Laboratório de Redes de Alta Velocidade
COPPE/UFRJ

Sistemas de Detecção de Intrusão
Seminários Ravel - CPS760

Aluno: André S. Barbosa
andre@ravel.ufrj.br
Orientador: Luis Felipe M. de Moraes
moraes@ravel.ufrj.br

Índice

Capítulo 1

Introdução

1.1 Introdução

Baseado nas informações de várias entidades de pesquisa em segurança tais como CERT[10] e ICSA[11] podemos afirmar que um sistema foi atacado ou invadido mais que uma vez por segundo no último ano (1999), só nos Estados Unidos a ICSA identificou que, em média, um site ou computador foi invadido a cada 11 minutos.

Estas estatísticas nos levam a desejar uma enorme necessidade de poder rastrear e identificar estes ataques. O sistema que possui a capacidade de fazer isso é um IDS, ou Sistema de Detecção de Intrusão.

Estamos definindo o termo *ataque* conforme a RFC 2828, onde ataque é uma ação inteligente que ameaça a segurança de um sistema, um ataque pode ter sucesso ou não e estará explorando uma vulnerabilidade no sistema alvo ou inerente aos protocolos. Um ataque bem sucedido pode caracterizar uma invasão ou até mesmo a negação de serviços no sistema alvo (DoS - *Denial of Service*).

No capítulo 2 são apresentados conceitos e definições envolvidas no estudo de sistemas de detecção de intrusão, e no seguinte classificamos os IDS quanto a sua forma de funcionamento, tecnologia empregada e sistema a ser monitorado, dentre as tecnologias empregadas a mais utilizada é a análise de assinaturas, que é o assunto do capítulo 5. O capítulo 4 apresenta algumas tentativas de padronização, muitas delas já adotadas por fabricantes ou desenvolvedores de software gratuito. O capítilo 6 explora a questão das vulnerabilidades que podem ocorrer em sistemas de detecção de intrusão. O capítulo 7 propõe algumas soluções para os problemas encontrados em IDS, abordando também algumas características já encontradas em sistemas comercias. No capítulo 8 são sugeridos alguns temas para pesquisas ou projetos. O capítulo 9 apresenta a implementação de um sistema básico de gerência para um IDS, o SnortWAT[16] (*Snort Web Administration Tool*), este sistema permite a gerência de um sensor Snort, através da Web, com sessões encriptadas usando SSL[17], com algumas das suas principais funcionalidades já implementadas. Nos capítulos 10 e 11 temos respectivamente as conclusões e as referências.

Ao longo do texto o acrônimo IDS (*Intrusion Detection System*) está sendo usado tanto no singular quanto no plural para identificar Sistema(s) de Deteção de Intrusão, o acrônimo em inglês foi escolhido por ser de uso corrente na literatura brasileira e estrangeira, além disso estaremos usando IDS para caracterizar sistemas baseados em rede ou em host, para os IDS baseados em rede utiliza-se também o acrônimo NIDS (*Network Intrusion Detection System*).

Capítulo 2

Definições e conceitos básicos

2.1 Definições

Antes de mais nada iremos fazer definições de alguns dos elementos envolvidos no estudo de sistemas de detecção de instrusão.

1. Detecção de Intrusão: *O processo de identificar e relatar atividade maliciosa agindo em computadores e recursos da rede*
2. Sistema de detecção de intrusão: *Sistema composto de hardware e software que trabalham juntos para identificar eventos inesperados que podem indicar que um ataque irá acontecer, está acontecendo ou aconteceu.*
3. Ataque: *É uma ação inteligente que ameaça a segurança de um sistema, um ataque pode ter sucesso ou não e estará explorando uma vulnerabilidade no sistema alvo ou inerente aos protocolos. Um ataque bem sucedido pode caracterizar uma invasão ou até mesmo a negação de serviços no sistema alvo.*
4. Vulnerabilidade: *É uma falha no sistema operacional, protocolo, serviços ou quaisquer outros componentes no sistema que permitem acesso ou intervenção de pessoas não autorizadas. A vulnerabilidade existe independente do ataque, ela não depende do tempo de observação.*
5. Sensor: *Agente principal de um IDS cuja função é monitorar um host ou rede a fim de identificar intrusões, gravar logs localmente e gerar mensagens alertando tais eventos, estas mensagens podem ou não serem enviadas a uma estação de gerenciamento.*
6. Estação de gerenciamento: *É uma estação encarregada de administrar um ou mais sensores espalhados pela rede, o software utilizado deve term uma interface gráfica que permita configuração e monitoração dos agentes (Sensores IDS).*
7. Evento: *Ocorrência na fonte de dados que é detectada pelo sensor, a qual pode resultar num alerta sendo transmitido ou gravado.*
8. Respostas ou contramedidas: *São ações que podem ser programadas na ocorrência de um determinado evento. Exemplos são: um aviso por e-mail, o fechamento da sessão que gerou o evento, o bloqueio de um usuário ou até a reconfiguração de um filtro de pacotes ou firewall.*
9. Assinatura: *É a regra usada pelo analisador de eventos (sensor) para identificar os tipos de atividade suspeita, o mecanismo de análise de assinaturas é o mais utilizado pelos IDS.*

O analisador de eventos é na verdade uma parte do sensor IDS e será estudado posteriormente e pode funcionar com outras tecnologias, como por exemplo: redes neurais.

2.2 Necessidade de um IDS seguro

IDS são alvos lógicos de ataque. Se um *hacker*[1] souber ou suspeitar da existência de um IDS numa rede de seu interesse, a primeira coisa que faria seria atacar a máquina na qual o IDS está, para desabilitá-lo ou reconfigurá-lo para que não identifique suas ações, se isto não fosse possível ele poderia utilizar técnicas de construção de pacotes para que o IDS não consiga identificar corretamente suas ações, falaremos mais sobre estas questões no capítulo 6.

As empresas que comercializam softwares em geral mantém o código fonte secreto, isto é um assunto muito controverso pois você deve confiar plenamente na empresa que desenvolveu o produto, ou seja, você sabe o que o software faz, mas não sabe se ele o faz bem, e muitas das vezes não sabe de coisas estranhas que ele possa fazer... Com software livre, além do código ter a vantagem de ser revisado por milhares de pessoas, você mesmo pode alterar, melhorar, adicionar funcionalidades, etc. por outro lado a descoberta de uma falha torna-se mais fácil. A questão é: O que é melhor? segurança por obscuridade ou não? Isso na maioria dos casos pode depender de conhecimento, tempo e dinheiro.

Devido a estes fatores, a escolha, instalação e configuração de um IDS deve ser feita com o máximo de cuidado, estudando diversas possibilidades que propiciem flexibilidade e segurança associadas, e esta é uma tarefa das mais difíceis.

2.3 Falsos positivos e falsos negativos

Um dos maiores problemas dos IDS hoje em dia, não contando com a questão das vulnerabilidades, são os problemas com falsos positivos. Falsos positivos ocorrem quando pacotes normais são identificados como tentativas de ataque, podem ocorrer tantos falsos positivos que irão até mesmo atrapalhar a análise de um arquivo de log por um administrador experiente, levando-o a não perceber ataques verdadeiros.

Dois requisitos são necessárias então: Primeiro o IDS deve ser bem configurado após a instalação para se adaptar às características da rede ou máquina a ser monitorada. Segundo: O IDS deve ter um bom sistema de gerenciamento que permita sua configuração e a análise dos logs de forma simples e amigável.

Os falsos negativos ocorrem quando os IDS deixam de alertar tentativas autênticas de ataques, este é um problema muito pior. No caso do IDS usar análise de assinaturas, o requisito aqui é que as assinaturas sejam atualizadas constantemente.

Podem ocorrem também que pacotes com determinados tipos de ataques sejam interpretados como de outro tipo, este caso exigiria que o mecanismo de análise de eventos pudesse contar com formas de reforço para poder alertar corretamente os ataques, estes reforços poderiam vir de outras heurísticas implementadas no analisador ou mesmo de outros sensores na rede.

---

1. O termo *hacker* foi utilizado neste contexto pois a maioria das pessoas e a mídia convencionaram que eles são os responsáveis por atividades maliciosas e que visam degradação ou roubo em sistemas. Na verdade o termo correto para estes é *cracker*. Um *hacker* é uma pessoa que é movida pelo desafio e interesse em aprender questões relacionadas a segurança de redes, sistemas operacionais, etc. São geralmente programadores, e são os responsáveis pelas maiores evoluções na área tecnológica.

Capítulo 3

Tipos de IDS

3.1 Classificação de IDS

3.1.1 Quanto ao momento do ataque

Podemos caracterizar os IDS quanto ao momento detecção de um ataque, o IDS pode enviar um alerta:

- Antes que um ataque aconteça;
- Quando um ataque está acontecendo;
- Depois que um ataque aconteceu;

Naturalmente podemos ter produtos que possuem mais de uma característica.

3.1.2 Quanto a tecnologia do analisador de eventos

Podemos também caracterizá-los quanto a tecnologia empregada no analisador de eventos, um IDS pode trabalhar com:

- Análise de assinaturas;
- Análise estatística;
- Sistemas adaptativos;

O mecanismo de análise de assinatura funciona de forma similar a um anti-vírus e por ser o método mais utilizado iremos estudá-lo com mais detalhes no capítulo 5. Um sistema de análise estatística constrói modelos estatísticos do ambiente se baseando em fatores tais como duração média de uma sessão de telnet por exemplo, qualquer desvio de um comportamento normal pode ser identificado como suspeito. Um sistema adaptativo começa por generalizar regras de aprendizado para o ambiente que está inserido e então determinar o comportamento dos usuários com o sistema, depois do período de aprendizado o sistema pode reconhecer determinados padrões como sendo acessos normais ou ataques, uma rede neural pode ser uma escolha natural para tais sistemas. Novamente podemos ter produtos que possuem mais de uma característica.

3.1.3 Quanto ao sistema que está monitorando ou agindo

As duas formas de classificação vistas anteriormente não visam separá-los em grupos, mas sim determinar sua forma de funcionamento, e podem ajudar como critérios para a escolha de um produto adequado.

A divisão clássica é quanto ao sistema que o IDS está inserido e monitorando, desta forma podemos ter:

- IDS baseados em rede (NIDS)
- IDS baseados em host
- Verificador de integridade de arquivos

Esta classificação foi baseada na sugerida pela ICSA[11].

Vamos examinar cada tipo separadamente, lembrando que produtos atuais já estão suportando algumas funções em comum, ou seja, IDS baseados em redes fazem algumas verificações no host onde estão instalados, IDS baseados em host fazem algum tipo de monitoramento na rede, e tanto os baseados em rede como os baseados em host podem já ter embutidos verificadores de integridade de arquivos.

3.2 IDS baseados em rede (NIDS)

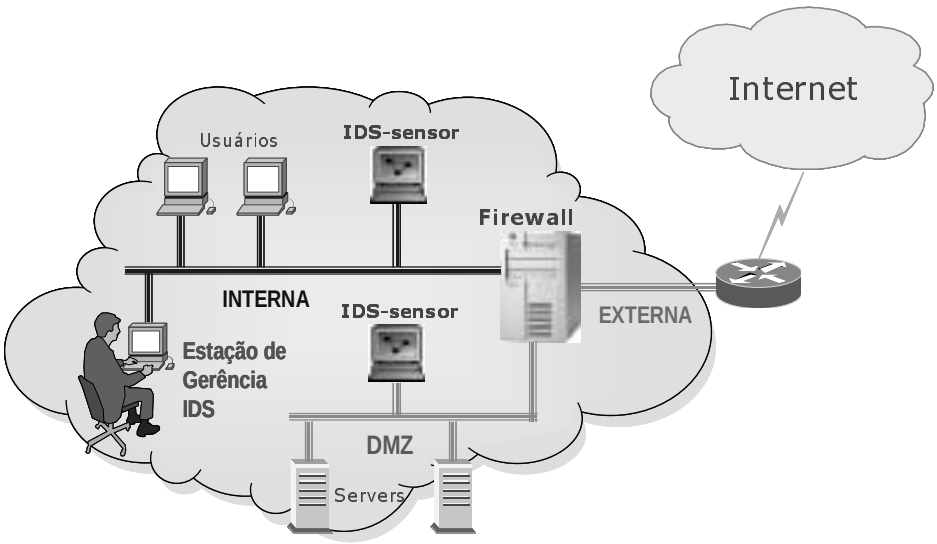
Os NIDS possuem geralmente dois componentes:

Sensores: São colocados em seguimentos de rede distintos, nos quais se deseja monitorar. Não entramos em detalhes por enquanto o uso de hubs ou switches.

Estação de gerenciamento: Possui uma interface gráfica e recebe os alarmes dos sensores informando ao operador.

Os sensores apesar de estarem instalados numa máquina específica monitoram todo o tráfego no seguimento de rede pois a interface com a rede é colocada em modo promíscuo, neste modo o software recebe todos os pacotes naquele seguimento de rede, não apenas os destinados ao IP da interface, por isso um único sensor tem a capacidade de monitorar atividade de rede em diversas máquinas.

FIGURA 1. IDS baseado em rede (NIDS)



3.2.1 Vantagens

- Detectam acessos com excesso de autoridade ou sem autoridade;
- Não necessitam de alteraçoes em servidores ou quaisquer outras máquinas;
- Não afeta diretamente o sistema onde está instalado;
- Tende a ser mais independente, tipo “soluçao caixa-preta”;

3.2.2 Desvantagens

- Visão localizada, monitora um segmento de rede, numa rede muito segmentada a quantidade de sensores será grande e isto se torna pior numa rede com switchs;
- Inadequado para tratar ataques mais complexos;
- Grande quantidade de dados podem trafegar entre os agentes e estações de gerência;
- Não consegue monitorar tráfego em sessões encriptadas;

Muitos softwares comerciais já diminuiram estas desvantagens;

3.3 IDS baseados em host

Os IDS baseados em host analisam sinais de intrusão na máquina nos quais estão instalados, eles frequentemente usam os mecanismos de log do sistema operacional e estão muito ligados aos recursos do sistema. Eles agem procurando por atividades não usuais em: tentativas de login, acesso à arquivos, alterações em privilégios do sistema, etc.

Ex.: Se algum usuário tentar usar o comando `su` do UNIX para tentar obter acesso como root então vamos gravar este evento, nome do usuário, hora, etc.

3.3.1 Vantagens

- Pode em várias circustâncias dizer exatamente o que o atacante fez;
- Menos falsos positivos do que IDS baseados em rede;
- Uso em ambientes onde largura de banda é crítica;
- Menor risco de fazer uma configuração errada;
- Mais difícil de ser enganado;

3.3.2 Desvantagens

- Depende das capacidades do sistema no qual está instalado;
- Requer instalação e possivelmente alteração no equipamento que se deseja monitorar;
- Necessita de maior atenção por parte dos administradores;
- São relativamente mais caros;
- Visão extremamente localizada, só monitora uma máquina;

3.4 Verificador de Integridade de Arquivos

São programas que examinam os arquivos num computador e determinam se eles foram alterados desde a última vez que o verificador rodou, eles são baseados em funções de hash. O processo consiste em gerar um hash para cada aquivo e armazenar estes hashes numa tabela contendo o nome do arquivo e seu respectivo hash, se após isso quaisquer dos arquivos que passaram por este processo forem alterados o seu hash será diferente e portanto o valor será diferente daquele armazenado, assim o programa poderá alertar o operador qual arquivo foi alterado.

Função de hash: *processo matemático que reduz uma sequência de bytes para uma cadeia de bits de comprimento fixo (ou não).*

Uma função muito utilizada é a MD5:

- Desenvolvida por Ron Rivest (o “R” do algoritmo RSA);
- A entrada é feita através de blocos de 512 bits;
- A saída é um hash fixo de 128 bits;
- Taxa do MD5 = 174 kbits/s num PC 486 33MHz
- Taxa do SHA = 75 kbits/s num PC 486 33MHz (outra função de hash, só que com saída de 160 bits)

Existem os utilitários em UNIX chamados `md5` ou `md5sum` que aceitam como entrada um arquivo e imprimem o hash do mesmo.

3.4.1 Vantagens

- É computacionalmente impraticável “derrotar” a matemática por trás de um verificador de integridade, a força bruta é mais fácil...
- São extremamente flexíveis, podem ser configurados para verificar quaisquer arquivos no sistema;
- Protege quanto aos *cavalos de tróia*, *backdors* e *linpeza de logs*[1];

3.4.2 Desvantagens

- Consome muitos recursos do sistema: CPU, memória e disco;
- Vulnerável à modificação pelo próprio atacante;
- Deve ser executado frequentemente e com isso irá reportar centenas de alterações se não for configurado corretamente. Ex.: Se todos os arquivos do sistema forem verificados existirão centenas de alterações feitas pelo proprio sistema operacional ou usuários legítimos, isso irá prejudicar a filtragem da informação;

1. Estes termos são parte do jargão hacker, cavalos de tróia são programas aplicados a outros programas normais fazendo com que a execução destes programas normais realizem outras ações, backdors são falhas colocadas escondidas nos sistemas invadidos que permitem acessos futuros com privilégios ao sistema, limpeza de logs ou encobrimento de rastros são precauções tomadas pelo invasor para que seus rastros não sejam descobertos, para isso ele necessita alterar ou remover aquivos de logs.

3.5 Implementações e seus fabricantes

Na tabela abaixo apresentamos os principais IDS, os fabricantes, o web site do fabricante e a classificação do produto.

FIGURA 2. IDS comerciais e sua classificação

Produto	Fabricante	Site	Tipo
Intruder Alert	Axent	www.axent.com	Host
NetProwler	Axent	www.axent.com	Rede
RealSecure	ISS	www.iss.net	Host/Rede
NetRanger	Cisco	www.cisco.com	Rede
NFR	NFR	www.nfr.net	Rede
SessionWall	C. Associates	www.ca.com	Rede
Snort (free)	Marty Roesch	www.snort.org	Rede
Abacus (free)	Psionic	www.psionic.com	Host

Capítulo 4

Padronizações

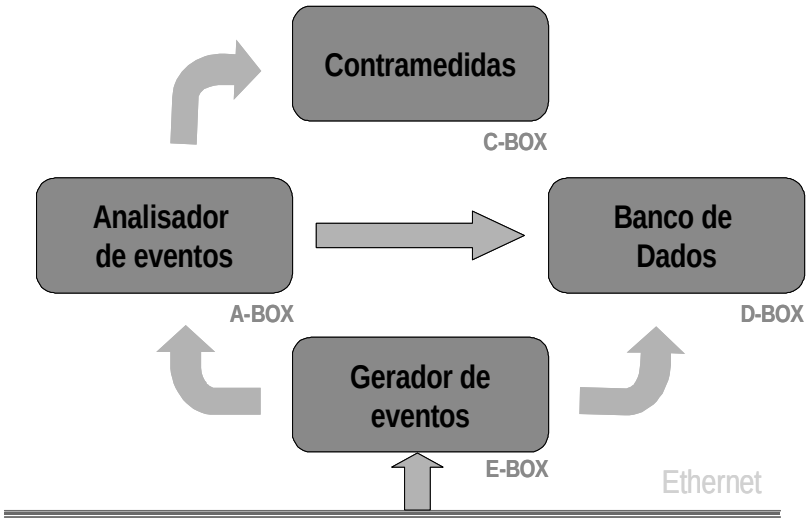
4.1 CIDF - Common Intrusion Detection Framework

O CIDF[14] é uma tentativa de padronização proposta (ainda em *draft*) por um grupo de voluntários de algumas empresas a fim de permitir interoperabilidade de IDS. É sugerido um modelo CIDF para projeto de um IDS e uma linguagem de especificação de eventos chamada CISL (*Common Intrusion Specification Language*). Esta padronização esta voltada para IDS baseados em rede.

O modelo para projeto de um IDS é composto dos seguintes componentes:

- E-Box – Gerador de eventos;
- A-Box – Analisador de eventos;
- D-Box – Banco de dados;
- C-Box – Contramedidas;

FIGURA 3. Interação dos componentes do modelo CIDF



O gerador de eventos captura todos os pacotes da rede (TCP, UDP, ICMP) e entrega ao analisador de eventos para que este aplique os critérios de identificação do ataque, estes dados são salvos num banco de dados, baseados em alguma forma de cache os dados podem ser automaticamente salvos do gerador para o banco de dados, o banco de dados contém todos os pacotes detectados como ataques. No momento da identificação o analisador de eventos pode disparar uma unidade de contramedidas.

4.1.1 Componentes do Modelo CIDF

Gerador de eventos

Geralmente a interface é colocada em modo promíscuo, no UNIX pode-se ter acesso facilitado através da biblioteca `libpcap` (biblioteca de captura de frames) que pode decodificar Ethernet, Token Ring, FDDI, PPP, SLIP e RAW e posteriormente IP, TCP, UDP e ICMP.

Analisador de eventos

É o cérebro do IDS. É o componente responsável por identificar o que e o que não é um ataque. A maioria dos fabricantes usam somente análise de assinaturas e alguns produtos estão apenas começando a utilizar análise estatística, os sistemas adaptativos são assunto de pesquisas atuais.

Banco de dados

Pode receber dados tanto do gerador de eventos quanto do analisador de eventos e além disso deve ter desempenho compatível com a aplicação sendo dimensionado de acordo com a capacidade da máquina e a capacidade do link a ser monitorado.

Contramedidas

É o componente responsável por tomar ações baseadas nos eventos, deve ter a capacidade de comunicar-se com outros IDS ou até um firewall, além disso outras ações podem ser sugeridas, não há limite.

- Pode agir sobre os servidores, desligando-os;
- Pode alertar através de email, bip, celular etc;
- Pode contra-atacar a origem do ataque;

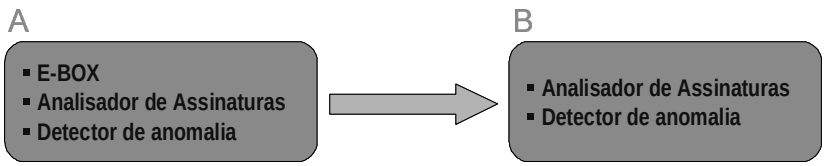
4.1.2 Interoperabilidade de IDS

São 6 os cenários nos quais os IDS podem interoperar:

Análise

Cada sensor representado na figura pode conter uma das tecnologias descritas nos tópicos no seu interior. O detector de anomalia pode ser um analisador estatístico ou sistema adaptativo. Quando em operação B tenta determinar se os eventos reportados por A estão relacionados de alguma forma, assim B decide se as informações de A são suficientes para gerar um alarme;

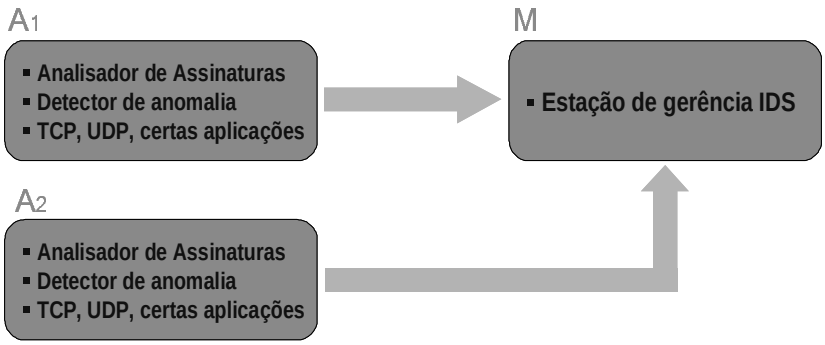
FIGURA 4. Interoperabilidade de IDS - Análise



Complemento

Esta configuração permite balanceamento de carga, A_1 e A_2 podem detectar diferentes tipos de ataque, ou analisar diferentes protocolos, podem também serem iguais mas em máquinas diferentes, em todo caso esta informação e envida em conjunto para uma estação de gerenciamento.

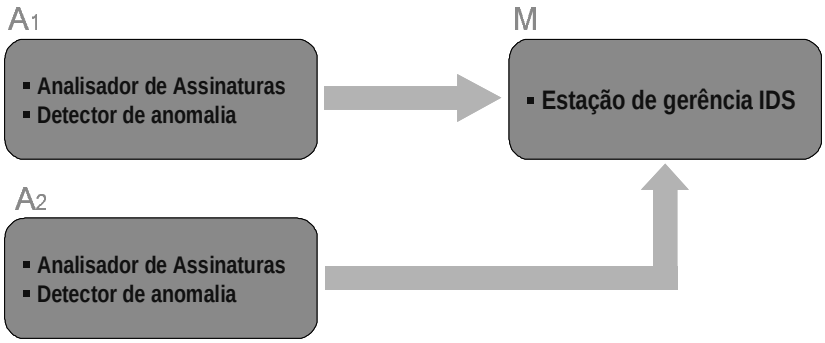
FIGURA 5. Interoperabilidade de IDS - Complemento



Reforço

Este caso possui uma sutil diferença em relação ao anterior, os dois sensores deverão analisar os mesmos tipos de dados, mas possuindo tecnologias diferentes, assim podemos supor que eles terão falsos positivos em tempos diferentes, e a estação M emitirá um alerta somente se os dois sensores indicarem um ataque.

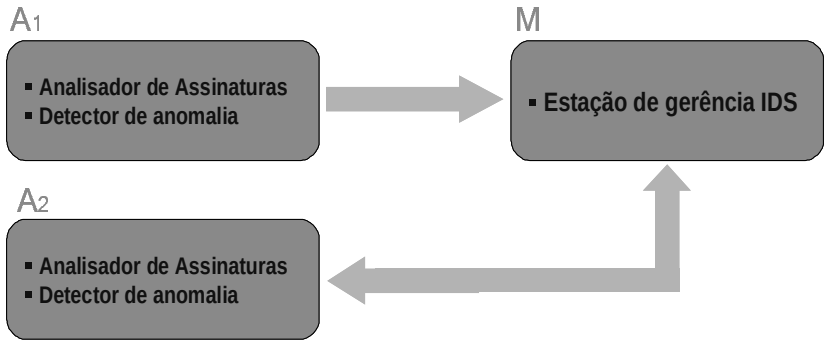
FIGURA 6. Interoperabilidade de IDS - Reforço



Verificação

Este caso é uma variação do anterior, sendo que desta vez, o segundo sensor só é consultado quando o primeiro reportar um evento, ou seja, depois que A_1 reporta o ataque, M "pergunta" a A_2 sobre sinais do mesmo ataque, isso irá reduzir a carga do segundo sensor fazendo com que perca menos pacotes;

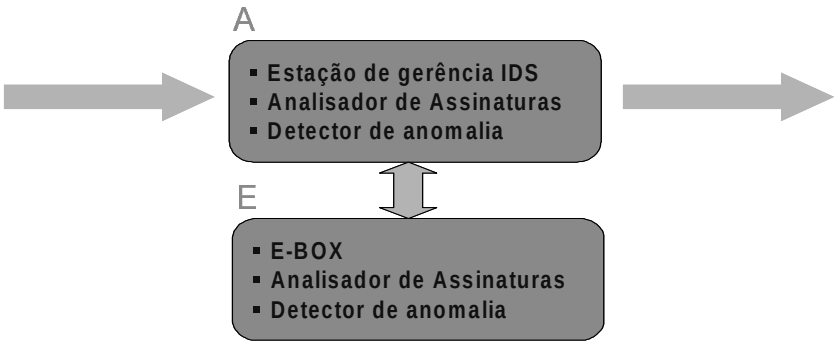
FIGURA 7. Interoperabilidade de IDS - Verificação



Ajuste de monitoração

Neste caso A envia instruções a E sobre o que monitorar, E recebe estas inormações, analisa e passa adiante, A tem o poder de reconfigurar o IDS.

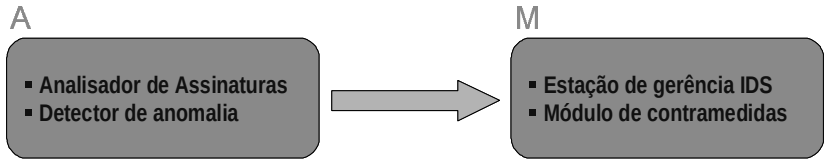
FIGURA 8. Interoperabilidade de IDS - Ajuste de monitoração



Resposta

M recebe informações de A e pode tomar ações automáticas, tais como: ajustar um firewall ou outros IDS; Agir sobre os servidores, desligando-os, por exemplo, alertar através de email, contra-atacar a origem do ataque, etc.

FIGURA 9. Interoperabilidade de IDS - Resposta



4.1.3 CISL - Common Intrusion Specification Language

A CISL é a tentativa de padronizar transferência de informações e interoperabilidade entre IDS, faz parte do CIDF e pode ser utilizada na comunicação dos sensores com a estação de gerenciamento, comunicação entre os sensores ou comunicação com firewall ou roteadores;

A CISL usa “S-expressions”, estas são expressões que agrupam recursivamente indicadores e dados.

- SID - Identificador Semântico (Semantic Identifier);
- Dado - Informação relacionada ao SID;

Exemplo:

```
(HostName ‘www.qualquersite.com.br’)
```

onde,

- HostName = SID;
- ‘www.qualquersite.com.br = Dado;

Através de um exemplo completo fica claro como a linguagem é definida:

4.1.4 Exemplo de especificação na CISL

```
(InSequence
    (Login
        (Context
            (Time ’14:57:36 24 Feb 2000’)
        )
        (Initiator
            (HostName ‘big.evil.com’)
        )
        (Account
            (Username ‘joao’)
            (RealName ‘Joao da Silva’)
            (Hostname ‘www.target.com)
            (ReferAs 0x12345678)
        )
    )
    (Delete
        (Context
            (Hostname ‘www.target.com’)
            (Time ’14:58:12 24 Feb 2000’)
        )
        (Initiator
            (ReferTo 0x12345678)
        )
```

```
        (Source
            (FileName (Extendedby UnixFullFileName) '/etc/passwd')
        )
    )
)
```

Interpretando o exemplo:

Primeiro alguem se conectou usando a conta de joao na máquina www.target.com vindo de big.evil.com, aproximadamente meio minuto depois, esta mesma pessoa apagou o arquivo /etc/passwd da máquina www.target.com.

4.2 IAP - Internet Intrusion Alert

O IAP foi proposto por um grupo do IETF[20] e é um protocolo a nível de aplicação para troca de dados entre agentes IDS. O IAP utiliza o TCP omo protocolo a nível de transporte é primariamente destinado a transmissão de dados do sensor/analisador para a estação de gerenciamento que informa a ocorrência, grava o evento e toma as determinadas contramedidas.

Uma biblioteca para o uso do IAP já existe e para citar um exemplo temos um pluguin para o Snort[13] chamado SnortNet[4] que integra varios sensores Snort criando uma estação de gerência que se comunica através do IAP.

Capítulo 5

Análise de assinaturas

5.1 Características

O mecanismo de análise de assinaturas é nada mais do que uma lista contendo uma assinatura de ataque e a respectivo alerta a ser enviado. A assinatura do ataque é construída com base nas características do pacote que contém o ataque, estas características podem ser:

- Portas de origem e destino;
- Números de sequência;
- Flags dos protocolo TCP, ex.: SYN, FIN, etc.
- Outros campos do protocolo e suas opções;
- E principalmente uma pequena parte do conteúdo da camada de aplicação do pacote;

Então fica claro que o mecansismo de análise de assinaturas age como um anti-vírus, analisando cada pacote contra o conjunto de regras criado. Este conjunto de regras, ou seja a assinatura e o alerta respectivo são armazenados em memória como uma lista encadeada, ou duplamente encadeada a fim de agilizar ao máximo o processo de verificação dos pacotes.

O banco de dados, que muita das vezes é um arquivo texto, contendo as regras deve ser atualizado frequentemente, assim como um anti-vírus.

Regras para IDS podem ser encontradas no site do Snort[13] e ArachNIDS[21].

5.2 Exemplos de regras para IDS

Dois exemplos de regra do Snort pode ser vistas abaixo:

```
alert tcp any any -> 10.1.1.0/24 80 (content: “/cgi-bin/phf”; msg:
“Este é um teste do bug PHF”;)

alert tcp any any -> 10.1.1.13 143 (content: “|E8C0 FFFF FF|/bin/sh”;
msg: “detectado buffer overflow IMAP”;)
```

A primeira irá gerar um alerta de um pacote vindo de qualquer máquina com qualquer porta de origem para a subrede 10.10.1.0 (classe C pois a máscara é 255.255.255.0) na porta 80 (httpd) cujo pacote contenha a string “/cgi-bin/phf”, este alerta irá gravar no log este ataque como “Este é um teste do bug PHF”, que é uma falha encontrada em algumas versões antigas do PHF quando instalado no diretório cgi-bin do servidor web.

O segundo exemplo irá gerar um alerta de um pacote vindo de qualquer máquina com qualquer porta de origem para a máquina 10.1.1.13 na porta 143 (imapd) cujo pacote contenha a string “|E8C0 FFFF FF|/bin/sh”, este alerta irá gravar no log este ataque como “Detectado buffer overflow IMAP”, que é uma falha encontrada em algumas versões antigas do servidor de e-mail IMAP que proporciona ao cracker se utilizar de um estouro de buffer no programa e executar comandos remotamente na máquina.

5.3 Tipos de logs e alertas (Snort):

O Snort pode gerar os seguintes tipos de logs baseados num pacote que foi detectado como um ataque pelo mecanismo de análise de assinaturas:

- Sem logs;
- Log em formato amigável: Os pacotes são arquivados em diretórios baseados no IP da máquina de origem;
- Estilo do tcpdump: Permite uma análise veloz dos dados coletados pelo sistema;

Os seguintes alertas são disponíveis:

- Sem alertas;
- Syslog: Podem ser enviados para o syslog;
- Alert.txt (1): Armazenado num arquivo texto (fast);
- Alert.txt (2): Armazenado num arquivo texto (full);
- WinPopup: Enviado para máquinas Windows, como mensagems através do Net-BIOS e podem gerar janelas informando diretamente em máquinas Windows.

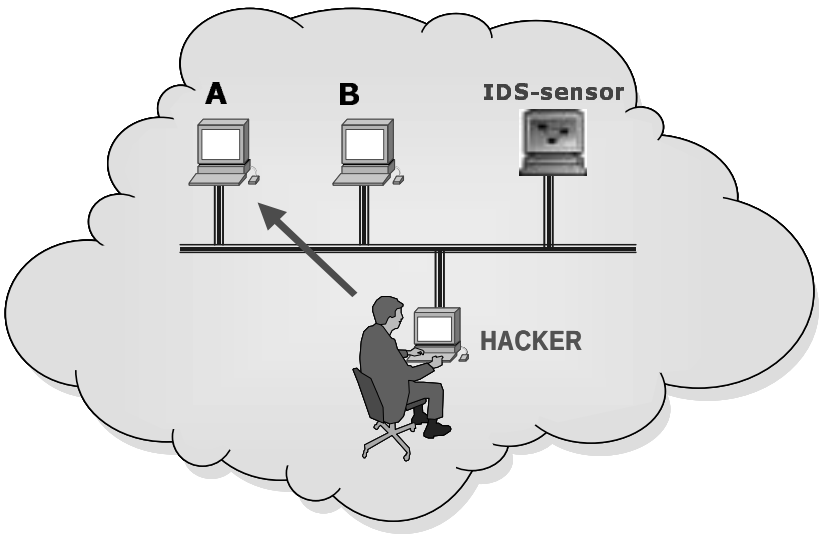
Capítulo 6

Vulnerabilidades

6.1 IP Spoofing

Esta é uma vulnerabilidade inerente ao protocolo TCP/IP, neste caso o atacante pode se fazer passar por outra máquina e assim enganar o IDS o qual não possui regras no seu banco de dados para detectar pacotes desta outra máquina.

FIGURA 10. Vulnerabilidads de IDS - IP Spoofing



Como mostra a figura o Sensor monitorando a rede com os máquinas A e B pode não ter regras para identificar ataques destas mesmas máquinas, então se o hacker consegue colocar pacotes na rede com endereço de origem destas máquinas ele consegue colocar pacotes que não sao detectados pelo IDS.

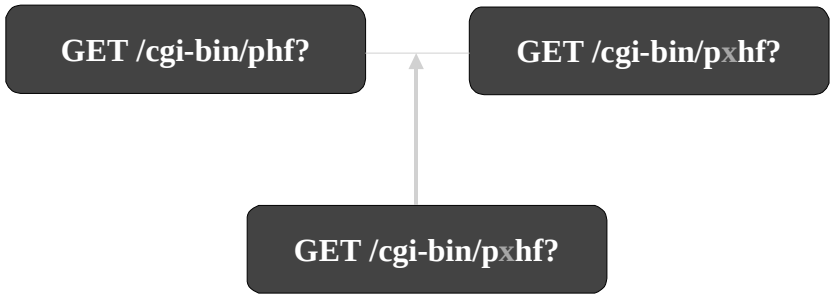
6.2 Inserção

Neste caso pacotes são propositalmente enviados a sistemas que irão rejeitá-los, mas estando o IDS configurado para analisá-los.

Desta forma, o hacker pode fazer com que o IDS aceite pacotes a mais na sequência de pacotes que ele envia, fazendo com que o mecanismo de análise de assinaturas seja enganado.

Na figura a seguir o hacker inclui um pacote especificamente construído contendo a letra *x*, este pacote é inserido na sequência de ataque de forma que o IDS detecte-o mas o sistema alvo não, assim o IDS é enganado e o ataque atinge o alvo com sucesso.

FIGURA 11. Vulnerabilidades de IDS - Inserção



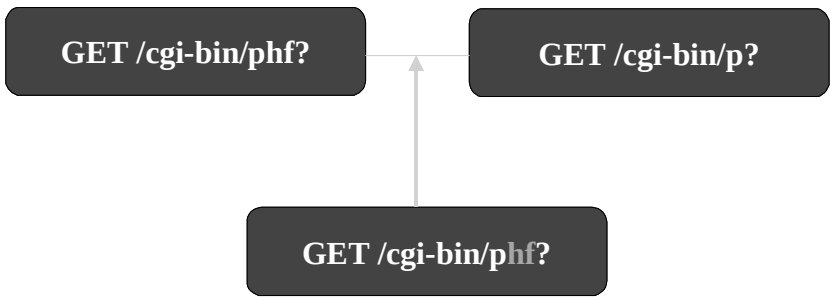
6.3 Evasão

Nesta modalidade pacotes são enviados ao sistemas (alvos) que irão aceitá-los, mas estando o IDS configurado para rejeitá-los, ataques deste tipo são realizados quando as regras na análise de assinaturas estão muito detalhadas (ou muito “amarradas”).

Este é o cao mais simples, onde pacotes são especificamente construídos para não serem detectados pelas regras do analisador de assinaturas.

No exemplo abaixo os pacotes contendo a string “hf” são criados a fim de que o IDS não os detecte corretamente.

FIGURA 12. Vulnerabilidades de IDS - Evasão



6.4 Denial of Service - DoS

Ataques de DoS (*Denial of Service*) ou DDoS (*Distributed Denial of Service*) são os mais perigos para IDS pois podem fazer com que o sistema não opere e portanto não detectem nenhuma intrusão.

Os ataques de DoS e DDoS visam consumir recursos do sistema operacional através de falhas existentes no próprio sistema ou em aplicativos nos mesmos. Podemos citar como recursos a serem explorados pelos ataque de DoS:

- Memória;
- CPU;
- Largura de banda;
- Espaço em disco;

Não existem meios eficazes de combater este tipo de ataque. Alguns IDS possuem mecanismos para detectar estes ataques, mas para combatê-los, bem, aí já é outra estória...

6.5 Defendendo-se contra os ataques aos IDS

Algumas regras podem ser estabelecidas no desenvolvimento de um IDS para que este seja menos vulnerável a tais ataques, são elas:

- Verificar todos os checksums;
- Examinar corretamente os cabeçalhos;
- Verificar existencia de flags, SYN, ACK, RST, etc.
- Realizar corretamente a remontagem de pacotes;
- IDS deve conhecer a topologia da rede!
- IDS deve conhecer os sistemas instalados!
- IDS deve conhecer a configuração dos sistemas!

Capítulo 7

Proposta de soluções aos problemas encontrados

7.1 Redes com Switch

IDS baseados em rede não funcionam em redes com switch.

Solução: Utilizar uma propriedade do Switch chamada port spam, com esta funcionalidade presente no equipamento pode-se configurar uma porta que irá receber pacotes de todas as outras, então o IDS deve ser colocado nesta porta. Deve-se observar que a capacidade desta porta do Switch bem como a robustez do IDS devem ser cuidadosamente analisadas.

7.2 Denial of Service no IDS

IDS são alvos lógicos de ataques e os ataques de DoS e DDoS são os mais perigosos, e não existem formas simples de combatê-los em sistemas em produção.

Solução: Para os IDS podemos tomar as seguintes precauções:

- Colocar IDS em modo Stealth - se o software tiver esta propriedade;
- Desabilitar IP e arp na interface se não tiver a propriedade Stealth;
- Modificar cabo de rede de forma a somente receber pacotes;
- Utilizar técnicas de redirecionamento de nomes no DNS;

O modo Stealth é conseguido desabilitando o IP e o arp na interface, a questão de modificação no cabo pode ser até paranóica, de qualquer forma a comunicação do IDS com outros agentes é sacrificada.

7.3 IDS em modo Stealth(Invisível)

IDS em modo Stealth não consegue se comunicar pela rede.

Solução: para resolver este problemas pode-se fazer esta comunicação através de outra sub-rede ou através de intrefaces seriais.

Uma outra solução seria fazer com que o IDS realiza-se uma configuração na interface desabilitando o modo Stealth em intervalos tempo a fim de trasnmitir informações para outros agentes.

Capítulo 8

Motivações para pesquisas

8.1 IDS e Redes Neurais

Redes neurais podem ser utilizadas como substituição ou reforço ao mecanismo de análise de assinaturas. Existem alguns estudos na área mas nenhum software com esta característica foi ainda lançado, ou é de comhecimento público.

As redes neurais poderiam utilizar técnicas como *backpropagation* para aprender os padrões nos pacotes na rede e então montar sistemas de reconhecimento quando padrões diferentes destes forem detectados.

8.2 IDS nativo em ATM

Sistemas de deteção de intrusão para redes ATM podem ser desenvolvidos, tanto para ATM nativo quanto para emulação de IP sobre ATM.

8.3 IDS + Detector de DoS

Podem se desenvolver IDS com características especiais para detecção de ataques de DoS e DDoS, analisando o comportamento do tráfego e a assinatura destes ataques.

8.4 IDS e Sistemas Distribuídos

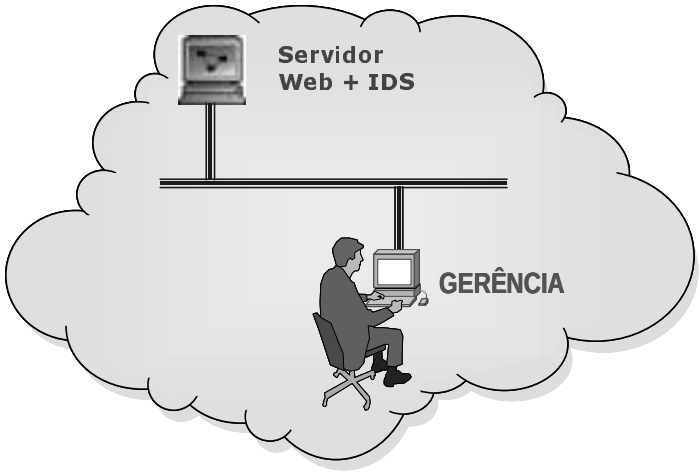
Sistemas de gerência de IDS podem ser contruídos tomando como base vários paradigmas de sistemas distribuídos, tais como:

- Sockets
- RPC
- CORBA
- Redes Ativas

8.5 Simples sistema de gerência de IDS

Um sistema simples de gerência de sensores IDS pode ser implementado através da Web, utilizando os browsers como estações de gerência, com comunicação encriptada usando SSL, conforme sugere a figura abaixo seguinte.

FIGURA 13. Sistema simples de gerência de IDS pela Web



O IDS poderia ser gerenciado através de um browser que se comunica por SSL com um servidor WeB, o servidor Web por sua fez aciona scripts que fazem a interface com o IDS instalado. O sistema implementado neste trabalho, que é assunto do capíuilo sequinte, utiliza em parte este conceito só que melhorias foram acrescentadas, principalmente o fato de se utilizar um servidor independente (Não precisa usar o servidor web existente), como veremos adiante.

Capítulo 9

SnortWAT - Sistema simples de gerência de IDS

9.1 Características

Dentre as principais características do sistema podemos citar:

- Não necessita de um Web-Server;
- Funciona sob o inetd numa porta específica;
- Utiliza o stunnel, provendo uma interface SSL para o programa, e portanto, encriptando as sessões;
- Utiliza o mecanismo de autenticação do HTTP;
- Arquitetura básica tem ampla utilização, até mesmo para configurar firewalls;

9.2 Componentes do sistema

- Browser - Internet Explorer ou Nestcape (testados);
- Stunnel - Programa (Wrapper) que faz a interface SSL com o SnortWAT;
- SnortWAT - Programa principal, escrito em Perl que é executado sobre o inetd no sistema operacional e faz a configuração e análise de logs do IDS;
- IDS - fui utilizado o Snort;

9.3 Funcionalidades

- Ativa e desativa o IDS;
- Exibe o arquivo de assinaturas (número de linhas);
- Exibe o log de todos os pacotes (número de linhas);
- Exibe o log por IP específico;
- Remove e compacta arquivos de logs;
- Resolve IPs e realiza traceroute nos mesmos;

9.4 Funcionalidades a serem implementadas

- Melhor design da interface Web;
- Melhor formatação e divisão dos logs;
- Configura regras (assinaturas) específicas;

- Atualiza regras (assinaturas) automaticamente;
- Realiza contramedidas na detecção de determinados pacotes (emails, alteração de regras, etc);

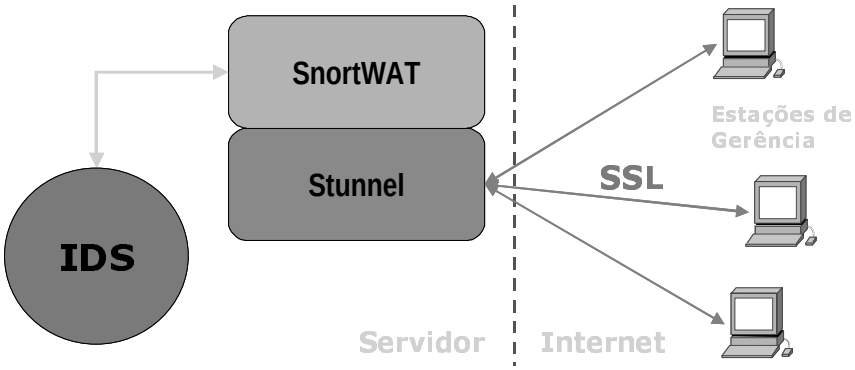
9.5 Arquitetura do Sistema

A figura 14 nos mostra a arquitetura adotada. As estações de gerências se comunicam com o SnortWAT via o programa `stunnel` (gratuito) proporcionando a interface SSL. O stunnel aciona diretamente o SnortWAT pois está rodando no `inetd`, conhecido como *super-server* ou *super-deamon* em sistemas UNIX.

O SnortWAT faz então a análise dos arquivos de log e configuração do IDS, repassando as informações para o stunnel e deste para o browser.

É utilizado o mecanismo de autenticação provido pelo protocolo HTTP.

FIGURA 14. Arquitetura do SnortWAT



9.6 Configuração do SnortWAT

1. O SnortWAT possui dois scripts principais que devem estar instalados no diretório / usr/local/sbin/, é possível a instalação em outro diretório mas os passos sequintes supõem este caminho.

- **snortwat.pl** - programa principal
- **snorwat_cf.pl** - arquivo de configuração

2. Deve-se saber a priori os arquivos utilizados pelo Snort no sistema, de uma forma geral:

- Programa: /usr/local/bin/snort
- Regras : /etc/snort.conf
- Logs : /var/log/snort

3. O programa traceroute também é utilizado, seu path deve ser conhecido, em geral:

- Programa: /usr/sbin/traceroute

Os dados acima devem ser configurados no arquivo de configuração do SnortWAT: **snortwat_cf.pl**

4. Deve-se estipular onde ficará o arquivo de senhas para autenticar os usuários do SnortWAT, bem como criar as suas senhas

- Senhas : /etc/snortwat.htpasswd
- Comando : htpasswd -c /etc/snortwat.htpasswd <user> (para criar o arquivo)
- Comando : htpasswd /etc/snortwat.htpasswd <newuser> (para adicionar novos usuarios)

5. Uma linha no /etc/inetd deve ser acrescentada

- snortwat stream tcp nowait root /usr/local/sbin/stunnel snortwat.pl -l /usr/local/ sbin/snortwat.pl -p /etc/ssl/stunnel.pem

6. Uma linha no /etc/services deve ser acrescentada

- snortwat 630/tcp # SnortWAT

7. Configuracao do arquivo contendo o certificado digital e achave privada usado pelo stunnel (/etc/ssl/stunnel.pem)

- O arquivo deve ter o segunte formato

<RSA PRIVATE KEY> linha em branco <CERTIFICATE> linha em branco

O script shell abaixo automatiza este procedimento:

```
Inicio do arquivo ----------------------------------------

#!/bin/sh

cd /etc/ssl

openssl req -x509 -nodes -newkey rsa:1024 -out rsacert.pem -keyout
stunnel.pem

echo "" >> stunnel.pem

cat rsacert.pem >> stunnel.pem

echo "" >> stunnel.pem

rm rsacert.pem

chmod 400 stunnel.pem
```

```
Fim do arquivo ----------------------------------------
```

8. Acesso:

- Agora é so acessar: https://www.seusite.com:630

9.7 Código fonte

9.7.1 snortwat_cf.pl

```
Inicio do arquivo ----------------------------------------

#!/usr/bin/perl


# CONFIGURACAO & PATHS

$server = 'https://192.168.1.1';

$port = '630';

$snort = '/usr/local/bin/snort';

$sweb_htpasswd = '/etc/snortwat.htpasswd';

$rules = '/etc/snort.conf';

$logdir= '/var/log/snort/';

$alert = 'snort.alert';

$traceroute = '/usr/sbin/traceroute';


# MESSAGES

$mess_progname = "Snort Web Administration Tool";

$mess_title = "SnortWAT";

$mess_welcome = "Benvido ao $mess_progname";

$mess_version = "Versão 1.0";

$mess_author = "by André S.B";

$mess_org = "Laboratório de Redes de Alta Velocidade (Ravel) - COPPE/
UFRJ";

$mess_snort_state = "Estado atual";

$mess_turn_on  =  "Snort ativado com sucesso";

$mess_turn_off = "Snort desativado - take care...";

$mess_on  = "Ativado";

$mess_off = "Desativado!";

$mess_rules_head = "Cabeçalho do arquivo de assinaturas";
```

```
$mess_rules_full = "Arquivo de assinaturas completo";

$mess_rules_info = "Arquivo de assinaturas informado";

$mess_log_full = "Arquivo de log completo";

$mess_log_head = "Arquivo de log - linhas mais recentes";

$mess_log_info = "Arquivo de log informado";

$mess_ip_log = "Arquivo de log por IP";

$mess_iplog_type = "Logs do tipo ";

$mess_iplog_ok  = "Arquivo de log por IP informado";

$mess_iplog_not = "Nenhum log deste IP";

$mess_cgi_error = "Erro nas entradas da CGI";

$mess_rm_1 = "Log removido";

$mess_rm_2 = "Salvo no servidor como";

$mess_ip_list = "Lista dos IPs logados";

$mess_full_list = "Lista completa";

$mess_click_ip = "Clique no IP para ver os logs correspondentes";

$mess_trace = "Aguarde o trace...";

$mess_trace_ok = "Trace completo.";

$mess_access_denied = "Acesso nao autorizado";

$mess_htpasswd_not = "Arquivo $sweb_htpasswd nao encontrado";

$mess_btn_lines = "Linhas";

$mess_btn_onoff = "Ativa/Des";

$mess_btn_head = "Assinaturas(Info)";

$mess_btn_full = "Assinaturas";

$mess_btn_tail = "Log recente";

$mess_btn_log = "Log Completo";

$mess_btn_list = "Listar IPs logados";

$mess_btn_remove = "Remover logs";

$mess_btn_tgzip = "Logs compactados";

$mess_logs_tgzip = "Logs compactados";

$mess_rm = "Clique no log do IP a ser removido";

$mess_rm_care = "Você está prestes a remover um log, no entanto o log
sera salvo de forma compactada no diretorio $logdir do servidor";

$mess_untar = "Desconpacta";
```

```
$mess_purge = "Apaga (purge!)";

$mess_untar_purge = "Clique nos links correspondentes para descompac-
tar ou eliminar os arquivos";

$mess_purged = "Arquivo eliminado";

$mess_untared = "Arquivo descompactado";

Fim do arquivo ----------------------------------------
```

9.7.2 snortwat.pl

```
Inicio do arquivo ----------------------------------------

#!/usr/bin/perl

# Laboratorio de Redes de Alta Velocidade (Ravel)

# by André S. Barbosa - 12/2000


use Socket;

require "/usr/local/sbin/snortwat_cf.pl";


# =====================================================

cgi_init();

my $info = $mess_welcome;

header("$mess_progname");

# Debug --------------------------

# foreach $nam (keys %NAME) {

#     print "<BR>$nam = $NAME{$nam}";

# }

# --------------------------------


$lines = $NAME{'lines'};

$ip    = $NAME{'ip'};

$file  = $NAME{'file'};

if ($lines && $lines !~ /\d+/) { error($mess_cgi_error); }

if ($lines > 1000) { error($mess_cgi_error); }

if ($ip && $ip !~ /\d+\.\d+\.\d+\.\d+/) { error($mess_cgi_error); }
```

```
if ($file && $file !~ /\d+\.\d+\.\d+\.\d+\-(.*?)/ && $file !~ /
^$alert\-/) { error($mess_cgi_error); }


if ($NAME{'action'} eq "onoff") { $info = set_state(); }

if ($NAME{'action'} eq "conf")  { $info = get_conf($lines); }

if ($NAME{'action'} eq "log")   { $info = get_log($lines); }

if ($NAME{'action'} eq "ip")    { $info = get_ip_log($ip); }

if ($NAME{'action'} eq "rmenu") { $info = rm_menu(); }

if ($NAME{'action'} eq "rm")    { $info = rm_log($ip); }

if ($NAME{'action'} eq "list")  { $info = list_ips(); }

if ($NAME{'action'} eq "trace") { $info = traceroute($ip); }

if ($NAME{'action'} eq "tgzip") { $info = list_logs_tgz(); }

if ($NAME{'action'} eq "untar") { $info = untar($file); }

if ($NAME{'action'} eq "purge") { $info = purge($file); }


$snort_state = check_state();


if ($snort_state) {

     print "<BR><B><FONT COLOR=blue>$mess_snort_state: </FONT><FONT
COLOR=red>$mess_on</FONT></B>";

} else {

     print "<BR><B><FONT COLOR=blue>$mess_snort_state: </FONT><FONT
COLOR=brown>$mess_off</FONT></B>";

}

print "<BR><BR>";


print $body;


footer($info);


exit;
```

```
# ========================================================
# SUBROTINAS
# ========================================================
sub cgi_init() {

 sysread (STDIN, $in, 5000);

 if ($in !~ /^GET/) {
  # So tratamos requisicoes tipo GET;
  exit;
 }

 @req = split("\n", $in);
 $req = $req[0];

 $auth_error = 1;

 foreach $linereq (@req) {
    if ($linereq =~ m/Authorization\:\sBasic\s(.*?)=/) {
        (my $user, my $pass) = split(':', decode_base64($1));

                  unless (-f $sweb_htpasswd) { $message = ":
$mess_htpasswd_not"; last; }
        @htpasswd = `cat $sweb_htpasswd`;
foreach $line (@htpasswd) {
chomp $line;
        (my $ruser, my $rpass) = split(':', $line);
                if ($user eq $ruser) {
                    $real_user = $ruser;
                    $real_pass = $rpass;
    last;
}
        }
```

```
        if (!$real_user || !$real_pass) { last; }
        $cryptpass = crypt($pass, $real_pass);
        if ($cryptpass eq $real_pass) { $auth_error = 0; }
        last;
   }
 }

 if ($auth_error) {
    $erro  = "401 Authorization Required";
    $dados = "WWW-Authenticate: Basic realm=\"$mess_progname\"\r\n";
   print "HTTP/1.0 $erro\r\n$dados<HTML><HEAD><TITLE>$mess_progname</
TITLE></HEAD><BODY>$mess_access_denied              $message</BODY></
HTML>\r\n\r\n";
    exit;
 }

 $req =~ m/GET \/\?(.*?)\sHTTP/;
 $data = $1;

 if (substr ($data, -1, 1) eq '/') {
    chop $data;
 }

 @pairs = split(/&/, $data);
 foreach $pair (@pairs) {
     ($name,$value) = split(/=/,$pair);
     $value =~ tr/+/ /;
     $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
     $NAME{$name} = $value;
 }

}
```

```
sub check_state() {

 @procs = `ps -axw`;

 foreach $proc (@procs) {
   if ($proc =~ m/$snort/) { return 1; }
 }

 return 0;
}




sub get_conf() {

 my $linhas = shift;

 $body = "<FORM METHOD=GET ACTION=\"$server:$port\/\"><INPUT TYPE=sub-
mit VALUE=\"$mess_btn_lines\">";
 $body .= " = <INPUT TYPE=hidden NAME=action VALUE=conf>";
 $body .= "<INPUT TYPE=text NAME=lines VALUE=50></FORM>";

 if ($linhas) {
     $body .= "<BR><B>$mess_rules_head (linhas = $linhas):</B><BR>";
     @header = `head -$linhas $rules`;
     foreach $line (@header) {
         chomp $line;
         $body .= "$line<BR>";
     }
 } else {
     $body .= "<BR><B>$mess_rules_full:</B><BR>";
     @rules = `cat $rules`;
```

```
     foreach $line (@rules) {
         chomp $line;
         $body .= "$line<BR>";
     }
 }

 return $mess_rules_info;

}

sub get_log() {

 my $linhas = shift;

 $body = "<FORM METHOD=GET ACTION=\"$server:$port\/\"><INPUT TYPE=sub-
mit VALUE=\"$mess_btn_lines\">";
 $body .= " = <INPUT TYPE=hidden NAME=action VALUE=log>";
 $body .= "<INPUT TYPE=text NAME=lines VALUE=200></FORM>";

 if ($linhas) {
     $body .= "<BR><B>$mess_log_head (linhas = $linhas):</B><BR>";
     my @alert = `tail -$linhas $logdir$alert`;
     foreach $line (@alert) {
         chomp $line;
         $body .= "$line<BR>";
     }
 } else {
     $body .= "<BR><B>$mess_log_full:</B><BR>";
     my @alert = `cat $logdir$alert`;
     foreach $line (@alert) {
         chomp $line;
         $body .= "$line<BR>";
     }
```

```
 }

 return $mess_log_info;

}

sub get_ip_log() {

  my $addr = shift;
  unless (-d "$logdir/$addr") { return $mess_iplog_not; }

  $body = "<BR><B>$mess_ip_log ($addr):</B><BR>";
  my @dir = `ls $logdir/$addr`;

  foreach $subdir (@dir) {
    chomp $subdir;
    my @subdir = `cat $logdir/$addr/$subdir`;
       $body  .=  "<BR><B><FONT  COLOR=blue>$mess_iplog_type  $subdir</
FONT></B><BR>";
    foreach $line (@subdir) {
      chomp $line;
      $body .= "$line<BR>";
    }
  }

 return $mess_iplog_ok;

}

sub list_ips() {

  $body = "<BR><B>$mess_ip_list:</B><BR>";
  my @dir = `ls $logdir`;
```

```
  foreach $line (@dir) {

    chomp $line;

      if (-d "$logdir$line" && $line ne $alert) {

      my $iaddr = inet_aton($line);

      my $hostname = gethostbyaddr($iaddr, AF_INET);

                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=ip&ip=$line\"><B>$line</B></A> ($hostname) ";

                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=trace&ip=$line\"><B>Trace</B></A><BR>";

    }

  }


             $body       .=       "<BR><A       HREF=\"$server:$port\/
\?action=log\"><B>$mess_full_list</B></A><BR>";


  return $mess_click_ip;

}


sub list_logs_tgz() {


  $body = "<BR><B>$mess_logs_tgzip:</B><BR>";

  my @dir = `ls $logdir`;


  foreach $line (@dir) {

    chomp $line;

      if (-f "$logdir$line" && $line ne $alert) {

                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=untar&file=$line\"><B>$line</B></A>($mess_untar)
  ";

                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=purge&file=$line\"><B>$mess_purge</B></A><BR>";

    }

  }
```

```
  return $mess_untar_purge;

}


sub purge() {
 my $file = shift;

 $dummy = `rm -f "$logdir$file"`;

 return "$mess_purged: $file";
}

sub untar() {
 my $file = shift;

 chdir $logdir;
 $dummy = `tar -xzf $file`;
 $dummy = `rm -f $file`;

 return "$mess_untared: $file";
}

sub traceroute() {

 my $addr = shift;

 $body = "<BR><B>$mess_trace</B><BR>";
 my @out = `$traceroute $addr 2>/dev/null`;

 foreach $line (@out) {
    chomp $line;
```

```
    $body .= "$line<BR>";
 }

 return $mess_trace_ok;

}

sub rm_menu {

 $body = "<BR><B>$mess_rm</B><BR>";
 my @dir = `ls $logdir`;

 foreach $line (@dir) {
    chomp $line;
      if (-d "$logdir$line" && $line ne $alert) {
      my $iaddr = inet_aton($line);
      my $hostname = gethostbyaddr($iaddr, AF_INET);
                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=ip&ip=$line\"><B>$line</B></A> ($hostname) ";
                          $body    .=    "<A    HREF=\"$server:$port\/
\?action=rm&ip=$line\"><B>Remover</B></A><BR>";
    }
 }

        $body        .=        "<BR><A        HREF=\"$server:$port\/
\?action=log\"><B>$mess_full_list</B></A> ";
  $body  .=  "<A  HREF=\"$server:$port\/\?action=rm\"><B>Remover</B></
A><BR>";

 return $mess_rm_care;
}

sub rm_log() {
```

```
  my $addr = shift;

  if ($addr) {
      unless (-d "$logdir/$addr") { return $mess_iplog_not; }
      chdir $logdir;
      ($sec, $min, $hour, $mday, $mon, $year) = localtime(time);
      $year+=1900;
      $mon++;
      $date = $hour.'-'.$min.'-'.$sec.'-'.$mday.'-'.$mon.'-'.$year;
      $dummy = `tar -czf "$addr-$date.tgz" "$addr"`;
      $dummy = `rm -rf $addr`;
      return "$mess_rm_1 ($addr) $mess_rm_2 $addr-$date.tgz";
  } else {
      chdir $logdir;
      ($sec, $min, $hour, $mday, $mon, $year) = localtime(time);
      $year+=1900;
      $mon++;
      $date = $hour.'-'.$min.'-'.$sec.'-'.$mday.'-'.$mon.'-'.$year;

      $dummy = set_state();
      $dummy = `tar -czf "$alert-$date.tgz" "$alert"`;
      $dummy = `rm -f $alert`;
      $dummy = set_state();
      return "$mess_rm_1 ($alert) $mess_rm_2 $alert-$date.tgz";
  }

}


sub set_state() {
 $snort_state = check_state();
```

```
 if ($snort_state) {
    @procs = `ps -axw`;
    foreach $proc (@procs) {
        if ($proc =~ m/$snort/) {
            ($pid) = ($proc =~ /(\d+)/);
            $dummy = `kill $pid`;
            last;
        }
    }
    sleep 2;
    return "$mess_turn_off";
 } else {
    $dummy = `$snort -D -c $rules`;
    return "$mess_turn_on : $dummy";
 }

}


sub header() {

 my $title = shift;

 print "HTTP/1.0 200 SNORTWEB_OK\r\n";
 print "Content-Type: text/html\r\n\r\n";
 print "<HTML><HEAD><TITLE>$title</TITLE></HEAD><BODY>";
 print "<FONT FACE=\"verdana,arial,helvetica\" SIZE=\"1\">";

 print "<H1>$mess_title</H1>";
 print "$mess_org<BR>";
 print "$mess_author - $mess_version<BR>";
```

```
 print "<BR>[";

 print " <A HREF=\"$server:$port\/\?action=onoff\">$mess_btn_onoff</
A> |";

 print " <A HREF=\"$server:$port\/\?action=list\">$mess_btn_list</A>
|";

  print  "  <A  HREF=\"$server:$port\/\?action=log\">$mess_btn_log</A>
|";

          print          "          <A          HREF=\"$server:$port\/
\?action=log&lines=100\">$mess_btn_tail</A> |";

 print " <A HREF=\"$server:$port\/\?action=conf\">$mess_btn_full</A>
|";

          print          "          <A          HREF=\"$server:$port\/
\?action=conf&lines=13\">$mess_btn_head</A> |";

 print " <A HREF=\"$server:$port\/\?action=rmenu\">$mess_btn_remove</
A> |";

 print " <A HREF=\"$server:$port\/\?action=tgzip\">$mess_btn_tgzip</
A> ";

 print "]<BR>";

}


sub footer() {


 my $nfo = shift;


 print "<BR><BR><B>INFO:</B> $nfo";

 print "</FONT></BODY></HTML>";


}


sub error() {


 my $erro = shift;


 print "<BR><BR><B>ERROR:</B> $erro";

 print "</FONT></BODY></HTML>";
```

```
 exit;


}


sub decode_base64 ($)
{
    local($^W) = 0; # unpack("u",...) gives bogus warning in 5.00[123]


    my $str = shift;
    my $res = "";


    $str =~ tr|A-Za-z0-9+=/||cd;            # remove non-base64 chars
    $str =~ s/=+$//;                        # remove padding
    $str =~ tr|A-Za-z0-9+/| -_|;           # convert to uuencoded for-
mat
    while ($str =~ /(.{1,60})/gs) {
        my $len = chr(32 + length($1)*3/4); # compute length byte
        $res .= unpack("u", $len . $1 );    # uudecode
    }
    $res;
}
Fim do arquivo ---------------------------------------
```

Capítulo 10

Conclusões

10.1 Conclusões

- Os IDS devem estar configurados, ou se comportar de forma mais próxima possível dos sistemas monitorados;
- Os IDS devem ter mecanismos de proteção contra ataques, principalmente DoS;
- Os IDS devem ter mecanismos para intercomunicação com outros agentes no sistema;
- Os IDS devem ter interfaces fáceis, práticas e intuitivas de configuração e análise de eventos;
- IDS são uma peça fundamental na segurança de uma organização, tanto quanto firewalls, mas devemos lembrar que de nada adianta se estes não estiverem configurados adequadamente. Por isso se justifica o desenvolvimento de interfaces de administração tais como as propostas neste trabalho.

10.2 Aprendizado

Durante o estudo dos IDS muito conhecimento e idéias novas surgiram, motivando a integração e gerência de Sistemas de Detecção de Intrusão e Firewalls, todo estudo levou a idealização de um sistema de gerência que permita isto, este sistema será provavelme assunto de uma tese de mestrado.

O desenvolvimento do SnortWAT trouxe a tona idéias que podem ser implementadas com sucesso a fim de propiciar o gerenciamento de IDS de forma simples, e grande parte de sua estrutura pode ser utilizada para o sistema de gerenciamento mais genérico descrito anteriormente.

Capítulo 11

Referências

[1] T. H. Ptacek, T. N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Secure Networks, Inc, January, 1998.

[2] C. Kahn, P. A. Porras, S. Staniford-Chen, B. Thung, "A Common Intrusion Detection Framework", July, 1998

[3] M. Roesch "Snort – Lightweight Intrusion Detection for Networks", Stanford Telecommunications, Inc, November, 1999.

[4] Y. Fyodor, "Snortnet – A Distributed Detection System", Kyrgyz Russian Slavic University, June 26, 2000

[5] B. Schneier, "Applied Cryptography", Second Edition, Wiley, 1999.

[6] W. Stallings, "Cryptography and Network Security", Second Edition, Prentice Hall.

[7] S. McClure, J. Scambray, "Hacking Exposed", Makron Books, 2000.

[8] Axent Site – http://www.axent.com

[9] ISS Site – http://www.iss.net

[10] CERT - Computer Emergency Respose Team - http://www.cert.org

[11] ICSA - Internet Consortium Security Agency - http://www.icsa.net

[12] NFR Site – http://www.nfr.net

[13] Snort Site – http://www.snort.org

[14] CIDF Site – http://www.gidos.org

[15] SecurityFocus – http://www.securityfocus.com

[16] André S. Barbosa (andre@ravel.ufrj.br) http://www.ravel.ufrj.br/~andre

[17] OpenSSL - http://www.openssl.com

[18] RFC2828 - Internet Security Glossary

[19] RFC2196 - Site Security Handbook

[20] IETF - Internet Engineering Task Force - – http://www.ietf.org

[21] ArachNIDS - http://www.whitehats.com