



Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação

Dissertação de Mestrado

Block

**Uma Ferramenta para Reconhecimento e Bloqueio de Ataques
a Redes de Computadores**

AUTOR: Flávio Marcelo Cavalcante Bandeira do Amaral

ORIENTADOR: Prof. Dr. Paulo Sérgio da Motta Pires

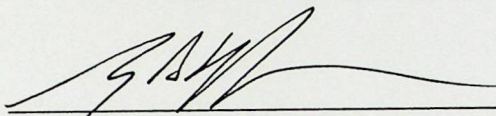
Natal/RN - Brasil

Outubro de 2002

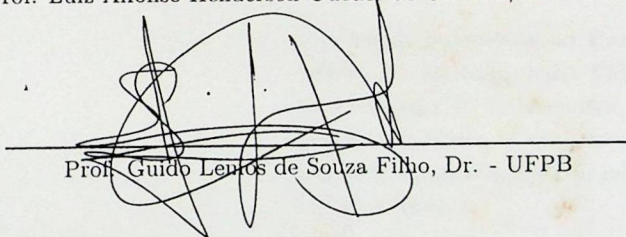
Block
Uma Ferramenta para Reconhecimento e Bloqueio de Ataques a
Redes de Computadores

Flávio Marcelo Cavalcante Bandeira do Amaral

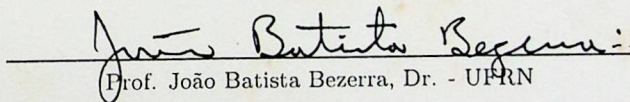
Aprovada, em 11 de Outubro de 2002, pela Banca Examinadora formada pelos seguintes membros:



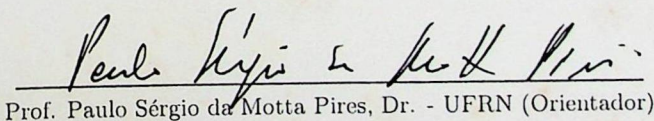
Prof. Luiz Affonso Henderson Guedes de Oliveira, Dr. - UFPA



Prof. Guido Leiros de Souza Filho, Dr. - UFPB



Prof. João Batista Bezerra, Dr. - UFRN



Prof. Paulo Sérgio da Motta Pires, Dr. - UFRN (Orientador)

NATAL, RN - BRASIL
Outubro de 2002

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação

Flávio Marcelo Cavalcante Bandeira do Amaral

Block
Uma Ferramenta para Reconhecimento e Bloqueio de Ataques
a Redes de Computadores

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Rio Grande do Norte, como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.).

ORIENTADOR: Prof Dr. Paulo Sérgio da Motta Pires

Natal/RN - Brasil
Outubro de 2002

RN000178368

MEC UFRN Biblioteca Central "Zila Mamede"

681.3.066

179.029/03

Nº de Classif

Registro

Fornecido

28/02/03

DISSERT

Forma de

Deapo Pat

Empenho

R\$ 15,00

Preço



L179029

Agradecimentos

Primeiramente a Deus, por sempre me mostrar o caminho certo a seguir e me prover de tudo o que possuo.

À minha família e a Rossana Rocha pelo constante suporte. Sem eles, esse trabalho não teria sido concluído.

Ao meu orientador, Prof. Dr. Paulo Sérgio da Motta Pires, por ter me aceitado como seu orientando.

Ao Dr. Sergio Vianna Fialho por ter me incentivado a fazer o mestrado.

A todos da Avalon Consultoria pelo roteador emprestado. Aos amigos Ricardo Dotta, Breno Roberto pela ajuda nos testes, Márcio Wilhelm pela ajuda na véspera da defesa e a Fernando Araújo e Bruno Fiorentini pelo apoio dado.

A Richard Stevens, pelo excelente livro *Unix Network Programming*, onde aprendi a programação em rede.

Dedicatória

Há muito tempo, em um pequeno lugarejo perdido no mundo, moravam duas famílias. Uma grande duna se aproximava dia após dia das duas casas. Uma família, prevendo o inevitável, mudou-se rapidamente, enquanto a outra ficou. Após cinco anos, a família que se mudou volta e vê a duna atrás da sua casa intacta. Curiosos, foram perguntar à família que ficou como a duna passara por cima da casa. Eles, então, disseram que ela não passara por cima: “como tínhamos um grande problema para resolver, unimos todos os esforços para resolvê-lo. Todo dia, cada um de nós tirava um balde de areia da duna e colocava atrás das casas. O tempo passou e a duna também e não precisamos nos mudar.”

Dedico esta tese a essa família e a todas as pessoas que sempre me incentivaram a prosseguir, mesmo quando o caminho era árduo demais. À minha mãe Vitória, minha avó Dóia, minha esposa Rossana Rocha, minha Tia Suerda e minha irmã Ana Flávia.

Sumário

Lista de Figuras	VIII
Lista de Tabelas	IX
Resumo	X
Abstract	XI
1 Introdução	1
2 Tipos de Ataques	4
2.1 Introdução	4
2.2 <i>Scans</i>	4
2.2.1 Técnicas de Varreduras	5
2.2.2 Identificação Remota de Sistemas Operacionais	6
2.3 Ataques por Falsificação de Endereços IP	9
2.3.1 Evitando Ataques por Falsificação de Endereços IP	9
2.4 Ataques por Negação de Serviços	10
2.4.1 Ataque <i>smurf</i>	10
2.4.2 Ataques com Pacotes IP Grandes	11
2.4.3 Ataques com <i>Land</i> e <i>Teardrop</i>	12
2.4.4 Ataque <i>SYN Flooding</i>	12
2.4.5 Ataques DDoS	13
3 Características da Ferramenta Block	15
3.1 Introdução	15
3.2 Sistemas Detectores de Intrusões em Rede - <i>NIDS</i>	15
3.3 O Programa watcher	16
3.4 O projeto do Block	17
3.5 Bloqueando Tráfego não Autorizado e Ataques	18
3.6 Listas de Acesso	18

3.6.1	Definindo uma Lista de Acesso	19
3.6.2	Aplicando uma Lista de Acesso	20
3.7	O Ambiente de Aplicação	21
3.7.1	Níveis de Proteção do Block	22
3.7.2	O Block e os Ataques DDoS	24
4	A Arquitetura da Ferramenta Block	26
4.1	Introdução	26
4.2	O Programa Principal	26
4.3	Identificando Ataques	27
4.3.1	Estruturas de Dados	28
4.3.2	Algoritmos de Identificação de Ataques	29
4.4	Gerando Relatórios e Construindo Listas	32
4.5	Conexão com o Roteador	35
4.6	Arquivo de Configuração	38
4.7	Limitações da Ferramenta	43
5	Detecção de Ataques com o Block	44
5.1	Introdução	44
5.2	Ambiente de Testes	44
5.3	Ataques	47
5.3.1	Ataques a Portas TCP	47
5.3.2	Ataques com a Ferramenta targa	50
5.4	Ataque por Varredura de Portas	53
5.5	Ataque <i>SYN Flooding</i>	56
6	Conclusões e Perspectivas	58
6.1	Conclusões	58
6.2	Perspectivas de Desenvolvimento	59
Apendice A - Hierarquia de Processos e Diagramas de Fluxo de Dados		61
Referências Bibliográficas		65

Lista de Figuras

2.1	Arquitetura de um ataque do tipo <i>smurf</i>	11
2.2	Arquitetura de um ataque com a ferramenta <i>Land</i>	12
2.3	Arquitetura do ataque distribuído de negação de serviço (DDOS).	14
3.1	Solução de IDS baseada em rede Ethernet.	16
3.2	Exemplo de construção de uma lista de acesso do tipo <i>standard</i>	19
3.3	Exemplo de construção de uma lista de acesso do tipo <i>extended</i>	20
3.4	Exemplo de construção de uma lista de acesso do tipo <i>extended</i> usando nomes.	20
3.5	Exemplo da aplicação de uma lista de acesso.	21
3.6	Bloqueio de estação para estação.	23
3.7	Bloqueio de rede para estação.	23
3.8	Bloqueio de rede para todos.	24
3.9	Bloqueio de todos para todos.	24
4.1	Arquitetura da ferramenta Block	26
4.2	Código do laço infinito para a leitura de pacotes na função <code>main()</code>	27
4.3	Códigos das assinaturas dos ataques <i>Land</i> e <i>SYN Flooding</i>	28
4.4	Código da função <code>addtarget()</code>	29
4.5	Código da assinatura de um <i>portscan</i> TCP.	30
4.6	Código da assinatura de um <i>portscan</i> UDP.	30
4.7	Código da assinatura do <i>Teardrop</i>	31
4.8	Código da assinatura do ataque <i>smurf</i> e ping da morte.	31
4.9	Código para a detecção de ataques em portas TCP.	32
4.10	Código para o registro de um ataque <i>Teardrop</i>	33
4.11	Código para a geração da lista de acesso.	33
4.12	Código para o bloqueio de um ataque de portas TCP.	34
4.13	Código para a conexão com o roteador.	36
4.14	Código para a retirada da lista.	37
4.15	Código para inclusão da lista gerada.	37
4.16	Código para o encerramento da conexão com o roteador.	37

4.17	Parâmetros do Block .	39
4.18	Configuração do tipo de bloqueio desejado.	40
4.19	Configuração da concatenação de listas de acesso.	40
4.20	Pré-seleção do número da lista de acesso.	40
4.21	Configuração do endereço IP do roteador.	40
4.22	Notificação da existência dos arquivos de portas TCP e UDP.	41
4.23	Configuração dos parâmetros de erro.	41
4.24	Configuração do endereço eletrônico do administrador.	41
4.25	Configuração das senhas do roteador.	41
4.26	Configuração do <i>prompt</i> do roteador.	42
4.27	Configuração do tempo de conexão com o roteador.	42
4.28	Configuração do tempo de <i>timeout</i> com o roteador.	42
4.29	Interface onde a lista de acesso será aplicada.	43
5.1	Topologias das redes de ataque e defesa.	44
5.2	Informações sobre o modelo e a versão do IOS do roteador utilizado no ambiente de testes.	45
5.3	Configuração inicial do roteador.	46
5.4	Linhas do arquivo gerado pelo tcpdump na rede de ataque.	48
5.5	Registro do ataque de portas TCP no arquivo do syslog .	48
5.6	Conexão do Block com o roteador.	49
5.7	Configuração do roteador contra ataques TCP proteção <i>host->host</i> .	49
5.8	Configuração do roteador contra ataques TCP proteção <i>net->host</i> .	49
5.9	Configuração do roteador contra ataques TCP proteção <i>net->any</i> .	50
5.10	Configuração do roteador contra ataques TCP proteção <i>any->any</i> .	50
5.11	Ataque <i>smurf</i> capturado pelo tcpdump .	51
5.12	Registro do ataque <i>smurf</i> no arquivo do syslog .	51
5.13	Lista de acesso do ataque <i>smurf</i> .	51
5.14	Configuração do roteador após o ataque <i>smurf</i> .	51
5.15	Ataque <i>Land</i> capturado pelo tcpdump .	52
5.16	Registro do ataque <i>Land</i> no arquivo do syslog .	52
5.17	Lista de acesso do ataque <i>Land</i> .	52
5.18	Configuração do roteador após o ataque <i>Land</i> .	52
5.19	Ataque <i>Teardrop</i> capturado pelo tcpdump .	53
5.20	Registro do ataque <i>Teardrop</i> no arquivo syslog .	53
5.21	Lista de acesso do ataque <i>Teardrop</i> .	53
5.22	Configuração do roteador após o ataque <i>Teardrop</i> .	53
5.23	Ataque de varredura de portas TCP capturado pelo tcpdump .	54
5.24	Ataque de varredura de portas UDP capturado pelo tcpdump .	55

5.25	Registro do ataque de varredura de portas TCP e UDP no arquivo syslog	55
5.26	Lista de acesso do ataque de varredura de portas TCP e UDP.	55
5.27	Configuração do roteador após o ataque de varredura de portas TCP e UDP. . .	56
5.28	Bloqueio de varredura de portas capturado pelo tcpdump	56
5.29	Registro do ataque <i>SYN Flooding</i> no arquivo syslog	56
5.30	Lista de acesso gerada pelo Block do ataque <i>SYN Flooding</i>	56
5.31	Listas no roteador após o ataque <i>SYN Flooding</i>	57
5.32	Bloqueio de ataque <i>SYN Flooding</i>	57
5.33	Mensagem enviada ao administrador da rede informando sobre a ocorrência do ataque <i>SYN Flooding</i>	57
1	Hierarquia de processos da ferramenta Block	61
2	Diagrama de Fluxo de Dados, nível 0.	62
3	Diagrama de Fluxo de Dados, nível 1.	62
4	Diagrama de Fluxo de Dados, nível 2.	63
5	Diagrama de Fluxo de Dados, nível 3.	64

Lista de Tabelas

3.1	Opções de linha de comando do programa watcher	17
5.1	Configurações e as funções de cada estação do ambiente de testes.	45
5.2	Parâmetros de configuração do arquivo block.conf	47

Resumo

Os sistemas operacionais e as máquinas em redes são passíveis de ataques eletrônicos. O aumento do número e da frequência dos ataques a instituições conectadas à Internet e o risco de se ter informações roubadas ou indevidamente alteradas é elevado. Os danos que podem ser causados variam da simples leitura de informações à modificação dos dados.

Os *NIDS*, *Network Intrusion Detection Systems*, são ferramentas utilizadas para detectar ataques a sistemas conectados em redes. Neste trabalho, desenvolvemos uma ferramenta, chamado **Block**, que utiliza um *NIDS free* e de código aberto para gerar listas de acesso para roteadores Cisco como resposta aos ataques detectados. Além de bloquear os ataques de forma automática, o programa desenvolvido envia mensagens de correio eletrônico ao administrador do sistema para notificá-lo sobre as ações realizadas pelo programa.

Abstract

Introdução

Operating systems and hosts connected to networks can be a target for attacks. The increasing number and frequency of those attacks aimed at Internet connected institutions are very high as well as the risks of steal and the unauthorized access of their information. The damage can range from a single reading to the modification of proprietary information.

NIDS, Network Intrusion Detection Systems, are tools used to detect attacks aimed at on-line systems. In this research, we have developed a tool named **Block**, which uses a free and open source NIDS to generate Cisco access lists as a response to detected attacks. Beyond automatically blocking attacks, the developed program sends e-mail messages to the system administrator to notify about the actions taken by the tool.

Capítulo 1

Introdução

Os sistemas operacionais e as redes são passíveis de ataques eletrônicos. O aumento do número e da frequência de ataques a instituições conectadas à Internet e o risco de se ter informações roubadas ou indevidamente alteradas é elevado. Os danos que podem ser causados variam da simples leitura de informações confidenciais à modificação dos dados.

Nesse cenário, há a necessidade de se catalogar e de se estudar as características dos ataques e de se gerar alertas quando padrões de tráfego em redes forem similares a padrões de ataques já catalogados. Com esses objetivos, surgiram os Sistemas Detectores de Intrusos em Redes, os *NIDS* (*Network Intrusion Detection Systems*). Esses programas coletam informações do tráfego da rede ou da configuração de uma estação, registrando-as em arquivos de *logs*.

Os *NIDS* podem ser vistos como um dos componentes de uma política de segurança mais abrangente. Eles implementam os recursos de auditoria, monitoração, reconhecimento e resposta aos ataques. Alguns tipos de *NIDS* atuam somente na máquina onde estão instalados; outros, conseguem investigar o tráfego que flui pela rede.

O objetivo desse trabalho é apresentar o desenvolvimento e a implementação de uma ferramenta, chamada **Block**, que detecta e bloqueia ataques de forma flexível e automática. Os ataques são bloqueados nos seus estágios iniciais através de listas de acesso colocadas em roteadores. As listas de acesso são construídas a partir de eventos gerados por um *NIDS*. A ferramenta desenvolvida também gera mensagens de correio eletrônico alertando ao administrador da rede sobre as ocorrências. A conexão agressiva é negada enquanto as demais continuarão a funcionar.

O programa desenvolvido permite que o roteador de bloqueio já possua uma lista de acesso previamente configurada. Caso exista, esta lista será concatenada com a lista referente à negação dos ataques detectados. O módulo de construção de listas de acesso foi desenvolvido de forma a ser portado para outros *NIDS* com poucas modificações.

As listas de acesso são voltadas para roteadores Cisco. Os roteadores Cisco foram escolhidos por estarem, atualmente, presentes na maioria das redes de instituições conectadas à Internet.

A ferramenta **Block** foi desenvolvida na linguagem de programação C. Ela consiste em um

módulo construtor de listas de acesso e outro para transmiti-la ao roteador. As listas de acesso são geradas a partir da ocorrência de eventos detectados por um *NIDS* e implementam níveis de bloqueio, ou proteção, aplicáveis às redes atacada e atacante. O *NIDS* escolhido foi o **watcher** [1], que também está implementado em C e é uma ferramenta *free* e de código aberto. O **watcher** permite trabalhar no nível dos protocolos de rede e de transporte (IP, TCP e UDP), dando flexibilidade para a construção de assinaturas de ataques que usam esses protocolos. Por trabalhar no nível desses protocolos, acrescentamos ao **watcher** uma função de reconhecimento de assinaturas de ataques que usam as portas TCP e UDP.

O ambiente de testes consistiu em duas estações de trabalho, um roteador Cisco modelo 1600 e um hub. Implementamos, com esses equipamentos, duas redes. Uma das redes era a rede de ataque, que consistia de uma estação com os programas de ataque. Na outra rede, foi colocada a outra estação, rodando o **Block**, que era, também, a vítima dos ataques. Para verificar a eficiência da ferramenta desenvolvida, lançamos os ataques *Land*, *Teardrop*, *Syn Flooding*, *portscan*, *smurf* e um ataque a uma porta TCP que simulava uma *backdoor*. Todos esses ataques foram reconhecidos e bloqueados através da geração de listas de acesso que foram enviadas para o roteador. O administrador da rede foi avisado via *e-mail*, demonstrando que a ferramenta executou a contento as tarefas para as quais foi projetada.

Algumas ferramentas com o mesmo propósito do **Block** estão disponíveis. Dentre elas, podemos citar a desenvolvida por Lance Sptizner [2] e os produtos Cisco *Secure* [3].

O trabalho de Lance Sptizner consiste em um *shell script* chamado **alert.sh** e é projetado para executar em plataformas Unix. Para que o *script* funcione, é necessária a Firewall-1, que deve ser configurada para direcionar seus alertas para o *script*. Quando algum intruso executa ataque por *scan*, o *script* é ativado. Esse método não exige que uma máquina fique escutando o tráfego da rede, pois são as saídas dos logs da firewall que são processadas. Quando uma máquina é classificada como um *portscanner*, o *script* pode bloqueá-la automaticamente ou não, através da criação de uma entrada nas regras da Firewall. Como sistema de alerta, uma mensagem de correio eletrônico é enviada ao administrador e **traps** de SNMP são gerados. Logs também são gerados em discos.

Os produtos Cisco *Secure* são, basicamente, constituídos pela adição de funcionalidades de uma Firewall e de um IDS ao IOS de um roteador e estão disponíveis para as plataformas 800, uBR900, 1400, 1600, 1700, 2500, 3600, 7100, 7200 e 7500.

O **Block** possui maior flexibilidade que a ferramenta desenvolvida pelo Lance Spitzer, já que não detecta apenas *portscans*. Além disso, pode ser utilizado nas plataformas Cisco citadas com custo bastante reduzido, é *free* e com código fonte disponível.

Esta Dissertação está dividida em seis capítulos e um apêndice. Este Capítulo, é uma apresentação da idéia central do trabalho desenvolvido. No Capítulo 2, apresentamos a descrição dos tipos de ataques que são detectados pelo **Block**. Estes ataques são decorrentes de falhas na implementação dos protocolos da pilha TCP/IP. No Capítulo 3, descrevemos as caracterís-

Capítulo 2

Tipos de Ataques

2.1 Introdução

O protocolo IP versão 4 pode ser usado de maneira hostil pois não possui mecanismos intrínsecos de autenticação e encriptação [4]. O uso desse protocolo, sem observar critérios de segurança, pode oferecer oportunidades que podem ser exploradas por atacantes.

Geralmente, antes de realizar o ataque propriamente dito, o atacante cumpre uma etapa que consiste no reconhecimento dos serviços que estão sendo oferecidos pelas estações da rede alvo. Nesta fase, o agressor procura identificar os tipos de sistemas operacionais envolvidos e as portas TCP/UDP ativas. As portas possuem uma numeração oficial, atribuída pela IANA (Internet Assigned Numbers Authority) [5]. Nesta fase do processo, o atacante utiliza programas conhecidos como *scans*.

Após ser identificada, a possível fragilidade será explorada através de ataques que utilizam a falsificação de endereços IP ou através de ataques que objetivam a negação de um serviço que esteja sendo oferecido.

Neste Capítulo, apresentamos algumas considerações sobre as técnicas de *scan*, dos ataques por falsificação de endereços IP e dos ataques visando a negação de serviços. Estas técnicas serão utilizadas, no Capítulo 5, para verificar a eficiência da ferramenta que desenvolvemos.

2.2 *Scans*

Descobrir os serviços disponíveis em uma determinada máquina é parte da etapa de reconhecimento que normalmente um intruso realiza antes dos preparativos para uma invasão. Os programas que executam automaticamente esse tipo de tarefa são conhecidos como *scans* e seu funcionamento baseia-se em tentativas de conexão com portas TCP ou UDP de uma determinada estação ou de todas as máquinas de uma rede inteira. Normalmente, a ferramenta verifica sequencialmente as portas de cada estação e apresenta os serviços que estão disponíveis. Entre as informações que são coletadas por um *scan* de portas, chamado de *portscanner*, destacam-se

os serviços que estão atualmente sendo oferecidos (ativos), que usuário os inicializou, quais as versões dos serviços, o tipo e a versão do sistema operacional.

Existem vários programas que permitem fazer uma varredura completa em máquinas de uma rede local. Alguns, também apontam as possíveis falhas existentes nas versões dos serviços identificados e como explorá-las. Com base nessas informações, é possível elaborar um ataque bem sucedido. Como exemplo, apresentamos o resultado do uso do programa *sscan* [6] em uma rede.

```
---<[ * report for host web.victim.com ]*>
<[ tcp port: 80 (http) ]>
<[ tcp port: 23 (telnet) ]>
<[ tcp port: 111 (sunrpc) ]>
<[ tcp port: 21 (ftp) ]>
<[ tcp port: 22 (unknown) ]>
<[ *OS*: web.victim.com: os detected: solaris 2.x
<[ *NFS exp0rt*: 10.1.1.1: to: web2.victim.com dir: /mnt/disk1 ]>-
-<[ *NFS exp0rt*:10.1.1.1: to: everyone! =) dir: /mnt/disk1/logs ]>-
--<[*VULN*: web.victim.com: world readable/writeable nfs exports
--<[ *VULN*: web.victim.com: solaris running nlockmgr.. remote overflow?{'u'}
--<[ *VULN*: web.victim.com: QPOP remote r00t buffer overflow
<[ *VULN*:win.victim.com: port 31337(tcp) open.. backdoor?
--<[ *VULN*: web3.victim.com: /cgi-bin/phf here.. shell escape?
```

Podemos observar que esse programa, além de apresentar as portas com serviços ativos, indica possíveis vulnerabilidades existentes na estação alvo.

2.2.1 Técnicas de Varreduras

Várias técnicas foram criadas e refinadas para facilitar a investigação de estações, protocolos e portas. Elas possuem prós e contras quanto ao grau de confiabilidade do resultado, alertas gerados e velocidade de execução. As técnicas mais comuns de varredura são [7]:

1. **TCP connect:** É a forma mais básica e fácil de ser realizada. A função *connect()*, disponível nos sistemas operacionais, é usada para estabelecer uma conexão com qualquer porta na máquina alvo. Se a porta estiver ativa, a função terá sucesso; caso contrário, o serviço pode estar desabilitado ou filtrado. Uma das vantagens dessa função é que não é necessário possuir privilégios de administrador para usá-la. Outra vantagem é a velocidade, pois é possível abrir muitos *sockets* em paralelo, ao invés de abrir um por vez, de forma sequencial. A maioria dos programas de varreduras usa esse método. A desvantagem é que ele é facilmente detectável e pode gerar grandes quantidades de *log* nas máquinas alvo.
2. **TCP SYN:** Essa técnica é também conhecida como varredura meio aberta, porque não abre uma conexão TCP completa. O atacante envia um pacote SYN, como na abertura

de uma conexão normal e, depois, espera uma confirmação. Uma resposta contendo um pacote **SYN+ACK** indica que a porta está ativa. Quando esse pacote chegar, um pacote **RST** é enviado para abortar a conexão. A maior vantagem desse método é que poucos *sites* irão registrar em *logs* essa conexão;

3. **TCP FIN**: Os pacotes **FIN** podem passar sem serem notados por vários dispositivos de segurança até o destino final. O fundamento dessa estratégia é que esses pacotes somente são respondidos com pacotes **RST**, por portas não ativas, enquanto as ativas simplesmente os ignoram. Assim, pelas respostas recebidas, podemos deduzir quais estão ativas e quais estão inativas.
4. **Fragmentação** : Esta não é uma nova técnica, mas uma modificação de outras. A idéia consiste em dividir o cabeçalho TCP em vários pacotes, ao invés de mandá-lo em um único. O objetivo é burlar os filtros de pacotes e conseguir penetrar na rede sem ser detectado. Esse método não consegue passar despercebido por NIDS e *firewalls* que fazem remontagem de pacotes.
5. **Reverse Ident Scanning**: O protocolo *ident* [8] permite descobrir que usuário é dono de processos conectados via TCP, mesmo que o processo não tenha iniciado a conexão. Dessa forma, por exemplo, é possível se fazer uma conexão à porta 53 (DNS) e usar o *daemon* do programa *ident* para descobrir se o servidor de DNS está executando como usuário *root*.
6. **UDP e ICMP Port unreachable** : Essa técnica de varredura possui um grau de dificuldade maior que a das portas TCP, pois as portas do protocolo UDP não enviam pacotes de confirmação e as portas desativadas não precisam necessariamente enviar respostas de erro. Entretanto, alguns sistemas operacionais enviam mensagens de erros do tipo *ICMP_PORT_UNREACH*, quando enviamos um pacote para uma porta UDP fechada. Assim, podemos distinguir entre as portas ativas e inativas.

2.2.2 Identificação Remota de Sistemas Operacionais

Como evolução, os programas de *portscan* podem identificar remotamente o tipo e a versão do sistema operacional de uma máquina. Com esse objetivo, uma das primeiras tentativas foi a utilização de métodos para a execução dos comandos **telnet** ou **ftp** para uma máquina remota e observar a mensagem de *login*. Por exemplo,

```
$> telnet alvo
trying 192.168.2.10
connected to alvo
Escape character is '^]'
Linux 2.1.14
```



```

login:
$> telnet alvo 21
trying 192.168.2.10
connected to alvo
Escape character is '^]'
220 alvo FTP server Version 6.00LS) ready.
ftp> syst
215 UNIX Type: L8 Version: BSD-199506

```

Os problemas na utilização dessas técnicas são os filtros de conexões e a eventual modificação de mensagens de abertura realizadas pelos administradores de sistemas. Devemos considerar que esconder ou dar informações falsas também é uma técnica de segurança. Estes fatos, dificultam o trabalho do invasor, fazendo com que novos procedimentos sejam tentados.

Outra alternativa para a busca de informações é o estudo de como cada sistema operacional implementa o protocolo TCP/IP. A tática consiste em enviar pacotes TCP, UDP ou IP, modificados intencionalmente, usando as portas que não estão sendo filtradas. O cerne desse processo é a observação das diferenças de comportamento entre os diversos sistemas operacionais. Em seguida, são escritos programas que exploram as diferenças na tentativa da identificação, agora com alta margem de sucesso, do sistema operacional da máquina alvo [9]. A seguir, destacamos algumas das técnicas utilizadas :

1. **Técnica FIN:** Quando enviamos um pacote FIN (ou qualquer outro pacote sem um ACK ou SYN) para uma porta desabilitada, não deveríamos receber qualquer resposta, segundo [10]. Contudo, algumas implementações do protocolo em sistemas operacionais específicos respondem.
2. **Técnica do Parâmetro não Definido:** A idéia consiste em enviar um parâmetro do TCP não definido (64 ou 128) no cabeçalho TCP de um pacote SYN. As versões de *kernel* de Linux anteriores a 2.0.35 mantinham esse parâmetro na resposta; outros sistemas operacionais terminam a conexão.
3. **Técnica do Número de Sequência Inicial:** O fundamento dessa técnica é encontrar padrões nos números de sequência inicial, escolhidos por algumas implementações TCP, ao responderem um pedido de abertura de conexão. Os valores podem ser os seguintes:
 - (a) Incrementos de 64k;
 - (b) Incrementos randômicos;
 - (c) Incrementos variáveis com a hora do dia;
 - (d) Constantes.

4. **Técnica do Bit de não-Fragmentação:** Muitos sistemas operacionais usam o bit de não-fragmentação e alguns o fazem em certos casos. Dessa forma, ao observarmos esse comportamento, podemos obter informações sobre a máquina analisada.
5. **Técnica da Janela Inicial do TCP:** Envolve a verificação do tamanho da janela em pacotes de resposta. Esses valores são constantes de acordo com o sistema operacional. A verificação do tamanho da janela do TCP é um dos testes mais importantes na descoberta do sistema operacional remoto, uma vez que cada implementação possui valores próprios para o tamanho da janela e isso é usado para diferenciá-los.
6. **Valores de ACK:** Os valores do campo ACK variam muito nas implementações. Alguns são o mesmo da sequência do emissor, quando este último envia um pacote com os campos FIN, PSH, URG habilitados. Outros colocam o valor acrescido de 1.
7. **ICMP Error Messages Quenching:** Alguns sistemas operacionais obedecem fielmente as especificações contidas na RFC-1812 [11], que sugere que a taxa de mensagens de erros ICMP possua um limite máximo. Por exemplo, o kernel do Linux limita a taxa de mensagens do tipo ICMP *destination unreachable* a 80 a cada 4 segundos, com uma penalidade de 1/4 de segundo se o limite for alcançado. A maneira de descobrirmos isso é enviar uma certa quantidade de pacotes UDP para uma porta qualquer e contarmos o número de mensagens recebidas.
8. **Respostas de Mensagens de Erros ICMP:** Algumas informações são acrescentadas nas mensagens de erros ICMP. Com base nessas informações adicionais, é possível identificar vários sistemas. Isso permite, por exemplo, descobrir o sistema operacional de uma estação, mesmo que ela não possua uma única porta ativa.
9. **Manipulação de Fragmentos de Pacotes IP:** Certos sistemas possuem métodos próprios de manipular fragmentos. Dessa forma, se enviarmos fragmentos que sobre-escrevem os anteriormente enviados, poderemos comparar as respostas e diferenciar os sistemas.
10. **Utilização de Opções no Cabeçalho TCP:** Assim como o tamanho da janela TCP é uma boa fonte de informação, as opções do cabeçalho TCP também fornecem dados particulares de cada implementação. As grandes vantagens desse método são:
 - (a) Os campos são opcionais, portanto nem todos os sistemas os implementam;
 - (b) Sabemos que uma estação implementa-os se enviarmos um pacote com os campos;
 - (c) Podemos enviar um pacote com todas as opções ativas para testar tudo de uma única vez.

Alguns sistemas suportam todas as opções de cabeçalho; outros, somente algumas delas. Podemos refinar mais ainda nossa pesquisa se observarmos os valores retornados.

11. **Cronologia de Falhas de Segurança** : Algumas versões de sistemas operacionais não modificaram muito suas pilhas TCP/IP ao longo dos anos, o que pode resultar em alguma dificuldade em diferenciá-los. Contudo, se as testarmos contra falhas de segurança apropriadas, poderemos ter uma idéia da versão e qual a última vez que correções foram aplicadas.
12. **Resistência a Ataques *SYN Flooding*** : As versões mais antigas de sistemas operacionais param de aceitar novas conexões, se um atacante executar um ataque *SYN Flooding* em uma porta específica. Esse tipo de ataque será analisado posteriormente.

2.3 Ataques por Falsificação de Endereços IP

Toda instituição conectada à Internet recebe uma faixa de endereços IP para colocar em suas máquinas. O protocolo IP é flexível o bastante para permitir que os endereços sejam alterados pelo administrador à sua conveniência. Alguns ataques consistem em atribuir a uma máquina remota um endereço fonte igual ao de uma máquina interna de uma rede local.

No cabeçalho do pacote IP existe um campo chamado *Source Address* (endereço fonte ou endereço de origem) [12]. Para realizar certos tipos de ataques, um atacante precisa criar um pacote contendo um endereço de origem falso e enviá-lo através da Internet. Alguns programas de ataque permitem que o endereço fonte seja especificado na linha de comando, no momento da execução do mesmo.

Um roteador tem como função principal rotear pacotes que chegam pelas suas interfaces. Quando um pacote com endereço de origem falso chega em uma delas, o roteador simplesmente o direciona para o seu destino, assim como faria com qualquer outro pacote.

A falsificação de endereços tem o objetivo de enganar serviços que autenticam máquinas baseados em endereços IP. Ao explorar esse método, os intrusos tentam ganhar acesso a uma conta de usuário, ou a conta do administrador, conta de *root*. Esse mesmo procedimento também pode ser usado para implicar uma determinada máquina como fonte de um ataque.

Os serviços que são mais vulneráveis aos ataques de falsificação de endereços são o *proxy* *firewalls*, o NFS (*Network File System*), os comandos *r* do Unix, o XWindow, os programas de filtros de acesso ou outros que façam uso do endereço IP como forma de autenticação.

2.3.1 Evitando Ataques por Falsificação de Endereços IP

Para minimizar as tentativas de ataques baseados na falsificação de endereços IP de origem, é aconselhável reduzirmos ao mínimo necessário a utilização de serviços que se baseiem em

autenticação via endereço IP. Como medida adicional, devem ser desabilitados todos os comandos `r`, removidos todos os arquivos `.rhosts` dos usuários e esvaziado o arquivo `/etc/hosts.equiv`.

Outro procedimento utilizado para prevenir a falsificação do campo de endereço origem do protocolo IP é a filtragem de pacotes. O filtro é implementado pelos roteadores dos provedores de acesso, que devem rotear apenas os pacotes que possuem cabeçalho IP com campo de endereço origem pertencentes à faixa atribuída às suas próprias redes locais. Os pacotes IP com o campo de endereço origem fora da faixa do provedor também devem ser bloqueados. Além disso, na confecção dos filtros de roteadores com conexão à Internet, é aconselhável o bloqueio de endereços privados. Estes endereços, definidos na RFC-1918 [13], são os endereços 10.x.x.x, 172.16-32.x.x e 192.168.x.x (onde x indica qualquer valor entre 0 a 255).

Como medida adicional de segurança, os administradores de rede devem configurar equipamentos para registrar em arquivo informações dos pacotes que sejam descartados por estarem utilizando endereços não permitidos. Essa atitude fornecerá subsídios para monitorar qualquer atividade suspeita.

Proteções adicionais devem ser implementadas em determinados equipamentos utilizados por provedores de acesso [14], como os servidores de acesso remoto (RAS - *Remote Access Server*). Estes equipamentos permitem que usuários domésticos acessem o provedor via acesso discado. Os únicos endereços IP que devem passar pelo RAS são aqueles fornecidos pelo provedor. O servidor de acesso deve verificar, através de filtros, todos os pacotes enviados pelos usuários para garantir que não haja falsificações.

2.4 Ataques por Negação de Serviços

Os ataques por negação de serviços (DoS - *Denial of Service*) nem sempre são precedidos de invasões para serem bem sucedidos. Este tipo de ataque aproveita falhas dos protocolos e programas servidores para deixarem uma máquina, ou até mesmo uma rede inteira, impossibilitada de executar sua tarefa.

2.4.1 Ataque *smurf*

O ataque *smurf* [15] tem seu nome baseado no programa criado para explorar uma falha no esquema de endereçamento do protocolo IP. Seu funcionamento [16], consiste no envio de um pesado tráfego de ICMP *echo* ping, no endereço IP broadcast, de uma rede intermediária, contendo o endereço da vítima como origem do ataque. Se o roteador da rede intermediária não bloquear os pacotes, as estações dessa rede os receberão e responderão a esses ECHO REQUESTs com um ECHO REPLY cada uma, multiplicando o tráfego pelo número de respostas. Em uma rede grande, pode haver centenas de máquinas respondendo, aumentando a intensidade desse ataque. A Figura 2.1 mostra a arquitetura desse ataque.

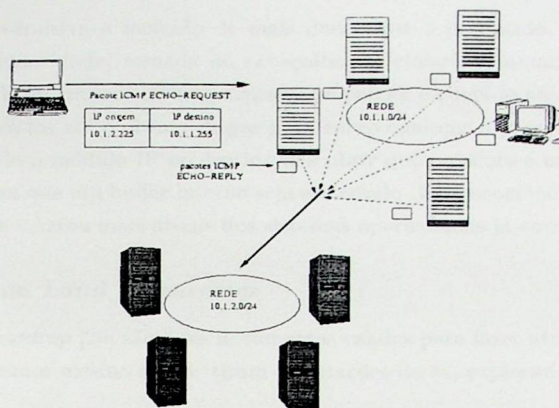


Figura 2.1: Arquitetura de um ataque do tipo *smurf*.

Existe uma variação do ataque *smurf* chamado *fraggle* [17], que usa os pacotes *echo* do protocolo UDP da mesma maneira que é usado o ICMP pelo *smurf*. O *fraggle* pode ser considerado como uma nova versão do *smurf*.

O ataque *smurf* consegue atingir duas redes distintas numa só ação. As redes atingidas são:

1. A rede intermediária: constituída pelas estações de uma rede local usada para receber o *broadcast* e respondê-lo. Também são conhecidas como máquinas amplificadoras;
2. A rede alvo: constituída pelas estações vítimas. Recebe todo o tráfego gerado pelas máquinas amplificadoras.

2.4.2 Ataques com Pacotes IP Grandes

O ataque com pacotes IP grandes, 64Kb, pode ser lançado através do comando *ping*, existente na maioria dos sistemas operacionais. Embora seja simples de ser feito, seus efeitos são bastante nocivos. São exemplos de alguns desses efeitos a paralisação do equipamento por algum tempo e volta ao normal, a paralisação total, o *core dump* (escrita do conteúdo da memória em disco) seguida de uma pane geral, a perda das conexões de rede ou a reinicialização abrupta do equipamento atacado [18]. Esse ataque explora uma falha no protocolo IP. Mais especificamente, na fragmentação e remontagem de pacotes. O tamanho de um pacote IP é determinado por um campo do cabeçalho de 16 bits de comprimento. Desse forma, um pacote pode conter até 65.535 bytes. O cabeçalho do IPv4 possui tamanho padrão de 20 bytes, ficando 65.515 bytes destinados aos dados. Como os protocolos de enlace não lidam com quantidades de dados desse tamanho, a fragmentação do pacote é necessária.

O lançamento do ataque não requer maiores esforços, pois o atacante constrói um pacote grande e o envia. O problema está na estação destino onde os pacotes serão remontados.

Algumas aplicações permitem a inclusão de mais dados que o permitido, como por exemplo 65.520 bytes. Essa quantidade, somada ao cabeçalho IP violará o tamanho máximo de um pacote. O protocolo IP no emissor irá fragmentar esse pacote e enviá-lo ao destino. O receptor irá recebê-los e enfileirá-los até o último chegar para então começar o processo de remontagem. O efeito surgirá quando o módulo IP no destino descobrir que o pacote é maior que o máximo permitido, fazendo com que um buffer interno seja estourado (há a ocorrência de um *overflow*). Devemos salientar que versões mais atuais dos sistemas operacionais já corrigem esta falha.

2.4.3 Ataques com *Land* e *Teardrop*

O *Land* [19] e o *Teardrop* [20] são duas ferramentas usadas para fazer ataques de negação de serviços. Ambas causam o mesmo efeito: tiram as estações do ar, explorando vulnerabilidades diferentes [21].

O *Teardrop* utiliza falhas na fragmentação e remontagem de pacotes IP. Ele tem seu funcionamento baseado no envio de pacotes IP grandes (maiores que 65.535 *bytes*), visando derrubar o sistema operacional da máquina alvo.

O ataque *Land* faz uso do serviço *echo* que algumas estações disponibilizam. Esse serviço permite que um cliente envie informações a um servidor, fazendo com que o mesmo responda ecoando a informação. O problema surge quando o endereço IP origem e a porta origem são alterados para ficarem iguais ao do servidor, fazendo-o responder para si mesmo em um processo infinito. A Figura 2.2 mostra o diagrama de um ataque utilizando a ferramenta *Land*.

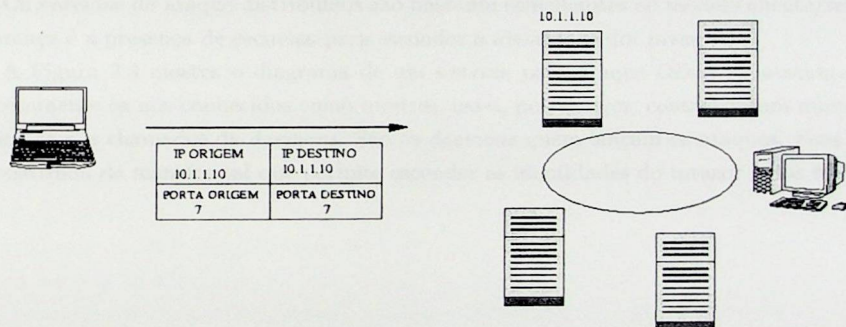


Figura 2.2: Arquitetura de um ataque com a ferramenta *Land*.

2.4.4 Ataque *SYN Flooding*

O ataque *SYN Flooding* consiste em enviar um fluxo de pacotes *SYN* para uma porta em um servidor numa velocidade maior que a mesma possa processar.

O problema surge quando o servidor envia uma confirmação (*SYN-ACK*) para o cliente, não recebendo de volta a mensagem confirmando esse *SYN-ACK*. Esse estado é conhecido como conexões

meio abertas e, geralmente, o tempo de espera dessas confirmações é de 75 segundos. O servidor possui uma estrutura de dados contendo todas as conexões pendentes. Essa estrutura possui, naturalmente, uma limitação e pode alcançar o máximo permitido através do recebimento de mais mensagens SYN.

Algumas conexões meio abertas são realizadas através da falsificação de endereços IP, com o endereço origem igual aos endereços de redes privadas da Internet. O atacante preenche o endereço fonte com um desses endereços, tornando o servidor incapaz de responder às mensagens SYN-ACK.

2.4.5 Ataques DDoS

Uma forma mais sofisticada de ataque por negação de serviço é chamada de DDoS (*Distributed Denial of Service Attack* - ataque distribuído de negação de serviço). De acordo com Mixer [22], os ataques DDoS possuem a forma de uma avalanche de pacotes, lançados com o objetivo de sobrecarregar canais de comunicação. Ataques DDoS podem servir, também, para quantificar o volume de tráfego que uma estação pode suportar. Como exemplo de ferramentas DDoS citamos o TFN2K [23], o *trino* [24], e o *stacheldraht* [25].

Os ataques DDoS baseiam-se na aplicação, em larga escala, de ataques DoS consistindo de uma avalanche de pacotes UDP, TCP (*SYN Flooding*), ICMP e *smurf* contra uma estação ou contra uma rede. Os atacantes invadem, preferencialmente, máquinas em redes que possuem conexões de alta capacidade com a Internet.

Os sistemas de ataque distribuídos são bastante semelhantes ao modelo cliente/servidor. A diferença é a presença de recursos para esconder a identidade dos invasores.

A Figura 2.3 mostra o diagrama de um sistema para ataque DDoS. O atacante controla remotamente os nós conhecidos como mestres, esses, por sua vez, controlam um número muito maior de nós chamados de *daemons*. São os *daemons* quem lançam os ataques. Essa estrutura é construída de maneira tal que permite esconder as identidades do invasor e dos mestres [26].

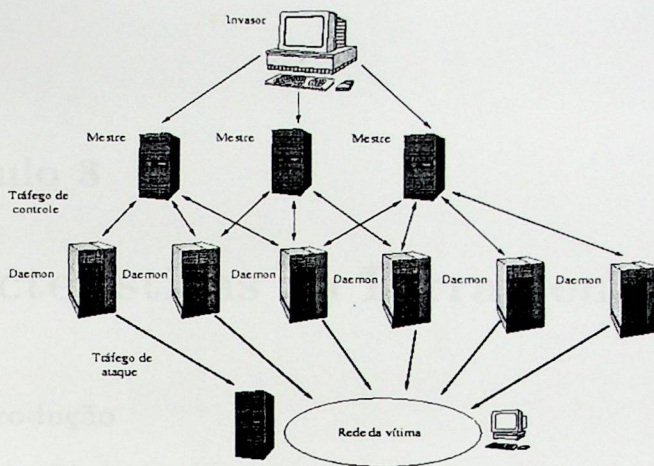


Figura 2.3: Arquitetura do ataque distribuído de negação de serviço (DDOS).

Para prevenir a descoberta dos mestres e a localização do atacante, os comandos entre eles são transmitidos de forma encriptada, usando o ICMP como protocolo de transporte. Isso dificulta a sua identificação por ferramentas de detecção, que monitoram o tráfego em busca de padrões de ataques.

Alguns tipos de ataques não requerem que a conta *root* seja comprometida, podendo ser lançados com os privilégios de usuários comuns. Somente quando o ataque requer a escrita em *sockets* brutos (*raw sockets*) é que os privilégios de administrador são necessários. Geralmente, uma estação invadida tem seus utilitários de sistemas (*ps*, *netstat* e *ifconfig*) modificados para esconder a presença dos programas. Os mestres também podem ser instalados em contas de usuários comuns, deixando ainda menos evidente a invasão.

Nesse Capítulo, apresentamos algumas técnicas de ataque a estações e a redes. Essas técnicas serão utilizadas, no Capítulo 5, para testar a eficiência da ferramenta objeto do presente trabalho de dissertação de mestrado.

No próximo Capítulo, apresentaremos as características da ferramenta **Block**.

Capítulo 3

Características da Ferramenta Block

3.1 Introdução

A ferramenta **Block** foi projetada para plataformas Linux e implementada sobre um *NIDS* (*Network Intrusion Detection system*). Na versão desenvolvida neste trabalho, o *NIDS* escolhido usa o paradigma de reconhecimento de ataques por assinatura. O **Block** recebe o resultado da análise do tráfego realizada pelo *NIDS*, prepara a lista de acesso correspondente e a envia para roteadores Cisco como resposta automática a possíveis ataques.

O *NIDS* selecionado como detector de ataques foi o **watcher** [1]. O **watcher** é um programa *free* e de código aberto. Ele está escrito em linguagem C e atua processando os cabeçalhos dos protocolos TCP, IP e UDP. Por ter o código fonte disponível, é possível se fazer a inserção de novas assinaturas de ataques no **watcher**.

3.2 Sistemas Detectores de Intrusões em Rede - *NIDS*

Os *IDS* baseados em rede, os *NIDS*, analisam os pacotes que trafegam em uma rede local. Cada pacote é aberto, seu conteúdo examinado e, dependendo do programa, a análise é feita até a camada de aplicação. As informações coletadas são comparadas com padrões de ataques ou assinaturas para verificar sua natureza. A vantagem desse modelo é que ele consegue cobrir uma rede inteira, independentemente dos tipos de sistemas operacionais e de dispositivos presentes. Para o funcionamento dos *NIDS*, é necessário que o computador onde ele esteja funcionando opere com sua placa de rede em modo promíscuo em uma rede Ethernet. Na Figura 3.1, mostramos o diagrama de funcionamento de um *NIDS*.

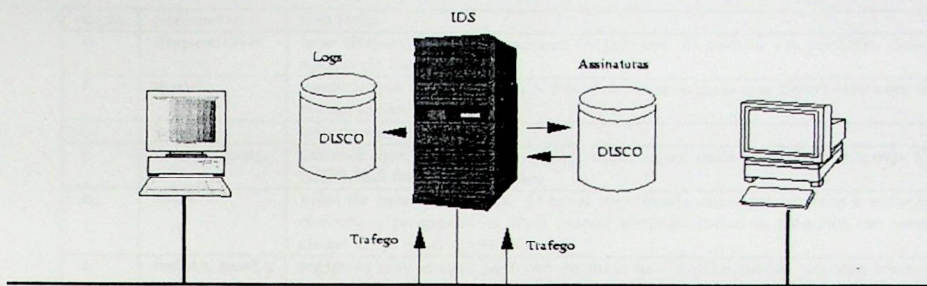


Figura 3.1: Solução de IDS baseada em rede Ethernet.

3.3 O Programa watcher

O **watcher** é um *NIDS* com um banco de assinaturas de ataques incluído no próprio código fonte. Ele possui parâmetros para a detecção de anomalias na rede (taxa de *SYN Flooding*, *echo_reply*), podendo ser configurados pelo usuário através de opções passadas no nível de linha de comando.

O **watcher** verifica todos os pacotes capturados por uma interface de rede e presume que eles originam-se de conexões hostis. Esse exame é realizado dentro de uma janela de tempo igual a 10 segundos. Após esse período, as informações sobre possíveis ataques são armazenadas no arquivo *syslog*. O **watcher** trabalha ativando gatilhos pré-configuráveis que indicam quando um ataque está ocorrendo. Esses gatilhos se baseiam em perfis de tráfego considerados normais. Se dentro da janela de monitoração, esses perfis se desviarem do usual, eventos suspeitos serão armazenados. Podemos, por exemplo, configurá-lo para reconhecer somente ataques do tipo DoS ou ataques por varreduras de portas. Nesse último caso, a ocorrência de recebimento de pacotes por 7 portas diferentes dentro da mesma janela de tempo, caracteriza um evento de *portscan*, por exemplo.

Na versão utilizada, são detectados os ataques por varreduras TCP e UDP, *SYN Flooding*, *Teardrop*, *Land*, *smurf* e ping da morte. As opções do programa **watcher** são mostradas na Tabela 3.1.

opção	parâmetro	descrição
-d	dispositivo	usar dispositivo para captura de pacotes. O padrão é o primeiro disponível após o de <i>loopback</i> .
-f	flood	assumir que um ataque <i>SYN Flood</i> ocorreu se mais que flood conexões incompletas forem recebidas.
-h	—	tela de ajuda.
-i	limite_icmp	assumir que um ataque <i>smurf</i> ocorreu se mais que limite_icmp ECHO REPLIES forem detectados.
-m	nível	nível de monitoramento. O nível <i>myselfonly</i> abrange somente a estação que executa o programa; o nível <i>subnet</i> abrange todas as estações em uma rede classe C; já o <i>all</i> abrange todos.
-p	limite_porta	registrar um ataque <i>portscan</i> se mais que limite_porta pacotes forem recebidos.
-r	tipo	se o tipo for igual a DoS, somente ataques de negação de serviços são registrados; caso seja <i>scan</i> , somente as varreduras serão registradas; se o tipo for igual a <i>all</i> , todos serão registrados.
-t	timeout	contar os pacotes e imprimir ataques a cada ciclo de timeout segundos.
-w	webcount	assumir um ataque <i>portscan</i> está ocorrendo se mais que webcount pacotes recebidos forem direcionados para a porta 80 (ver detalhes na seção 4.3.2).

Tabela 3.1: Opções de linha de comando do programa **watcher**.

O banco de assinaturas de ataques do **watcher** foi ampliado para incluir as assinaturas de ataques a portas TCP e UDP utilizadas pelas ferramentas de ataque *backorifice* [27] e *netbus* [28].

O usuário pode configurar um arquivo contendo portas TCP e outro contendo portas UDP que devam ser observadas. Caso alguma porta presente em um desses arquivos seja encontrada, o pacote correspondente é classificado como ataque.

O tempo de resposta da ferramenta a um ataque depende do valor colocado no parâmetro **-t** (*timeout*) somado ao tempo de colocação da lista de acesso no roteador. Isso pode variar entre 4 e 5 segundos.

3.4 O projeto do Block

O **Block** pode proteger a própria estação onde ele está instalado, redes /24 ou todas as estações destino. Esses perímetros de proteção definem níveis de bloqueio contra ataques que são baseados nas regras de construção de listas de acesso dos roteadores Cisco. A capacidade de filtragem oferecida pelas listas de acesso permite definir quatro níveis de proteção contra os ataques. Estes níveis oferecem proteção de estação para estação, de uma rede para uma estação, de uma rede para todas as estações e de todas as estações para todas as estações.

Após um prévio conhecimento do perfil do tráfego da rede a ser protegida e feitos os ajustes necessários na sua configuração, o **Block** detecta as formas de ataques mencionadas no Capítulo 2 e os ataques a portas TCP e UDP, gerando listas de acesso para bloqueá-las segundo os níveis de proteção especificados no parágrafo anterior. A ferramenta implementa diferentes níveis de

proteção dependendo do tipo de ataque detectado, como veremos no Capítulo 4.

Ao ser detectado um ataque, uma mensagem de correio eletrônico é enviada ao administrador, é gravada uma entrada no arquivo *syslog*, gerada a lista de acesso correspondente e enviada ao roteador.

3.5 Bloqueando Tráfego não Autorizado e Ataques

Basicamente, existem duas formas de bloquear ataques de tráfego não autorizado. A primeira, é protegendo cada máquina de uma rede local individualmente. Nesse caso, a proteção é feita a partir da instalação e/ou configuração do suporte adequado disponível para cada estação. A segunda, é a proteção da rede inteira sendo realizada através da implantação de listas de acesso ou filtros em roteadores e *firewalls*. Estas listas funcionam como bloqueadores que abrangem toda a rede, protegendo suas máquinas, independentemente do sistema operacional que elas possuam. Pode-se, se necessário, construir filtros personalizados para cada máquina presente na rede a partir de um ponto central de entrada do tráfego. A ferramenta **Block** baseia a sua proteção em listas de acesso implementadas em roteadores Cisco.

3.6 Listas de Acesso

As listas de acesso são mecanismos de segurança que podem ser implementados em roteadores. As listas permitem a filtragem do tráfego na entrada de uma rede local, aliviando o processamento em possíveis mecanismos de filtragem internos.

A filtragem de pacotes dos roteadores permite a um administrador de rede restringir o acesso a certas estações, segmentos de redes, blocos de endereço e portas TCP/UDP baseados numa variedade de critérios. Esses filtros são a primeira barreira para bloquear ataques vindos de redes externas e, quando bem implementados, podem evitar um grande número deles.

Os recursos de filtragem do sistema operacional dos roteadores Cisco, o IOS (*Internetwork Operating System*), permitem a criação de listas de acesso para filtrar pacotes baseados nos seguintes critérios [29]:

1. Endereços IP de origem e de destino;
2. Protocolos TCP, UDP e ICMP;
3. Portas de origem e/ou destino do TCP e UDP, e
4. Serviços do protocolo ICMP.

A implantação de uma lista de acesso em um roteador Cisco é feita em duas etapas. Na primeira, há a definição dos critérios do filtro de pacotes e, na segunda etapa, ocorre a aplicação dos critérios de filtragem nas interfaces do roteador.

3.6.1 Definindo uma Lista de Acesso

Os critérios de filtragem são definidos em uma lista de declarações de permissão (*permit*) e negação (*deny*). Cada linha da lista é avaliada em sequência até que haja uma coincidência. Assim que a coincidência ocorre, a lista é abandonada e a conexão pretendida será negada ou permitida. Por ser sequencial, esse processo é dependente da ordem na qual as declarações são colocadas. Uma lista mal configurada, pode oferecer uma falsa sensação de segurança, pois uma entrada que negue um tráfego pode não ser processada devido a uma outra entrada que esteja posicionada mais acima na lista e que permita tal tráfego.

As listas de acesso podem ser identificadas por nomes ou números. As listas nomeadas possuem um nome fornecido pelo administrador. É recomendável que o nome da lista seja referente à função executada para facilitar a identificação.

O comando para criar uma lista nomeada é:

```
ip access-list
```

enquanto o comando para criar uma lista numerada é :

```
access-list
```

As listas nomeadas ou numeradas podem possuir duas categorias: padrão (*standard*) ou estendida (*extended*). Uma lista padrão somente avalia o endereço IP origem. Uma lista estendida possui declarações que permitem avaliar endereços origem/destino, o tipo de protocolo e as portas origem/destino.

O comando de criação de listas numeradas exige como parâmetro o número da lista. As listas padrão são numeradas de 1 a 99, inclusive. Para as listas estendidas, a numeração é entre 100 e 199, inclusive. Após o número da lista de acesso, segue-se uma das palavras chaves *permit* ou *deny* seguidas por um endereço IP, pela máscara de rede, pelo protocolo e pelo número da porta.

Para ilustrar os procedimentos apresentados nos parágrafos precedentes, mostramos na Figura 3.2 um exemplo de uma lista padrão (número de sequência igual a 2) implementada em um roteador Cisco. Após a digitação das declarações da lista, é necessário digitar a sequência de teclas *control* e a letra *z* ou a palavra *exit*.

```
router# configure t
Enter configuration commands, one per line. End with CNTL/Z
router(config)# access-list 2 deny 10.0.0.1
router(config)# access-list 2 permit 10.0.0.0 0.255.255.255
router(config)#~z
```

Figura 3.2: Exemplo de construção de uma lista de acesso do tipo *standard*.

No exemplo mostrado, a lista proíbe pacotes com endereço IP origem igual a 10.0.0.1 e permite qualquer pacote com endereço IP origem da rede 10.0.0.0. Como mencionado anteriormente, a ordem das declarações é importante em uma lista de acesso. Por exemplo, a inversão

das declarações da lista de acesso mostrada na Figura 3.2 fará com que nenhum pacote seja filtrado. Para que uma lista seja efetiva, devemos associá-la a uma interface.

O exemplo mostrado na Figura 3.3 refere-se a uma lista numerada estendida (número de sequência 110) que permite o tráfego de SMTP para a máquina 10.0.0.20 vindo de qualquer máquina da Internet (identificador *any*) e o tráfego de **pop3** para a máquina 10.0.0.22, também vindo de qualquer origem na Internet. Por padrão, o roteador inclui uma regra que nega a passagem de todos os pacotes. Esta regra é colocada no final da lista, não aparecendo. Caso seja necessário, coloca-se o comando `permit ip any any` no final da lista para permitir a fluência normal do tráfego.

```
router# configure t
Enter configuration commands, one per line. End with CNTL/Z
router(config)# access-list 110 permit tcp any host 10.0.0.20 eq smtp
router(config)# access-list 110 permit tcp any host 10.0.0.22 eq pop3
router(config)#^z
```

Figura 3.3: Exemplo de construção de uma lista de acesso do tipo *extended*.

Para criarmos uma lista IP rotulada com nome, o comando `ip access-list` exige dois parâmetros: o tipo da lista, que pode ser *standard* ou *extended* e o nome da lista. Após a criação da lista, o IOS aceitará somente as palavras *permit* ou *deny* junto com os critérios do filtro. O nome da lista não necessita ser repetido em todas as linhas. Na Figura 3.4, apresentamos um exemplo de lista estendida usando nomes.

```
router# configure t
Enter configuration commands, one per line. End with CNTL/Z
router# ip access-list extended lista-correio
router(config-ext-nacl)# permit tcp any host 200.200.200.20 eq smtp
router(config-ext-nacl)# permit tcp any host 200.200.200.22 eq pop3
router(config-ext-nacl)#^z
```

Figura 3.4: Exemplo de construção de uma lista de acesso do tipo *extended* usando nomes.

3.6.2 Aplicando uma Lista de Acesso

Como dissemos anteriormente, uma lista de acesso, para ser efetiva, deve ser aplicada a uma das interfaces de rede de um roteador. A aplicação pode ser feita tanto no tráfego de saída como de entrada da interface. Uma lista de acesso é aplicada através do comando:

```
ip acces-group
```

Esse comando requer dois parâmetros: o número ou nome da lista e palavra chave *in* (entrada) ou *out* (saída).

No exemplo mostrado na Figura 3.5, apresentamos a aplicação de uma lista de acesso de saída na interface FastEthernet 0 de um roteador.


```

router# configure t
Enter configuration commands, one per line. End with CNTL/Z
router(config)# interface FastEthernet 0
router(config-if)# ip access-group 1 out
router(config-if)# ^z

```

Figura 3.5: Exemplo da aplicação de uma lista de acesso.

Após configuradas, as listas de acesso podem ser vistas e verificadas através dos comandos: `show acces-lists` e `show ip acess-lists`.

3.7 O Ambiente de Aplicação

O **Block** precisa ser instalado em uma máquina que consiga escutar todo o tráfego da rede para poder detectar os ataques, podendo ser usado tanto na DMZ (*Demilitarized Zone* - Zona Desmilitarizada) quanto na rede interna. É importante notar que o dispositivo utilizado pelo **Block** deve ser um roteador Cisco.

Quando usada a sua configuração padrão, os eventos serão rotulados como ataques nas seguintes condições:

1. Se mais de 8 pacotes SYN para a mesma porta são vistos na janela de tempo , sem nenhum ACK ou FIN associado, um evento de *SYN Flooding* é registrado;
2. Se um pacote UDP fragmentado com o valor do campo IP id igual a 242 for visto, é registrado um ataque de *Teardrop*;
3. Pacotes TCP SYNs com endereços fonte e destino e portas iguais são classificados como ataque *Land*;
4. Se mais de 5 mensagens ICMP ECHO REPLY são vistas, é assumido um ataque *smurf*;
5. Qualquer pacote ICMP fragmentado é visto como um ping da morte;
6. Se for detectada uma porta destino de uma conexão TCP ou UDP que esteja contida no arquivo `tcp.dat` ou `udp.dat`. Estes arquivos serão definidos no Capítulo 4;
7. Se mais que 7 conexões forem abertas vindas de um mesmo endereço origem para portas distintas, é assumido um ataque *portscan*.

Todos esses parâmetros são configuráveis em um arquivo de configuração, dessa forma, adaptando-se ao perfil do tráfego da rede local.

É importante notar que a ferramenta trata todos os pacotes como potencialmente agressivos. Isso pode gerar alguns falsos positivos, pois parte do tráfego normal pode ser rotulado como

ataque. É importante que o administrador conheça o perfil do tráfego da sua rede para minimizar esses erros. Há opções que permitem aumentar o nível de tolerância com relação a certas atividades. Infelizmente, isso também pode elevar a taxa na qual as varreduras podem ser feitas antes de serem descobertas. As respostas falsas mais comuns são varreduras TCP e *SYN Flooding*, a maioria resultante de acesso a servidores WWW. Isso é devido a páginas que possuem muitas imagens e outros tipos de dados, causando uma nova conexão TCP para obter cada arquivo. O **Block** enxerga isso como uma varredura TCP. Para minimizar essa situação, a ferramenta somente registrará varreduras TCP se mais de 40 requisições forem recebidas na janela de tempo e a porta origem for igual a 80. Todos esses valores são configuráveis. Esse mesmo expediente é válido para ataques *SYN Flooding*, pois um cliente pode abrir várias conexões com o servidor WWW antes da expiração da janela de 10 segundos, fazendo o **Block** atuar com se esse cliente estivesse fazendo um *SYN Flooding* na porta 80 de um servidor localizado na rede sob proteção.

Os relatórios produzidos a cada duração da janela são gravados no arquivo *syslog*, consistindo do ataque detectado e do endereço IP origem e destino. Dependendo do tipo de ataque, mais informações como o número de pacotes e portas envolvidas são arquivadas. Se o ataque for resultante de um endereço IP falso, o endereço MAC também é guardado. No caso de um ataque externo, o endereço MAC do roteador será guardado.

3.7.1 Níveis de Proteção do Block

O **Block** foi projetado objetivando fornecer flexibilidade no bloqueio de ataques explorando os recursos presentes nas listas de acesso. Nas Figuras 3.6, 3.7, 3.8, e 3.9, a porta 12345 foi utilizada como exemplo. Quatro níveis possíveis de proteção estão disponíveis :

1. **Proteção estação para estação** : É o tipo mais básico de proteção. A ferramenta funciona bloqueando conexões entre a máquina origem do ataque e a máquina alvo. Caso o atacante decida atacar máquina a máquina, a ferramenta irá gerar uma entrada na lista de acesso para proteger cada uma das máquinas alvo. Neste caso, a lista de acesso criada pode ser grande. Convém lembrar que quanto menor a lista, mais rápidas fluirão as conexões, pois a lista inteira sempre é consultada em cada conexão. Esse nível não é recomendado para ambientes conectados a redes públicas como a Internet, mas pode ser implementado em ambientes de *intranets*, visando descobrir ataques internos como mostrado na Figura 3.6.

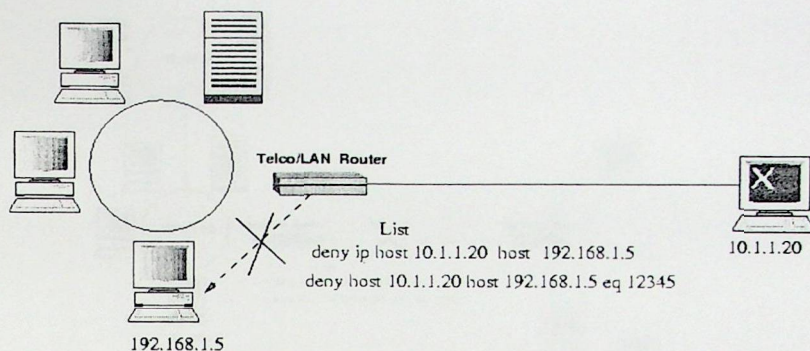


Figura 3.6: Bloqueio de estação para estação.

2. **Proteção rede para estação** : Esse nível bloqueia todo o tráfego da rede do atacante direcionado à máquina alvo. É um bom nível de proteção, pois toda a rede hostil fica sem comunicação com a máquina atacada. Entretanto, caso o atacante vise outra máquina da rede local, será criada uma nova entrada na lista, aumentando-a. Essa modalidade também pode ser útil entre os departamentos de uma grande empresa, mas não é muito apropriada a conexões com redes externas. A Figura 3.7 mostra um diagrama dessa proteção.

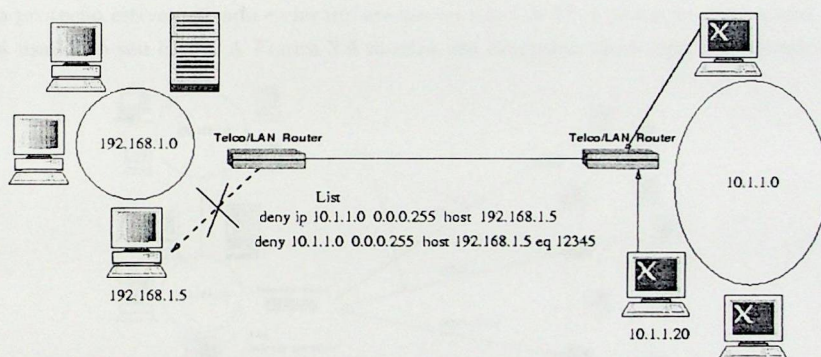


Figura 3.7: Bloqueio de rede para estação.

3. **Proteção rede para todos** : Essa proteção é a mais efetiva de todas. Ela bloqueia toda a rede do atacante protegendo, desta maneira, qualquer máquina da rede local. A grande vantagem desse nível de proteção é que a rede local pode até estar segmentada em várias subredes, mas o filtro *any* irá bloquear as conexões para todos os segmentos, independente delas terem endereços de rede diferentes. Esse é o filtro ideal para redes com conexões à Internet. Na Figura 3.8, mostramos um diagrama representado esse nível de proteção.

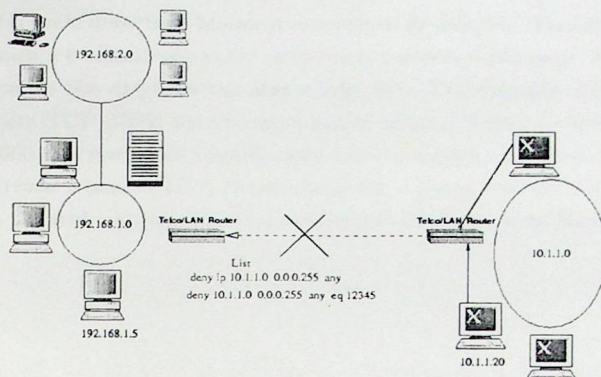


Figura 3.8: Bloqueio de rede para todos.

4. **Proteção todos para todos** : Esse tipo de proteção é eficiente no bloqueio de portas usadas por *exploits* ou para *backdoors* e também quando a rede está sob o ataque *smurf*. Vale salientar que esse filtro só é útil para bloquear tráfegos direcionados para portas TCP ou UDP e não para tráfego no nível do protocolo IP. A entrada *deny ip any any* bloqueia todo o tráfego vindo de qualquer direção para qualquer máquina da rede, assim, quando essa proteção estiver ativada e vier um ataque no nível de IP, a proteção **rede para todos** será usada no seu lugar. A Figura 3.9 mostra um diagrama desse tipo de proteção.

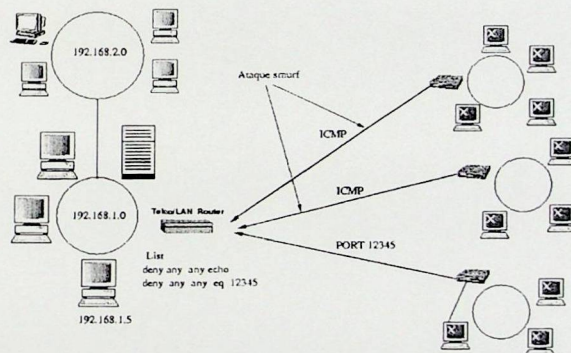


Figura 3.9: Bloqueio de todos para todos.

3.7.2 O Block e os Ataques DDoS

Algumas ferramentas de ataques DDoS podem ser configuradas para lançar ataques do tipo *SYN Flooding*, *ICMP flood* e *smurf* [30]. Esses tipos de ataques foram descritos no Capítulo 2.

O **Block** também pode detectar e bloquear esses tipos de ataques. Também, é possível filtrar as portas usadas para a comunicação entre os clientes, mestres e *daemons*. Para isso, essas portas devem ser colocadas nos arquivos `tcp.dat` e `udp.dat`. Por exemplo, as ferramentas *trino* e TFN usam a porta TCP 27665 para a comunicação entre o cliente e o mestre. No *stacheldraht*, essa porta é 16660. As portas de comunicação entre o mestre e os *daemons* são 31335 e 27444 (UDP) para o *trino* e para o TFN. No *stacheldraht*, a porta é 65000 (UDP).

No próximo Capítulo, apresentamos a arquitetura da ferramenta **Block**.

A Arquitetura da Ferramenta Block

4.1 Introdução

Neste Capítulo, apresentamos a arquitetura da ferramenta **Block** e descrevemos as suas principais funcionalidades. Também, apresentamos as ferramentas que foram usadas para a criação da ferramenta **Block** e a forma de instalá-la no sistema de arquivos do usuário.

4.2 O Programa Principal

O programa principal que controla a ferramenta **Block** é o arquivo `block.c`, localizado no diretório `src`.

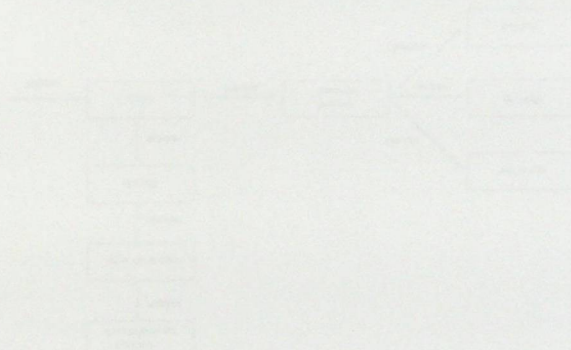


Figura 4.1: Arquitetura da ferramenta **Block**

Os dados gerados pelo programa **Block** são armazenados no diretório principal de trabalho do usuário, que é o diretório `~/.block`. Este diretório contém os arquivos de configuração e os arquivos de log. O programa **Block** também cria um arquivo de configuração no diretório `~/.block`, que é o arquivo `block.conf`. Este arquivo contém as configurações da ferramenta **Block**, como o endereço IP do servidor e o endereço IP do cliente.

Capítulo 4

A Arquitetura da Ferramenta Block

4.1 Introdução

Neste Capítulo, apresentamos a arquitetura do **Block** e descrevemos as suas principais características funcionais. A hierarquia dos processos que compõem a ferramenta e os diagramas de fluxos de dados, DFDs, são apresentados no Apêndice A.

4.2 O Programa Principal

Os principais módulos que compõem a ferramenta **Block** são apresentados na Figura 4.1.

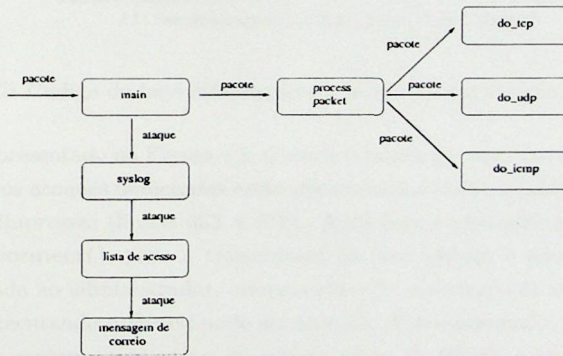


Figura 4.1: Arquitetura da ferramenta **Block**.

Em linhas gerais, um pacote é processado pelo programa principal dependendo do tipo de protocolo que ele representa (TCP, UDP, ICMP). Em seguida, se for classificado como pertencente a um ataque, é gerado um registro no arquivo *syslog*, é criada a lista de acesso, que é enviada para o roteador Cisco, e é remetida uma mensagem de correio para o administrador

da rede. Esses procedimentos serão detalhados a partir da apresentação de trechos do código fonte da ferramenta **Block**.

A função `main()` inicializa suas variáveis globais a partir da leitura de um arquivo de configuração e verifica o nível de proteção desejado. Depois, verifica se existe um arquivo de lista de acesso previamente configurada no diretório onde a ferramenta é executada. Em seguida, abre os arquivos de configuração de portas TCP e UDP que devem ser observadas, criando uma lista encadeada com as portas lidas nesses arquivos.

O programa, então, fica em um laço infinito lendo pacotes presentes na rede a ser protegida, como mostrado no código apresentado na Figura 4.2.

```
344     t = time((time_t *)0);
345     while(pkt = readdevice(netfd, &pktlen))
346     {
347         process_packet(pkt, pktlen);
348         if(time((time_t *)0) - t > Gtimer)
349         {
350             /* Times up. Print what we found and clean out old stuff. */
351
352             for(Gsi = Gsilist, i = 0; Gsi; Gsi = Gsi->next, i++)
353             {
354                 clear_pktin(Gsi);
355                 print_info();
356                 Gsi->icmpcnt = 0;
357             }
358             time(&start_time);
359             if(Gattack) /* if some attack has happened, put the list */
360             {
361                 result=connect_cisco(); /* if result != 0 ocorreu um erro */
362                 if(!em_message(0,Gfirst_mess,start_time))
```

Figura 4.2: Código do laço infinito para a leitura de pacotes na função `main()`.

Pelo código apresentado na Figura 4.2, quando a janela de tempo expira, a estrutura de lista encadeada, onde os ataques detectados estão armazenados (`Gsi`), é percorrida e as informações dos ataques são impressas (linhas 352 a 357). Após isso, é chamada a função de conexão ao roteador, `cisco_connect()`, para a transmissão da lista gerada e uma mensagem de correio eletrônico é enviada ao administrador, informando-o da ocorrência de um ataque.

A máquina executando o **Block** pode ser atacada. É recomendado, portanto, que a mesma não disponibilize nenhum outro tipo de serviço além do **Block** e que seja sempre mantido atualizado o seu sistema operacional. Dessa forma, os riscos de invasões serão minimizados.

4.3 Identificando Ataques

As assinaturas de ataques que o **Block** é capaz de identificar são inseridas no próprio código fonte da ferramenta. Por exemplo, na Figura 4.3, mostramos as assinaturas correspondentes

aos ataques *Land*, linhas 772 a 789, e *SYN Flooding*, linhas 806 a 817.

```
772     struct tcphdr *thdr;
773         u_short sport, dport;
774
775     thdr = (struct tcphdr *) pkt;
776     if(thdr->th_flags & TH_RST) /* RST generates no response */
777         return;                /* Therefore can't be used to scan.*/
778     sport = ntohs(thdr->th_sport);
779     dport = ntohs(thdr->th_dport);
780
781     if(thdr->th_flags & TH_SYN)
782     {
783         if(Gsaddr == Gdaddr && sport == dport)
784         {
785             Gsi->land.atktime = Gtimein;
786             Gsi->land.saddr = Gsaddr;
787             memcpy(Gsi->land.eaddr, ep->h_source, ETH_ALEN);
788         }
789     }
790     ... ..
791
806 /* See if this packet relates to other packets already received.*/
807
808     for(pi = Gsi->tcpin; pi; pi = pi->next)
809     {
810         if(pi->saddr == Gsaddr && pi->dport == dport)
811         {
812             if(flags == TH_SYN)
813                 addfloodinfo(pi, sport);
814             else if((flags & TH_FIN) || (flags & TH_ACK))
815                 rmfloodinfo(pi, sport);
816             return;
817         }
818     }
```

Figura 4.3: Códigos das assinaturas dos ataques *Land* e *SYN Flooding*.

4.3.1 Estruturas de Dados

As estruturas mais importantes são as listas encadeadas que guardam as informações dos ataques. Elas são variáveis globais. Um exemplo, mostrado na Figura 4.4, é a variável `Gsilist`, que guarda informações das máquinas a serem protegidas. As entradas nessa lista são criadas pela função `addtarget()`, como mostrado na Figura 4.4.


```

692 struct scaninfo *addtarget(addr)
693 u_long addr;
694 {
695     struct scaninfo *si;
696
697     if((si = (struct scaninfo *)malloc(sizeof(struct scaninfo))) == NULL)
698     {
699         perror("malloc scaninfo");
700         exit(-1);
701     }
702     memset(si, 0, sizeof(struct scaninfo));
703     si->addr = addr;
704     si->next = Gsilib;
705     Gsilib = si;
706     return(si);
707 }

```

Figura 4.4: Código da função `addtarget()`.

Para cada máquina a ser protegida, uma nova entrada na lista `Gsilib` é criada. Dessa forma, somente os ataques detectados e direcionados para máquinas que constam nessa lista são reportados. A variável global `Gsi`, mostrada na Figura 4.3, também guarda informações de ataques. Sempre que um ataque é detectado contra uma estação protegida, os dados são armazenados nos campos dessa estrutura, como por exemplo no campo `Gsi->land.atktime`, Figura 4.3. Essa variável global é usada em todas as funções que processam os protocolos, guardando informações sobre os ataques a serem registrados no **syslog** e na lista de acesso.

4.3.2 Algoritmos de Identificação de Ataques

Cada protocolo é tratado por um módulo específico, que trabalha com assinaturas para reconhecer os ataques. Descrevemos, em seguida, essas funções de análise.

Ataques TCP

A função `do_tcp()` recebe o pacote e processa os campos do cabeçalho TCP. A primeira ação é obter as portas origem e destino `sport` e `dport` para detectar um possível ataque *Land*.

O código descrito nas linhas 772 a 789 (Figura 4.3) é a assinatura de um ataque *Land*. Caso o mesmo seja detectado, informações como hora, endereço IP e MAC origem são armazenadas.

Caso não seja um ataque *Land*, pode ser um *portscan* ou um ataque *SYN Flooding*. Inicialmente, descrevemos como o **Block** trata ataques *SYN Flooding*. Em parágrafo posterior, mostramos como a ferramenta trata de ataques do tipo *portscan*.

No caso de um ataque *SYN Flooding*, o pacote é passado para a função `addtcp()`. O endereço IP origem e a porta destino são analisados para ver se já não fazem parte de alguma conexão anterior. Caso façam parte de uma conexão anterior, pode estar ocorrendo o ataque *SYN Flooding*. A indicação da ocorrência desse ataque depende da superação do limite de

conexões para a mesma porta dentro da janela de tempo estabelecida. As linhas 806 a 817 da Figura 4.3, pertencentes à função `addtcp()`, correspondem à assinatura desse tipo de ataque. É feita uma comparação entre o endereço IP origem e porta de destino do pacote atual com os valores de conexões anteriores armazenados em uma estrutura de dados. Caso o pacote seja do tipo FIN ou ACK, a informação de um possível ataque *SYN Flooding* é removida da estrutura `floodinfo` imediatamente.

Para detectar um ataque *portscan*, é necessário que o número de conexões vindas de um mesmo endereço IP tenha excedido o limite da variável `Gportlimit`, definida pelo código mostrado na Figura 4.5, dentro da janela de tempo. Somente o tipo `connect()` é detectado. Porém, outros tipos de *portscan* poderiam ser detectados com modificações no código fonte do **Block**. Existem parâmetros que visam minimizar o número de falsos positivos resultantes do tráfego HTTP direcionados para a porta 80, onde uma nova conexão é aberta para cada elemento de uma página. A estrutura `al` guarda informações de conexões do tipo HTTP e também de outros tipos de tráfego vindas de um mesmo endereço origem. Caso o número de conexões não HTTP menos o número de conexões HTTP vindas do mesmo endereço IP seja maior que o valor definido em `Gportlimit`, ou o número de conexões HTTP seja maior que o valor definido em `Gwebcount`, um evento de *portscan* é reportado. As linhas entre 1151 a 1159, na Figura 4.5, mostram a assinatura de um ataque *portscan*. A assinatura de ataques do tipo *portscan* usando o protocolo UDP é semelhante e está descrita nas linhas 1209 a 1215 na Figura 4.6.

```

1151 if(Greportlevel == REPORTALL || Greportlevel == REPORTSCAN)
1152 {
1153     for(al = tcplist; al; al = al->next)
1154     {
1155         if((al->cnt - al->wwwcnt) > Gportlimit || al->wwwcnt > Gwebcount)
1156         {
1157             sprintf(buf, "Possible TCP port scan from %s (%dports) against %s\n",
1158                 anetaddr(al->saddr), al->cnt, abuf);
1159             syslog(LOG_ALERT, buf);

```

Figura 4.5: Código da assinatura de um *portscan* TCP.

```

1209 for(al = udplist; al; al = al->next)
1210 {
1211     if(al->cnt > Gportlimit)
1212     {
1213         sprintf(buf, "Possible UDP port scan from %s (%dports) against
1214             %s\n", anetaddr(al->saddr), al->cnt, abuf);
1215         syslog(LOG_ALERT, buf);

```

Figura 4.6: Código da assinatura de um *portscan* UDP.

Ataques UDP

A função `do_udp()` recebe o pacote e verifica se o mesmo faz parte de um ataque *Teardrop*. A assinatura desse ataque é aparecimento do valor 242 no campo de fragmentação do cabeçalho IP, (*Segmentation Offset*), como mostra a linha 908 da Figura 4.7. Caso esse valor seja alterado por alguma ferramenta de ataque, poderá gerar um falso negativo. O código do **Block** pode ser alterado para reportar essa modificação.

```
907     uhdr = (struct udphdr *) pkt;
908     if(Gid == 242 && Gisfrag) /* probable teardrop */
909     {
910         Gsi->teardrop.saddr = Gsaddr;
911         memcpy(Gsi->teardrop.eaddr, ep->h_source, ETH_ALEN);
912         Gsi->teardrop.atktime = Gtimein;
913     }
```

Figura 4.7: Código da assinatura do *Teardrop*.

Ataques ICMP

Os ataques do tipo ICMP que podem ser detectados são o ping da morte e o *smurf*. A detecção é feita pela função `do_icmp()`. A assinatura de um ping da morte é a fragmentação do ICMP como mostra a linha 973 da Figura 4.8.

```
966 void do_icmp(ep, pkt)
967 struct ethhdr *ep;
968 u_char *pkt;
969 {
970     struct icmphdr *icmp;
971
972     icmp = (struct icmphdr *) pkt;
973     if(Gisfrag) /* probable ICMP attack (i.e. Ping of Death) */
974     {
975         Gsi->icmpfrag.saddr = Gsaddr;
976         memcpy(Gsi->icmpfrag.eaddr, ep->h_source, ETH_ALEN);
977         Gsi->icmpfrag.atktime = Gtimein;
978     }
979     if(icmp->type == ICMP_ECHOREPLY)
980         Gsi->icmpcnt++;
981     return;
982 }
```

Figura 4.8: Código da assinatura do ataque *smurf* e ping da morte.

Se o pacote for do tipo *ICMP_ECHOREPLY*, o campo `Gsi->icmpcnt` é atualizado (linhas 979 e 980). Caso esse valor ao final da janela de tempo seja maior que o definido na variável `Gicmplimit`, um ataque *smurf* é reportado. O valor máximo para `Gicmplimit` é dado pelo parâmetro `icmp_limit` lido do arquivo de configuração do **Block** conforme mostrado na linha

Ataques de Portas TCP e UDP

A busca de padrões de portas é feita na função `print_info()`, onde as listas encadeadas são percorridas. As portas TCP e UDP capturadas durante a janela de monitoração serão comparadas com a lista de portas dos arquivos de configuração TCP e UDP. A estrutura `pi`, do tipo `pktin`, contém informação da porta destino no campo `pi->dport`. As linhas 1111 a 1120 da Figura 4.9 mostram como a porta sob análise é comparada com as portas na lista `ctcports`. O código relativo a ataques UDP é semelhante.

```

1105 if (Gtcpports)
1106 {
1107     ctcports=Gfirst_tcp->next;
1108     /* read the configured ports */
1109     while((ctcports) && (!found))
1110     {
1111         if(pi->dport == ctcports->cport)
1112         {
1113             found = 1;
1114             eaddr = ether_ntoa(pi->eaddr);
1115             sprintf(buf, "Possible dangerous tcp port (%d)attack from %s (%s),against%s.\n",
1116                 pi->dport, anetaddr(pi->saddr),eaddr,abuf);
1117             syslog(LOG_ALERT, buf);
1118             build_list(anetaddr(pi->saddr),abuf,TY_DANGEROUS_PORT,pi->dport,pi->dport,PROTO_TCP);
1119             list_attacks(buf);
1120         }
1121         ctcports= ctcports->next;
1122     }
1123 }
```

Figura 4.9: Código para a detecção de ataques em portas TCP.

4.4 Gerando Relatórios e Construindo Listas

A função `print_info()` é a responsável por registrar no `syslog` todas as informações dos ataques detectados. As estruturas onde as informações de ataques estão armazenadas são percorridas e seu conteúdo impresso. A função `print_info()` é chamada pela função `main()` após a janela de tempo expirar. A Figura 4.10 mostra um trecho da função `print_info()` com o código de geração de logs do ataque *Teardrop*.


```

1045 strcpy(abuf, anetaddr(Gsi->addr));
1046 if(Greportlevel == REPORTALL || Greportlevel == REPORTDOS)
1047 {
1048     if(Gsi->teardrop.atktime)
1049     {
1050         eaddr = ether_ntoa(Gsi->teardrop.eaddr);
1051         sprintf(buf, "Possible teardrop attack from %s (%s)against %s",
1052             anetaddr(Gsi->teardrop), eaddr, abuf);
1053         syslog(LOG_ALERT, buf);
1054         build_list(anetaddr(Gsi->teardrop), abuf, TY_TEARDROP, 0, PROTO_IP);
1055         list_attacks(buf);
1056         memset(&Gsi->teardrop, 0, sizeof(struct atk));
1057     }

```

Figura 4.10: Código para o registro de um ataque *Teardrop*.

As listas de acesso são geradas através de informações que são passadas pela função `print_info()` para a função `build_list()` (ver a linha 1054, Figura 4.10). Os parâmetros passados são: IP origem e destino, tipo de ataque, porta destino (quando houver) e protocolo (IP, UDP ou TCP). Esses parâmetros são usados para construir a lista. Outro parâmetro importante é o `Gparanoid`, Figura 4.12, que informa ao programa o nível de proteção usado numa lista de acesso que será construída. A construção da lista também depende do tipo de ataque detectado, sendo que a primeira linha é um comentário sobre esse ataque. O código mostrado na Figura 4.11 identifica o ataque, escrevendo a linha de comentário.

```

1417 switch(attack)
1418 {
1419     case 1 : fprintf(flistge, "!synflooding attack detected \n");
1420         break;
1421     case 2 : fprintf(flistge, "!teardrop attack detected \n");
1422         break;
1423     case 3 : fprintf(flistge, "!tcp or udp scan detected \n");
1424         break;
1425     case 4 : fprintf(flistge, "!smurf attack detected \n");
1426         /* denies any->any no more icmp echo traffic*/
1427         fprintf(flistge, "access-list %ld deny %s any any echo log\n", Glist, proto_name);
1428         break;
1429     case 5 : fprintf(flistge, "!ping of death attack detected \n");
1430         break;
1431     case 6 : fprintf(flistge, "!IP spoofing attack detected - Land\n");
1432         break;
1433     case 7 : fprintf(flistge, "!dangerous port detected \n");
1434     default:
1435         break;

```

Figura 4.11: Código para a geração da lista de acesso.

Como o ataque *smurf* pode prejudicar todo o tráfego da rede local, o programa bloqueia todos os pacotes ICMP recebidos após a detecção do ataque. As linhas 1425 a 1428, mostradas na Figura 4.11, implementam a proteção.

Os ataques de portas TCP e UDP podem ser bloqueados no nível do protocolo de transporte. Um ataque desse tipo pode ser barrado de várias maneiras, dependendo do nível de proteção escolhido.

```

1442 if (Gparanoid==1) /* host -> host */
1443 {
1444     if (attack==7) /* tcp/udp port attack */
1445         fprintf(flistge,"access-list %ld deny %s host %s host%s eq %d log \n",
            Glist,proto_name,ipor,ipdest,port);
1446     else
1447         fprintf(flistge,"access-list %ld deny ip host %s host %s log \n",Glist,ipor,ipdest);
1448 }
1449 else if (Gparanoid==2) /* net->host */
1450 {
1451     if (attack==7) /* tcp/udp port attack */
1452         fprintf(flistge,"access-list %ld deny %s %s %s host %s eq %d log \n",
            Glist,proto_name,snet_class,net_mask,ipdest,port);
1453     else
1454         fprintf(flistge,"access-list %ld deny ip %s %s host %s log \n",
            Glist,snet_class,net_mask,ipdest);
1455 }
1456 else if (Gparanoid==3) /* net->any */
1457 {
1458     if (attack==7)
1459         fprintf(flistge,"access-list %ld deny %s %s %s any eq %d log \n",
            Glist,proto_name,snet_class,net_mask,port);
1460     else
1461         fprintf(flistge,"access-list %ld deny ip %s %s any log \n",Glist,snet_class,net_mask);
1462 }
1463 else if (Gparanoid==4) /* any -> any */
1464 {
1465     if (attack==7)
1466         fprintf(flistge,"access-list %ld deny %s any any eq %d log \n",
            Glist,proto_name,port);
1467     else
1468         /* it can not generate deny ip any any, there will not be any communication, so*/
1469         /* it generates net->any */
1470         fprintf(flistge,"access-list %ld deny ip %s %s any log \n",Glist,snet_class,net_mask);
1471 }

```

Figura 4.12: Código para o bloqueio de um ataque de portas TCP.

A Figura 4.12 contém o nível de proteção que define como a rede do atacante é bloqueada através da variável `Gparanoid` (linhas 1442, 1449, 1456 e 1463). Em seguida, é identificado o tipo de ataque através do teste `if attack == 7`, caso seja um ataque de portas TCP ou UDP (tipo 7), será gerada uma lista com o número da porta (linhas 1445, 1452, 1459 e 1466). Os outros tipos de ataques não precisam do parâmetro do número da porta.

Os tipos de ataques mencionados no parágrafo anterior, são bloqueados no nível de IP de acordo com o nível de proteção escolhido.

As linhas 1447, 1454, 1461 e 1468 da Figura 4.12 geram a entrada na lista de acesso para os outros ataques. Uma observação importante sobre bloqueio de tráfego no nível de IP é que

proteção **any** -> **any** iria barrar todas as conexões, não sendo, portanto, utilizada.

Como podemos observar no código mostrado na Figura 4.12, a lista gerada é salva no arquivo **flistge** para ser transmitida para o roteador. O tipo de lista usado é a lista numerada estendida e o número da lista é definido pela variável **Glist**.

4.5 Conexão com o Roteador

Como vimos na seção 3.6, existem procedimentos para criar e aplicar uma lista de acesso em um roteador Cisco. As rotinas descritas a seguir executam esses procedimentos.

A rotina de conexão possui um limite de 10 segundos (valor padrão) de tolerância de espera por uma resposta do roteador. Caso a resposta não seja enviada dentro desse limite, a conexão é cancelada. Após a conexão ser estabelecida, as senhas do roteador são enviadas. As linhas de código na Figura 4.13, comparam os caracteres recebidos do roteador aos do arquivo de configuração. As linhas de 1563 a 1572 recebem os primeiros dados do roteador e verificam se existe a palavra *Password*. Caso exista, envia a senha do usuário. A variável **pass_count** auxilia no controle de quantas vezes a palavra *Password* foi vista, já que se deve vê-la duas vezes, senhas de usuário e *enable*, durante o processo de conexão.

A sequência de comandos apresentada entre as linhas 1586 e 1596, executa a abertura de conexão e elevação de privilégio para o nível de administrador. O código apresentado na Figura 4.14 retira a lista antiga, se houver. A nova lista é então aplicada, uma linha por vez, como mostrado pelo código da Figura 4.15. As linhas da Figura 4.16 encerram a conexão com o roteador. A lista está inserida no roteador e o ataque, bloqueado.

```

1563 if((strstr(recvline,"Password:")!=NULL) && (pass_count ==0))
1564 { /* user level password */
1565     sprintf(sendline,"%s\r\n",Gpass_term);
1566     if(send(sockfd,sendline,strlen(sendline),0) == -1)
1567     {
1568         perror("Send");
1569         return(5);
1570     }
1571     pass_count++;
1572 }
1573 if((strstr(recvline,Gprompt_us)!=NULL) && (init < 12))
1574 { /* prompt de usuario */
1575     if(init == 11)
1576     { /* disconnecting */
1577         sprintf(sendline,"quit\r\n\r\n");
1578         if(send(sockfd,sendline,strlen(sendline),0) == -1)
1579         {
1580             perror("Send");
1581             return(5);
1582         }
1583         init++; /* init = 12 */
1584         file_eof = 1;
1585     }
1586     else
1587     {
1588         strcpy(sendline,"enable\r\n");
1589         if(send(sockfd,sendline,strlen(sendline),0) == -1)
1590         {
1591             perror("Send");
1592             return(5); /* error num 5 - error during sending data */
1593         }
1594         pass_count++;
1595     }
1596 }
1597 if((strstr(recvline,"Password:")!=NULL) && (pass_count ==2))
1598 {
1599     sprintf(sendline,"%s\r\n",Gpass_sec); /* adm. level password*/
1600     if(send(sockfd,sendline,strlen(sendline),0) == -1)
1601     {
1602         perror("Send");
1603         return(5);
1604     }
1605     pass_count++;
1606 }

```

Figura 4.13: Código para a conexão com o roteador.


```

1623 sprintf(sendline,"conf term\r\n\n"); /* adm. level prompt*/
1624 if(send(sockfd,sendline,strlen(sendline),0) == -1)
1625 {
1626     perror("Send");
1627     return(5);
1628 }

... ..

1647 sprintf(sendline,"no access-list %d\r\n",Glist);
1648 if(send(sockfd,sendline,strlen(sendline),0) == -1)
1649 {
1650     perror("Send");
1651     return(5);
1652 }

```

Figura 4.14: Código para a retirada da lista.

```

1663 while(fgets(file_line,sizeof(file_line),flistge))
1664 {
1665     sprintf(sendline,"%s\r\n",file_line);
1666     if(send(sockfd,sendline,strlen(sendline),0) == -1)
1667     {
1668         perror("Send");
1669         return(5);
1670     }
1671     if(numbytes=recv(sockfd,recvline,MAXLINE,0)==0)
1672     {
1673         perror("Connect_cisco: router terminated prematurely");
1674         return(4);
1675     }
1676 }
1677 fclose(flistge);

```

Figura 4.15: Código para inclusão da lista gerada.

```

1573 if((strstr(recvline,Gprompt_us)!=NULL) && (init < 12))
1574 { /* user prompt */
1575     if(init == 11)
1576     { /* disconnecting */
1577         sprintf(sendline,"quit\r\n\n");
1578         if(send(sockfd,sendline,strlen(sendline),0) == -1)
1579         {
1580             perror("Send");
1581             return(5);
1582         }

```

Figura 4.16: Código para o encerramento da conexão com o roteador.

4.6 Arquivo de Configuração

O arquivo de configuração está escrito em inglês já que a ferramenta **Block** será disponibilizado na Internet. Os parâmetros do arquivo de configuração são apresentados nas Figura 4.17 a 4.28.

O caracter **#** é usado para inserir comentário nas linhas do arquivo. Pode-se usar a tecla *tab* ou deixar um espaço em branco entre a variável e o seu valor.

As linhas de 7 a 52 da Figura 4.17 são usadas para as configurações dos parâmetros de tolerância de ataques e configuração da placa de captura.

As linhas 56 a 61 da Figura 4.18 configuram o nível de proteção da lista de acesso. Os valores possíveis são: 1 (estação para estação), 2 (rede para estação), 3 (rede para todos) e 4 (todos para todos).

A linha 71 da Figura 4.19 é utilizada para indicar à ferramenta a existência (valor 1) ou não (valor 0) de uma lista de acesso prévia. Caso exista uma lista, a ferramenta irá concatená-la com as novas entradas. Essas entradas são inseridas sem nenhuma verificação prévia nas outras entradas já existentes da lista.

Cada lista de acesso possui um número e as linhas de 75 a 77 da Figura 4.20 informam à ferramenta o seu valor. A ativação da lista é feita conforme apresentado na seção 3.6.2.

O endereço IP do roteador é informado na linha 82 da Figura 4.21.

As linhas de 86 a 97 da Figura 4.22 informam à ferramenta a existência de um arquivo contendo uma lista de portas usadas por programas *backdoors* e *exploits* para o protocolo TCP e UDP. O parâmetro *tcp_ports* e *udp_ports* igual a 1 indica a existência dos arquivos; o valor 0 indica a ausência dos mesmos.

Caso aconteçam erros durante o processo de conexão com o roteador como erro de criação do *socket*, endereço do roteador inválido, impossibilidade de conexão com o roteador, término prematuro da conexão ou erros no envio de dados, uma mensagem de correio eletrônico é enviada ao administrador. O parâmetro *errors_number*, na linha 104, do código mostrado na Figura 4.23, indica o número de erros aceitáveis até que essa mensagem seja enviada. Isso evita que o administrador receba uma mensagem a cada janela de tempo (10 segundos) caso aconteçam erros nesse intervalo.

A aplicação envia mensagens para o endereço eletrônico do administrador definido no parâmetro *e-mail_address*, na linha 110, do código mostrado na Figura 4.24. Se essa linha estiver comentada, nenhuma mensagem é enviada. Vale salientar que é para esse endereço que são enviadas as mensagens de erro e de ataques.

A senha de *terminal* do roteador é definida na linha 114; a linha 119 define a senha de *enable*, ambas mostradas na Figura 4.25.

Da mesma forma como os sistemas operacionais de linha de comando, o IOS possui um *prompt*, sendo o mesmo configurado no momento da instalação do roteador. Cada nível de privilégio possui um *prompt* diferente, assim como o modo de configuração. A linha 124, da


```

1 # Configuration file
2 # Values are delimited by a tab or blank space
3 # A # means a remark
4 #
5 #
6 #
7 #device - Use device as the network interface device
8 #The first non-loopback interface is the default
9 device eth0
10 #
11 #
12 #
13 #flood - Assume a synflood attack occurred if more than 'flood' uncompleted
14 # connections are received
15 flood 8
16 #
17 #
18 #
19 #icmplimit - Assume we may be part of a smurf attack if more than icmplimit
20 # ICMP ECHO REPLIES are seen
21 icmp_limit 5
22 #
23 #
24 #
25 #level - Monitor more than just our own host(level 1). A level of 2 watches
26 #all addresses in our subnet and 3 watches all addresses
27 level 3
28 #
29 #
30 #
31 #
32 #portlimit - Logs a portscan alert if packets are received for more than
33 #portlimit ports in the timeout period.
34 port_limit 7
35 #
36 #
37 #
38 #reporttype - If reporttype is (2)dos, only Denial Of Service
39 #attacks are reported. If reporttype is scan(3) then only scanners are
40 #reported. If reporttype is 1 all is reported.
41 report_type 1
42 #
43 #
44 #
45 #timeout - Count packets and print potential attacks every timeout seconds.
46 timeout 15
47 #
48 #
49 #
50 #webcount - Assume we are being portscanned if more than webcount packets
51 # are received from port 80
52 web_count 40
53 #
54 #
55 #

```

Figura 4.17: Parâmetros do **Block**.

```

56 #paranoid - Paranoid level. 1 blocks the connection (host->host).
57 # 2 - blocks the connection (net->host)
58 # 3 - blocks the connection (net->any)
59 #
60 # 4 - blocks the connection (any->any)
61 paranoid_level 3
62 #
63 #
64 #
65 #

```

Figura 4.18: Configuração do tipo de bloqueio desejado.

```

66 #access-list - informs if there is a pre-configured access list. The program
67 #will put it together with the generated one. 0 Means no access-list, 1
68 #means that there is an access-list.
69 #OBS - the file name must be access-list.dat and must be in the same
70 #directory of the program.
71 access-list 0
72 #
73 #
74 #

```

Figura 4.19: Configuração da concatenação de listas de acesso.

```

75 #list_number - number of the access list. Must be the same of the existing
76 #one.
77 list_number 111
78 #
79 #
80 #

```

Figura 4.20: Pré-seleção do número da lista de acesso.

```

81 #router_ip - IP address of the router.
82 router_ip 192.168.2.1
83 #
84 #
85 #

```

Figura 4.21: Configuração do endereço IP do roteador.


```

86 #tcp_ports - a file containing a list of dangerous TCP ports to be blocked
87 #when they are seen inside the network. Ex: NetBus 12345, BO 313337 etc.
88 #The value 0 disable the search, 1 enable it.
89 #OBS: the file name must be tcp.dat
90 tcp_ports      1
91 #
92 #
93 #udp_ports - a file containing a list of dangerous UDP ports to be blocked
94 #when they are seen inside the network. Ex: NetBus 12345, BO 313337 etc.
95 #The value 0 disable the search, 1 enable it.
96 #OBS: the file name must be udp.dat
97 udp_ports      1
98 #
99 #
100 #

```

Figura 4.22: Notificação da existência dos arquivos de portas TCP e UDP.

```

101 #errors_number - send e-mail messages to the system administrator if more
102 #than errors_number errors have happened. This avoids a new message to be
103 #sent every 10 seconds to the mailbox.
104 errors_number  1
105 #
106 #
107 #

```

Figura 4.23: Configuração dos parâmetros de erro.

```

108 #e-mail_address - where messages will be sent. If this value is commented, no
109 #e-mail messages will be sent.
110 e-mail_address  root@localhost
111 #
112 #

```

Figura 4.24: Configuração do endereço eletrônico do administrador.

```

113 #pass_term - password of the user level in the router
114 pass_term    senha
115 #
116 #
117 #
118 #pass_secret - password of the enable level in the router
119 pass_secret   senha
120 #
121 #
122 #

```

Figura 4.25: Configuração das senhas do roteador.

```

123 #prompt_us - terminal prompt of the router
124 prompt_us      router>
125 #
126 #
127 #
128 #prompt_sec - enable prompt of the router
129 prompt_sec      router#
130 #
131 #
132 #
133 #prompt_conf - configuration prompt of the router
134 prompt_conf      router(config)#
135 #
136 #
137 #
138 #prompt_int - interface configuration prompt
139 prompt_int      router(config-if)#

```

Figura 4.26: Configuração do *prompt* do roteador.

```

143 #connection timeout - how long shall we wait to connect to the router
144 connect_time      13
145 #
146 #
147 #

```

Figura 4.27: Configuração do tempo de conexão com o roteador.

Figura 4.26, define o *prompt* com nível de privilégio mais baixo. A linha 129 informa o *prompt* de segurança, após a senha de *enable* ter sido enviada, a linha 134 informa o *prompt* de configuração a nível de administrador e, finalmente, a linha 139 refere-se à configuração da interface onde a lista de acesso será aplicada.

O valor da linha 144, do código mostrado na Figura 4.27, indica o tempo de espera de abertura de conexão entre a ferramenta e o roteador. Esse intervalo evita que a ferramenta fique mais de um minuto esperando que o tempo de expiração do *socket* finalize para voltar a executar novamente.

```

148 #send list - how long it takes to put large lists in the router -in seconds-
149 send_list        30

```

Figura 4.28: Configuração do tempo de *timeout* com o roteador.

Durante o envio de listas muito grandes, a ferramenta pode levar algum tempo, por isso, o valor definido na linha 149, do código mostrado na Figura 4.28, define o tempo máximo que a ferramenta deve esperar para poder enviar uma lista de acesso.

A linha 152 da Figura 4.29 informa em qual interface do roteador a lista será aplicada.


```
151 #interface
152 interface      eth0
153 #
```

Figura 4.29: Interface onde a lista de acesso será aplicada.

4.7 Limitações da Ferramenta

Como limitações principais da ferramenta, apontamos o banco de assinaturas restrito, não sendo previstas variações dos ataques detectados, *scans* espaçados no tempo e mascarados com a falsificação de endereços IP. Contudo, isso pode ser sanado por novas versões que usem como suporte uma linguagem de construção de assinaturas que permitiria aos usuários montarem seu próprio banco.

As assinaturas poderiam ser retiradas do código fonte e colocadas em um arquivo texto, onde seriam lidas pelas funções de reconhecimento de ataques (*do_tcp*, *do_udp* e *do_icmp*).

O módulo de construção de listas e de conexão com o roteador poderia ser também portado para outros dispositivos. Para tanto, ele leria um arquivo de configuração contendo a sintaxe de conexão com o dispositivo e a sintaxe de construção de suas listas de acesso.

Outra limitação da ferramenta, é a janela de tempo. Ela é muito pequena para detectar algumas variações de *scans*, mas isso poderia ser sanado caso a função de análise de pacotes tivesse acesso a um banco de dados, o qual aumentaria a sua capacidade investigativa.

O envio da senha do roteador pela estação onde a ferramenta é executada é feita em texto pleno. Isso gera um problema de segurança. Um usuário que esteja com a placa de rede em modo promíscuo e um programa de captura de senhas poderá obtê-la. Essa conexão poderia se tornar mais segura caso o roteador e o **Block** implementassem o protocolo ssh ou uma outra interface de rede fosse instalada no roteador e na estação que executa a ferramenta e ambas fossem ligadas por um cabo *crossover*. A implementação do protocolo ssh v1 já é feita pela Cisco a partir da versão do IOS 12.0.(5)S. Esse recurso é passível de restrições de exportação pelo governo dos Estados Unidos da América e mais informações podem ser obtidas no site da Cisco [31]. O **Block** pode implementar esse protocolo.

Também é necessário um módulo que verifique cada nova entrada na lista para garantir sua funcionalidade. Esse módulo percorreria toda a lista sempre que uma nova entrada fosse criada e, caso haja conflito com uma entrada mais antiga, a lista seria refeita, garantindo a funcionalidade.

Capítulo 5

Detecção de Ataques com o Block

5.1 Introdução

Neste Capítulo, apresentamos os testes que fizemos para verificar a eficiência da ferramenta desenvolvida. Simulamos os vários tipos de ataques descritos no Capítulo 2.

5.2 Ambiente de Testes

Para realizar os testes da ferramenta **Block**, utilizamos duas estações de trabalho, um roteador Cisco 1601 e um hub. A topologia das redes é mostrada na figura 5.1.

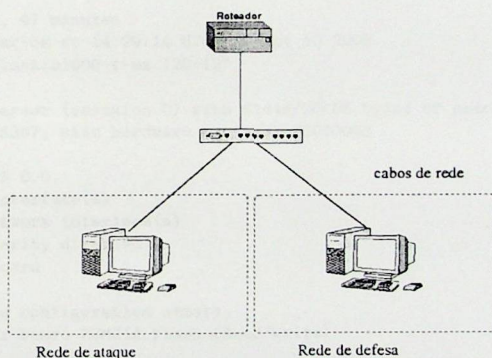


Figura 5.1: Topologias das redes de ataque e defesa.

Uma rede foi denominada de rede de ataque, com uma estação executando os programas de ataque e a outra rede foi denominada de rede de defesa. Na rede de defesa, a estação vítima rodava a ferramenta **Block**. A interface Ethernet do roteador Cisco 1601, foi configurada com dois endereços IP para que as duas redes se comunicassem. Um hub foi utilizado para conectar

todas as estações e o roteador. A máquina da rede de ataque foi conectada, também, ao roteador Cisco via saída Aux para monitoração.

A configuração das estações utilizadas e suas funções são apresentadas na Tabela 5.1

Nome da Estação	Endereço IP	Função	Configuração
ataque	192.168.1.10/24	Lançar ataques utilizando os programas targa , nmap , hping2 e ping .	Computador Novadata monoprocessado (Intel Pentium, 166 MHz), 64Mb RAM, disco rígido 2.1Gb, Sistema Operacional Linux Slackware 8.1, <i>kernel</i> 2.4.18
vitima	192.168.2.10/24	Verificar o tráfego através do programa tcpdump	Computador DFI (Cyrix, 133MHz), 192Mb RAM, disco rígido de 4Gb, Sistema Operacional Linux Slackware, <i>kernel</i> 2.4.18.

Tabela 5.1: Configurações e as funções de cada estação do ambiente de testes.

As informações sobre o modelo e sobre a versão do IOS do roteador utilizado nos testes são mostradas na Figura 5.2.

```
Cisco Internetwork Operating System Software
IOS (tm) 1600 Software (C1600-Y-M), Version 12.0(12), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2000 by cisco Systems, Inc.
Compiled Mon 10-Jul-00 19:59 by htseng
Image text-base: 0x02005000, data-base: 0x0245E484

ROM: System Bootstrap, Version 12.0(3)T, RELEASE SOFTWARE (fc1)
ROM: 1600 Software (C1600-RB00T-R), Version 12.0(3)T, RELEASE SOFTWARE (fc1)

router uptime is 1 hour, 47 minutes
System restarted by power-on at 14:29:14 UTC Thu Oct 10 2002
System image file is "flash:c1600-y-mz.120-12"

cisco 1601 (68360) processor (revision C) with 6144K/2048K bytes of memory.
Processor board ID 22636367, with hardware revision 00000003
Bridging software.
X.25 software, Version 3.0.0.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial(sync/async) network interface(s)
System/IO memory with parity disabled
8192K bytes of DRAM onboard
System running from RAM
7K bytes of non-volatile configuration memory.
4096K bytes of processor board PCMCIA flash (Read/Write)

Configuration register is 0x2102
```

Figura 5.2: Informações sobre o modelo e a versão do IOS do roteador utilizado no ambiente de testes.

A configuração inicial do roteador é mostrada na Figura 5.3.

```

Current configuration:
!
! Last configuration change at 16:15:42 UTC Thu Oct 10 2002
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname router
!
enable secret 5 $1$EyxY9EMxmSz5sjxHxHY70pMVq0
enable password flavio
!
ip subnet-zero
!
!
buffers verybig max-free 20480
!
interface Ethernet0
ip address 192.168.2.1 255.255.255.0 secondary
ip address 192.168.1.1 255.255.255.0
!
interface Serial0
no ip address
no ip directed-broadcast
shutdown
no fair-queue
!
ip classless
!
snmp-server community public RO
!
line con 0
transport input none
line vty 0 4
password flavio
login
!
end

```

Figura 5.3: Configuração inicial do roteador.

Para realizar os ataques, utilizamos os programas targa [32], nmap [33], hping2 [34] e o comando ping, presente nos sistemas operacionais utilizados.

A rede de ataque consistia do computador **ataque** e a rede de defesa composta pelas estação **vitima**.

Foi implementado um servidor que escutava na porta TCP 12345 com o intuito de testar a detecção de portas usadas por cavalos de tróia. Cada rede possuía uma estação executando o programa **tcpdump** [35] para registrar os ataques e a inclusão da lista com o roteador.

Para analisar o comportamento da ferramenta, a cada ataque, a configuração do roteador será mostrada, bem como linhas dos arquivos gerados pelo **tcpdump** exibindo o tráfego na

variável	valor
device	eth0
flood	8
icmp_limit	5
wlevel	3
port_limit	7
report_type	1
timeout	15
web_count	40
paranoid_level	3
access_list	0
list_number	110
router_ip	192.168.1.1
tcp_ports	1
udp_ports	1
errors_number	1
e-mailAddress	root@localhost
pass_term	flavio
pass_secret	flavio
prompt_us	router>
prompt_sec	router#
prompt_conf	router(config)#
connect_time	63
send_list	65
prompt_int	router(config-if)#
interface	eth0

Tabela 5.2: Parâmetros de configuração do arquivo `block.conf`.

rede, arquivo **syslog** com a detecção dos ataques e a lista gerada para bloqueá-los. Em alguns casos, exibiremos o efeito do bloqueio da lista com linhas dos arquivos **tcpdump** mostrando as tentativas de ataques após a lista ser implementada no roteador. Também, apresentamos um exemplo de mensagem de correio eletrônico que a ferramenta envia ao administrador informando sobre as ocorrências detectadas.

5.3 Ataques

Nos testes realizados, simulamos cada um dos tipos de ataques descritos no Capítulo 2. Quando apropriado, os níveis de proteção disponíveis no **Block** foram alterados. A máquina atacante é a máquina **ataque**.

Os parâmetros do arquivo de configuração do **Block**, `block.conf`, são mostrados na Tabela 5.2. Para executar a ferramenta basta editar seu arquivo de configuração com os valores desejados e digitar o nome da ferramenta na linha de comando.

5.3.1 Ataques a Portas TCP

Os ataques a portas TCP foram testados usando todos os níveis de proteção da ferramenta **Block**. Na máquina alvo, **vítima**, foi instalado um servidor que escutava na porta TCP 12345 e

que respondia para o cliente apenas com a mensagem conectou. Os arquivos `tcp.dat` e `udp.dat` estavam selecionados no arquivo de configuração `block.conf`. Previamente, não incluímos qualquer lista de acesso no roteador.

O primeiro ataque foi feito usando a proteção `host->host`. A máquina **ataque** executou o comando `telnet 192.168.2.10 12345` e conseguiu se conectar com a máquina **vítima** como mostram as linhas do `tcpdump` na Figura 5.4.

```
17:07:29.069528 192.168.1.10.1106 > 192.168.2.10.12345: S 3891624314:3891624314(0) win 5840
<mss 1460,sackOK,timestamp 9514710,nop,wscale 0> (DF) [tos 0x10]
17:07:29.071170 192.168.2.10.12345 > 192.168.1.10.1106: S 3887667373:3887667373(0)
ack 3891624315 win 5792 <mss 1460,sackOK,timestamp 951949 951471,nop,wscale 0> (DF)
17:07:29.072514 192.168.1.10.1106 > 192.168.2.10.12345: . ack 1 win 5840
<nop,nop,timestamp 951472 951949> (DF) [tos 0x10]
17:07:29.075507 192.168.2.10.12345 > 192.168.1.10.1106: P 1:15(14) ack 1 win
5792 <nop,nop,timestamp 951949 951472> (DF)
17:07:29.077118 192.168.1.10.1106 > 192.168.2.10.12345: . ack 15 win 5840
<nop,nop,timestamp 951472 951949> (DF) [tos 0x10]
17:07:29.077948 192.168.2.10.12345 > 192.168.1.10.1106: F 15:15(0) ack 1 win
5792 <nop,nop,timestamp 951949 951472> (DF)
17:07:29.079679 192.168.1.10.1106 > 192.168.2.10.12345: F 1:1(0) ack 16 win
5840 <nop,nop,timestamp 951472 951949> (DF) [tos 0x10]
17:07:29.081236 192.168.2.10.12345 > 192.168.1.10.1106: . ack 2 win 5792
<nop,nop,timestamp 951950 951472> (DF)
```

Figura 5.4: Linhas do arquivo gerado pelo `tcpdump` na rede de ataque.

As primeiras linhas mostram o processo de abertura de uma conexão TCP, com informações de número de seqüência dos pacotes, tamanho da janela e tipo de pacote (SYN ou ACK). Após a abertura da conexão, observamos o servidor enviar informações para o cliente e finalizar a conexão nas últimas linhas.

O registro desse ataque aparece no arquivo do `syslog`, como mostrado na Figura 5.5.

```
Oct 20 17:07:46 vitima BLOCK: Possible dangerous tcp port (12345) attack
from 192.168.1.10 (00:80:ad:c8:97:76), against 192.168.2.10.
```

Figura 5.5: Registro do ataque de portas TCP no arquivo do `syslog`.

A lista de acesso gerada contém a declaração,

```
access-list 111 deny tcp host 192.168.1.10 host 192.168.2.10 eq 12345 log
```

A ferramenta **Block** abre, então, uma conexão com o roteador, como mostram as linhas geradas pelo `tcpdump` apresentadas na Figura 5.6.


```

17:07:46.570184 192.168.2.10.32786 > 192.168.1.1.telnet: S 3912202997:3912202997(0)
win 5840 <mss 1460,sackOK,timestamp 9536720,nop,wscale 0> (DF)
17:07:46.574057 192.168.1.1.telnet > 192.168.2.10.32786: S 884907977:884907977(0)
ack 3912202998 win 4128 <mss 1460>
17:07:46.574177 192.168.2.10.32786 > 192.168.1.1.telnet: . ack 1 win 5840 (DF)
17:07:46.582613 192.168.1.1.telnet > 192.168.2.10.32786: P 1:13(12) ack 1 win 4128
17:07:46.582674 192.168.2.10.32786 > 192.168.1.1.telnet: . ack 13 win 5840 (DF)
17:07:46.591803 192.168.1.1.telnet > 192.168.2.10.32786: P 13:55(42) ack 1 win 4128
17:07:46.591859 192.168.2.10.32786 > 192.168.1.1.telnet: . ack 55 win 5840 (DF)
17:07:46.592097 192.168.2.10.32786 > 192.168.1.1.telnet: P 1:9(8) ack 55 win 5840 (DF)
17:07:46.602013 192.168.1.1.telnet > 192.168.2.10.32786: P 55:64(9) ack 9 win 4120
17:07:46.602171 192.168.2.10.32786 > 192.168.1.1.telnet: P 9:17(8) ack 64 win 5840 (DF)
...

```

Figura 5.6: Conexão do **Block** com o roteador.

O roteador foi configurado pelo **Block** com a proteção **host->host** como mostrado na Figura 5.7.

```

%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1112) -> 192.168.2.10(12345), 1 packet
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1113) -> 192.168.2.10(12345), 1 packet
router#sh access-lists 111
Extended IP access list 111
  deny tcp host 192.168.1.10 host 192.168.2.10 eq 12345 log (2 matches)
  permit ip any any log (27 matches)

```

Figura 5.7: Configuração do roteador contra ataques TCP proteção **host->host**.

Devemos observar, pelo apresentado na Figura 5.7, que o roteador foi configurado pela estação 192.168.2.10, que é a estação que contém o **Block**. Nesta Figura, verificamos, também, que o roteador nega duas tentativas de conexão com a porta 12345.

Na outra simulação de ataque, utilizamos o nível de proteção **net->host**. A linha da lista de acesso gerada neste caso foi

access-list 111 deny tcp 192.168.1.0 0.0.0.255 host 192.168.2.10 eq 12345 log.
e configuração do roteador passou a ser a mostrada na Figura 5.8.

```

%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1121) -> 192.168.2.10(12345), 1 packet
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1122) -> 192.168.2.10(12345), 1 packet
router#sh access-lists
Extended IP access list 111
  deny tcp 192.168.1.0 0.0.0.255 host 192.168.2.10 eq 12345 log (2 matches)
  permit ip any any log (27 matches)

```

Figura 5.8: Configuração do roteador contra ataques TCP proteção **net->host**.

Na simulação do ataque com a proteção **net->any** ativada, a linha da lista de acesso gerada

foi

`access-list 110 deny tcp 192.168.1.0 0.0.0.255 any eq 12345 log` e a configuração do roteador passou a ser a apresentada na Figura 5.9.

```
%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1134) -> 192.168.2.10(12345), 1 packet
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1135) -> 192.168.2.10(12345), 1 packet
router#sh access-lists
Extended IP access list 111
    deny tcp 192.168.1.0 0.0.0.255 any eq 12345 log (2 matches)
    permit ip any any log (27 matches)
```

Figura 5.9: Configuração do roteador contra ataques TCP proteção `net->any`.

O ataque com a proteção `any->any` ativada gerou a a declaração

`access-list 110 deny tcp any any eq 12345 log` na lista de acesso e a configuração do roteador é a mostrada na Figura 5.10.

```
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.2.10(32795) -> 192.168.1.1(23), 1 packet
%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1146) -> 192.168.2.10(12345), 1 packet
%SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.1.10(1147) -> 192.168.2.10(12345), 1 packet
router#sh access-lists
Extended IP access list 111
    deny tcp any any eq 12345 log (3 matches)
    permit ip any any log
```

Figura 5.10: Configuração do roteador contra ataques TCP proteção `any->any`.

Nas três últimas simulações de ataques à porta TCP, deixamos de apresentar as linhas capturadas pelo `tcpdump` por serem semelhantes às já mostradas anteriormente.

5.3.2 Ataques com a Ferramenta *targa*

O programa *targa* foi utilizado para produzir os ataques *Land*, *smurf* e *Teardrop* contra a estação que rodava o *Block*. A proteção escolhida para todos esses ataques foi a `net->any`. Os ataques foram lançados um por vez.

Ataques *smurf* e Ping da Morte

O ataque *smurf* foi gerado com a seguinte linha de comando:

```
targa 192.168.2.10 192.168.2.10 -t 2 -n 1.
```

Esse ataque consiste em gerar várias mensagens ICMP fragmentadas para a máquina alvo como mostra a Figura 5.11. Como o *Block* possui um gatilho de número de mensagens ICMP (`icmp_limit`) dentro da janela de captura igual a 5, conforme mostrado na Tabela 5.2, e a quantidade de mensagens o excedeu, o ataque foi classificado como *smurf*.


```

17:50:24.019888 179.86.76.107 > 192.168.2.10: (frag 4321:380026976+)
17:50:24.020257 179.86.76.107 > 192.168.2.10: (frag 4321:380027360+)
17:50:24.020664 179.86.76.107 > 192.168.2.10: (frag 4321:380027736+)
17:50:24.021030 179.86.76.107 > 192.168.2.10: (frag 4321:380028120+)
17:50:24.021398 179.86.76.107 > 192.168.2.10: (frag 4321:380028496+)
17:50:24.021762 179.86.76.107 > 192.168.2.10: (frag 4321:380028880+)
17:50:24.022129 179.86.76.107 > 192.168.2.10: (frag 4321:380029256+)
...

```

Figura 5.11: Ataque *smurf* capturado pelo tcpdump.

É importante observar que o programa **targa** falsifica o endereço IP de origem dos pacotes.

Esse ataque é registrado no arquivo **syslog** como mostrado na Figura 5.12.

```

Oct 20 17:50:45 vitima BLOCK: ICMP fragment detected from 244.60.253.65
(00:04:4d:31:88:da) against 192.168.2.10
Oct 20 17:50:45 vitima BLOCK: ICMP ECHO threshold exceeded, smurfs up. I
saw 588 packets
...

```

Figura 5.12: Registro do ataque *smurf* no arquivo do syslog.

A lista de acesso gerada é mostrada na Figura 5.13.

```

!ping of death attack detected
access-list 111 deny ip 244.60.253.0 0.0.0.255 any log
!smurf attack detected
access-list 111 deny icmp any any echo log

```

Figura 5.13: Lista de acesso do ataque *smurf*.

Um detalhe interessante a ser notado é a classificação desse ataque também como ping da morte pelo **Block**, conforme os comentários (declarações iniciadas com o símbolo !) colocados quando da geração da lista de acesso. Isso ocorre porque os pacotes ICMP estavam fragmentados.

A configuração do roteador após esse ataque é mostrada na Figura 5.14.

```

router#sh access-lists
Extended IP access list 111
  deny ip 244.60.253.0 0.0.0.255 any log
  deny icmp any any echo log
  permit ip any any log (39 matches)

```

Figura 5.14: Configuração do roteador após o ataque *smurf*.

Ataque *Land*

O ataque *Land* é gerado com a linha de comando

```
targa 192.168.2.10 192.168.2.10 -t 3 -n 1
```

O padrão desse ataque é a coincidência de endereços IP (origem e destino) e portas, que nesse caso, era a porta 113. A Figura 5.15 mostra o tráfego capturado pelo **tcpdump**.

```
17:56:28.661892 192.168.2.10.113 > 192.168.2.10.113: S 3868:3868(0) win 2048
17:56:28.661938 192.168.2.10.113 > 192.168.2.10.113: S 3868:3868(0) win 2048
```

Figura 5.15: Ataque *Land* capturado pelo **tcpdump**.

As entradas no arquivo **syslog** referentes a esse ataque são mostradas na Figura 5.16.

```
Oct 20 17:56:32 vitima BLOCK: Possible land attack from (00:04:4d:31:88:da)
against 192.168.2.10
```

Figura 5.16: Registro do ataque *Land* no arquivo do **syslog**.

O endereço MAC exibido na Figura 5.16 pertence à interface ethernet do roteador.

A lista de acesso gerada é mostrada na Figura 5.17, enquanto que a configuração do roteador é apresentada na Figura 5.18.

```
!IP spoofing attack detected - Land
access-list 110 deny ip 192.168.2.0 0.0.0.255 any log
```

Figura 5.17: Lista de acesso do ataque *Land*.

```
00:13:47: %SEC-6-IPACCESSLOGP: list 111 denied tcp 192.168.2.10(0) ->
192.168.1.1(0), 1 packet
00:13:51: %SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
router#sh access-lists
Extended IP access list 111
  deny ip 192.168.2.0 0.0.0.255 any log (30 matches)
  permit ip any any log
```

Figura 5.18: Configuração do roteador após o ataque *Land*.

Ataque *Teardrop*

O ataque *Teardrop* é lançado com o **targa** através do comando:

```
targa 192.168.2.10 192.168.2.10 -t 7 -n 1.
```

O padrão desse ataque, capturado pelo **tcpdump**, é mostrado na Figura 5.19.


```

17:59:56.055610 104.227.137.42.3570 > 192.168.2.10.1737: udp 28 (frag242:3600+)
17:59:56.055710 104.227.137.42 > 192.168.2.10: (frag 242:4024)
17:59:56.056912 104.227.137.42.3570 > 192.168.2.10.1737: udp 28 (frag242:3600+)
17:59:56.057779 104.227.137.42 > 192.168.2.10: (frag 242:4024)
17:59:56.073507 104.227.137.42.3570 > 192.168.2.10.1737: udp 28 (frag242:3600+)
17:59:56.073613 104.227.137.42 > 192.168.2.10: (frag 242:4024)
17:59:56.074844 104.227.137.42.3570 > 192.168.2.10.1737: udp 28 (frag242:3600+)
17:59:56.075667 104.227.137.42 > 192.168.2.10: (frag 242:4024)

```

Figura 5.19: Ataque *Teardrop* capturado pelo tcpdump.

O registro do ataque no arquivo `syslog` é mostrado na Figura 5.20.

```

Oct 20 18:00:21 vitima BLOCK: Possible teardrop attack from 51.15.46.80
(00:04:4d:31:88:da) against 192.168.2.10

```

Figura 5.20: Registro do ataque *Teardrop* no arquivo `syslog`.

A lista de acesso gerada é mostrada na Figura 5.21, enquanto que a configuração do roteador após o ataque é apresentada na Figura 5.22.

```

!teardrop attack detected
access-list 111 deny ip 51.0.0.0 0.255.255.255 any log

```

Figura 5.21: Lista de acesso do ataque *Teardrop*.

```

%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
router#sh access-lists
Extended IP access list 111
    deny ip 51.0.0.0 0.255.255.255 any log
    permit ip any any log (32 matches)

```

Figura 5.22: Configuração do roteador após o ataque *Teardrop*.

5.4 Ataque por Varredura de Portas

Para simular o ataque por varredura de portas, *scan*, foi utilizado o programa **nmap** versão 3.00 [33]. Foram feitas varreduras de portas dos protocolo TCP e UDP.

Para executar uma varredura nas portas TCP, o **nmap** é executado com a seguinte linha de comando:

```
nmap -sT 192.168.2.10 -p 1-12347
```

A linha acima executa uma varredura das portas TCP entre 1 e 12347 usando o método `connect()`, descrito no Capítulo 2. A janela de tempo de captura do **Block** (parâmetro `time-out`) foi aumentada para 25 segundos no arquivo de configuração para permitir uma maior

visibilidade do ataque. Na Figura 5.23, mostramos como esse tipo de ataque é capturado pelo `tcpdump`.

```
18:06:14.599654 192.168.1.10 > 192.168.2.10: icmp: echo request
18:06:14.601396 192.168.2.10 > 192.168.1.10: icmp: echo reply
18:06:14.955368 192.168.1.10.1024 > 192.168.2.10.5723: S 3330593914:3330593914(0)
win 5840 <mss 1460,sackOK,timestamp 1419330,nop,wscale 0> (DF)
18:06:14.956003 192.168.1.10.blackjack > 192.168.2.10.4661: S 3328323776:3328323776(0)
win 5840 <mss 1460,sackOK,timestamp 1419330,nop,wscale 0> (DF)
18:06:14.956496 192.168.1.10.1026 > 192.168.2.10.5341: S 3331678596:3331678596(0)
win 5840 <mss 1460,sackOK,timestamp 1419330,nop,wscale 0> (DF)
18:06:14.956902 192.168.1.10.1024 > 192.168.2.10.5723: S 3330593914:3330593914(0)
win 5840 <mss 1460,sackOK,timestamp 1419330,nop,wscale 0> (DF)
...
18:06:14.957026 192.168.2.10.5723 > 192.168.1.10.1024: R 0:0(0) ack 3330593915 win 0 (DF)
18:06:14.958650 192.168.2.10.4661 > 192.168.1.10.blackjack: R 0:0(0) ack 3328323777
win 0 (DF)
18:06:14.960272 192.168.2.10.5341 > 192.168.1.10.1026: R 0:0(0) ack 3331678597 win 0 (DF)
18:06:14.962439 192.168.2.10.5723 > 192.168.1.10.1024: R 0:0(0) ack 1 win 0 (DF)
...
```

Figura 5.23: Ataque de varredura de portas TCP capturado pelo `tcpdump`.

As primeiras linhas da Figura 5.23 mostram o `nmap` testando, através de mensagens ICMP, se a máquina alvo está acessível ou não. As linhas seguintes exibem as várias tentativas de conexões SYN com as portas da máquina alvo. Em seguida, pacotes RST são enviados para informar que portas não estão ativadas. Na sequência, é exibida, resumidamente, uma conexão com a porta 12345 que estava escutando na máquina alvo.

Para varreduras de portas UDP, o `nmap` é executado com a linha de comando `nmap -sU 192.168.2.10 -p 1-12347`

As linhas que detectam a varredura de portas UDP são mostradas na Figura 5.24. Podemos notar que o procedimento para verificar se a máquina está ativa é feito usando mensagens ICMP.


```

18:12:02.239087 192.168.1.10 > 192.168.2.10: icmp: echo request
18:12:02.239408 192.168.1.10.39379 > 192.168.2.10.http: . ack 2908772535 win
2048
18:12:02.240400 192.168.1.10 > 192.168.2.10: icmp: echo request
18:12:02.240509 192.168.2.10 > 192.168.1.10: icmp: echo reply
18:12:02.586099 192.168.1.10.39359 > 192.168.2.10.10590: udp 0
18:12:02.586245 192.168.1.10.39359 > 192.168.2.10.6027: udp 0
18:12:02.586370 192.168.1.10.39359 > 192.168.2.10.12115: udp 0
18:12:02.586493 192.168.1.10.39359 > 192.168.2.10.8479: udp 0
18:12:02.586670 192.168.1.10.39359 > 192.168.2.10.3431: udp 0
...
18:12:02.598615 192.168.2.10 > 192.168.1.10: icmp: 192.168.2.10 udp port 10590
unreachable [tos 0xc0]
18:12:02.599432 192.168.2.10 > 192.168.1.10: icmp: 192.168.2.10 udp port 6027
unreachable [tos 0xc0]
18:12:02.600294 192.168.2.10 > 192.168.1.10: icmp: 192.168.2.10 udp port 12115
unreachable [tos 0xc0]
18:12:02.601110 192.168.2.10 > 192.168.1.10: icmp: 192.168.2.10 udp port 8479
unreachable [tos 0xc0]
18:12:02.601928 192.168.2.10 > 192.168.1.10: icmp: 192.168.2.10 udp port 3431
unreachable [tos 0xc0]
...

```

Figura 5.24: Ataque de varredura de portas UDP capturado pelo **tcpdump**.

A Figura 5.25 mostra como os ataques por varreduras de portas TCP e UDP são registrados no **syslog**.

```

Oct 20 18:06:30 vitima BLOCK: Possible TCP port scan from 192.168.1.10 (5673 ports)
against 192.168.2.10
...
Oct 20 18:12:11 vitima BLOCK: Possible UDP port scan from 192.168.1.10 (82 ports)
against 192.168.2.10
Oct 20 18:12:35 vitima BLOCK: Possible UDP port scan from 192.168.1.10 (96 ports)
against 192.168.2.10
...

```

Figura 5.25: Registro do ataque de varredura de portas TCP e UDP no arquivo **syslog**.

A lista de acesso gerada pelo **Block** após essa seção de ataques é mostrada na Figura 5.26. Ressaltamos que a lista de acesso é idêntica para os dois tipos de ataques.

```

!tcp or udp scan detected
access-list 111 deny ip 192.168.1.0 0.0.0.255 any log
!tcp or udp scan detected
access-list 111 deny ip 192.168.1.0 0.0.0.255 any log

```

Figura 5.26: Lista de acesso do ataque de varredura de portas TCP e UDP.

A configuração do roteador é mostrada na Figura 5.27, tendo sido mudado o nível de proteção para **host->host**.

```
%SYS-5-CONFIG_I: Configured from console by vty0 (192.168.2.10)
router#sh access-lists
Extended IP access list 111
    deny tcp any any eq 12345 log
    deny ip 192.168.1.0 0.0.0.255 any log
    permit ip any any log (30 matches)
router#
%SEC-6-IPACCESSLOGDP: list 111 denied icmp 192.168.1.10 -> 192.168.2.10 (8/0),
1 packet
router#
```

Figura 5.27: Configuração do roteador após o ataque de varredura de portas TCP e UDP.

Os ataques foram efetivamente bloqueados, como mostra a Figura 5.28. Nesta Figura, as tentativas de conexão entre as máquinas 192.168.1.10 (máquina atacante) e 192.168.2.10 (máquina alvo) foram capturadas pelo **tcpdump**.

```
18:09:52.202123 192.168.1.10 > 192.168.2.10: icmp: echo request (DF)
18:09:52.203699 192.168.1.1 > 192.168.1.10: icmp: host 192.168.2.10
unreachable - admin prohibited filter
18:09:53.200195 192.168.1.10 > 192.168.2.10: icmp: echo request (DF)
18:09:53.201739 192.168.1.1 > 192.168.1.10: icmp: host 192.168.2.10
unreachable - admin prohibited filter
```

Figura 5.28: Bloqueio de varredura de portas capturado pelo **tcpdump**.

5.5 Ataque *SYN Flooding*

O ataque *SYN Flooding* foi feito usando o programa **hping2** [34]. A linha de comando usada para lançar o ataque é

```
hping2 -S 192.168.1.10
```

A ferramenta **Block** detectou o ataque. O correspondente registro no arquivo **syslog** é mostrado na Figura 5.29.

```
Oct 21 10:36:24 vitima BLOCK: Possible SYNFLOOD from 192.168.1.10 (00:80:ad:c8:97:76),
against 192.168.2.10. I saw 29 packets
```

Figura 5.29: Registro do ataque *SYN Flooding* no arquivo **syslog**.

A lista de acesso gerada pelo **Block** é mostrada na Figura 5.30.

```
!synflooding attack detected
access-list 111 deny ip 192.168.1.0 0.0.0.255 any log
```

Figura 5.30: Lista de acesso gerada pelo **Block** do ataque *SYN Flooding*.

Na Figura 5.31, mostramos a lista de acesso no roteador, enquanto que na e na Figura 5.32 mostramos o ataque sendo bloqueado e a geração correspondente das mensagens ICMP.

```
router#sh access-list
Extended IP access list 111
  deny ip 192.168.1.0 0.0.0.255 any log (15 matches)
  permit ip any any log (30 matches)
router#
```

Figura 5.31: Listas no roteador após o ataque *SYN Flooding*.

Na Figura 5.32, mostramos o bloqueio do ataque,

```
HPING 192.168.2.10 (eth0 192.168.2.10): S set, 40 headers + 0 data bytes
ICMP Packet filtered from ip=192.168.1.1 name=router.exemplo.org
ICMP Packet filtered from ip=192.168.1.1 name=router.exemplo.org
ICMP Packet filtered from ip=192.168.1.1 name=router.exemplo.org
```

Figura 5.32: Bloqueio de ataque *SYN Flooding*.

A mensagem de correio eletrônico enviada ao administrador informando-o do ataque ocorrido é mostrada na Figura 5.33.

```
Date: Mon, 21 Oct 2002 10:36:28 -0200
From: Block - Attacks Block <Block@vitima.exemplo2.com>
To: root@vitima.exemplo2.com
Subject: Attacks!

Dear Sysadmin

There were detected 1 attacks
Time taken to block them in the router: 4.00 seconds
Possible SYNFLLOOD from 192.168.1.10 (00:80:ad:c8:97:76), against 192.168.2.10.
I saw 29 packets

Block - Attacks Block
```

Figura 5.33: Mensagem enviada ao administrador da rede informando sobre a ocorrência do ataque *SYN Flooding*.

Podemos observar, pela mensagem mostrada na Figura 5.33, que este ataque foi bloqueado em cerca de 4.0 segundos.

Como ficou demonstrado, em todos os testes realizados o **Block** foi efetivo e executou as tarefas para o qual foi projetado.

Capítulo 6

Conclusões e Perspectivas

6.1 Conclusões

Nesta Dissertação, apresentamos o projeto e o desenvolvimento de uma ferramenta de bloqueio automático de ataques através da implantação de listas de acesso em roteadores Cisco. A ferramenta desenvolvida, chamada **Block**, é baseada em código aberto e funciona em plataformas Linux.

A ferramenta desenvolvida, recebe as informações necessárias para o seu funcionamento através de um *NIDS*. O *NIDS* escolhido, **watcher**, gera relatórios dos vários tipos de ataques que ele detecta e baseia-se no paradigma de assinaturas de ataques. Foram aproveitadas as funções de reconhecimento de ataques, sendo adicionada mais uma assinatura para portas TCP e UDP. As modificações implementadas foram o módulo construtor de listas de acesso, módulo de conexão com o roteador e o módulo de envio de mensagens de correio eletrônico para o administrador sobre os ataques detectados. O **Block**, entretanto, pode ser adaptado a qualquer ferramenta *NIDS* que gere relatórios dos ataques detectados.

O **Block** pode detectar tanto ataques internos quanto externos, bloqueando ambos, além de possuir o recurso de respostas automáticas a ataques que pode ser útil quando os administradores não estiverem presentes. Quatro níveis de proteção estão disponíveis para serem utilizados. Como o *NIDS* sob o qual ele está implementado processa o cabeçalho a nível de protocolos, a inclusão de novas assinaturas, no modelo atual, é um procedimento que exige modificação do código fonte e recompilação da ferramenta. A ferramenta **Block** é uma ajuda importante no combate a ataques, pois impede que seus danos se alastrem, uma vez que o ataque é barrado na fronteira de entrada e em seus momentos iniciais. Esse procedimento elimina o lapso de tempo entre a descoberta e o bloqueio de atividades suspeitas feito pelos administradores de sistemas. A ferramenta pode ser utilizado como suporte importante de uma política de segurança.

De acordo com os testes realizados, as listas de controle de acesso foram criadas com base no nível de proteção escolhido e as conexões envolvidas foram bloqueadas. Assim, todos os ataques simulados foram adequadamente tratados pelo **Block** e os objetivos de resposta automática a

ataques foram alcançados.

6.2 Perspectivas de Desenvolvimento

Algumas sugestões para a evolução da ferramenta desenvolvida :

1. **Encriptação da Senha do Roteador:** Para que essa sugestão seja implementada, o IOS do roteador utilizado deve suportar algum protocolo de encriptação. A implementação dessa sugestão impedirá que as senhas do roteador sejam capturadas por *sniffers*;
2. **Extensão do Banco de Assinaturas:** Como o banco de assinaturas do **Block** é restrito, faz-se necessário deixá-lo atualizado para os novos ataques. A melhor abordagem seria a definição de uma linguagem de construção de assinaturas, agilizando o bloqueio de ataques recém-descobertos. A definição da linguagem deve levar em consideração as informações de atualização de assinaturas disponibilizadas por organizações de segurança;
3. **Variações de Ataques:** O sistema conheceria protocolos de rede o suficiente para detectar perfis de tráfego que fogem ao padrão considerado normal, como por exemplo, taxas de abertura de conexões fora do usual, pacotes que fogem das especificações do protocolo etc. Dessa forma, variações de ataques e novos ataques poderão ser detectados;
4. **Integração com Banco de Dados:** Para melhorar sua capacidade investigativa, a integração com um banco de dados é recomendada. Esse recurso permite ao analista trabalhar com um volume maior de informações. O problema de detectar *scans* espaçados no tempo poderia ser resolvido com esse recurso;
5. **Integração com Outros Dispositivos:** Além de roteadores Cisco, outros dispositivos como *firewalls* poderiam ser suportados, pois eles possuem uma melhor capacidade de filtragem que os roteadores. A interação do **Block** com eles resultaria em um nível de segurança ainda maior.
6. **Acompanhamento dos Ataques:** Fazendo com que o roteador gere *logs* para uma estação, é possível acompanhar o efeito do bloqueio pelas lista de acesso, e
7. **Sistema de Detecção de Intrusões Distribuídos:** Algumas soluções de IDS estão migrando para esse modelo, onde vários IDS ficam distribuídos em várias redes, comunicando-se com uma estação central de análise (*Distributed Intrusion Detection System - dIDS*). Essa estação conteria um banco de dados e uma interface visando acelerar a análise dos dados recebidos. Os IDS, ou agentes, localizados em diversos pontos de rede, reportam informações de ataques ao servidor central. Esse modelo permite personalizar a segurança de cada segmento, tornando-a homogênea por todas as redes. O **Block** poderia se encaixar nesse modelo, bloqueando o avanço de ataques internos ou externos.

8. **Desativação de Entradas na Lista de Acesso:** o **Block** pode evoluir para acompanhar os ataques a partir da verificação de *logs* enviados pelo roteador para uma estação servidora de *logs*. Esse acompanhamento permitirá retirar as entradas referentes a ataques que deixaram de ser realizados, refazendo a lista de acesso.

Apendice A - Hierarquia de Processos e Diagramas de Fluxo de Dados

Neste Apêndice, apresentamos a hierarquia dos processos e os diagramas de Fluxo de dados (DFD) da ferramenta **Block**. Esses diagramas permitem visualizar as relações entre **Block** e outros sistemas utilizados.

A Figura 1 apresenta a hierarquia dos processos que compõem a ferramenta. Abaixo, apresentamos os fluxos de dados entre o programa e o computador e o segundo modo de operar, usando o computador. O diagrama apresentado dá uma visão mais detalhada de como a ferramenta é desenvolvida hierarquicamente.



Figura 1: Hierarquia de processos da ferramenta **Block**.

A Figura 2 apresenta o diagrama de fluxo de dados da ferramenta. As setas representam os fluxos de dados entre o programa e o computador e o segundo modo de operar, usando o computador.

A seguir, apresentamos os processos:

1. Atualizar lista de acesso de IP:

Apendice A - Hierarquia de Processos e Diagramas de Fluxo de Dados

Neste Apêndice, exibiremos a hierarquia dos processos e os diagramas de fluxo de dados (DFDs) da ferramenta **Block**. Esses diagramas podem ser utilizados para realizar as expansões ao **Block** sugeridas nesse trabalho.

A Figura 1, mostra a hierarquia de processos que compõem a ferramenta. Como apresentamos no Capítulo 4, o programa é modularizado e seguiu o modelo de programação estruturada. O diagrama apresentado dá uma visão mais detalhada de como os módulos estão distribuídos hierarquicamente.

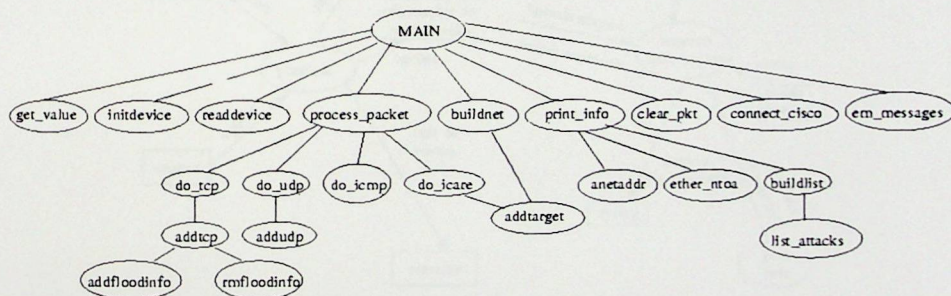


Figura 1: Hierarquia de processos da ferramenta **Block**.

A Figura 2, mostra o diagrama de fluxo de dados no nível 0. As entidades externas que se comunicam com o sistema são:

1. O tráfego da rede através dos pacotes;
2. O usuário, através das opções de configuração;

3. O roteador, que recebe as listas de acesso geradas, e
4. O arquivo de *logs*, onde as informações de possíveis ataques são armazenadas.

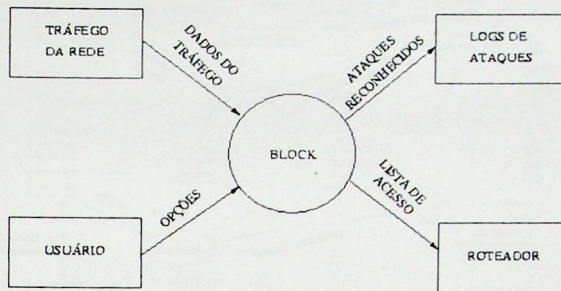


Figura 2: Diagrama de Fluxo de Dados, nível 0.

A Figura 3 mostra o DFD de nível 1. Nesse nível de abstração, apresentamos os detalhes sobre o fluxo de dados entre os diversos módulos da ferramenta.

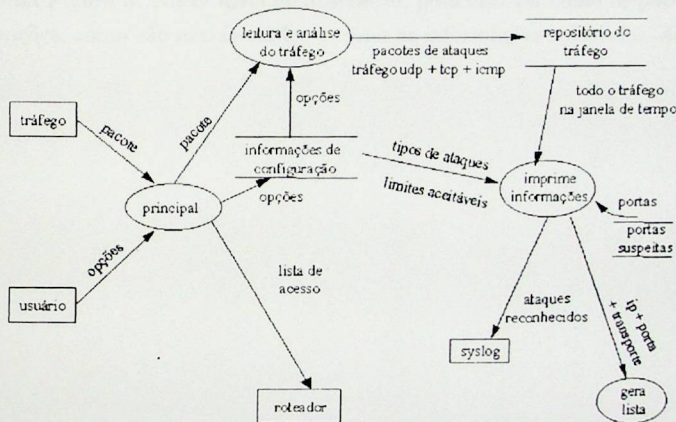


Figura 3: Diagrama de Fluxo de Dados, nível 1.

A Figura 4, mostra o DFD de nível 2, onde são apresentados os detalhes dos tipos de dados que fluem pelos diferentes módulos da ferramenta.

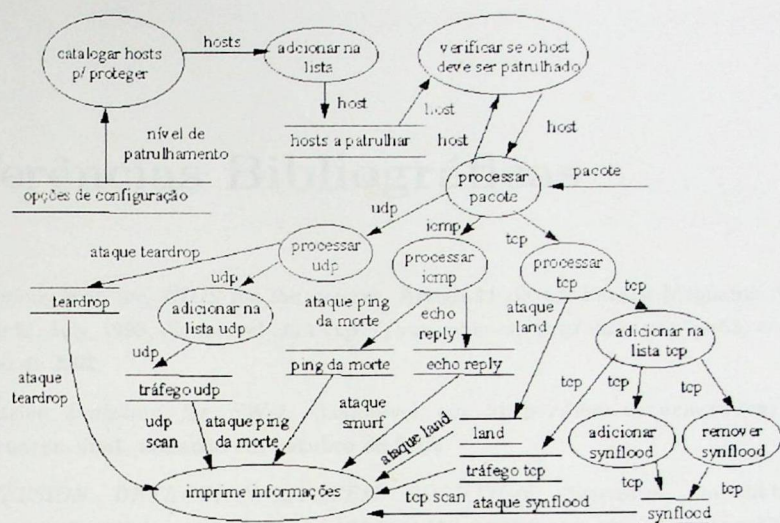


Figura 5: Diagrama de Fluxo de Dados, nível 3.

Referências Bibliográficas

- [1] hyperion, *Watcher, NIDS for the masses*, Artigo 11 de 15, Phrack Magazine, Volume 8, Issue 53, July, 1998, disponível em <http://www.phrack.org/show.php?p=53>, acessado em junho de 2002.
- [2] *Intrusion Detection for FW-1*, disponível em <http://www.enteract.com/~lspitz/intrusion.html>, acessado em outubro de 2002.
- [3] *INTRUSION DETECTION SYSTEM SOLUTION*, disponível em <http://www.cisco.com/en/US/netsol/ns110/ns129/ns131/ns292/networking\protect\T1\textunderscoresolutions\protect\T1\textunderscorepackage.html>, acessado em outubro de 2002.
- [4] Chambers, C., Dolske, J., Iyer, J., *TCP/IP Security*, Department of Computer and Information Science, Ohio State University, Columbus, Ohio, disponível em <http://www.linuxsecurity.com/resource-files/documentation/tcpip-security.html>, acessado em julho de 2002.
- [5] *Port Numbers*, Internet Assigned Numbers Authority (IANA), disponível em <http://www.iana.org/assignments/port-numbers/>, acessado em junho de 2002.
- [6] *sscan*, disponível em <http://packetstormsecurity.org/9901-exploits/sscan.tar.gz>, acessado em maio de 2002.
- [7] Fyodor, *The Art of Port Scanning*, disponível em <http://www.insecure.org/nmap/nmap-doc.html>, acessado em maio de 2002.
- [8] Johns, M., *Identification Protocol*, RFC-1413, Feb., 1993, disponível em <ftp://ftp.isi.edu/in-notes/rfc1413.txt>, acessado em julho de 2002.
- [9] Fyodor, *Remote OS Detection via TCP/IP Stack Finger Printing by Fyodor*, April 1999, disponível em <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>, acessado em julho de 2002.
- [10] Postel, J., *TRANSMISSION CONTROL PROTOCOL*, RFC-793, September., 1981, disponível em <ftp://ftp.isi.edu/in-notes/rfc793.txt>, acessado em junho de 2002.

- [11] Baker, F., *Requirements for IP Version 4 Routers*, RFC-1812, June 1995, disponível em <ftp://ftp.isi.edu/in-notes/rfc1812.txt>, acessado em maio de 2002.
- [12] DARPA, *Internet Protocol*, RFC-791, September, 1981, disponível em <ftp://ftp.isi.edu/in-note/rfc791.txt>, acessado em julho de 2002.
- [13] Rekhter, Y., et al, *Address Allocation for Private Internet*, RFC-1918, February, 1996, disponível em <ftp://ftp.isi.edu/in-note/rfc1918.txt>, acessado em junho de 2002.
- [14] Ferguson P., Sennie D., *Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing*, RFC-2267, January, 1998, disponível em <ftp://ftp.isi.edu/in-note/rfc2267.txt>, acessado em maio de 2002.
- [15] *smurf*, disponível em <http://packetstorm.decepticons.org/exploits/DoS/smurf.c>, acessado em julho de 2002.
- [16] Huegen, C. A., *The Latest in Denial of Service Attacks: 'Smurffing' Description and Information to Minimize Effects*, July, 1998, disponível em <http://www.pentics.net/denial-of-service/white-papers/smurf.txt>, acessado em julho de 2002.
- [17] *Fraggle*, disponível em <http://packetstorm.decepticons.org/exploits/DoS/fraggle.c>, acessado em julho de 2002.
- [18] daemon9, *Large Packet Attacks (AKA Ping of Death)*, disponível em <http://www.phrack.org/phrack/50/p50-03>, acessado em julho de 2002.
- [19] *Land*, disponível em <http://www.attrition.org/security/denial/w/latierra.dos.html>, acessado em julho de 2002.
- [20] *Teardrop*, disponível em <http://www.attrition.org/security/denial/w/teardrop.dos.html>, acessado em julho de 2002.
- [21] CERT Coordination Center, *IP Denial-of-Service Attacks*, CERT Advisory CA-1997-28, May, 1998, disponível em <http://www.cert.org/advisories/CA-1997-28.html>, acessado em agosto de 2002.
- [22] Mixer, *Tribe Flood Network 3000*, disponível em <http://www.apocalypseonline.com/security/text/tribeflood.asp>, acessado em julho de 2002.
- [23] Mixer, *Tribe Flood Network 2000*, disponível em <http://packetstorm.decepticons.org/groups/mixer/tfn2k.tgz>, acessado em julho de 2002.
- [24] Mixer, *Tribe Flood Network*, disponível em <http://packetstorm.decepticons.org/groups/mixer/tfn.tgz>, acessado em julho de 2002.

- [25] *Stacheldraht V4*, disponível em <http://packetstormsecurity.org/distributed/stachel.tgz>, acessado em julho de 2002.
- [26] Yardley, T., *Distributed Attacks and the Way to Deal With Them*, disponível em http://packetstormsecurity.org/papers/contest/Tim_Yardley.doc, acessado em agosto de 2002.
- [27] *Back Orifice*, disponível em <http://packetstormsecurity.org/trojans/bo/bo.zip>, acessado em julho de 2002.
- [28] *Netbus*, disponível em <http://packetstormsecurity.org/trojans/NetBus170.zip>, acessado em julho de 2002.
- [29] Allan, L. et al., *Cisco Router Configuration*. Macmillian Technical Publishing, 1998.
- [30] NightAxis, Rain Forest Puppy, *Purgatory 101: Learning to cope with the syns of the Internet*, disponível em <http://packetstormsecurity.nl/papers/contest/RFP.doc>, acessado em agosto de 2002.
- [31] Secure Shell Version 1 Support, disponível em <http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s5/sshv1.htm>, acessado em outubro de 2002.
- [32] Mixer, *targa*, versão 1.0, June 1998, disponível em <http://packetstormsecurity.org/DoS/targa.c>, acessado em julho de 2002.
- [33] Fyodor, *nmap*, versão 3.0 , disponível em <http://download.insecure.org/nmap/dist/nmap-3.00.tgz>, acessado em setembro de 2002.
- [34] *hping2*, disponível em <http://www.kyuzz.org/antirez/software.html>, acessado em outubro de 2002.
- [35] *Tcpdump*, disponível em www.tcpdump.org, acessado em agosto de 2002.

IMPRESSOS