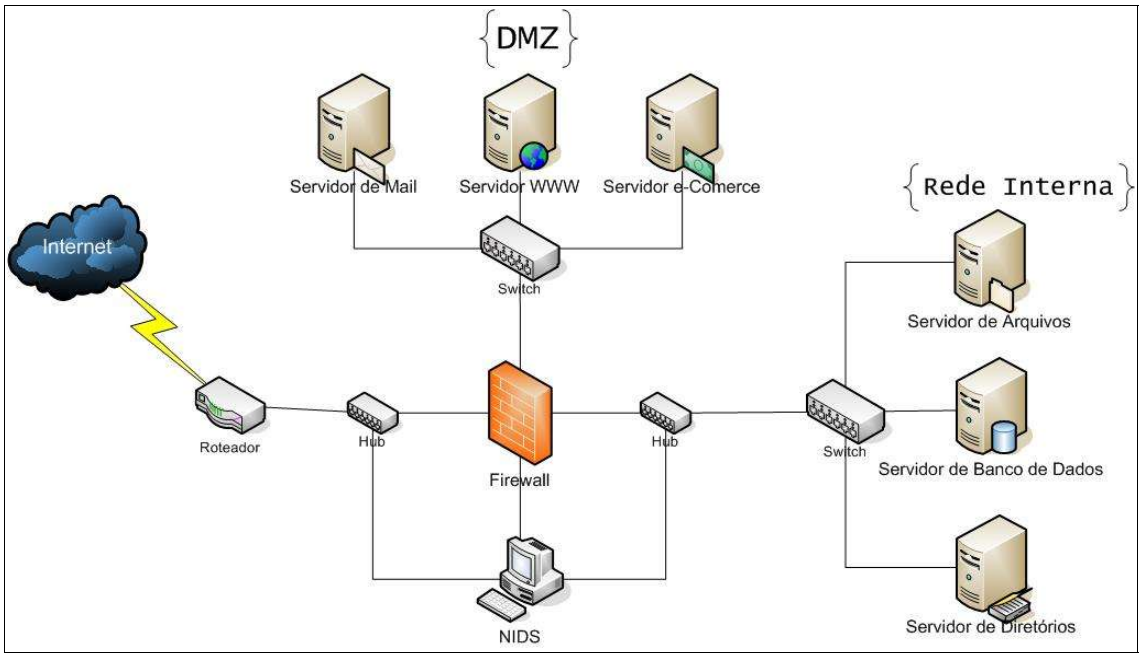


Figura 1 – Implementação de solução NIDS.

3.5 – HIDS – Detecção de Intrusos baseado no Host

Este sistema tem sua funcionalidade através da análise dos dados gerados pelos eventos que ocorrem no sistema, normalmente armazenados nos arquivos de *log* do sistema.

10 Hub – Concentrador para o cabeamento que interliga *hosts* em um ambiente de rede.

Apesar do NIDS ser mais popular, principalmente devido ao grande advento da Internet, o HIDS tem mostrado sua importância e necessidade no monitoramento de possíveis ameaças originadas na rede interna, situação responsável por uma grande parcela das invasões de sistemas. O HIDS apresenta uma maior eficiência para detectar a má utilização do sistema por um usuário, uma vez que efetua a monitoração das ações dos usuários conectados. Os "*log*" gerados pelo sistema operacional e pelas aplicações, auxiliados pelos históricos dos usuários, apresentam-se como as principais fontes de informação para o HIDS.

Um HIDS, normalmente, efetua a análise dos arquivos de *log* em intervalos de tempo predeterminados pelo administrador de sistemas. Certamente que, se essa análise fosse feita em tempo-real, as respostas poderiam ser mais rápidas, proporcionando um melhor índice de segurança ao sistema, mas, neste caso, deve-se levar em conta que o preço pago por essa maior utilização de recursos em prol do resultado seria um *overhead*[11] do sistema. (LAING, 2000)

Além de analisar eventos, o HIDS pode, também, monitorar e identificar possíveis tentativas de varredura de porta no sistema. (LAING, 2000)

Vejamos alguns cenários de ataque, enumerados por (PROCTOR, 2001), que o HIDS pode detectar:

Abuso de privilégios - ocorre quando um usuário possui direitos de administrador, e faz um mau uso desses direitos, podendo vir a colocar em risco toda a segurança do sistema, além da possibilidade de restringir ou desativar o HIDS.

Ex-empregados utilizando uma conta antiga - quando uma organização demora um certo tempo para remover a conta de um empregado demitido, este

11 Overhead - utilização excessiva dos recursos do sistema, como memória e cpu, dentre outros.

pode fazer uso da mesma para tentar prejudicar a empresa, como forma de vingança.

Administrador cria contas de backdoor – pode ocorrer quando o administrador não cria contas conforme os procedimentos normais, como por exemplo, durante a instalação de um software, este mesmo gerencia a criação de uma conta necessária para seu funcionamento. Isso pode implicar em uma conta de usuário não documentada e que pode ficar esquecida.

Falsificação de resultados – é um problema muito sério, e quando ocorre, o HIDS pode verificar a extensão do dano, bem como identificar o responsável por esse ato.

Usuários anônimos procurando por arquivos críticos – a observação dos acessos que estiverem à procura de arquivo ditos críticos, como arquivos de senhas, por exemplo, pode indicar que usuário está querendo tais arquivos.

3.5.1 – Arquitetura

O HIDS faz uso de agentes instalados nos *hosts*, os quais, uma vez que bem gerenciados, podem gerar resultados satisfatórios sem ocasionar uma queda significante na performance do sistema, aproveitando os benefícios que esse sistema pode prover. Esses agentes são pequenos programas que estão em execução nos *hosts* e se comunicam com um computador central, responsável pelo gerenciamento desses agentes.

Podemos destacar alguns modelos de arquitetura disponíveis para implementar uma solução de HIDS.

3.5.1.1 – Arquitetura Centralizada

Nesse modelo, os dados coletados pelos agentes são enviados e analisados por uma console de gerenciamento, um outro computador central de controle desses agentes. Uma vez que o processamento da análise desses dados é feita na console de gerenciamento, o impacto na performance do *host* é insignificante, melhorando, assim, a performance do agente. A console de gerenciamento armazena a base de assinaturas de todos os agentes. Porém, esse modelo arquitetural apresenta como desvantagem, a grande dificuldade, quando não impossibilidade, de efetuar a detecção e resposta em tempo–real, exceto na existência de um número muito pequeno de agentes ou no uso de um computador de processamento extremamente rápido como console de gerenciamento. Outro fator que deve ser considerado, é um possível aumento no tráfego da rede em questão. (PROCTOR, 2001)

Tabela 1 – Vantagens e Desvantagens da Arquitetura Centralizada.

Vantagens	Desvantagens
– Não há degradação de performance no *host* alvo. – Informações estatísticas de comportamento. – Assinaturas *Multi–host*.	– Não tem detecção em tempo–real. – Não tem resposta em tempo–real. – Gera mais tráfego na rede.

3.5.1.2 – Arquitetura Distribuída de Tempo–Real

Os dados gerados pelos eventos são analisados, em tempo–real, no próprio *host*, propiciando os devidos alertas e respostas. Sua principal vantagem é o processamento em tempo–real, porém têm–se a degradação da performance do sistema com preço a ser pago por esse modelo, além de não ser muito efetivo no processo de detecção.

Já que um HIDS pode operar em diferentes modos, torna–se essencial selecionar aquele que melhor se adapte aos ambientes onde deverão ser insta–

lados, bem como às necessidades de detecção da organização. Pode, ainda, operar em modos os quais requerem uma certa quantidade de pessoal, como pode operar em modo que permita minimizar os recursos.

Tabela 2 – Vantagens e Desvantagens da Arquitatura em Tempo–real

Vantagens	Desvantagens
– Alerta em tempo–real. – Resposta em tempo–real.	– Degradação da performance do *host* alvo. – Não tem estatisticas de comportamento. – Não tem assinaturas *multi–host*.

3.5.2 – Gerenciamento de políticas

Através desse procedimento, segundo (PROCTOR, 2001), é possível re–duzir consideravelmente a degradação da performance e os custos associados com a operação de um HIDS. Existem duas políticas que deve ser gerenciadas, primeiramente, políticas de auditoria e de detecção.

3.5.2.1 – Políticas de Auditoria

Define quais ações do usuário irá resultar em um registro no *log* de eventos. A redução dos registros armazenados no arquivo de *log* pode acarre–tar na queda de performance do IDS baseado no *host*.

Muitos acham que habilitando as políticas de auditoria irá inundar o sistema com eventos para serem gravados nos arquivos de *log* podendo, assim, encher todo o espaço livre do disco rígido. Portanto, como os eventos ocorrem quando cada objeto do sistema é acessado, a melhor maneira de controlar o espaço no disco é restringir a auditoria de eventos apenas nos acessos feitos aos arquivos de missão crítica, pastas e alguns objetos do sistema.

Uma política de controle de acesso é algo muito difícil de administrar, pois é possível abrir as portas para usuários que podem ganhar privilégios

para acessar dados aos quais lhes faltaria a autorização necessária para acessá–los.

3.5.2.2 – Políticas de Detecção

Através desta, define–se os padrões a serem seguidos na análise dos dados na busca de um mau uso do sistema. No caso, o reconhecimento de assinaturas é o mecanismo mais comum no uso do HIDS. Assinaturas são um conjunto de regras que define uma seqüência de eventos e um grupo de transições entre os eventos. O foco para uma boa política de detecção é um conjunto de assinaturas devidamente configurado, que proporciona um número apropriado de assinaturas detectados em tempo–real. Uma configuração com muitas assinaturas ou com poucas, evidencia o resultado de uma má aplicação das políticas de detecção.

3.6 – NIDS x HIDS

Uma dúvida muito comum entre os administradores de sistema é sobre qual dessas tecnologias implementar no seu ambiente de trabalho. Existe uma grande concorrência entre o NIDS e o HIDS, porém é sempre necessária a avaliação sobre o que se deseja monitorar e proteger, de forma a poder efetuar a escolha adequada.

3.6.1 – Diferenças entre HIDS e NIDS

O NIDS processa as informações coletadas nos pacotes TCP/IP que trafegam pela rede.

O HIDS analisa os dados dos arquivos de *log* de eventos gerados pelo sistema operacional, banco de dados e algumas aplicações.

Dessa forma, percebe-se que possuem arquitetura e técnicas diferentes. Além disso, o HIDS pode trabalhar de forma distribuída, enquanto o NIDS é implementado de forma mais centralizada.

Os dados das aplicações são as principais fontes de informações para o HIDS, porém, a dificuldade de gerenciamento desses dados demonstra um crescimento exponencial na sua complexidade, se comparados com os dados analisados por um NIDS. Por esta razão, as organizações tendem a implementar, inicialmente, uma solução NIDS, para só então focarem suas atenções para as soluções HIDS.

Essa diferença, existente entre os dois modelos de IDS, é o que torna interessante a implementação dessas duas soluções em uma organização propiciando, assim, todos os benefícios que podem ser adquiridos de cada um deles.

3.6.2 – Vantagens do NIDS

Vamos citar, agora, algumas vantagens que os sistemas baseados em rede apresentam.

- Os agentes de rede podem monitorar e detectar ataques de rede, *SYN Flood*[12];

- A inserção de novos agentes de rede não afeta a atual origem dos dados para análise;

- Identificação de erros nas camada de rede (GOELDENITZ, 2002);

12 SYN Flood – Envio de milhares de pacotes por segundo ao servidor para esgotar os recursos do sistema.

- Os dados são coletados sem nenhum requisito especial, na maioria dos casos sendo coletados simplesmente configurando a interface de rede para o modo promíscuo, de forma a receber pacotes oriundos de qualquer *host*, mesmo que não lhe tenham sido endereçados.

3.6.3 – Desvantagens do NIDS

Existe, porém, algumas situações onde o uso do NIDS poderá não ser bem sucedida, de modo a não surtir o efeito desejado. Podemos citar alguns exemplos:

- Não pode capturar e analisar os protocolos se estiverem trafegando criptografados pela rede (BACE, 2000);

- Alertas de Falso/Positivo;

- A utilização de switches, em ambientes de rede, dificulta o monitoramento e detecção baseada em rede, pois uma rede com switch cria um segmento de rede para cada *host*, fazendo com que o NIDS fique restrito à monitorar apenas um único *host* (BACE, 2000);

- Os NIDS atuais podem ter sua performance muito prejudicada ao manipular redes de banda larga, com alta velocidade de transmissão de dados (BACE, 2000);

- Latência entre tempo de ataque e tempo de alerta (GOELDENITZ, 2002);

- Pode deixar rede vulnerável, senão inacessível, no caso de sofrer um ataque de DoS;

- Sua força está no uso das últimas atualizações das assinaturas, pois novas variações ou padrões de ataques não serão registrados (GOELDENITZ, 2002).

3.6.4 – Vantagens do HIDS

O HIDS apresenta uma série de vantagens que o tornam uma solução muito interessante. Vejamos alguns exemplos citados por (BACE, 2000):

- Pode monitorar informações de acesso, do tipo “quem acessou o que”;
- Habilidade para associar o usuário à um determinado evento (GOELDENITZ, 2002);
- Pode detectar ataques que não são detectados pelo NIDS (GOELDENITZ, 2002);
- Monitorar mudanças de comportamento através da verificação de mau uso dos recursos;
- Fornecer informação sobre o *host* durante um ataque;
- Pode operar em ambientes de rede compostos por “switches”;
- Pode distribuir os agentes e os analisadores de informações através de *hosts* instalados em grandes redes, reduzindo custos de distribuição.

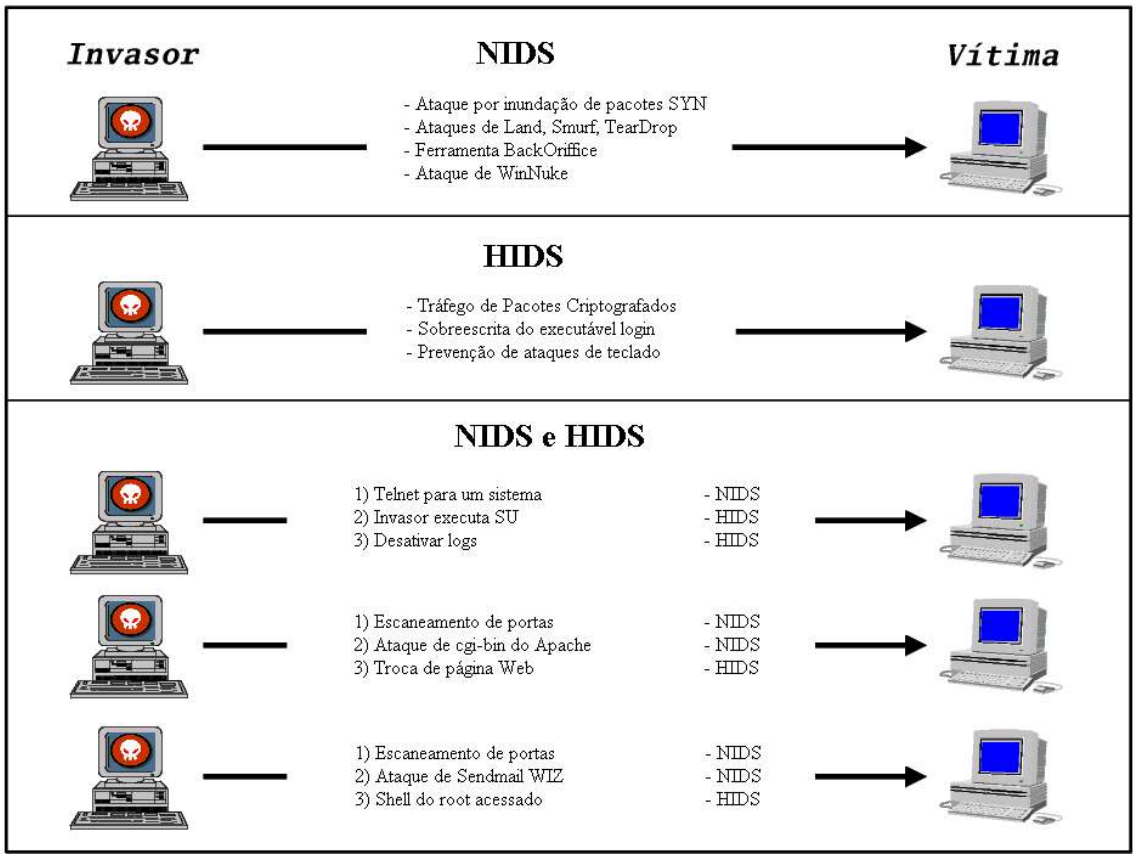
3.6.5 – Desvantagens do HIDS

Um sistema HIDS também possui algumas desvantagens. Eis alguns exemplos:

- Não podem detectar atividades de rede;
- Os mecanismos de auditoria podem gerar um *overhead* no sistema;
- As vulnerabilidades do sistema operacional podem comprometer a integridade dos analisadores e agentes do HIDS;
- Se um ataque derrubar o sistema operacional, o HIDS cai com o sistema;
- Pode ser ineficiente durante um ataque DoS;
- Os agentes devem ser mais específicos quanto à plataforma utilizada, o que pode gerar um custo adicional;
- Os custos com gerenciamento e distribuição dos HIDS são, geralmente, maiores que um NIDS, por exemplo.

3.6.6 – Técnicas de Aplicabilidade

A Figura X, tem o propósito de ilustrar algumas técnicas de ataque e o respectivo sistema IDS que pode detectá–la.



Fonte: (LAING, 2000)

Figura 2 – Tipos de ataques detectados pelo IDS.

3.7 – Características para um bom IDS

Um IDS deve satisfazer algumas características que o tornam uma solução.

1 – Deve ser estável e de confiança, de tal forma que execute continuamente sem depender de qualquer tipo de supervisão por parte do administrador de sistemas.

2 – Deve ser tolerante a falhas, para que não seja necessário a reconstrução da base de dados após uma queda do sistema.

3 – O IDS deve proteger a si mesmo, ou seja, monitorar a si próprio para que não seja corrompido.

4 – Uma das principais características, gerar o menor *overhead* possível no sistema. Afinal, um IDS que degrada a performance do computador é uma solução inviável.

5 – Precisa perceber desvios de comportamento de um determinado usuário.

6 – Precisa ser direcionado para o sistema ao qual deverá monitorar.

7 – Deve adaptar–se às mudanças que ocorrem no sistema quando uma nova aplicação é instalada.

3.8 – Distribuição do IDS

A implantação de um IDS requer um estudo elaborado do caso, onde deve ser feito um planejamento cuidadoso e toda uma preparação do processo. Montar o protótipo a ser testado, para então efetuar o treinamento especializado da solução.

Estratégicamente, (BACE & MELL, 2002) sugerem, em uma implementação onde seja almejada uma segurança relativamente boa, que seja adotada uma combinação entre os modelos NIDS e HIDS.

O HIDS pode contribuir com um nível avançado de proteção para o sistema, enquanto o NIDS propicia uma boa segurança à nível de rede. Entretanto, instalação de um HIDS em cada *host* da rede pode demandar um gasto desnecessário de tempo, pois deve ser configurado em cada um desses *hosts*. Assim, a instalação do HIDS apenas nos servidores e do NIDS em computadores instalados de forma estratégica na rede, proporciona que as atenções possam ser concentradas nos alarmes gerados por aquelas que podem ser considerados os *hosts* mais importantes da rede.

3.9 – Métodos de Detecção

Citamos, agora alguns dos principais métodos utilizados pelos IDS's no processo de identificação de uma provável intrusão no sistema.

3.9.1 – Sistemas de Detecção por Anomalias

Esse é um dos métodos de análise de mais comuns em um IDS, através do qual uma análise é feita mediante um perfil, previamente gerado, do comportamento apresentado por cada usuário do sistema. A geração desse perfil baseia-se através de uma abordagem estatística das ações que o usuário toma no sistema, juntamente com uma previsão dos padrões tendenciosos que poderiam ser adotados por esse usuário.

3.9.1.1 – Abordagem Estatística

Este método cria um perfil de comportamento para cada usuário, o qual é totalmente baseado nas ações por este efetuadas. Durante o uso do sistema, uma variante desse perfil é gerada a partir do perfil dito original. Dessa forma, vários fatores podem afetar o comportamento desse perfil, tais como tempo de uso da CPU, número e tipos de conexões em um período, arquivos acessados, entre outros. Assim, este método permite que o sistema efetue uma espécie de aprendizado sobre o comportamento do usuário.

Porém, existe um problema. Um possível invasor, uma vez de posse do *logon* de um usuário, pode efetuar ações específicas para “treinar” o IDS, para que esse comece a interpretar estas ações como normais, apesar de, originalmente, não fazerem parte do cotidiano do referido usuário.

A detecção de anomalias apresenta uma boa funcionalidade, mas apenas como uma ferramenta de auxílio a um suporte de decisões, pois pode

mostrar–se uma solução falha quando utilizada como forma de um mecanismo automatizado para detecção. Os modelos atuais não conseguem distinguir usuários confiáveis em muitos sistemas, o que acaba por gerando uma quantia significante de falso–positivos[13]. (PROCTOR, 2001)

3.9.2 – Geração de padrões previstos

Como o próprio nome sugere, consiste na tentativa de prever eventos futuros com base em eventos já ocorridos. Baseia–se na análise probabilística de que um determinado evento possa vir a ocorrer, mediante algum evento que já tenha ocorrido anteriormente.

Esse método possui suas vantagens, tais como: encontrar atividades anormais que seria difícil através dos métodos tradicionais. Sistemas construídos com base nesse método mostram–se mais aptos a eventuais modificações. É fácil detectar usuários que estão tentando "treinar" o sistema durante seu período de aprendizagem. Atividades anormais podem ser detectadas e respondidas em questão de segundos a partir do recebimento das informações dos eventos.

3.9.3 – Detecção por Mau Uso

Este método, como o próprio nome sugere, consiste em monitorar as ações que indiquem a utilização inadequada dos recursos do sistema. Uma lista de assinaturas de mau uso pode ser construída baseada em duas formas de comportamento, aceitável e não aceitável.

13 Falso–positivo – Um evento normal do sistema, que pode ser interpretado como um mau uso do sistema.

3.9.3.1 – Comportamento Aceitável

Nessa implementação, a lista é construída com todas ações que são permitidas aos usuários. Desta forma, qualquer evento que não se enquadre nesta categoria pode ser considerado como não aceitável, ou seja, um mau uso do sistema.

A construção deste modelo é feito através de dados históricos que serão usados para definir os comportamentos aceitáveis. (PROCTOR, 2001)

3.9.3.2 – Comportamento Não Aceitável

Contrariamente ao exemplo anterior, esta lista é construida definindo todas as ações que não são permitidas aos usuários. Assim, caso algum evento se enquadre nesta categoria, é gerado um alarme de mau uso do sistema.

Este é um modelo determinístico, cuja construção é baseada em regras. Apesar dessas regras serem limitadas, sua utilização é grande nos IDS's comerciais.(PROCTOR, 2001)

4 – TRABALHOS RELACIONADOS

O tema Detecção de Intrusos em questão, tem sido bastante pesquisado tanto em Universidades como em empresas privadas, ambos objetivando o aperfeiçoamento nas técnicas de segurança de sistemas. Destacamos, então alguns exemplos que são mais facilmente encontrados na Internet.

4.1 – HIDS baseado em Agentes Autônomos

Projeto em desenvolvimento na University of Purdue, USA, cuja arquitetura baseia-se no uso de agentes autônomos para efetuar o serviço de detecção e resposta à possíveis intrusões, enquadrando-se na arquitetura de agentes distribuídos.

Neste projeto, os agentes autônomos são definidos, conforme (ZAMBONI & SPAFFORD, 2000), como *"um software que proporciona uma certa segurança monitorando funções em um host"*. Sua idéia central constitui-se em entidades que executam independentemente sendo gerenciada apenas pelo sistema operacional, ao invés de algum outro processo.

4.1.1 – Características desse IDS

Como já mencionado, os agentes desse modelo são entidades que executam independentemente umas das outras, podendo ter seus componentes alterados sem a necessidade de reiniciar o IDS, eis alguns problemas, dos IDS's comuns, que essa arquitetura pode resolver:

- Se um agente parar de executar o dano estará restrito apenas àquele agente, enquanto os demais continuarão em funcionamento;

- Habilidade de inicializar e parar os agentes independentemente dos demais, proporcionando, por exemplo, a inserção de regras para coletar novos tipos de dados sem a necessidade de parar todo o IDS;

- A coleta de informação de rede relacionada ao *host* pode reduzir a possibilidade de ocorrerem suposições erradas sobre essas informações;

4.2 – Projeto ACME

O projeto ACME (Advanced Counter Measures Environment) (CANSIAN, 2002) é desenvolvido na Unespe de São José do Rio Preto, iniciado por Adriano Mauro Cansian. Consiste em um sistemas e detecão de intrusos em redes de computadores, baseado em captura de pacotes e com a análise das assinaturas de ataque utilizando rede neural. Vale destacar que esse projeto é pioneiro na utilização de rede neural como forma de análise de assinaturas.

4.2.1 – Modelo ACME

O modelo ACME é formado por um sistema de captura de pacotes e outro para análise dos dados. Ambos são divididos em módulos, que tratam desde a coleta até o tratamento da rede neural.

- Módulo de captura de pacotes – utiliza um método semelhante à um *sniffer*, para capturar pacotes em um ambiente de rede tipo broadcast;

- Módulo de pré–seleção e sistema especialista – funciona como um filtro inicial, decidindo se uma conexão pode ser considerada suspeita;

- Módulo de conexão – cria os vetores de conexão, que irão armazenar os dados coletados pelo primeiro módulo;

- Módulo de rede neural – munido dos dados gerados pela codificação binária dos vetores de conexão e, com base no treinamento da rede neural, um valor numérico é retornado, indicando o grau de suspeita da sessão.

4.3 – PortsEntry

O PortsEntry (SENTRYTOOLS, 2002) é um programa que monitora qualquer atividade em portas TCP/IP específicas. Essas atividades são relatadas e várias ações podem ser tomadas, como por exemplo, proibir futuras conexões com o *host* de onde as atividades eram originadas.

Constitui-se em um importante mecanismo de defesa, uma vez que um *hacker* podem investigar o sistema por vários dias, através de uma varredura nas portas, antes de tentar uma invasão ao sistema.

4.4 – LogCheck

O LogCheck (LOGCHECK, 200) consiste em uma ferramenta utilizada para efetuar uma varredura nos arquivos de "*log*" do sistema, buscando ocorrências estranhas. Essa tarefa é feita com base em algumas assinaturas que o mesmo possui para efeito de comparação com as informações dos arquivos de *log*. Uma vez detectado algo suspeito, o administrador de sistema é notificado via e-mail.

O logcheck possui dois arquivos, logcheck.hacking e logcheck.violations, onde estão armazenadas as assinaturas que podem referenciar-se à possíveis atividades suspeitas no sistema.

4.5 – HostSentry

O HostSentry (PSIONIC, 2001) é uma ferramenta utilizada para detectar anomalias nas contas de usuários que conectam–se ao *host* através de *shell*[14] por meio de serviços conhecidos, Telnet, SSH – Secure Shell, e outros. Sua análise consiste na leitura dos arquivos de *log* wtmp/utmp, além dos arquivos *history*[15] dos usuários.

O HostSentry (PSIONIC, 2001) não verifica as conexões que são feitas através de outros serviços, como IMAP (Interim Mail Access Protocol) ou POP3 (Post Office Protocol), bem como quaisquer outros que não escreva nos arquivos wtmp/utmp.

4.6 – Snort

O Snort (SNORT, 2001), consiste em um NIDS open source, o qual possui a capacidade de realizar análise em tempo–real do tráfego e armazenar os *logs*[16] dos pacotes analisados. O *snort* pode realizar uma análise de protocolos, buscando informações que indiquem uma variedade de ataques, como *buffer overflow*, port scan, ataques de CGI e muitos outros.

Os alertas são reportados em tempo–real para o administrador, seja através de e–mail, WinPopup, UNIX sockets, entre outros.

4.7 – LIDS (Linux Intrusion Detection System)

Apesar do que o nome sugere, o LIDS (LIDS,2002) não é necessariamente um IDS na sua forma explícita, possui outras funções além dessa. A

14 Shell – Interface entre o usuário e o computador. Nome comum dado ao acesso em modo texto, via teclado, efetuado aos sistemas UNIX.

15 History – Arquivo que armazena os comandos executados pelo usuário no shell

16 Logs – Informações dos eventos realizados no sistema.

única característica de IDS que apresenta é um mecanismo para detectar varredura de portas no sistema. Além disso, pode ser integrado ao *PortsEntry* e ao *LogCheck*, os quais podem fornecer uma funcionalidade de IDS ao LIDS.

Na realidade, o LIDS se propõe a incrementar a segurança ao sistema em si. Para isso é necessário uma atualização dos fontes do *kernel*[17] com o patch do LIDS e a recompilação do *kernel*, selecionando a opção de suporte ao LIDS.

Essa ferramenta, uma vez instalada e configurada restringe os poderes, até mesmo, do superusuário *root*. Dessa forma, uma vez que algum invasor consiga acesso à conta de *root*, não poderá fazer muita coisa no sistema, pois as características adicionadas pelo LIDS não permitirão, afinal, além da senha do *root*, é necessário, também, a senha administrativa para ativar e desativar o LIDS, que somente o verdadeiro administrador do sistema deverá possuir.

4.8 – Tripwire

O tripwire (TRIPWIRE, 2002) é uma ferramenta com duas formas de distribuição, comercial e domínio público. É utilizada para detectar os problemas causados por uma intrusão, já que sua tarefa principal é verificar a integridade do sistema de arquivos em sistemas UNIX e semelhantes.

Seu funcionamento consiste na criação de uma base de dados com informações do sistema de arquivos, como o tamanho dos arquivos por exemplo. O *tripwire* utiliza essa base de dados para verificar a ocorrência de quaisquer alterações no sistema de arquivos, comparando, periodicamente, as informações atuais dos arquivos com essa base de dados. Uma vez encontrada alguma alteração, essa informação é reportada ao administrador de sistema, que então, deverá decidir se essa alteração era ou não autorizada. Normalmente os arquivos verificados são arquivos que não tem a necessidade de serem altera-

17 Kernel – Parte central do Sistema Operacional, que gerencia os recursos do computador.

dos e, caso isso ocorra, com certeza pode ser considerado um ato não autorizado.

4.9 – Dragon

O Dragon© (ENTERASYS, 2002) é outra solução comercial, sendo uma das mais populares e robustas do mercado. Consiste em um sistema de detecção de intrusos produzido pela Enterasys Networks™, sendo composto por dois módulos, o *Sensor* e o *Squire.*

- *Sensor* – este módulo monitora, passivamente, as atividades no ambiente de rede, gerando alarmes com base em um conjunto de assinaturas e configurações de rede;

- *Squire* – este exerce a função de HIDS. Instalado nos *hosts* que podem ser alvo de ataques, monitora os atributos e conteúdo dos arquivos, informações do sistema, entre outras.

- *Police Manager* – utilizado no gerenciamento das políticas e do status dos Sensores, assim como gerenciar as configurações de rede e das bibliotecas de assinaturas dos sensores.

- *Event Flow Processor* – esta ferramenta é utilizada para processar as informações dos eventos, tais como: eventos de banco de dados, alarmes, gerar de *logs*, *checksum*, entre outros.

Ambos reportam os alarmes gerados para uma console central de gerenciamento, a qual poderá notificar o administrador de sistemas por e-mail, pager, informações no monitor, e outras que estajam habilitadas e configuradas.

5 – ABORDAGEM PROPOSTA

O trabalho aborda o modelo HIDS, centrando a idéia na proposição de viabilizar um conceito para o desenvolvimento de um protótipo para uma ferramenta de utilidade administrativa, de tal forma que possa funcionar como um sistema de gerenciamento para o administrador de sistemas, similar aos modelos utilizados por feramentas para monitoramento de redes e serviços, como por exemplo o Nagios[18].

A idéia consiste em agentes estáticos distribuídos e uma console central de gerenciamento. Esses agentes, uma vez instalados nos *hosts*, reportam informações à uma console central. Esses agentes deverão ter um certo limite na sua capacidade de tomar ações automáticas, cabendo ao administrador de sistemas participar desse processo, enviando os comandos, aos agentes, para que estes efetuem as devidas ações.

5.1 – Agente

Consiste em um programa em execução no *host* a ser monitorado. Sua funcionalidade deve ter como base os eventos que ocorrem no sistema, bem como a utilização de seus recursos. O enfoque principal para este agente é a utilização de uma ACL (*Access Control List*). Essa ACL classifica, basicamente, o mau uso, referenciando ações específicas que não podem ser efetuadas por usuários que não possuam status de superusuário. Como a proposta está inicialmente enfocada no uso de ACL, vamos minimizar o uso de uma lista de assinaturas, devendo esta conter uma quantidade relativamente reduzida de informações.

Quando algum evento se enquadrar nas regras da ACL ou na lista de assinaturas, o agente se deverá executar as medidas necessárias a fim de mini–

18 Nagios – Sistema de gerenciamento de redes. Site – http://www.nagios.org

mizar algum tipo de dano. Essas medidas podem ser tanto parar o processo que gerou o evento como desconectar o usuário que executou esse processo. Essas respostas, por parte do agente, podem ser configuradas, de tal maneira que para alguns eventos duvidosos, o agente notifique a console central, a qual deverá conter um conjunto maior de regras, assim como propiciar a atuação do administrador de sistemas, que poderá tomar alguma resolução para o fato informado, seja manter a execução do evento referido, como enviar um comando para finalização do mesmo.

Quanto à composição do agente, divide-se em dois módulos, Scanner e Analyzer.

5.1.1 – Módulo Scanner

Este módulo tem a função de verificar, em um determinado intervalo de tempo, o diretório virtual */proc*, onde estão armazenadas as informações sobre o que está acontecendo com o sistema naquele momento. Além disso, o módulo também verifica os arquivos de log *messages*[19] e *secure*[20], buscando informações sobre ações que possam ser consideradas suspeitas, mediante a consulta à uma base de assinaturas comuns.

5.1.1.1 – Estrutura do */proc*

O diretório */proc* é um pseudo-sistema de arquivos, onde suas informações são utilizadas pelo *kernel* como uma interface entre o mesmo e as estruturas de dados do sistema. Essas informações não são reais, são geradas mediante as situações que o sistema se encontra naquele momento.

---

19 Messages – log que armazena as ações ocorridas em sistemas UNIX/Linux.

20 secure – log que armazena autenticações de usuários, usado na distribuição Linux Slackware.

A listagem do diretório */proc*, em um ambiente Linux, irá revelar uma série de arquivos e diretórios, como pode ser observado na Figura 3. Dentre os diretórios, podemos destacar aqueles que são nomeados numéricamente, os quais correspondem aos processo em execução no sistema, naquele momento.

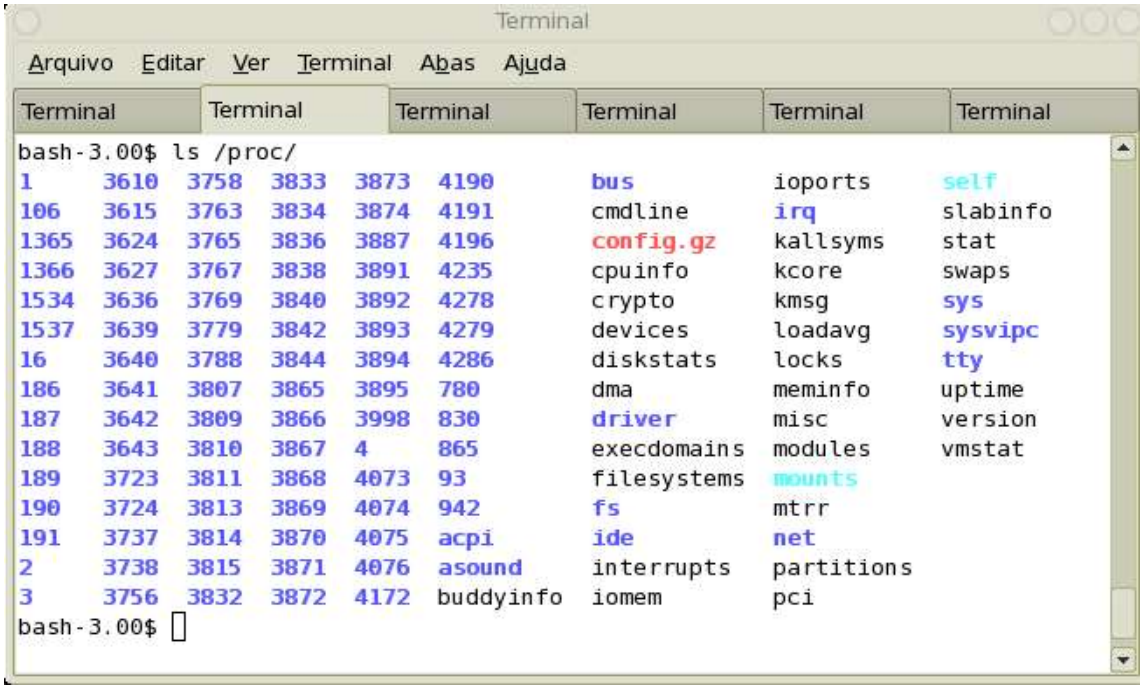


Figura 3 – Listagem do diretório */proc*

Normalmente, se houver algum *backdoor*[21] ou um *sniffer* sendo executado, o diretório correspondente ao seu processo irá aparecer na listagem, mas a única forma de descobrir um programa desses é através dos arquivos existentes dentro desses diretórios, como o *cmdline*, por exemplo, que armazena a linha de comando que executou o processo. Os demais são diretórios com informações sobre configurações e variáveis de sistema, *net, sys, ide*, e os demais.

21 Backdoor – Uma porta de conexão aberta, por um invasor, para envio de informações e/ou conexão remota ao sistema.

5.1.1.2 – Funcionamento do Módulo

O objetivo é verificar, em pequenos intervalos de tempo, o diretório */proc*, onde poderá ler, além do conteúdo do arquivo *cmdline*, também o arquivo *status* existente dentro dos diretórios dos processos obtendo-se as informações como Name[22], State[23], UID[24], PID[25] e PPID[26] de cada processo. Na figura que segue, podemos verifica um exemplo do conteúdo dos referidos arquivos.

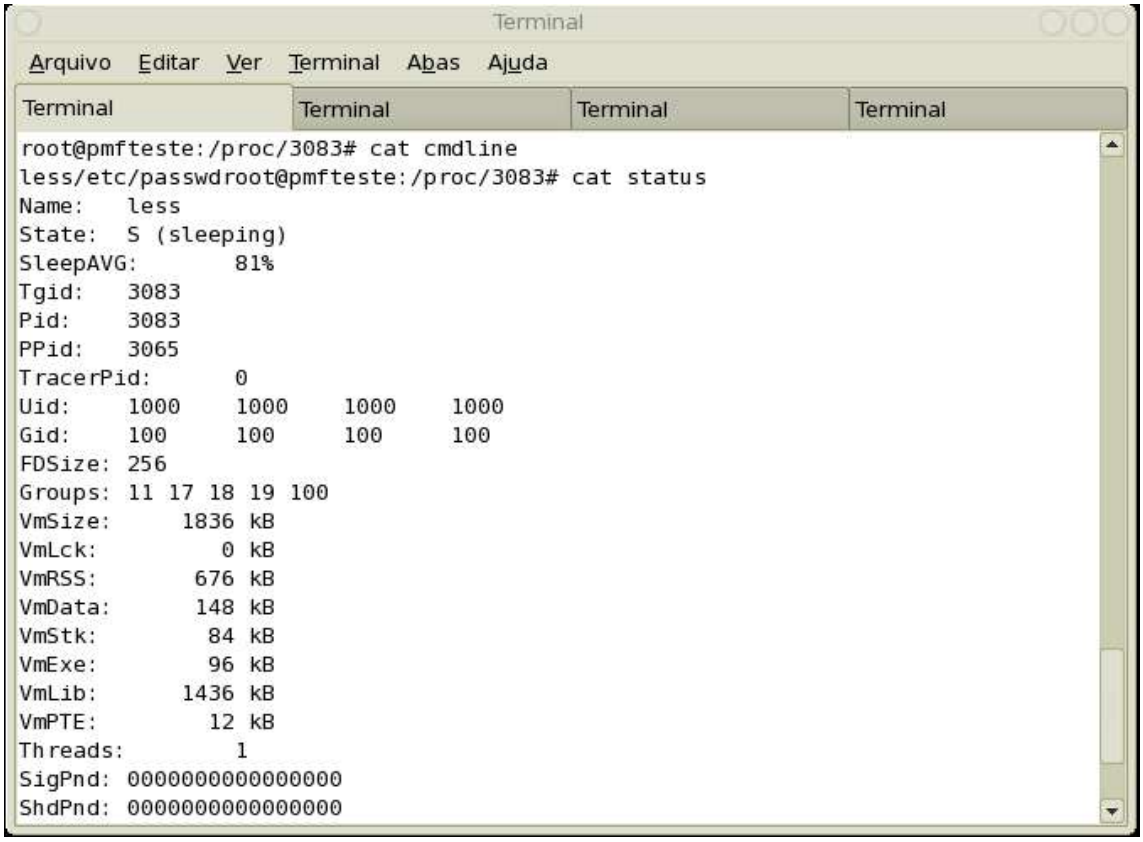


Figura 4 – Arquivos cmdline e status de um determinado processo.

22 Name – Nome do processo em execução.

23 State – Demonstra o estado atual do processo, *running* ou *sleeping*.

24 UID – Identificação do usuário que está executando determinado processo.

25 PID – Identificação de um processo em execução.

26 PPID – Identificação do processo-pai, que tenha chamado a execução de outro processo.

A figura exemplifica o conteúdo dos dois arquivos referidos, pertencentes a um determinado processo sendo executado, no caso "less /etc/passwd", conforme o arquivo *cmdline* demonstra, o que pode ser verificado, também, no nome do processo que está registrado dentro de *status*, assim como outras informações importantes, Pid (ID do processo), PPid (ID do processo pai), Uid (ID do usuário) e Gid (ID do grupo ao qual pertence o usuário). Todas essas informações nos permitem determinar quem é o usuário que executou esse processo.

Verificando-se os arquivos *tcp* e *udp* dentro do diretório */proc/net*, pode-se obter as tabelas de *socket*[27] TCP e UDP respectivamente, de forma a descobrir quais os serviços de rede que estão habilitados, para cada protocolo, naquele momento. Com essas informações, pode-se descobrir, também, o comando que iniciou um determinado serviço, bem como o usuário que executou esse comando.

5.1.2 – Módulo Analyzer

Munido das informações sobre os processos, coletadas pelo módulo *scanner*, essas são verificadas nas ACLs e, posteriormente, com a base de assinaturas. Tais ações estão ilustradas na Figura 5.

Uma vez que esses dados analisados coincidam com uma regra da ACL ou com uma assinatura, o evento correspondente pode ser classificado como intrusivo ou suspeito, devendo então ser tomada a ação correspondente ao respectivo campo na tabela da ACL. Por exemplo, se a consulta à ACL retornar um *DENY*, como ação, um sinal *SIGKILL*[28] é enviado ao PID correspondente ao processo que gerou essa ação, para que o mesmo tenha sua execução finaliza-

27 Socket – Dispositivo de comunicação entre processo, seja no computador local, como computadores remotos.

28 SIGKILL – Sinal informando ao kernel para *matar* a execução de determinado processo.

da. Essa ocorrência é notificada ao administrador de sistemas, através de uma mensagem enviada, pelo agente, à console central.

As ações do Analyzer são definidas pela ACL, podendo este parar um processo ou notificar o administador da rede através da console central, repassando as informações do processo em questão. Feito isso, o administrador de redes pode analisar o problema e tomar a devida ação para o mesmo.

5.1.3 – ACLs

Como mencionado, este é o enfoque principal da abordagem. Uma ACL armazena informações referentes aos arquivos e diretórios, assim com as ações correspondentes a cada um deles. Essas ações definem como o agente deve proceder mediante a ocorrência de uma ação intrusiva ou suspeita no sistema.

5.1.3.1 – Controle de arquivos e diretórios

Segue, abaixo, uma tabela exemplificando a construção da ACL de arquivos e diretórios.

Tabela 3 – ACL de acesso à arquivos

Acesso	*Ação*
/etc/shadow	DENY
/etc/lilo.conf	DENY
/etc/inetd.conf	NOTIFY
/etc/passwd	NOTIFY

Como podemos perceber, na tabela, no caso de algum usuário tentar acessar diretamente, copiar ou listar, o arquivo de senhas *shadow*[29], por exemplo, terá como resposta uma ação DENY.

29 shadow – Armazena as senhas criptografadas dos usuários cadastrados no passwd.

Essa regra pode ser aplicar a outros arquivos considerados importantes, tais como arquivos de configurações, o próprio *passwd*[30]. Este, por exemplo, pode conter informações sobre o usuário.

5.1.3.2 – Controle de Serviços

Esta lista deve conter os serviços configurados por padrão pelo administrador do sistema. A intensão dessa lista é armazenar todos os serviços que devem estar executando no referido *host*, de forma que qualquer outro serviço que seja executado possa ser tratado como atividade suspeita. Caso seja necessário a execução de algum serviço adicional, cabe ao usuário administrador atualizar essa lista.

Dessa forma, se um serviço for inicializado e não constar nessa lista, o agente deverá tomar a ação de notificar o administrador de sistemas, enviando uma mensagem à console de gerenciamento, relatando o serviço que foi inicializado.

Segue, abaixo uma tabela com exemplos de serviços e portas, respectivamente.

Tabela 4 – Descrição de serviços e portas relacionadas.

Serviço	***Porta***
sshd	22
httpd	80
ftpd	21
smtpd	25

30 passwd – Cadastro dos usuários no sistema Linux.

5.1.4 – Assinaturas

A análise de assinaturas consiste no mecanismo mais utilizado para detecção de intrusos em sistemas baseados em *host* (HIDS). Essas assinaturas são definidas mediante padrões de comportamento pré–definidos pelo administrador de sistema. Existem vários tipos de assinaturas como, eventos simples, eventos múltiplos, *hosts* múltiplos, entre outros.

5.1.4.1 – Eventos Simples

Caracteriza–se por ações simples, que podem indicar atividades suspeitas. Uma vez que a maioria dessas atividades suspeitas podem ser verificadas através de eventos simples, esse tipo responde por noventa porcento das assinaturas para HIDS.

Um exemplo a ser citado é **escrita em arquivos executáveis**. Esse tipo de ação pode ocorrer por atividades do próprio sistema, como uma atualização, por exemplo, porém, ações como um potencial atacante que esteja tentando colocar um *trojan*[31] no sistema ou uma infecção de arquivos executáveis por vírus podem ser detectadas através desse padrão.

5.1.4.2 – Múltiplos Eventos

Essa assinatura baseia–se na verificação de uma sequência contendo dois um mais eventos, assim como em um conjunto de transições entre esses eventos. Representa um número bem menor de assinaturas se comparadas às assinaturas por eventos simples.

O exemplo mais comum desse modelo são as **três falhas de login**. Essa assinatura em questão, pode gerar um falso–positivo, pois o usuário pode, no

31 Trojan – Programa malicioso utilizado por um hacker, para obter acesso à *hosts* remotos.

momento, ter esquecido ou confundido com outras senhas, e falhar em três tentativas de logon. Mas, por outro lado, pode indicar e previnir tentativas de força–bruta para descobrir senhas de usuários, o que normalmente é feito através do uso de ferramentas que fazem sucessivas tentativas de conexão FTP, Telnet ou SSH ao *host* alvo.

5.1.5 – Comportamento do agente

Na Figura 5, podemos visualizar o comportamento do agente instalado no *host*. O módulo scanner monitora os eventos do sistema e os repassa ao módulo analyzer, o qual verifica as regras de ACL e a base de assinaturas, reagindo de acordo com o retorno da consulta, seja reagindo à ação suspeita ou informando à console central sobre o ocorrido.

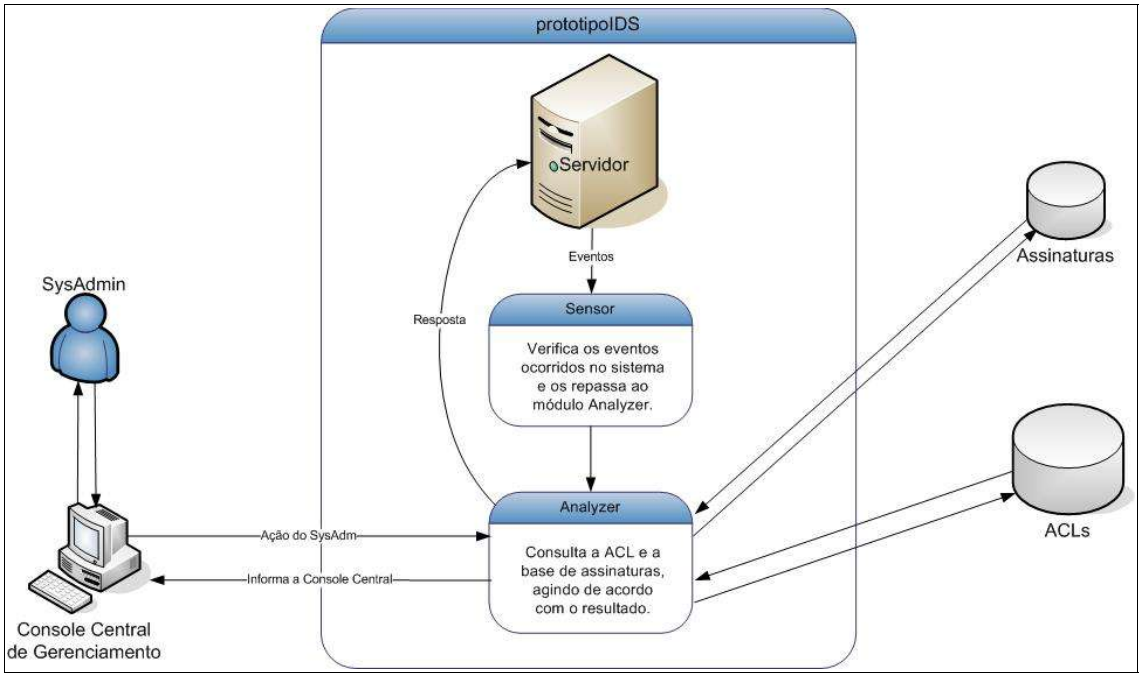


Figura 5 – Comportamento do agente.

5.2 – Console Central de Gerenciamento

O protótipo da console central de gerenciamento, constitui–se de uma interface onde o administrador de sistemas e de redes tem acesso às informações provenientes dos agentes remotos.

Toda a mensagem recebida de um agente é armazenada no banco de dados, gerando uma base de eventos ocorridos. Nessa base também são armazenadas as informações referentes ao *host* que originou a mensagem, horários e as ações tomadas pelo administrador de sistemas.

No monitor da console são visualizadas as informações enviadas pelos agentes. Casa uma seja selecionada, surge no display aquelas informações específicas, bem como uma lista de opções de ações a serem tomadas. Selecionando uma dessas opções o administrador de sistemas pode mandar uma mensagem para que o agente envie um sinal *SIGKILL* para um processo em execução, ou crie uma nova regra de *firewall*, por exemplo: negando acesso à porta do serviço *SSH*[32].

Após o administrador de sistemas efetuar uma determinada ação, o campo correspondente à ação tomada é atualizado no banco de dados. Dessa forma, além de uma base com ocorrências, temos também as ações ocorridas, gerando assim uma boa base para consultas de histórico e para geração de relatórios.

Figura 6 – Comportamento da Console Central.

5.3 – Experimento

Foi utilizado um ambiente de teste constituido de um servidor alvo, onde o agente está instalado, duas estações, sendo uma para a Console de Ge–

32 SSH – Protocolo para comunicação segura entre *hosts* remotos.

renciamento, e dois usuários de teste. Como a concepção do modelo proposto é bem abrangente, os resultados obtidos foram pelas informações dos eventos foram registrados na forma de arquivo de *log*, onde registrou-se as ações suspeitas efetuadas pelos usuários de teste, ações essas baseadas na aplicação da ACL.

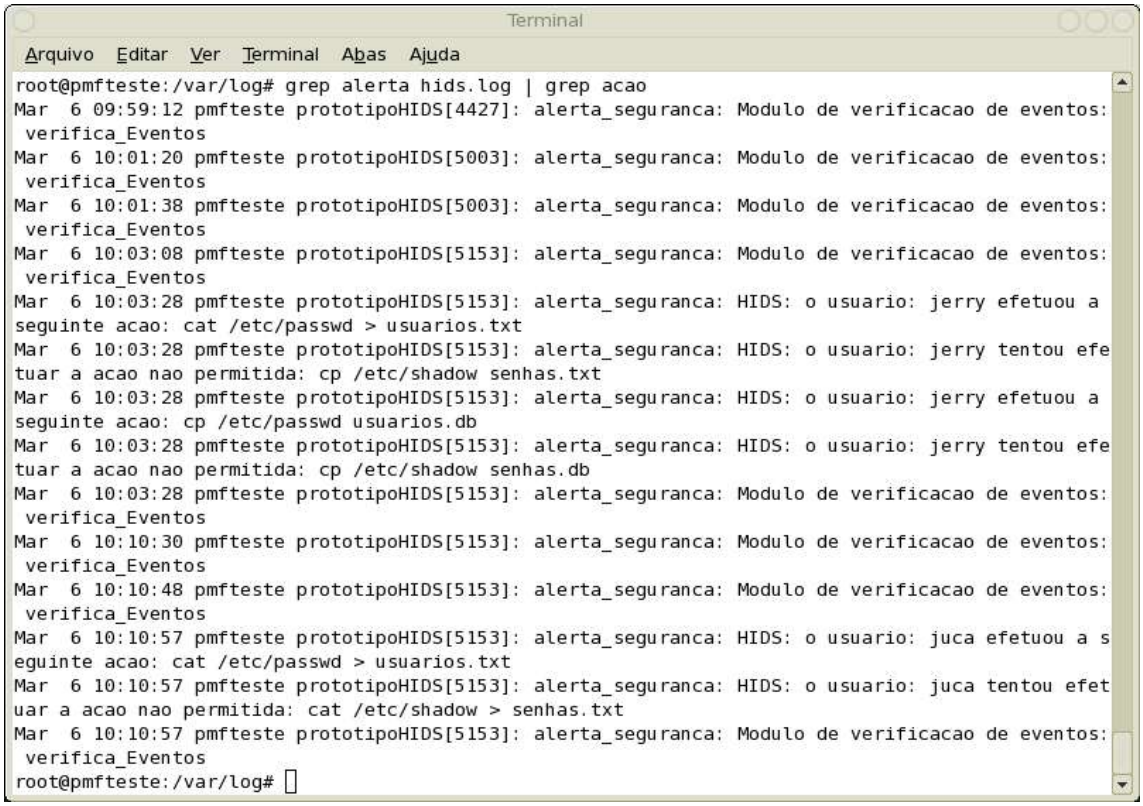


Figura 7 – ProtótipoHIDS com ACL

Para o experimento com assinaturas, foi utilizado o modelo de assinatura que reconheçe as três tentantivas de login, que pode ser verificado com maior facilidade nos arquivos de log do sistema, conforem a consulta feita no arquivo de log messages na Figura 8.

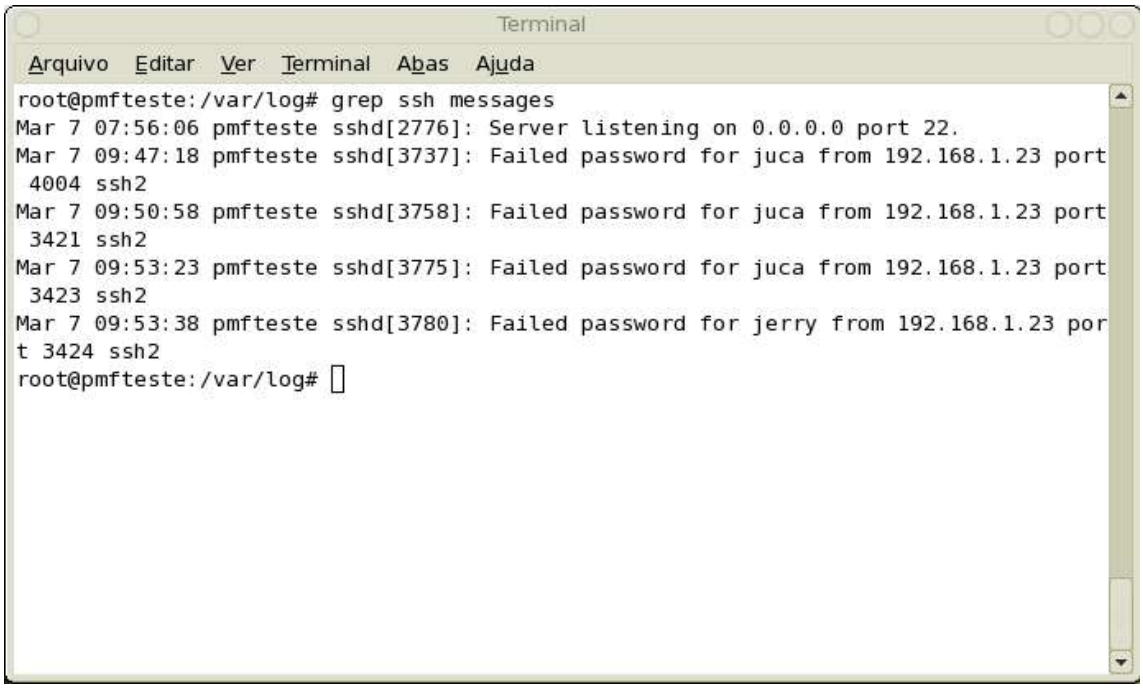


Figura 8 – Log do sistema demonstrando falha de SSH

Na Figura 8 pode–se verificar, através do log gerado pelo prototipoHIDS, que o agente registrou as tentativas falhas de login do usuário. As falhas foram provocadas com a finalidade de testar essa funcionalidade no módulo de verificação de eventos do agente. Percebe–se que o usuário “juca” efetuou três tentativas de conexão SSH sem sucesso, podendo ser classificado como atividade suspeita, enquanto o usuário “jerry” efetuou apenas uma fallha de login, o que não o enquadra na assinatura referida.

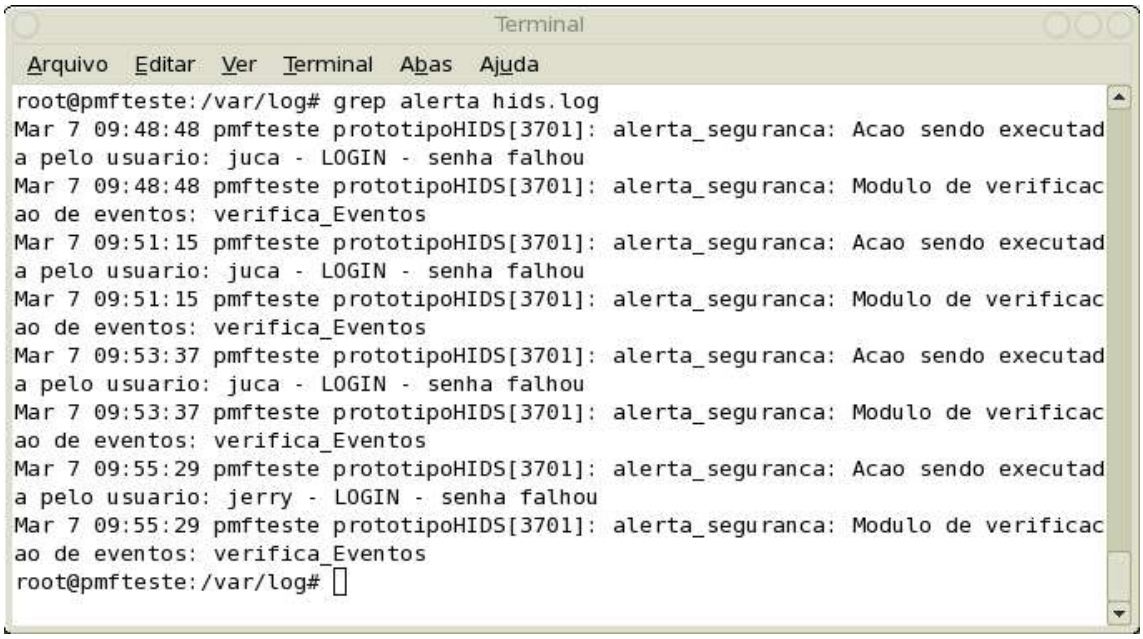


Figura 9 – Log falha de logon

6 – CONCLUSÃO

A necessidade de proteção da informação é cada vez maior, tanto em empresas quanto nos computadores particulares, principalmente com a descoberta das facilidades proporcionadas pela Internet.

A abordagem proposta enfatiza o uso de ACLs como uma forma de agilizar a detecção de *mau uso* do sistema, através da construção de uma política de acesso dos usuários onde os ponto mais críticos são enfocados, ou seja, aqueles que realmente podem vir a apresentar problemas, como o acesso a determinados arquivos do diretório */etc*[33] ou execução de serviços que abram portas de conexão não autorizadas.

Um HIDS não pode ter um número muito grande de ações, mediante algum alarme, pois algum desses alarmes pode ser uma tarefa real e necessária para o usuário, e que ainda poderia não fazer parte do perfil de comportamento do usuário.

6.1 – Porque essa conclusão?

Embasado no fato de que o HIDS é a última linha de defesa em um ambiente de rede corporativa, onde o *mau uso* por parte de algum usuário pode causar danos, a definição de regras para controle de quais diretórios e arquivos não podem ser acessados por usuários comuns, é de grande valia para detecção de *mau uso* do sistema. Exemplificando, um usuário comum do sistema não tem motivo nenhum para querer visualizar ou copiar o conteúdo dos arquivos *passwd* e *shadow*.

33 /etc – Diretório padrão onde são armazenadas as configurações de um sistema UNIX/Linux.

6.2 – Implementações Futuras

Como proposta para implementações futuras, podemos citar a evolução do agente para possibilitar a coleta de dados através dos arquivos de *log*, gerados por mensagens do kernel do Linux ao serviço *syslog*, em tempo de execução, proporcionando um maior enfoque no uso de *assinaturas* como base comparativa, uma vez que alguns padrões podem ser reconhecidos nos *logs*.

A verificação, em tempo–real, dos arquivos de *log wtmp* e *utmp*, que armazenam os usuários que efetuam *logon* no sistema, para agilizar o processo comparativo, já que os usuários a serem monitorados serão conhecidos nessa situação.

Verificar possibilidade de integração com bases de assinaturas mais populares e renomadas, como o CVE (Common Vulnerabilities Consortium) e BUGTRAQ.

Implementação e utilização de MIB (Management Information Base), destinada a fornecer informações dos eventos utilizando SNMP (Simple Network Management Protocol).

Otimização para melhorar o gerenciamento na utilização dos recursos do sistema, para que possibilite seu funcionamento de forma eficiente e com baixo consumo de CPU e memória.

7 – REFERÊNCIAS BIBLIOGRÁFICAS

BACE, Rebeca **An Introdution to Intrusion Detection & Assessment**, ICSA Inc.

BACE, Rebecca; MELL, Peter. **Intrusion Detection Systems**, National Institute of Standards Technology, Disponível na Internet via www.url: http://csrc.ncsl.nist.gov/pub/nistpubs/800-31/sp800-31.pdf, arquivo capturado em outubro de 2002.

BROOK, Jon-Michael C. **Intrusion Detection Systems, Network IDS: To Tailor, or Not to Tailor**, Disponível na Internet via www.url:http://verificar.na.internet.com SANS Institute, 2002.

BRUCE, Glen; DEMPSEY, Rob. **Security in Distrubuted Computing**, Prentice Hall, 1997.

CANSIAN, Adriano Mauro, **ACME – Advanced Counter-Measures Environment**, Disponível na Internet via www.url: http://www.acme-ids.org/, arquivo capturado em abril de 2002.

CARR, Jim, Good News/Bad News in DoS Struggle, **Network Magazine**, URL: http://www.networkmagazine.com/shared/article/showArticle.jhtml;jsessionid=BF0UVZ1GW43O4QSNDBGCKHSCJUMEKJVN?articleId=8703386&pgno=1, julho, 2002.

ELSON, David Del. **Intrusion Detection on Linux**, URL: http://www.securityfocus.com/infocus/1416, SecurityFocus, 2000.

ENTERASYS, **Enterasys Intrusion Detection**, Disponível na Internet via www.url: http://www.enterasys.com/ids, arquivo capturado em novembro de 2002.

GARFINKEL, Simson; SPAFFORD, Gene. **Practical UNIX and Internet Security**, O ´Reilly, 1996.

GOELDENITZ, Thomas. **IDS – Today and Tomorrow**, 2002.

HATCH, Brian; LEE, James; KURTZ, George. **Hackers Linux Expostos – Segredos e Soluções para a Segurança do Linux**, Makron Books, 2002.

ILGUN, Koral. **USTAT – A Real–Time Intrusion Detection System for UNIX**, University of Califórnia, 1992.

KIM, Gene H.; SPAFFORD, Eugene H. **The Design and Implementation of Tripwire: A File System Integrity Checker**, Purdue University, 1995.

LAING, Brian. **How To Guide – Implementing a Network Based Intrusion De–tection Systems**, Internet Security Systems, 2000.

LIDS, **Linux Intrusion Detection System**, URL: http://www.lids.org, arquivo capturado em novembro de 2001.

LOGCHECK, LogCheck, URL: http://logcheck.org, arquivo capturado em setembro de 2002.

MOURANI, Gerhard. **Securing and Optimizing Red Hat Linux**, versão 1.2, 2000.

NAGIOS, **Nagios**, URL: http://www.nagios.org, arquivo capturado em dezembro de 2002.

NORTHCUTT, Stephen; NOVAK, Judy. **Network Intrusion Detection – Third Edition**, New Riders, 2002.

SENTRYTOOLS, **PortSentry**, URL: http://sourceforge.net/projects/sentrytools/, arquivo capturado em setembro de 2002.

PROCTOR, Paul E. **The Practical Intrusion Detection Handbook**, Prentice Hall PTR, 2001.

PSIONIC, **Ferramentas de Monitoramento**, URL: http://www.psionic.com, aquivo capturado em novembro de 2001.

PTACEK, Thomas H.; NEWSHAM, Timothy N. **Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection**. 1998.

SCHNEIER, Bruce. **Segurança.com: segredos e mentiras sobre a proteção na vida digital**, Rio de Janeiro, Campus, 2001.

SNORT, **The Open Source Network Intrusion Detection System**, URL: http://www.snort.org, arquivo capturado em novembro de 2001.

SUNDARAM, Aurobindo. **An Introduction to Intrusion Detection**. URL: http://www.acm.org/crossroads/xrds2-4/intrus.html, janeiro, 2001.

SYMANTEC, White Paper, Symantec, **Detecção de Intrusão em Toda a Empresa – Uma abordagem multi-nível**, Symantec, 2001.

TRIPWIRE, **Tripwire Open Source Project**, URL: http://www.tripwire.org, arquivo capturado em agosto de 2002.

ZAMBONI, Diego; SPAFFORD, Eugene H. **Intrusion detection using autonomous agents**, Purdue University, 2000.