ÁKIO NOGUEIRA BARBOSA

UM SISTEMA PARA ANÁLISE ATIVA DE COMPORTAMENTO DE *FIREWALL*

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

São Paulo
2006

ÁKIO NOGUEIRA BARBOSA

UM SISTEMA PARA ANÁLISE ATIVA DE COMPORTAMENTO DE *FIREWALL*

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. Pedro Luís Próspero Sanchez

São Paulo
2006

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 22 de novembro de 2006.

Ákio Nogueira Barbosa

Prof. Dr. Pedro Luís Próspero Sanchez

FICHA CATALOGRÁFICA

Barbosa, Ákio Nogueira
Um sistema para análise ativa de comportamento de *firewall* / A.N. Barbosa. -- ed.rev. -- São Paulo, 2006.
120 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Redes de computadores (Segurança) 2.Proteção e segurança de sistemas de computação I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

A MINHA FAMÍLIA, PRINCIPALMENTE AOS MEUS PAIS E A NÚBIA, PELO CARINHO, COMPREENSÃO E INCENTIVO PARA QUE ESTA DISSERTAÇÃO PUDESSE SER ELABORADA.

AGRADECIMENTOS

Ao Grande Criador do Universo por conceder-me saúde e disposição para realização deste mais importante passo em minha vida.

A minha família e a Engª. Núbia Cristina Lisboa pela paciência, incentivo e compreensão de minha ausência das incontáveis horas em que não foi possível estar presente, mas que foram recompensadas pela conclusão deste trabalho.

Ao Professor Doutor Pedro Luís Próspero Sanchez, pela orientação e incentivo nesta empreitada.

Ao Professor Doutor Volnys Borges Bernal, pelas inestimáveis discussões técnicas, sugestões e revisões.

Ao Professor Doutor Moacyr Martucci Júnior, pela atenção nesta etapa importante de minha vida.

Ao amigo Engº. Marcelo Succi de Jesus Ferreira, pelo auxílio nas implementações.

Aos amigos e colegas do Núcleo de Segurança de Redes de Alta Velocidade (NSRAV).

A todas bibliotecárias da EPUSP especialmente a Maria Cristina Martinez Bonésio, pelas revisões bibliográficas.

A todos as funcionárias da secretaria de pós-graduação, especialmente a Regina.

A todos os amigos e colegas que contribuíram diretamente ou indiretamente para a realização deste trabalho, cujos nomes não foram citados aqui.

RESUMO

Devido à importância dos *firewalls* para proteção de redes de computadores, muito se estuda no sentido do aprimoramento das técnicas de proteção e no desenvolvimento de novas técnicas para serem utilizadas na análise destes. Com enfoque neste tema, esta dissertação trata a respeito da viabilidade da técnica de injeção de pacotes e observação dos resultados para analisar o comportamento de *firewalls* de rede para a pilha TCP/IP, resultando em uma técnica alternativa para análise de *firewalls*. Para mostrar a validade da técnica foi proposta uma arquitetura e, como prova de conceito, foi implementado um protótipo do sistema de análise. Foram também efetuados alguns testes. A técnica de injeção de pacotes e observação dos resultados mostrou-se viável para algumas situações. Para outras, são necessárias estudos adicionais para redução da explosão combinatória.

ABSTRACT

Due to the importance of the firewalls for protection of network computers, a lot of studies has been done in order of the improvement of the protection techniques and in the development of new techniques to be used in the analysis of them. With focus in this theme, this thesis considers the viability of the technique of injection of packages and observation of the results to analyze the behavior of network firewalls for stack TCP/IP, resulting in an alternative technique for analysis of firewalls. To show the validity of the technique an architecture was proposed and, as a concept proof, a prototype of the analysis system was implemented. Also was implemented some tests. The technique of injection of packages and observation of the results reveled viable for some situations. For others, addictionals studies are necessary for reduction of the combinatory explosion.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACK - Acknowledgment

ARP - Address Resolution Protocol

BGP - Border Gateway Protocol

CERT - Computer Emergency Response Team

CPU - Central Processing Unit

DNS - Domain Name System

DoS - Denial of Service

DDoS - Distributed Denial of Service

EGP - Exterior Gateway Protocol

FACE - Firewall Analysis and Configuration Engine

FANG - Firewall Analysis Engine

FO - Fragment Offset

FTP - File Transfer Protocol

HTML - Hyper Text Markup Language

ICMP - Internet Control Message Protocol

IDS - Intrusion Detection System

IP - Internet Protocol

ISN - Initial Sequence Number

LFA - Lumeta Firewall Analyser

MAC - Media Access Control

MDL - Model Definition Language

MTU - Maximum Transmission Unit

NAT - Network Address Translation

ns	-	nano segundo
PING	-	Packet Internet Groper
RAM	-	Random Acess Memory
RECV	-	Receive
RFC	-	Request For Comments
RIP	-	Routing Information Protocol
RST	-	Reset
s	-	segundos
SERV	-	Servidor
SNA	-	Sequence Number Attack
SNPA	-	Sequence Number Prediction Attack
SYN	-	Synchronize Sequence Numbers Flag
TCP	-	Transfer Control Protocol
UDP	-	User Datagram Protocol
VPN	-	Virtual Private Network
WWW	-	World Wide Web

CAPÍTULO 1

INTRODUÇÃO

A segurança de dados em sistemas de informação é um assunto de extrema importância e uma das áreas mais críticas nas ciências da computação. Um dos fatores de maior preocupação é a proteção e controle do acesso as informações que trafegam entre as diferentes redes de computadores (BURK, 2004).

Ataques a computadores interligados em rede objetivando roubo ou acesso não autorizado às informações têm causado enormes prejuízos e danos a instituições dos mais variados segmentos. Este cenário exige recursos com mecanismos eficazes e com capacidade de elevar a proteção e o grau de segurança dos ambientes computacionais.

Equipamentos de restrição e controle de tráfego, como roteadores e *firewalls*, são utilizados para reforçar a segurança de redes de computadores (CHESWICK, 2005; POUW, 2003; STREBE, 2002; WOLL, 2004b; ZWICKY, 2000).

Um sistema de *firewall*[1], que geralmente incorpora técnicas de filtragem de pacotes[2], *proxy*, tradução de endereços de redes (NAT - *Network Address Translation*) e portas ou VPN (*Virtual Private Network*) tornou-se um elemento

[1] Sistema de *firewall* ou *firewall* é utilizado no contexto de hardware, software ou uma combinação destes.

[2] Pacote é a denominação dada a qualquer dado transmitido pela rede.

indispensável para proteger redes privadas contra acessos não autorizados, isolando uma intranet corporativa da *Internet* pública, ou ainda, isolando as diversas redes de uma mesma corporação entre si (ADI, 2003; AL-TAWIL, 1999; PERMPOONTANALARP, 2001b; STREBE, 2002).

Freqüentemente existe a necessidade de conhecer o nível de proteção oferecido por um *firewall*, bem como suas deficiências. *Firewalls* não são softwares ou equipamentos que podem simplesmente ser retirados da caixa, conectados na rede e utilizados instantaneamente. Precisam ser adequadamente configurados, geralmente seguindo uma política de segurança da informação corporativa, para que possam atender necessidades específicas de cada rede. Além disso, a configuração é dinâmica e precisa ser revista periodicamente, seja quando novas vulnerabilidades são descobertas, quando são efetuadas alterações na arquitetura da rede ou ainda quando a política de segurança da informação corporativa é modificada (AL-SHAER, 2003; 2004b).

1.1 MOTIVAÇÃO

Uma das grandes dificuldades encontradas é a validação da configuração de sistemas de *firewalls* para que operem de maneira satisfatória seguindo as especificações estabelecidas na política de segurança da informação corporativa que, por sua vez, contempla a política de restrição de comunicação, sendo descrita em linguagem de alto nível (ADI, 2003; AL-SHAER, 2003).

Os administradores de rede são responsáveis por implementar a política de segurança usando, na maioria das vezes, uma linguagem de baixo nível para configurar as bases-de-regras desses equipamentos de proteção de redes de computadores (WOLL, 2001). Além disso, determinadas funcionalidades dos *firewalls* necessitam de regras específicas para habilitá-las, não sendo geralmente simples determinar sua correta configuração ou, até mesmo, se alguma funcionalidade de proteção está ativa.

A configuração das bases-de-regras destes equipamentos para que

operem de forma adequada pode ser uma tarefa extremamente complexa, exaustiva, onerosa e propensa a erros. Além disso, as configurações destas bases-de-regras para cada interface tipicamente permitem aos administradores de segurança definir vários parâmetros tais como: intervalos de endereços IP e grupos de serviços (conjunto de protocolos e número de portas correspondentes) que formam as bases-de-regras, como no exemplo apresentado na Tabela 1.1. Uma única regra tipicamente especifica: endereço IP de origem, porta TCP ou UDP de origem, endereço IP de destino, porta TCP ou UDP de destino, protocolo utilizado, direção (entrada ou saída) e uma ação correspondente.

Tabela 1.1 – Base de regras implementada a partir da política de segurança.

REGRA Nº	IP DE ORIGEM	PORTA ORIGEM	IP DE DESTINO	PORTA DESTINO	PROTOCOLO	DIREÇÃO	AÇÃO
1	10.0.161.12	>1023	143.107.254.714	110	TCP	SAÍDA	PERMITIR
2	10.0.161.12	>1023	QUALQUER	443	TCP	SAÍDA	PERMITIR
3	143.107.254.71	110	10.0.161.12	>1023	TCP	ENTRADA	PERMITIR
4	192.168.0.0/24	22	10.0.161.12	>1023	TCP	ENTRADA	PERMITIR
5	QUALQUER	80	10.0.161.12	>1023	TCP	ENTRADA	PERMITIR
6	143.107.254.73	143	10.0.161.12	>1023	TCP	ENTRADA	PERMITIR
N							

Nos *firewalls* as regras são sensíveis à ordem em que aparecem na base de regras: o *firewall* verifica se a primeira regra da base de regras se aplica ao pacote, então o pacote é aceito (*accept*), rejeitado (*reject*) ou descartado (*drop*) de acordo com a ação especificada na primeira regra, caso contrário, o *firewall* continua verificando se a segunda regra se aplica, e assim sucessivamente até que seja verificada uma ação a ser tomada.

Quando as bases-de-regras são examinadas, observa-se que o procedimento de implementação, geralmente conduz diferentes tipos de erros de configuração (WOLL, 2004a), que podem ser justificados principalmente pelos seguintes motivos:

- Redundância de regras na base-de-regras;
- A diversidade de sintaxes e parâmetros utilizados por cada fabricante;

- A quantidade de regras a serem implementadas nos diversos *firewalls* e as inconsistências entre as bases-de-regras (no caso onde exista mais de um *firewall*);

Devido aos motivos mencionados anteriormente é possível que persistam brechas de segurança na configuração do *firewall* (KAMARA, 2003; NOURELDIEN, 2000; WOLL, 2004a), o que permite perceber a complexidade de projetar, implementar, gerenciar e testar uma ou mais bases de regras de *firewalls*, que dependendo da dimensão da rede, quantidade e modelos de equipamentos existentes, pode tornar essa tarefa ainda mais complexa.

Como conseqüência dessas dificuldades, pode-se ter uma configuração inadequada do *firewall*, resultando em uma falsa sensação de segurança.

Dessa forma, existe a necessidade de conhecer efetivamente que tipo de pacotes o que o *firewall* aceita (permite), descarta ou restringe (bloqueia) e mecanismos de proteções existentes, ou seja, de conhecer o comportamento do *firewall*.

Essa necessidade pode surgir em diversas situações como, por exemplo, quando é efetuada uma análise pela equipe de segurança de redes, por uma equipe de auditoria para confirmação de que as normas definidas na política de segurança foram aplicadas conforme a especificação, ou até mesmo para se descobrirem falhas de segurança de rede (FLYNT, 2004; PERMPOONTANALARP, 2001a; 2001b).

Existem diversos fabricantes de *firewalls* no mercado: (CISCO *SYSTEMS*, 2002; *CHECK POINT*, 2001a; LUCENT *TECHNOLOGIES*, 1998; SUN *MICROSYSTEMS*, 2002), apenas para mencionar alguns. Muitos desses fabricantes oferecem equipamentos com sofisticadas ferramentas para configuração de *firewalls* e roteadores, mas nenhum deles direciona o foco para a tecnologia de ferramentas que efetuam testes para a verificação das funcionalidades de proteção ativas em *firewalls*.

Algumas ferramentas efetuam testes de análise passiva, ou seja, análise que não envolvem o envio de pacotes.

A motivação para o desenvolvimento deste trabalho provém dos seguintes fatos:

- A grande importância de um esquema de testes para descoberta do comportamento de mecanismos de proteção ativos em *firewalls*, razão pela qual existem tantas ferramentas disponíveis visando efetuar testes de *firewalls* e tanto esforço por parte dos pesquisadores para aprimorar essas ferramentas.
- A lacuna deixada quando se trata de ferramentas de análise ativa, por injeção de pacotes e observação de resultados aplicadas a testes de *firewalls*, já que a maioria das ferramentas são desenvolvidas para descoberta de erros de sintaxe nas bases de regras, geração automática de regras, efetuam análise passiva e geralmente se aplicam a *firewalls* de fabricantes específicos.
- A escassez, bem como a limitação apresentada pelas ferramentas de testes existentes, quando se deseja verificar a aderência das regras implementadas à política de restrição de acesso a rede da empresa, impedindo uma estimativa mais precisa sobre o comportamento dos *firewalls*.
- Determinar a reação do *firewall* mediante ataques específicos.

1.2 OBJETIVOS

O principal objetivo deste trabalho consiste em um estudo para verificar a viabilidade da utilização de técnicas de análise ativa, por injeção de pacotes e observação dos resultados, para descoberta do comportamento de *firewalls* para redes TCP/IP que operem na camada de redes.

Para isso, é proposto um modelo de arquitetura e, neste âmbito, a implementação de um sistema protótipo, cujos resultados servirão para comprovar ou não a viabilidade da técnica de injeção de pacotes e observação dos resultados na verificação das funcionalidades de proteção ativas em um *firewall*. Como prova de conceito, foram escolhidos alguns testes para serem implementados: Teste para análise de filtragem UDP, Teste

para análise de filtragem TCP não precedido de estabelecimento de conexão, Teste para análise de filtragem TCP no estabelecimento de conexão, Teste para análise de proteção SYN *Flood*, etc.

1.3 ESCOPO

O escopo deste trabalho é ilustrado na Figura 1.1, e está direcionado ao estudo de técnicas alternativas utilizadas para diagnóstico de comportamento de *firewalls* de rede, visando identificar mecanismos de proteção e configurações que estejam ativas nesses dispositivos.

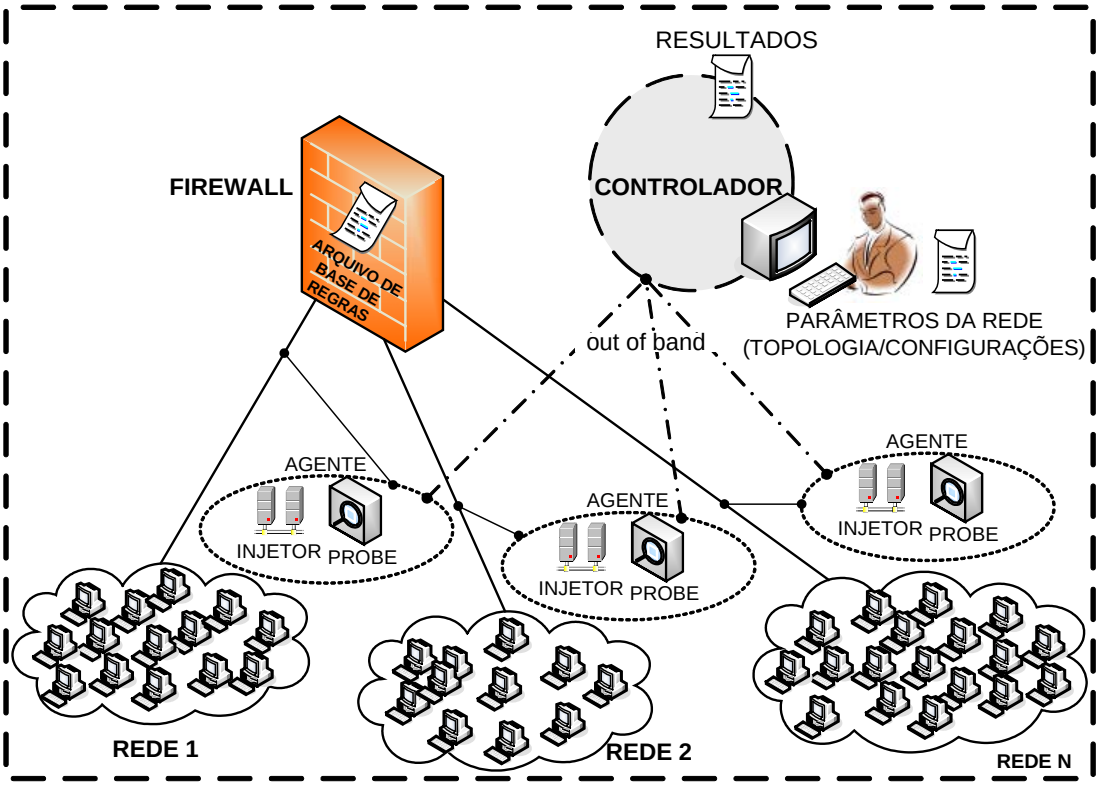

Figura 1.1 – Abrangência do trabalho proposto

Portanto a abrangência deste trabalho não está direcionada a:

- Identificação ou descoberta de problemas e erros relativos à sintaxe de regras;
- Análise da qualidade da política de segurança;

- Confrontação de consistência entre regras;
- Definição de regras ou conjunto de regras;
- Análise de *firewalls* de aplicação;
- Implementação do módulo Controlador;
- Comparação dos resultados com a política de segurança;

1.4 ORGANIZAÇÃO DO TRABALHO

A seguir são apresentados os assuntos tratados nos capítulos subseqüentes do presente trabalho:

Capítulo 2: Apresenta um estudo das principais vulnerabilidades da pilha TCP/IP, constatando as conseqüências das falhas encontradas nesta pilha de protocolos e de que forma estas vulnerabilidades podem ser exploradas interferindo no comportamento ou segurança de redes protegidas por *firewalls*. São apresentadas ainda algumas formas de como as vulnerabilidades encontradas na pilha de protocolos TCP/IP podem ser amenizadas.

Capítulo 3: Fornece uma visão geral e discute os modelos e as principais funcionalidades dos *firewalls*.

Capítulo 4: Apresenta os resultados dos estudos realizados até o momento, relacionados às ferramentas para testes de *firewalls*.

Capítulo 5: No capítulo é detalhada a arquitetura do sistema proposto e a metodologia de avaliação.

Capítulo 6: Apresenta a implementação do protótipo do sistema e descrição dos testes efetuados como prova de conceito.

Capítulo 7: São apresentadas as análises dos resultados obtidos e as conclusões do trabalho.

CAPÍTULO 2

VULNERABILIDADES DOS PROTOCOLOS DA PILHA TCP/IP

A pilha de protocolos TCP/IP (Postel, 1981a; Postel, 1981b), cujo projeto original possui mais de 25 anos de existência, é largamente utilizada para interligação de redes atualmente. O propósito do projeto original não previa uma variedade de aplicações e um crescimento na escala que podemos notar nos dias atuais. Desta forma, apresenta diversas vulnerabilidades[3] de segurança inerentes aos protocolos originais e também aos que foram sendo implementados e incorporados desde então, para atender as novas tecnologias e necessidades dos usuários das redes de computadores (BELLOVIN, 1989; BELLOVIN; 2004; MORIS, 1985; VIVO, 2004).

Este capítulo apresenta uma abordagem sucinta de algumas das vulnerabilidades de segurança encontradas no conjunto de protocolos da pilha TCP/IP e de que forma os impactos causados por essas vulnerabilidades podem ser explorados na tentativa de evasão ou para subverter *firewalls ou* computadores interligados em redes.

[3] Vulnerabilidade é definida aqui como sendo uma falha de um ou mais protocolos da pilha TCP/IP que permita que uma pessoa não autorizada execute uma operação que não deveria poder executar e obtenha privilégios que não deveria ter (HATCH, 2003).

As vulnerabilidades encontradas na pilha de protocolos TCP/IP podem ser caracterizadas basicamente em: intrínsecas (de projeto) ou de implementação. As vulnerabilidades de projeto são inerentes à arquitetura da concepção original do projeto da pilha TCP/IP, enquanto que as vulnerabilidades de implementação referem-se às falhas decorrentes da adequação dos protocolos para funcionarem em diferentes plataformas.

Aqui a ênfase está voltada para as vulnerabilidades do projeto e não especificamente em falhas de implementações, embora estas sejam ligeiramente comentadas em certos trechos do texto, para auxiliar na compreensão. Portanto, estamos interessados em abordar problemas relacionados à segurança que podem ser explorados devido aos erros de concepção do projeto original encontrados nos protocolos da pilha TCP/IP.

2.1 SEGURANÇA NA AUTENTICAÇÃO DO PARCEIRO

Uma das principais deficiências no aspecto de segurança do protocolo IP é sua fraqueza em identificar e autenticar[4] um computador na rede (BELLOVIN, 1989), ou seja, com base no endereço IP de origem de um datagrama IP recebido, nem sempre é possível determinar com certeza a identidade do computador que o tenha originado. Além disso, também há poucas garantias de que o conteúdo de um datagrama IP recebido não tenha sido modificado ou observado quando em tráfego pela rede, ou seja, que a integridade dos dados contidos no pacote tenha sido preservada.

Os ataques que exploram tal falha têm como tática mais comum a personificação de um computador na rede. A finalidade consiste desde obter informações sigilosas como senhas, abuso da confiança que as máquinas mantêm entre si, até alterações do conteúdo dos dados que estejam de passagem para outros computadores de destino.

[4] Confirmar que as entidades envolvidas numa comunicação são elas mesmas, validando a identidade de um utilizador, dispositivo ou processo.

2.2 IP

O protocolo IP (*Internet Protocol*) foi projetado para interconectar computadores em redes orientadas a pacotes. O protocolo IP provê a transmissão de blocos de dados (datagramas), de um computador de origem para um computador de destino (POSTEL, 1981b).

Um nível de consideração importante para o protocolo IP do ponto de vista para a filtragem de pacotes é a fragmentação e remontagem de datagramas, se necessário, para transmissão por uma rede de computadores. Ou seja, o protocolo IP tem capacidade de dividir um grande datagrama que, caso contrário, não conseguiria atravessar algum enlace de rede (devido a limitações de MTU do enlace), em datagramas menores, denominados fragmentos, e remontá-los posteriormente. A RFC 791 (POSTEL, 1981b) descreve um algoritmo de remontagem de fragmentos que assume a sobreposição de fragmentos, caso o valor do campo FO (*Fragment Offset*) seja inferior ao tamanho do fragmento anterior.

Normalmente, qualquer roteador pode fragmentar um datagrama, a não ser que um sinalizador no cabeçalho IP esteja negando esta permissão. Porém, se um roteador precisar fragmentar um datagrama e encontrar esta permissão negada, descartará o pacote, possivelmente causando uma falha na comunicação. Geralmente este fato é menos desejável do que ter o pacote fragmentado (ZIEMBA, 1995).

O problema com a fragmentação é que apenas o primeiro fragmento de cada pacote contém as informações de cabeçalho de protocolos de nível mais alto (como o TCP) de que o sistema de filtragem de pacotes necessita para decidir se deve ou não deixar passar o pacote completo.

Originalmente, a abordagem comum de filtragem de pacotes para trabalhar com a fragmentação era permitir a passagem de quaisquer fragmentos que não fossem os primeiros e fazer a filtragem de pacotes apenas no primeiro fragmento de um pacote. Isso era considerado seguro porque, se o sistema de filtragem de pacotes decidisse descartar o primeiro fragmento do pacote, o sistema de destino não seria capaz de reunir o restante dos fragmentos no pacote original, não importando quantos

fragmentos restantes ele recebesse. Se não fosse possível reconstruir o pacote original, o pacote parcialmente reunido não seria aceito.

Entretanto, existem problemas com pacotes fragmentados. Caso sejam repassados todos os fragmentos que não sejam os primeiros, o computador de destino guardará os fragmentos na memória por algum tempo, esperando pelo fragmento que falta. Isso possibilita aos atacantes o uso de pacotes fragmentados em ataques de Negação de Serviço (DoS - *Denial of Service*) ou ataques de negação de serviço distribuído (DDoS - *Distribute Denial of Service*) (FARROW, 2005).

Quando o computador de destino desistir de remontar o pacote, ele enviará uma mensagem ICMP "*Time Exceeded*" ao computador de origem.

Além disso, atacantes podem usar pacotes fragmentados de modo malicioso para ocultar dados. Cada fragmento contém informações sobre a remontagem, então, um atacante pode forjar pacotes fragmentados no quais os fragmentos realmente se sobrepõem.

Cada implementação pode responder de uma forma diferente a fragmentos sobrepostos, podendo resultar em alguns casos, na queda do sistema operacional. Sobreposições de fragmentos possibilitam certos tipos de ataques, tais como:

- Ataques de negação de serviço contra computadores ou *firewalls* baseados em filtros de pacotes que não possuem mecanismos de tratamento de pacotes sobrepostos;
- Ataques de ocultação de informações. Se um atacante obtiver determinados tipos de informações, como por exemplo, antivírus, sistemas para detecção de intrusão ou outros sistemas que controlam o conteúdo dos pacotes que estão sendo utilizados e puder determinar que método de montagem os sistemas usam para fragmentos superpostos, o atacante poderá construir fragmentos superpostos que irão ocultar o conteúdo destes fragmentos dos sistemas de inspeção;
- Um atacante pode construir um pacote com cabeçalhos aceitáveis no primeiro fragmento, porém sobrepor o próximo fragmento de modo que também tenha cabeçalhos. Como os filtros de pacotes não esperam

cabeçalhos TCP em fragmentos que não sejam os primeiros, eles não os filtrarão.

2.3 ICMP

O ICMP *(Internet Control Message Protocol*), especificado na RFC 792 (POSTEL, 1981c), é capaz de reportar problemas com uma rota, terminar uma conexão por detectar problemas na rede, entre outras funções de controle. Existe um conjunto de tipos de mensagens ICMP definidas. A tabela 2.1 mostra exemplos de mensagens ICMP. Muitos sistemas de filtragem de pacotes permitem filtrar pacotes ICMP com base no campo do tipo de mensagem ICMP.

Tabela 2.1 – Exemplos de tipos e códigos de mensagens ICMP

Tipo	Código	Mensagem	Significado
0	0	*Echo Reply* (resposta de eco)	Resposta de um computador a uma mensagem "*Echo Request*"
3	0 a 15	*Destination unreachable* (destino inalcançável)	Resposta de um roteador quando o destino de um pacote por alguma razão não pode ser alcançado (por exemplo, um link da rede está inativo).
4	0	*Source Quench* (enviar mais devagar)	Se um roteador estiver sobrecarregado com tráfego de um computador, ele pode enviar esta mensagem de controle de congestionamento, solicitando ao computador de origem que reduza ou pare a transmissão de datagramas IP.
5	0 a 3	*Redirect* (redirecionar)	Resposta que um roteador envia a um computador devido a um pacote que o computador deveria ter enviado a um roteador diferente. Mesmo assim, o roteador encaminha o pacote para o roteador onde ele deveria ter ido à primeira vez, e o redirecionamento informa ao computador sobre o caminho mais eficiente a usar da próxima vez.
8	0	*Echo Request* (solicitação de eco)	Utilizada para verificar conectividade entre dois computadores.
11	0 ou 1	*Time exceeded* (tempo excedido)	Se um roteador receber um pacote com TTL (*time to live*) 0, enviará uma mensagem de tempo excedido ao computador de origem.

O protocolo ICMP pode ser usado para efetuar vários tipos de ataques, como por exemplo:

- Através do utilitário PING (*Packet Internet Groper*), que é um utilitário utilizado na maioria das vezes para testar conectividade entre computadores, um atacante pode enviar uma enxurrada de solicitações de *Echo Request*, permitindo inundar a rede ou o computador atacado;
- Mensagens ICMP *Redirect* também podem ser usadas para interceptação de pacotes IP. Mensagens ICMP *Redirect* são comumente utilizadas por um roteador quando um computador envia erroneamente um datagrama IP para um roteador diferente daquele que deveria. Se um atacante falsificar uma mensagem ICMP *Redirect*, ele pode indicar para um computador (alvo do ataque) para enviar os datagramas IP para certos destinos. Este tipo de ataque só pode ser realizado estando o atacante e o atacado na mesma rede local;
- Através de mensagens ICMP "*Time exceeded*" ou "*Destination unreachable*". A mensagem *Time exceeded* indica que o campo *time-to-live* no cabeçalho IP foi expirado. Isto normalmente é causado por *loops* no roteamento ou tentativa de alcançar um computador extremamente distante. Uma Mensagem "*Destination unreachable*" pode ter vários significados (baseado no sub campo da mensagem ICMP), mas basicamente indicam que o pacote não pode ser enviado com sucesso ao computador desejado. Ambas as mensagens de erro do protocolo ICMP podem levar um computador a descartar imediatamente uma conexão. Um atacante pode fazer uso disto simplesmente forjando mensagens ICMP, e enviando-as para um ou ambos os computadores envolvidos na comunicação, conseqüentemente a comunicação será fechada (CHAMBERS, 2005).

As especificações atuais do protocolo ICMP não recomendam qualquer tipo de teste de validação em mensagem de erro ICMP recebidas, deixando assim a possibilidade para ocorrência de uma variedade de ataques. Ataques

contra o TCP por meio do protocolo ICMP, que incluem: *Blind connection-reset*, *Blind throughput-reduction* e *Blind performance-deggrading attacks* (GONT, 2005).

2.4 VULNERABILIDADES DOS PROTOCOLOS DE ROTEAMENTO

Os protocolos de roteamento são utilizados por roteadores para descobrir dinamicamente as melhores rotas para cada destino e também para detectar problemas relacionados a alguma de suas rotas, caso exista problema com alguma rota (PLUMMER, 1982).

Aproveitando-se de vulnerabilidades existentes nos protocolos de roteamento, um atacante pode tentar sobrepujar os roteadores existentes entre ele e o computador alvo do ataque. A dificuldade em se personificar um computador está em fazer com que todos os pacotes enviados a um computador legítimo cheguem ao falso computador.

Determinados protocolos de propagação de rotas, a exemplo do RIP - *Routing Information Protocol* (HEDRICK, 1988), não possuem mecanismos para verificação se as informações recebidas de roteadores vizinhos são verdadeiras (CHESWICK, 2005). Deste modo, um atacante pode inserir informações de roteamento nas tabelas de rotas, modificando-as de acordo com seus objetivos e intenções. Vale ressaltar que ataques a roteadores usando o protocolo de roteamento podem propagar as informações forjadas pelos vários roteadores da rede.

Além de usar as deficiências dos protocolos de roteamento e de ICMP *Redirect*, descrito na próxima seção, um atacante pode facilmente manipular rotas usando a opção *source routing* do protocolo IP (COMER, 2005; STEVENS, 1994). Pacotes IP com esta opção por padrão são roteados não com base nas tabelas de rota dos roteadores e sim conforme o caminho especificado pelo computador que originou tais pacotes. Este tipo de ataque a rotas pode ser evitado utilizando uma funcionalidade existente na maioria dos roteadores que permite rejeitar pacotes IP contendo esta opção.

2.4.1 *SOURCE ROUTING*

Um ataque DoS baseado em roteamento envolve atacantes manipulando entradas de tabela de roteamento para negar serviço a sistemas de redes legítimos. Segundo Kamara (2003), a maioria dos protocolos de roteamento, tais como o RIP (*Routing Information Protocol*) versão 1 e o BGP (*Border Gateway Protocol*) versão 4 (REKHTER, 1995), possuem autenticação muito fracas ou inexistente (o mecanismo de autenticação que eles fornecem, são considerados fracos e raramente são usados, mesmo quando implementados). Isto cria um cenário perfeito para que atacantes alterem rotas legítimas (freqüentemente falsificando seu IP de origem) para criar uma condição de DoS. As vítimas desses ataques normalmente terão seu tráfego roteado para a rede do atacante ou para uma rede que não existe.

2.5 UDP

O UDP (*User Datagram Protocol*) (POSTEL, 1980) é um protocolo não orientado a conexão, não confiável (pois não há retransmissões, tratamento de datagramas duplicados nem reordenação dos datagramas) e que oferece para a aplicação um serviço semelhante ao oferecido pelo IP. Além disso, o UDP não oferece nenhum mecanismo para controle de fluxo, o que pode gerar congestionamento na rede ou inundação em um computador com menor capacidade de processamento e, conseqüentemente, um aumento da quantidade de pacotes perdidos na rede.

Por não existir procedimento para o estabelecimento de conexão nem números de seqüência, como no protocolo TCP, aplicações baseadas em UDP são mais suscetíveis à falsificação de endereços IP (CHESWICK, 2005; POUW, 1996). Conseqüentemente, pacotes UDP são largamente utilizados para ataques de DoS ou ataques de DDoS, através de inundação de pacotes UDP (UDP *Flooding*). Ressaltando que ataques de DoS ou DDoS que realizam inundação não dependem necessariamente de falsificação de

endereços IP (IP *Spoofing*), porém o uso de endereços forjados permite que o atacante esconda a origem do ataque e, até mesmo, dificulte a detecção dos ataques em redes que utilizam sistemas de detecção de intrusão IDS (*Intrusion Detection System*).

2.5.1 UDP *FLOOD*

O ataque de inundação UDP ou UDP *Flood* consiste em um computador de origem enviar rapidamente uma grande quantidade de datagramas UDP para um computador alvo a ser atacado (SILANDER, 1999). Quando o computador alvo do ataque recebe os pacotes UDP enviados pelo atacante, ele tenta determinar qual aplicação está aguardando pelo pacote naquela determinada porta em que foi recebido o pacote, então emite uma mensagem ICMP para o destinatário (que pode ser um endereço IP forjado) informando que o pacote enviado não encontrou seu destino. Dessa forma, o computador alvo do ataque começa a descartar pacotes pela impossibilidade de tratar todas as requisições, podendo ficar comprometido.

2.5.1.1 DEFESA PARA UDP *FLOOD*

Firewalls e outros dispositivos de filtragem podem descartar ou negar datagramas UDP em portas não solicitadas (SCHUBA, 1993).

2.6 TCP

O protocolo TCP (*Transport Control Protocol*) (POSTEL, 1981a), ao contrário do UDP, é orientado a conexão e mais confiável, pois trata de perda de pacotes, retransmissão e reordenação dos pacotes. Além disso, o protocolo TCP apresenta maior estabilidade em redes congestionadas e na comunicação entre máquinas com grande diferença de processamento (apresenta mecanismos próprios para controle de fluxo). Estas características

são possíveis porque é mantido o controle de todas as conexões em andamento. O controle das conexões é realizado, principalmente, através de números de seqüência e números de reconhecimento (ACK - *Acknowledgment Numbers*) que são introduzidos no cabeçalho TCP (COMER, 2005; POSTEL, 1981a, STEVENS, 1994). Dessa forma, o protocolo TCP é menos susceptível a ataques de falsificação quando comparado com os protocolos UDP e ICMP. Entretanto, será visto adiante que em algumas implementações antigas, estes números de seqüência são passíveis de previsão e, com isso, esta “proteção extra” torna-se ineficaz (MORIS, 1985).

Embora os números de seqüência e o controle das conexões ofereçam maior proteção, o primeiro segmento TCP enviado no processo de estabelecimento da conexão (*three way handshaking*), ilustrado na figura 2.1, pode ter seu endereço de origem falsificado (IP *Spoofing*). Esta propriedade é explorada, por exemplo, no ataque de TCP SYN *Flood* (NEONSURGE, 1996).

No processo de estabelecimento de uma conexão o cliente, envia um segmento TCP SYN para o servidor. O servidor recebendo um segmento com o *flag* SYN ativo, deve responder com um segmento SYN/ACK ao cliente que iniciou a conexão. A conexão é finalmente estabelecida quando o cliente que enviou o segmento SYN e recebeu como resposta um SYN/ACK, enviar um ACK final, reconhecendo o segmento SYN/ACK recebido, conforme mostrado na figura 2.1. Observa-se também, nesse processo de estabelecimento de conexão, a importância dos números de seqüências iniciais utilizados por cada máquina, que não são detalhadamente abordados neste trabalho.

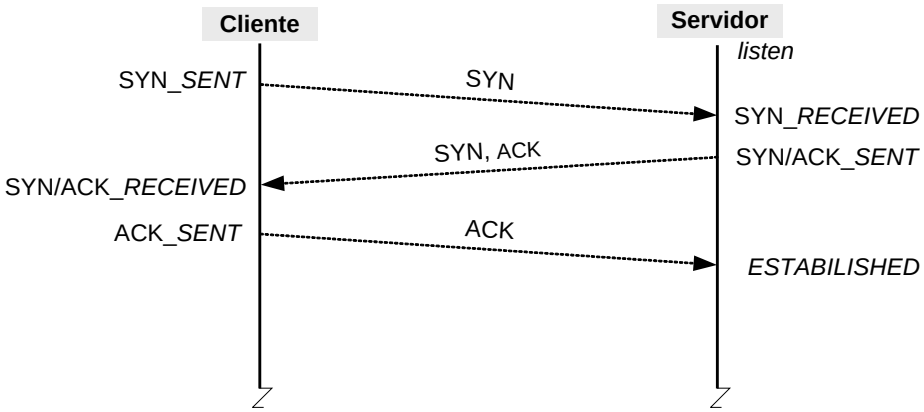

Figura 2.1 – Estabelecimento de conexão TCP

2.6.1 SYN *FLOOD*

No ataque de SYN *Flood* (CERT, 2000) o objetivo do atacante é tornar o computador ou *firewall* indisponível. Para isso, envia uma quantidade de seqüências de segmentos TCP SYN para o computador ou *firewall* alvo do ataque simulando a solicitação de abertura de conexão TCP, maior do que estes têm a capacidade para responder.

Assim que estes segmentos são recebidos pelo computador alvo, o mesmo responde enviando as respectivas respostas com o *flag* SYN/ACK ativo e fica aguardando as confirmações ACK que nunca serão enviados. O fluxo de segmentos TCP trocados neste ataque é apresentado na figura 2.2.

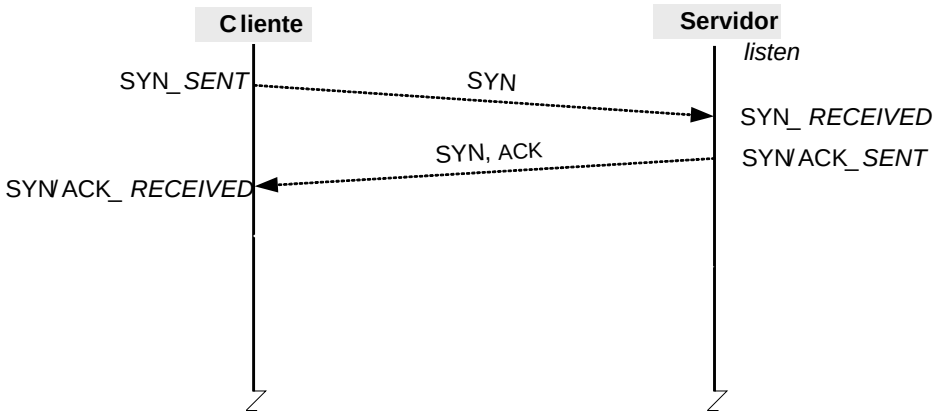

Figura 2.2 – Conexão TCP/IP semi-aberta

Neste caso, as conexões ficam em estado de conexões semi-abertas pendentes ocupando espaço de memória na estrutura do núcleo do sistema operacional da máquina alvo. Isto pode se intensificar até que a estrutura alocada seja exaurida, impedindo que outras conexões legítimas possam ser estabelecidas.

Eventualmente, estas entradas de conexões semi-abertas são removidas por expiração de tempo. Porém o atacante pode permanecer gerando rapidamente novos segmentos TCP SYN indefinidamente.

Geralmente os atacantes utilizam o endereço IP falsificado ou de um computador que não possa ser alcançado, por exemplo, um endereço IP inexistente ou um computador desligado. Dessa forma, jamais receberá de

volta um segmento ACK do computador de origem, permanecendo esta conexão no estado SYN_RECV.

2.6.1.1 DEFESAS PARA ATAQUES DE SYN *FLOOD*

A defesa contra ataques de SYN *Flood* é dificultada principalmente porque segmentos TCP SYN fazem parte do tráfego normal TCP, endereços IP de origem podem ser falsificados e neste ataque segmentos TCP SYN são pequenos.

Como tentativas para conter ataques SYN *Flood*, pelo menos durante um intervalo de tempo, são utilizados alguns métodos de defesas. Estes métodos de defesas podem ser agrupados em duas categorias: aqueles para defesa do próprio equipamento e aqueles para serem utilizadas em um equipamento de *firewall*. Os principais métodos de proteção são (EDDY, 2006; NEONSURGE, 1996):

- Proteção do próprio equipamento
  - SYN *Cookies*.
- Proteção em *firewalls*
  - SYN *Threshold;*
  - SYN *Proxy;*
  - SYN *Protection.*

2.6.1.1.1 SYN *COOKIES*

Este método procura eliminar a necessidade de armazenar estado , ou seja, informações de conexões incompletas ou semi-abertas pelo servidor (BERNSTEIN, 2004).

Quando um servidor recebe um segmento TCP com o *flag* SYN ativo solicitando estabelecimento de conexão TCP, é gerado um *cookie* que será transmitido no segmento TCP SYN/ACK, mais especificamente codificado no campo reservado ao seu ISN (número seqüencial inicial). Na geração deste

cookie é utilizado um algoritmo de *hash* criptográfico, tendo como entrada as informações do cabeçalho IP e TCP do pacote recebido com o pedido de conexão (endereço IP e porta de origem, número de seqüência usado pela máquina de origem, endereço e porta de destino) e um “segredo”, conhecido somente pelo este computador que recebeu a solicitação de conexão.

Observe que estas mesmas informações, obtidas do cabeçalho deste pacote SYN, seriam mantidas na estrutura de conexões semi-abertas no núcleo no modo de operação normal. Porém, com este novo procedimento nada é mantido pelo sistema, estas informações simplesmente geram um código *hash* com no máximo 32 *bits*[5], que é transmitido pela rede codificado junto ao ISN usado pelo servidor na nova conexão. O “segredo” é usado para que o servidor possa validar um *cookie* que ele tenha criado anteriormente.

Com o uso do *cookie*, nenhuma ou pouca informação precisa ser mantida no núcleo do servidor sobre estas conexões semi-abertas. Dessa maneira, uma tentativa de ataque de SYN *Flood* não causará nenhum “prejuízo”. O número de seqüência, será incrementado pelo cliente e retornará ao servidor no campo reservado ao número de reconhecimento do segmento de TCP ACK, concluindo o processo de *three way handshake*.

Portanto, para o cliente tudo ocorrerá de maneira transparente, sem a necessidade de mudanças no protocolo TCP original. Este segmento TCP ACK do cliente, ao chegar no servidor, é verificado facilmente, usando a mesma função *hash*, as informações dos cabeçalhos no pacote e o segredo do servidor. Caso o código *hash* obtido neste processo seja idêntico ao *cookie* retornado pelo cliente, pode-se concluir que se trata de um segmento TCP ACK válido estabelecendo uma conexão. As informações relativas a esta nova conexão são extraídas deste pacote, mantidas na tabela de conexões do núcleo do sistema operacional e a conexão prossegue normalmente. Caso contrário, o pacote será rejeitado e um pacote RST transmitido, como definido pelo protocolo TCP (POSTEL, 1981a; STEVENS, 1994).

---

[5] Tamanho do campo reservado para números de seqüência.

2.6.1.1.2 SYN *THRESHOLD*

O SYN *Threshold* é um método para ser utilizado por firewalls, estabelecendo um limite ou quota no número de conexões incompletas (semi-abertas) e descarta pacotes SYN se o número de conexões incompletas alcançarem o limite. Este é um dos tipos de técnicas de defesa contra ataques de SYN *Flood* sendo implementado em *firewalls* e roteadores de diversos fabricantes.

Uma conseqüência do SYN *Threshold* é a possibilidade de que ocorra o descarte de uma tentativa de conexão válida, ou seja, uma conexão sem fins maliciosos.

2.6.1.1.3 SYN *PROXY*

O SYN *proxy* ilustrado na figura 2.3, é um método de proteção para ser utilizado por firewalls. No SYN *proxy*, quando o *firewall* que implementa esta tecnologia recebe um segmento com o flag SYN ativo do cliente, envia o segmento SYN/ACK de volta para o cliente, e espera o segmento TCP com o ACK final, estabelecendo a conexão. Após algum tempo, se o ACK final não for recebido, o *firewall* descarta a tentativa de estabelecimento de conexão.

Depois que o *firewall* recebe o pacote ACK do computador cliente, o *firewall* então repassa a seqüência do passo 3 do *three way handshake* para o computador cliente, e os dados podem ser enviados.

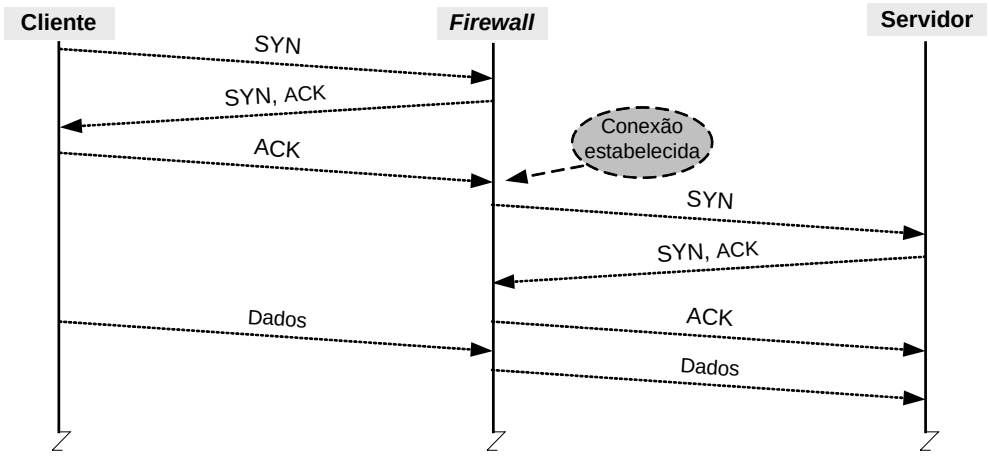

Figura 2.3 – *Firewall* que implementa SYN *Proxy*

2.6.1.1.4 SYN *PROTECTION*

O SYN *Protection* é um método de proteção para ser utilizado por firewalls. Este método utiliza o método SYN *Proxy* em conjunção com o método SYN *Cookies* para oferecer proteção de ataques de SYN *Flood*.

O mecanismo é mostrado na figura 2.4. O cenário é um computador cliente tentando estabelecer uma conexão TCP com um computador servidor, como por exemplo, um servidor de *HTTP* protegido por um *firewall* que implementa a SYN *Protection*.

O cliente tenta estabelecer uma conexão TCP com o servidor enviando um segmento SYN, o *firewall* intermédia a conexão e retorna o segmento SYN/ACK com um *cookie* para o cliente. O cliente envia um segmento ACK final com o *cookie* solicitado e a conexão é estabelecida entre o cliente e o *firewall*. Somente após o recebimento deste segmento ACK o *firewall* repassa a conexão para o servidor. Então os dados podem ser enviados.

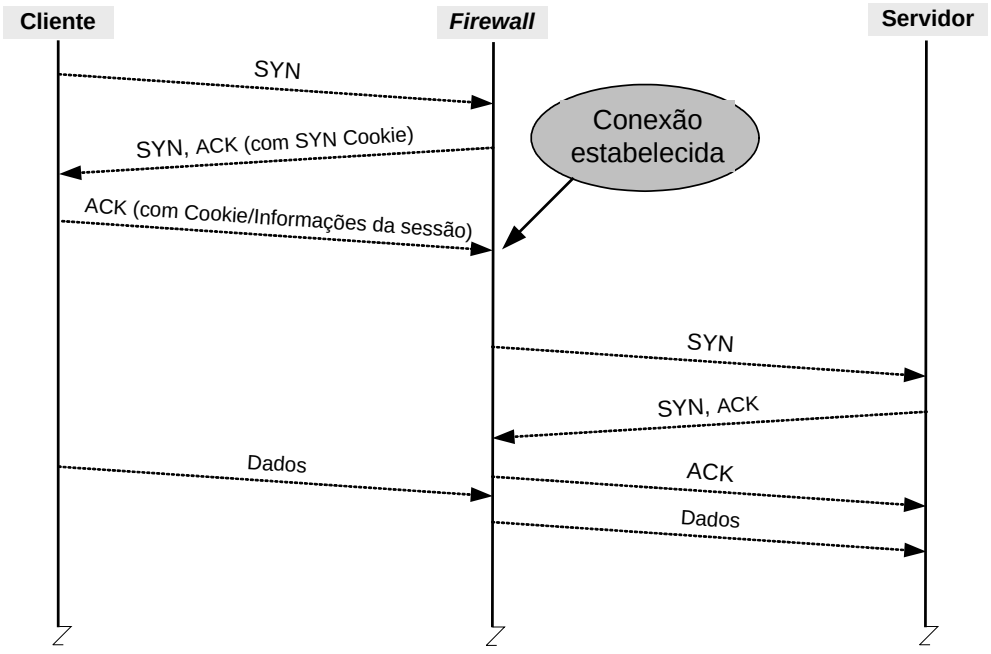

Figura 2.4 – *Firewall* que implementa SYN *Protection*

Dentre as alternativas apresentadas, o uso de SYN *cookie* é utilizado como principal opção na tentativa de se conter ataques de SYN *Flood*.

2.6.2 ATAQUE *LAND*

Outro ataque de DoS, usando os pacotes TCP de solicitação de conexão, que merece destaque é o *Land* (CERT, 2004). Este ataque explora a facilidade de *spoofing*, um problema na implementação da pilha de protocolos TCP/IP presente em alguns sistemas operacionais. O ataque consiste no envio de um pacote modificado com SYN ativo, especialmente montado (solicitação de conexão) para um computador alvo usando o endereço IP desta máquina como endereço de origem e destino e um mesmo número de porta como portas de origem e destino. No instante que o computador tenta responder à solicitação de conexão, ele gera um *loop*, podendo ocasionar a paralisação do sistema operacional (STREBE, 2002).

2.7 CONSIDERAÇÕES FINAIS

Como vimos, há diversos tipos de ataques que exploram as falhas das vulnerabilidades existentes nos protocolos da pilha TCP/IP. A exploração de tais vulnerabilidades pode ser utilizada com as mais diferentes finalidades, algumas inclusive, podem deixar computadores, *firewalls* ou outros equipamentos de rede completamente inutilizáveis, através do consumo de largura de banda.

Mesmo com as falhas e vulnerabilidades existentes nos protocolos da pilha TCP/IP, é possível proteger uma rede interna, de maneira a minimizar consideravelmente as conseqüências de exploração das vulnerabilidades apresentadas neste capítulo. Tal objetivo pode ser alcançado pela correta instalação e configuração de *firewalls*. Técnicas utilizadas em *firewall* serão apresentadas no Capítulo 3.

CAPÍTULO 3

TECNOLOGIAS DE *FIREWALLS*

Para proteger uma rede contra ameaças provenientes de outra rede, explorando, por exemplo, algumas das vulnerabilidades descritas no capítulo anterior podem ser utilizadas técnicas de proteção de perímetro.

Em tecnologia da informação, *firewall* é um termo utilizado para identificar um conjunto de sistemas e equipamentos que implementam mecanismos de proteção de perímetro entre redes.

Normalmente é inserido em pontos de concentração, nas fronteiras entre as redes, de forma a possibilitar o gerenciamento (monitoração e controle) de todo e qualquer tráfego, aplicando as restrições definidas em sua configuração (GODDARD, 2001; GOUDA 2004).

Para isto, o *firewall* deve interceptar o tráfego entre as redes e, baseado na política de restrição de comunicação, permitir ou bloquear a passagem dos pacotes (CRONJE, 2003; ZWICKY, 2000). Dessa maneira, o *firewall* pode auxiliar na redução de riscos de ataques a computadores localizados em uma rede protegida pelo *firewall*, controlando quais computadores de uma rede podem estabelecer sessões com quais computadores de outra rede.

Neste trabalho a ênfase é voltada para *firewalls* de camada de rede para a pilha TCP/IP.

3.1 FUNCIONALIDADES DOS *FIREWALLS*

Os *firewalls* podem operar desempenhando diversas funcionalidades, principalmente (LIU, 2004):

- Filtragem de pacotes;
- Conversão de endereços de rede (NAT);
- *Proxy*;

Adicionalmente, os *firewalls* também podem incorporar outras funcionalidades, tais como:

- Autenticação criptografada;
- Túneis criptografados (VPN);
- Módulo de proteção contra ataques de inundação, como por exemplo, proteção contra SYN *Flood* ou UDP *Flood*;
- Módulo de controle de fragmentação;
- Módulo de proteção contra ICMP *Redirect;*
- Limitação de banda;
- Etc.

3.1.1 FILTROS DE PACOTES

Os filtros de pacotes funcionam controlando os dados que fluem entre suas interfaces de rede. Atuam principalmente nas camadas de rede e de transporte da pilha TCP/IP, comparando, os endereços IP de origem e de destino, o protocolo e, no caso do TCP e do UDP, os números de porta de origem e de destino, com um conjunto de regras contidas em uma base de regras e, então determinam uma ação a ser tomada com os pacotes: aceitar, rejeitar ou descartar.

Essas regras são, geralmente, baseadas em informações existentes nos cabeçalhos de cada pacote. Por exemplo, no caso de datagramas IP, a filtragem pode ser efetuada a partir de informações de endereços IP de origem e destino, portas de origem e destino, protocolo de transporte utilizado,

tipo e código de mensagens ICMP, entre outras informações.

3.1.1.1 FILTROS DE PACOTES COM ESTADOS

Sistemas de filtragem de pacotes um pouco mais avançados oferecem o controle de estado. A filtragem de pacotes com informação de estado, também é chamada de filtragem dinâmica, pois o comportamento do *firewall* muda de acordo com o tráfego que ele detecta.

Os filtros de pacotes com estados utilizam no seu processo de filtragem, um conjunto de regras de filtragem igual aos filtros de pacotes tradicionais e as informações de estados obtidas das conexões e sessões (CHECK POINT, 1998). Se o primeiro pacote de uma sessão ou conexão é permitido, o filtro de pacotes com estados, cria uma entrada para esta conexão ou sessão em sua tabela de estados e a aceitação dos demais pacotes referentes a esta sessão ou conexão está condicionada a existência desta entrada na tabela de estados (LIMA, 2000).

3.1.1.2 PRINCIPAIS VANTAGENS DA FILTRAGEM DE PACOTES

Um único *firewall* que efetua filtragem de pacotes posicionado estrategicamente pode proteger a rede inteira. A filtragem de pacotes quando comparada com outras técnicas, como por exemplo, *proxy* de aplicação, que será vista adiante, apresenta uma baixa sobrecarga.

3.1.1.3 PRINCIPAIS DESVANTAGENS DA FILTRAGEM DE PACOTES

As regras de filtragem de pacotes tendem a ser difíceis de configurar. Uma vez configuradas, as regras de filtragem de pacotes tendem a ser difíceis de testar. Dependendo do método utilizado, pode ser difícil ou impossível a implementação de certos tipos de filtros altamente desejáveis. Por exemplo, os pacotes informam o endereço IP de origem dos pacotes, mas não

informam de qual usuário, impossibilitando então, impor restrições sobre usuários específicos.

3.1.2 TRADUÇÃO DE ENDEREÇOS DE REDE

A conversão ou tradução de endereços de rede, também denominada de NAT, permite que endereços IP e número de pacotes sejam trocados no momento em que um pacote flui pelo *firewall*. Os sistemas de conversão de endereços de rede podem usar diferentes esquemas para realizar a conversão entre endereços de uma rede para outra:

- Alocar estaticamente um único endereço IP do *firewall* para cada endereço IP interno e sempre aplicar a mesma conversão. Isso não propicia nenhuma economia de endereços IP e torna as conexões mais lentas;
- Alocar dinamicamente um endereço IP do *firewall* toda vez que um computador interno inicia uma sessão de comunicação, sem modificar os números de portas. Isso limita o número de computadores internos que pode acessar simultaneamente a Internet ao número de endereços IP disponíveis no *firewall*;
- Criar um mapeamento fixo de endereços internos para endereços visíveis externamente, mas empregar mapeamento de portas de tal forma que várias máquinas internas utilizem os mesmos endereços externos;
- Alocar dinamicamente um endereço de um computador externo e um par de portas toda vez que um computador interno inicia uma conexão. Isso propicia o uso mais eficiente possível dos endereços de hosts externos;
- *Masquerade*, que permite que uma rede utilize internamente um conjunto de endereços de rede e converta estes números em outros diferentes ao lidar com redes externas. Dessa maneira, a NAT oculta os endereços IP internos, convertendo todos os endereços dos computadores internos para o endereço, por exemplo, do *firewall*. Este

então retransmite os dados dos computadores internos a partir de seu próprio endereço. Para a rede externa, todo o tráfego da rede parece vir de um único computador. Além disso, ajuda a ocultar a topologia da rede interna e a forçar as conexões a passarem por um único ponto controlado da rede.

3.1.3 *PROXIES* DE APLICAÇÃO

Proxies representam entidades que atuam na camada de aplicação da pilha TCP/IP intermediando a comunicação entre os parceiros de forma transparente ou explícita. Existem entidades de *proxy* genéricas ou específicas para cada serviço ou protocolo de aplicação. Além disso, *proxies* podem agregar outras características importantes de proteção, tais como, filtragem de conteúdo, autenticação de usuários, *proxy* reverso e etc.

No entanto, as funcionalidades extras têm um preço: os *proxies* de aplicação exigem mais memória e ciclos de CPU em comparação com a filtragem de pacotes.

3.1.3.1 PRINCIPAIS DESVANTAGENS DOS *PROXIES* DE APLICAÇÃO

Muito embora exista software de *proxy* disponível para os serviços mais antigos e mais simples como o FTP (File Transfer Protocol), pode ser mais difícil encontrar software testado para serviços mais novos ou menos utilizados. Até surgir *software proxy* adequado, um sistema que necessite de novos serviços talvez tenha de ser colocado fora do *firewall*, abrindo possíveis brechas de segurança.

3.1.4 REDES PRIVADAS VIRTUAIS

Redes Privadas Virtuais, também denominadas de VPN, permitem a conexão com segurança de duas redes separadas fisicamente sem que

espiões possam observar o conteúdo dos pacotes que trafegam entre estas redes.

Uma VPN pode estar sujeita a tentativas de redirecionamento ou outros tipos de interceptações enquanto o túnel estiver sendo estabelecido, mas, quando implementadas como parte integrante de um *firewall*, os serviços de autenticação e segurança do *firewall* podem ser utilizados para evitar a exploração enquanto o túnel estiver sendo estabelecido.

A segurança deste túnel está diretamente relacionada com o nível de segurança da criptografia utilizada.

Fundamentalmente, as VPNs empregam o mesmo princípio: o tráfego é criptografado, tem sua integridade protegida e é encapsulado em novos pacotes. Para a utilização de VPNs, é preciso ter o cuidado de como esta interage com o *firewall*. Em alguns casos, o *firewall* não pode controlar o tráfego que chega sobre a VPN, o que faz dela um modo para evitar os controles do *firewall* e abrir novas brechas de segurança.

Pelo fato de VPNs dependerem de criptografia. A criptografia pode ser efetuada quando o tráfego é gerado, ou como um túnel, onde o tráfego é criptografado e descriptografado em algum lugar entre a origem e o destino. A questão de onde é executada a criptografia e descriptografia em relação a filtragem de pacotes é importante. Se a criptografia e descriptografia for realizada dentro do perímetro de filtragem de pacotes (rede interna), então os filtros somente terão de permitir que os pacotes entrem e saiam. Isso pode ser relativamente simples se houver tunelamento, porque todos os pacotes serão endereçados para o mesmo endereço remoto e número de porta no outro extremo do túnel. Por outro lado, fazer a criptografia e descriptografia dentro de seu perímetro de filtragem significa que a chegada de pacotes criptografados não está sujeita ao escrutínio dos filtros de pacotes.

Se a criptografia e descriptografia forem efetuadas fora do perímetro da filtragem de pacotes (na rede de perímetro ou em um roteador externo), então os pacotes que entram do outro site poderão estar sujeitos ao escrutínio total de qualquer pessoa que possa ler o tráfego em sua rede de perímetro, inclusive atacantes.

3.1.4.1 PRINCIPAIS VANTAGENS DA VPN

As principais vantagens da utilização de redes privadas virtuais são os fatores econômicos e a segurança. É mais econômico utilizar redes públicas compartilhadas. Uma VPN oculta todo o tráfego que passa por ela, dessa forma ela garante não apenas que todas as informações serão criptografadas, mas também impede que as pessoas saibam quais máquinas internas estão sendo usadas e com quais protocolos.

3.1.4.2 PRINCIPAIS DESVANTAGENS DA VPN

Embora as redes privadas virtuais representem um importante mecanismo de segurança, elas apresentam problemas em um ambiente de *firewall*. Os computadores de uma VPN são conectados através de uma rede real, dessa forma estão sujeitos a ataques. Um exemplo disso é a utilização de uma VPN para oferecer conectividade a uma rede interna para usuários móveis que se conectam a Internet. As máquinas desses usuários podem ser atacadas a partir da Internet.

3.1.5 AUTENTICAÇÃO

A autenticação permite que usuários estejam em uma rede externa e possam acessar a rede interna, provando ao *firewall* que são usuários autorizados e, assim, possam abrir uma conexão por meio do *firewall* com a rede interna.

3.1.6 OUTRAS FUNCIONALIDADES

3.1.6.1 MÓDULO DE PROTEÇÃO CONTRA ATAQUES DE INUNDAÇÃO

Conforme já explicado no capítulo anterior, esta funcionalidade permite a proteção de computadores contra ataques de inundação de pacotes utilizando a técnica de SYN *Flood* e limitação de tráfego.

3.1.6.2 MÓDULO DE CONTROLE DE FRAGMENTAÇÃO

Funcionalidade que permite a proteção do *firewall* e do ambiente de rede contra ataques de fragmentação maliciosa de pacotes.

3.1.6.3 MÓDULO DE PROTEÇÃO CONTRA ICMP *REDIRECT*

Funcionalidade que permite controlar o aceite e envio de mensagens do tipo ICMP *Redirect*.

3.2 TESTES DE *FIREWALLS*

Testar um *firewall* é fundamentalmente diferente de testar qualquer outro sistema de hardware ou software. Por exemplo, em testes de software em geral, duas técnicas comuns são: o Teste da Caixa Preta e o Teste da Caixa Branca. Testes de Caixa Preta não presumem conhecimento algum sobre os detalhes internos do sistema e testam seu comportamento com relação à especificação e muitas entradas diferentes. O segundo utiliza conhecimento do código para testar como o estado interno responde a várias entradas. Estas técnicas, com algumas variações também podem ser utilizados ao testar *firewalls* (CHESWICK, 2005; JÜRJENS, 2001; KRISHNAN, 2003).

Neste trabalho, a abordagem dos testes deve conduzir para a descoberta do comportamento efetivo do *firewall*. Portanto a ênfase está direcionada às técnicas relacionadas com de teste de caixa preta, no qual não se deseja necessariamente conhecer o conteúdo da base de regras do *firewall*, cujo comportamento se deseja estudar. Como a arquitetura proposta neste trabalho é baseada na técnica de injeção de pacotes, os tópicos relacionados com este assunto são abordados com maior ênfase no Capítulo 5.

3.2.1 TÉCNICAS DE ANÁLISES

Técnicas são utilizadas para analisar o que um *firewall* está permitindo ou restringindo, principalmente: inspeção visual, análise automática de regras e injeção de pacotes (AL-TAWIL, 1999; BARTAL, 1999; LEE, 2004; 2001).

3.2.1.1 ANÁLISE POR INSPEÇÃO VISUAL

Esta técnica consiste na verificação visual de cada regra existente na base de regras, com o objetivo de descobrir principalmente a existência de erros de sintaxe e conflitos entre regras.

3.2.1.2 ANÁLISE AUTOMÁTICA DE REGRAS

Neste caso, ferramentas automatizadas, geralmente de fabricantes específicos são utilizadas na verificação de sintaxe e funcionalidade, ou seja, se a regra está seguindo o padrão de sintaxe para o equipamento testado (ERONEN, 2001; LEE, 2003, URIBE, 2004).

3.2.1.3 ANÁLISE POR INJEÇÃO DE PACOTES

Nesta técnica, pacotes são injetados em cada interface (de entrada ou

de saída) e, posteriormente são realizadas verificações em determinados registros de quais pacotes foram permitidos, bloqueados ou descartados pelo *firewall* (VERMA, 2005).

Se a análise for realizada, por exemplo, de uma rede pública (Internet) para uma rede privada (intranet), pacotes são injetados na interface de entrada (externa) do *firewall* com blocos de endereços IP válidos na Internet e combinados com os protocolos nas portas possíveis (0 a 65535), aplicando as regras existentes na base de regras. O resultado obtido fornece as ações que o *firewall* está tomando na direção da interface externa para as interfaces internas. A figura 3.1 representa a análise descrita anteriormente.

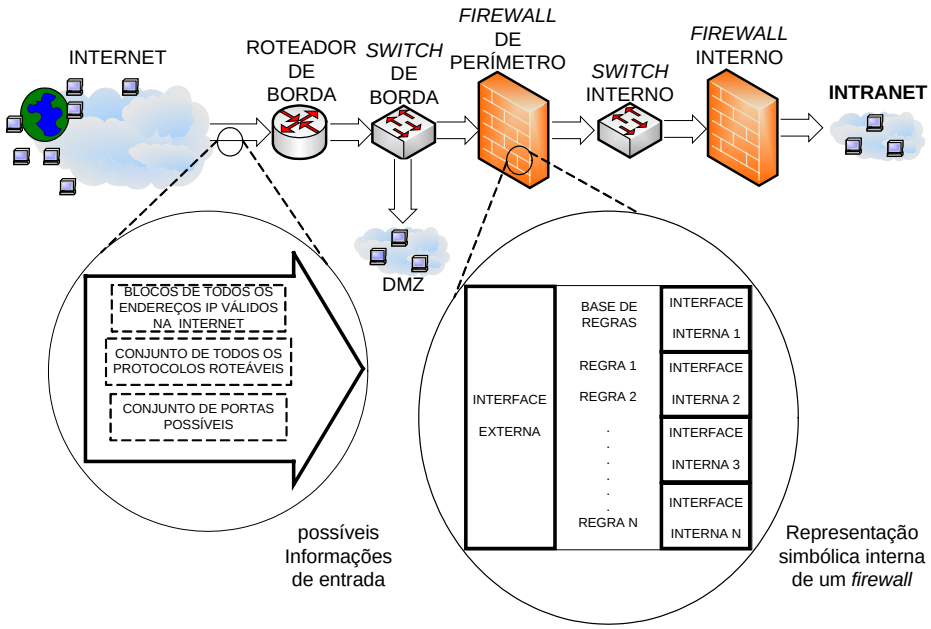

Figura 3.1 – Injeção de pacotes de informações aplicadas no *firewall*

3.3 PROBLEMAS NAS TÉCNICAS DE TESTES

Em cada uma das técnicas mencionadas anteriormente é possível identificar aspectos negativos, que podem dificultar a análise. A seguir serão destacados, os principais problemas encontrados em cada uma das técnicas de testes apresentadas.

3.3.1 PROBLEMAS NA ANÁLISE POR INSPEÇÃO VISUAL

Segundo Cheswick (2005), o número de regras é o melhor indicador da complexidade de um *firewall*. "*Se na base de regras existirem 10 regras, talvez possa analisá-las; se existirem 250, porque tantas? Talvez uma série de administradores tenha gerenciado o firewall e cada um teve receio de desfazer o que o outro fez*". Portanto, há um limite para a quantidade de testes que podem ser feitos manualmente. *Firewall* com grande quantidade de regras pode ser complexo demais para ser analisado utilizando esta técnica, consistindo em um exercício necessário, mas não suficiente.

Na técnica de análise por inspeção visual é possível a ocorrência de erros por falhas humanas, podendo permanecer problemas tais como: erros de sintaxe e parâmetros ou inconsistências entre regras ou bases-de-regras (VERWOERD, 2002).

3.3.2 PROBLEMAS NA ANÁLISE AUTOMÁTICA DE REGRAS

As ferramentas que apresentam esta característica cobrem apenas equipamentos de fabricantes específicos. Dessa forma seriam necessárias diversas ferramentas específicas ou módulos específicos para cada fabricante/tipo de *firewall*.

3.3.3 PROBLEMAS NA ANÁLISE POR INJEÇÃO DE PACOTES

Na técnica por injeção de pacotes, independente do fabricante ou tipo de *firewall*, nota-se a carência de ferramentas especificamente desenvolvidas para injeção de falhas em sistemas de *firewalls* e observação dos resultados. A maioria das ferramentas encontradas são inadequadas para a realização de testes ou análises de *firewalls*, tendo sido projetadas geralmente para outras finalidades.

Outro problema conseqüente desta técnica é a possibilidade de ocorrer uma explosão combinatória, que pode inviabilizar a análise devido o tempo

gasto ou análises de parâmetros desnecessários que não se aplicam à rede analisada. Para exemplificar a quantidade de testes e o problema da explosão combinatória, ou seja, testando todos os endereços IP (origem e destino), todas as possibilidades de portas (origem e destino) e ainda todos os protocolos conforme mostrado anteriormente na figura 3.1, é possível verificar que existem em torno de 2^{104} combinações possíveis:

- Endereço IP de origem: 2^{32}
- Endereço IP de destino: 2^{32}
- Porta de origem: 2^{16}
- Porta de destino: 2^{16}
- Protocolos: 2^{8}

O grau de complexidade desta técnica de análise pode se elevar dependendo da arquitetura da rede, quantidade de regras e bases-de-regras, ou ainda quando existem diversos *firewalls* de fabricantes distintos (YUE, 2003).

Dessa forma, pode-se notar a necessidade de ferramentas e metodologias desenvolvidas especificamente para este tipo de análise, ou seja, com capacidades e características para realizar análises do comportamento de *firewall* do ponto de vista ativo, informando que o mesmo está efetivamente permitido ou restringindo, se o mesmo possui mecanismos de proteções contra determinados tipos de ataques e ainda funcionalidades para auxiliar no problema da explosão combinatória.

CAPÍTULO 4

TRABALHOS RELACIONADOS

A aplicação de técnicas e ferramentas automatizadas utilizadas em testes de *firewalls* é um assunto cada vez mais explorado e que tem despertado o interesse dos pesquisadores da área de segurança de redes de computadores. Este fato pode ser justificado principalmente pelo grau de importância que o *firewall* representa para a proteção dos sistemas de redes de computadores.

Os resultados obtidos nas pesquisas confirmam as buscas constantes de aprimoramentos das técnicas e funcionalidades das ferramentas utilizadas, visando encontrar novas técnicas e soluções alternativas para testes de *firewalls*.

4.1 TÉCNICAS PARA TESTES DE *FIREWALLS*

Para posicionar o modelo proposto neste trabalho frente aos sistemas já desenvolvidos até o momento, este capítulo apresenta os resultados dos trabalhos de pesquisas realizados:

- Proposta de Gutman
- Sistema Firmato
- Sistema Fang

- Sistema Lumeta
- Sistema Face

4.1.1 PROPOSTA DE GUTMAN

Gutman (1997) apresenta um sistema, que utiliza uma linguagem tipo *Lisp*, para expressar a política de restrição de acesso à rede que devem ser implementadas nos *firewalls*. A utilização desta linguagem visa ser uma linguagem simples para os administradores de *firewalls*, principalmente para os que programam em linguagem *Lisp*.

São propostos dois algoritmos. No primeiro, dada a topologia da rede e a política de restrição de acesso à rede, gera automaticamente um conjunto de regras para os *firewalls.* O principal objetivo da geração automática do conjunto de regras é assegurar a correta implementação da política de restrição de acesso a rede. O segundo algoritmo, caso já exista uma base de regras implementada, permite a conversão desta base de regras para uma nova base de regras que utilize a sintaxe da linguagem. A partir disto, compara este novo conjunto de regras com a política de restrição de acesso à rede já descrita na linguagem, para determinar a existência de violações na política de segurança, ou para reportar que não existem violações. No trabalho, o autor salienta que é dado um importante passo em direção da geração automática de regras de *firewalls*.

Conforme os resultados obtidos, os algoritmos apresentados consistiram em uma eficiente alternativa para auxiliar administradores de rede em problemas relativos a erros de sintaxe gerados no momento da implementação das bases de regras, observando que as regras são geradas automaticamente a partir da política de restrição de acesso à rede.

No trabalho, o autor destaca a necessidade de conhecer os mecanismos de proteção existentes para o próprio *firewall*, porém, não propõe nenhuma sugestão ou alternativa.

4.1.2 SISTEMA FIRMATO

Bartal (1999) propõe o sistema FIRMATO (*Firewall Management Toolkit*) que consiste em um conjunto de ferramentas de análise passivo, que não injeta pacotes, para gerenciamento de *firewalls* distribuídos em redes onde existem mais de um *firewall*. O autor demonstra as propriedades deste conjunto de ferramentas considerando que o gerenciamento do *firewalls* pode ser realizado com um nível apropriado de abstração.

Esse conjunto de ferramentas foi implementado para trabalhar com produtos de *firewalls* disponíveis comercialmente, pois o autor defende a idéia de estabelecer uma linguagem de alto nível que seja válida para analisar *firewalls* de forma genérica, independente de marca ou modelo.

Segundo o autor, o sistema FIRMATO se diferencia das demais ferramentas por possuir principalmente as seguintes propriedades:

a) É uma ferramenta capaz de separar a política de segurança das especificidades de cada fabricante de *firewall*, permitindo assim ao administrador de segurança focar no projeto e política de segurança apropriada sem preocupar-se com a complexidade, ordenação e outras questões de configurações das regras de *firewalls*, consideradas de baixo nível. Isto também permite um gerenciamento unificado de *firewalls* de diferentes fabricantes e maior facilidade na transição, caso exista a substituição de um *firewall* por um modelo de outro fabricante.

b) É capaz de separar o projeto da política de segurança do projeto da topologia da rede em questão, isso permite ao administrador manter a consistência da política em face das mudanças de topologia da rede. Além disso, esta separação também permite ao administrador reusar uma política semelhante em diversas empresas com diferentes topologias de rede, ou para permitir pequenas companhias a usar projetos de políticas padrão.

c) Gera arquivos de configuração do *firewall* automaticamente a partir da

política de segurança, simultaneamente para múltiplos equipamentos de *firewalls*. Isto reduz a probabilidade de introduzir brechas na segurança causadas por erros em arquivos de configuração.

Para alcançar estes objetivos e como forma de resultados, foram implementados os seguintes componentes do conjunto de ferramentas do sistema Firmato:

- Um modelo entidade-relacionamento, que é uma estrutura para representação tanto da política de segurança quanto da topologia de rede.
- Uma Linguagem de Definição de Modelo (MDL), que foi utilizada para definir instâncias do modelo de entidade-relacionamento e o analisador (*parser*) associado.
- Um compilador de modelos, que traduz a MDL de uma instância do modelo para arquivos de configuração específicos para cada *firewall*. O conjunto de tais arquivos inclui informações sobre a topologia e base de regras.

Após a política de segurança ser descrita na linguagem MDL, o compilador de modelos gera os arquivos com as políticas de restrição de acesso para cada *firewall*.

A abordagem dada ao projeto foi um importante passo em direção à agilização no processo de monitoramento e gerenciamento de configuração de política de acesso em *firewalls*, especialmente em redes complexas com múltiplos *firewalls*. Outro importante avanço foi o esforço para a utilização de uma linguagem com um nível de abstração mais elevado, em contra partida a linguagem assembler. Uma consideração importante feita pelo autor, é que o modelo é baseado em *firewalls* que efetuam filtragem de pacotes, e não é aplicável a *firewalls* que efetuam filtragem de conteúdo, evidenciando assim a necessidade de estender esta característica para o FIRMATO.

4.1.3 SISTEMA FANG

Mayer (2000) ratificou o problema da dificuldade de configurar, gerenciar e testar *firewalls*, principalmente quando se possui mais de um e, são de fabricantes distintos. Para mitigar algumas dessas dificuldades, propõe o sistema FANG (*Firewall Analysis Engine*), um sistema para análise de *firewalls* que permite aos administradores testar uma política de segurança, seja esta uma política implementada (que já está em uso) ou planejada (que se pretende implementar).

O sistema utiliza também uma descrição mínima da topologia de rede e analisa diretamente os vários arquivos de configuração. O sistema interage com o usuário em um nível de abstração mais alto por meio de uma sessão de perguntas e respostas.

Dessa forma, o autor destaca a intenção de complementar as ferramentas de análise existentes, considerando que o sistema FANG pode ser utilizado antes mesmo que uma política de segurança seja de fato empregada, possibilitando ainda tratar todos os *firewalls* de uma única vez.

Além das características citadas anteriormente, o sistema FANG apresenta as seguintes funcionalidades:

- Propicia ao administrador a capacidade de interagir com a ferramenta em um nível elevado de abstração, isto é, em um nível no qual a política de segurança da empresa é definida ou expressa. Em uma rede ampla, a ferramenta pode permitir rapidamente focar nos aspectos mais importantes para testar;
- Não danificar: Análise de política deve ser possível sem ter que mudar ou ajustar a atual configuração da rede, que por sua vez pode tornar a rede vulnerável a ataques;
- Ser passivo: Análise de política não deve envolver o envio de pacotes e deve complementar os recursos das ferramentas de testes existentes;
- Ser atualizado: A análise deve refletir com precisão a política que é efetivamente aplicada no momento ou que está prestes de ser

aplicada;

- Ser eficiente: O tempo requerido para executar os testes comuns não deve depender do número de máquinas da rede. Deve depender somente do número de regras nas várias bases-de-regras.
- Ser fácil para usar: A interface de uso interativo deve permitir executar os testes com alguns cliques de mouse.

Para satisfazer esses objetivos, o sistema FANG coleta e lê os arquivos de configuração relevantes, e constrói uma representação da política de segurança e da topologia da rede, provê uma interface gráfica para o usuário colocar questões de quais serviços podem ser utilizados em um computador e quais computadores podem estabelecer sessões entre si.

O software foi desenvolvido como um módulo complementar ao conjunto de ferramentas de gerenciamento de *firewalls* FIRMATO (BARTAL, 1999), e utiliza técnicas de modelagem orientada a objetos. Isto significa que um componente pode ser usado independentemente dos outros componentes do conjunto de ferramentas.

O sistema FANG oferece vantagens e recursos não encontrados em nenhuma ferramenta até então, e complementa as ferramentas existentes.

Antes de o sistema FANG ser utilizado, é necessário ter um modelo de topologia de rede definido. Então a primeira coisa que o usuário do sistema FANG precisa fazer era escrever um arquivo que descreva a topologia da rede. A linguagem que o autor utiliza para descrever a topologia é um subconjunto da linguagem MDL do sistema FIRMATO.

O autor preocupa-se somente com os dispositivos de filtragem de pacotes que possuam bases-de-regras instaladas, e a respeito das zonas que estes dispositivos isolam. O autor destaca a necessidade da topologia da rede deve estar completamente estável. O arquivo de topologia somente precisa ser alterado se *firewalls* forem adicionados ou substituídos na rede.

Uma vez que o arquivo de topologia estiver escrito, a GUI (*Graphical User Interface*) do sistema FANG pode ser iniciada.

Como parte do arquivo de topologia, o usuário especifica o nome do

arquivo de configuração do *firewall* que contém a base de regras para todas as interfaces do *firewall*. Após ler o arquivo de topologia, a ferramenta FANG analisa cada um desses arquivos de configuração, usando um módulo "*front-end*" específico para cada marca de fabricante de *firewall*, e analisa a estrutura de dados de base de regras interna para cada dispositivo.

Depois de todos os arquivos serem analisados gramaticalmente, o sistema FANG cria um menu "*drop-down*", e está pronto para receber perguntas de usuários.

Em resumo o sistema FANG lê os arquivos de configurações de todos os fabricantes específicos e constrói uma representação da política contida. O sistema FANG simula então a política do *firewall* comparando com a questão, e apresenta o resultado na tela do usuário. Antes do sistema ser utilizado é necessário ter um modelo de conectividade do *firewall*, que contém detalhes como quantidade de interfaces que o *firewall* possui, quais sub-redes estão conectadas a cada interface, e onde a Internet está situada no respectivo *firewall*.

Dessa forma, é possível considerar que a principal contribuição do protótipo FANG foi à interface de perguntas e respostas para o usuário.

4.1.4 SISTEMA LUMETA

Woll (2001) apresenta o analisador de *firewall* LUMETA - LFA (*Lumeta Firewall Analyser*), que teve como ponto de partida o sistema FANG (MAYER, 2000). De acordo com o autor o sistema LFA melhora o sistema FANG em muitos aspectos, os avanços mais significativos estão na interação com o usuário. O sistema LFA automaticamente emite a questão de interesse do usuário e mostra o resultado de forma a ressaltar riscos sem comprometer a visão de alto nível. Isso resolve um importante problema encontrado no uso do sistema FANG, onde na maioria das vezes os usuários não sabiam que pergunta fazer.

A entrada do sistema LFA consiste em uma tabela semelhante a uma tabela de roteamento e arquivos de configuração do *firewall*. A ferramenta

analisa esses dados de baixo nível, e simula o comportamento do *firewall* contra todos os pacotes que são possíveis de receber. A simulação é realizada sem o envio de pacotes, ou seja, a verificação é efetuada por uma varredura de cada regra existente na base de regras. O administrador recebe um relatório completo de quais tipos de tráfego o *firewall* permite entrar da Internet para a intranet, e quais tipos de tráfego são permitidos sair da intranet. O relatório do sistema LFA é apresentado em um formato semelhante a páginas *web*. Além disso, outras características podem ser observadas:

- O usuário não precisa mais escrever o arquivo de conectividade do *firewall*. LFA possui um novo módulo de início que carrega uma tabela de roteamento formatada e, cria automaticamente um arquivo de conectividade do *firewall*.
- A utilização de uma GUI como mecanismo de entrada tem se mostrado difícil para os usuários. No lugar disso, LFA é agora uma seqüência de processos que simula a política do *firewall* comparando com praticamente todas as possibilidades de pacotes.
- Uma parte crucial do processamento é a seleção automática de perguntas. A seleção de perguntas precisa assegurar uma ampla cobertura de possibilidades.
- A saída das respostas do LFA é formatada como uma coleção de páginas *web* (*HTML*). Este formato possibilita apresentar a saída em múltiplos níveis de abstração e de múltiplos pontos de vista, permitindo facilmente notar detalhes.
- Abrange um maior número de fabricantes de *firewalls*. Para este propósito, o sistema LFA usa uma linguagem de configuração intermediária, para a qual é possível converter a configuração de vários fabricantes.

Com relação ao sistema FIRMATO, o autor identifica principalmente dois problemas:

Primeiro, o usuário tem que aprender a sintaxe e a semântica da linguagem MDL, o que leva algum tempo e esforço.

Segundo, a informação que é necessária para descrever a conectividade de um *firewall* não é prontamente disponibilizada aos administradores do *firewall* em um formato satisfatório. Esta informação é tipicamente codificada somente na tabela de roteamento do *firewall*. Porém, tabelas de roteamento de entradas são usualmente interligadas: é comum ter muitas sobreposições de tabelas de roteamento de entradas que cobrem o mesmo endereço IP. A semântica da tabela de roteamento determina qual a rota é utilizada para um dado endereço IP: é o mais específico, por exemplo, a menor entrada para uma sub-rede que contém determinado endereço IP é o que determina a rota àquele endereço IP. A tarefa de acessar a tabela de roteamento, e manualmente converter isto em listas separadas de intervalos de endereços IP, tem se mostrado difícil e propenso a erros.

Para resolver ambos os problemas o sistema LFA introduziu um novo módulo *front-end*, chamado route2hos, que converte automaticamente a tabela de roteamento em um arquivo de conectividade de *firewall*. Tudo que é requerido do usuário é o fornecimento da tabela de roteamento do *firewall* (semelhante ao formato de saída do comando `netstat` dos sistemas Unix).

O módulo route2hos usa um mecanismo que implementa a semântica da tabela de roteamento. Em outras palavras, para um dado endereço IP, é possível determinar sobre qual interface de rede um pacote com este endereço de destino será roteado. Usando este mecanismo adequadamente contra as sub-redes listadas na tabela de roteamento, route2hos é capaz de criar listas distintas de intervalos de endereços IP que o mecanismo de perguntas do sistema FANG requer. A saída do route2hos é um arquivo de descrição de conectividade do *firewall*.

Como parte deste processamento feito por route2hos, são produzidas definições para dois grupos especiais, chamados Internos e Externos. O grupo de `computadores` externo consiste de todos os endereços IP que conectam pela interface padrão, de acordo com a tabela de roteamento do

firewall. Este grupo tipicamente inclui a Internet e sub-redes de corporações que são externas ao *firewall*. Já o grupo Interno abrange os que não pertencem ao grupo Externo.

4.1.5 SISTEMA FACE

Verma (2005) apresenta o sistema FACE (*Firewall Analysis and Configuration Engine*), uma ferramenta que auxilia na análise e configuração de *firewalls*. Ao usar o sistema FACE, os administradores de redes podem automaticamente gerar e analisar configurações para um ou mais *firewalls* da rede especificando a política de filtragem e o modelo de ameaça aos quais um *firewall* deve prover defesa, por exemplo, contra tráfego forjado (*spoofed*) em pontos específicos da rede.

O sistema FACE gera configurações do *firewall* para implementar uma política de segurança, sempre que for possível. Quando não for possível realizar a implementação da política especificada, sistema FACE indica o motivo e sugere alterações, para que a política se torne possível de ser implementada. Sugestões incluem mudanças de confianças e ajuste de conexões. Uma consideração feita pelo autor em seu trabalho é que as técnicas descritas são restritas para políticas que rejeitam todo o tráfego por padrão e especifica somente o que é permitido.

Dessa forma o autor considera que um importante avanço do sistema FACE é prover ajuda para configurar *firewalls* da rede de uma organização para rejeitar pacotes que podem ser forjados. Segundo o autor as principais formas de ataques a *firewalls* são os ataques de *Spoof* e *Smurf*.

4.2 TRABALHO PROPOSTO E OS TRABALHOS RELACIONADOS

Neste capítulo foi constatada a necessidade de eficientes mecanismos para avaliação do nível de proteção oferecido pelos *firewalls*. A maioria dos trabalhos focam nos problemas encontrados na configuração de *firewalls*,

principalmente devido às sintaxes utilizadas para implementação das regras.

Com exceção do Sistema Face que envia pacotes na realização dos testes, os demais sistemas utilizam análise passiva, na qual não ocorre a injeção de pacotes, por isso não é intrusiva, em comparação com análise ativa. Porém, não foram observadas enfaticamente técnicas que possibilitem determinar se o *firewall* está ou não efetuando o bloqueio de determinados tipos de pacotes específicos. Também não foi observada a preocupação para descobrir se existe proteção do próprio *firewall* contra ataques de inundação ou exaustão de recursos e ainda, um mecanismo que possa ser utilizado para testar a maioria dos *firewalls*, características existentes no modelo proposto neste trabalho.

Assim, o sistema de análise proposto se diferencia dos trabalhos vistos anteriormente, principalmente do ponto de vista da arquitetura proposta e metodologia utilizada.

4.3 CONSIDERAÇÕES

Este capítulo procurou apresentar o estado da arte da utilização de ferramentas automatizadas testes de *firewalls*.

Através dos trabalhos descritos neste capítulo, ratifica-se o interesse e a necessidade de ferramentas eficientes para análise e testes de *firewalls*. Sendo possível notar a lacuna existente quando se trata de ferramentas para análise ativa.

4.3.1 TÉCNICA DE VARREDURA DE PORTAS

Existem vários tipos de varreduras de portas, e diversas ferramentas utilizadas para esta finalidade. Um problema observado é que os resultados produzidos pelas ferramentas utilizadas para varredura de portas geralmente são baseados em pacotes devolvidos ou mensagens retornadas pelos computadores alvo da varredura. Porém, tais resultados podem não ser

confiáveis e precisos, pois os *firewalls* podem enviar respostas com o propósito de confundir e enganar as ferramentas de varredura de portas, ou ainda não retornar resposta alguma, deixando dúvidas do que pode ter ocorrido com os pacotes enviados: se foram perdidos na rede por algum problema, aceitos, descartados ou rejeitados.

Isto faz com que os resultados sejam contestados, pois não há como confirmar se o *firewall* está ou não permitindo a passagem de determinados pacotes.

Uma ferramenta com tais características é o Nmap (Nmap, 2005), amplamente utilizada para análise de estado de portas de computadores. O autor do Nmap descreve, por exemplo, o funcionamento da varredura UDP como:

"...*O scan UDP funciona enviando um datagrama UDP vazio (sem conteúdo no campo de dados) para cada porta especificada no intervalo de portas. Se um erro ICMP de porta inalcançável (tipo 3, código 3) é retornado, a porta está fechada. Outros tipos de mensagens ICMP, como por exemplo: tipo 3, códigos 1, 2, 9, 10 ou 13, marcam a porta como filtrada. Ocasionalmente um serviço irá responder com um pacote UDP, provando que está aberta. Se nenhuma resposta é recebida após as retransmissões, a porta é classificada como aberta/filtrada. Isto significa que a porta poderia estar aberta, ou talvez que filtros de pacotes estejam bloqueando a comunicação*...".

Ainda segundo o autor da ferramenta Nmap: "…*Um grande desafio para o escameamento UDP é fazê-lo rapidamente. Portas abertas e filtradas raramente enviam alguma resposta, deixando o Nmap esgotar o tempo (time out) e então efetuar retransmissões para o caso de a sondagem ou a resposta ter sido perdida. Portas fechadas são um problema ainda maior, elas costumam enviar de volta uma mensagem ICMP de porta inalcançável. Porém muitos sistemas operacionais, por exemplo, o Linux e o Solaris, limitam a taxa de mensagem ICMP de porta inalcançável por padrão. O Nmap detecta a limitação de taxa e diminui o ritmo de envio de mensagens para evitar inundar a rede. Dessa forma, uma varredura das 65535 portas de um computador com sistema operacional Linux pode levar mais de 18 horas. Idéias para acelerar o escaneamento UDP incluem fazer uma varredura rápida*

das portas mais comuns primeiro e utilizar o parâmetro < --host--timeout > para pular computadores lentos…".

De acordo com o que foi afirmado anteriormente, o método proposto não deve ser confundido com a técnica de varredura de portas, utilizadas por algumas ferramentas de análise de redes, pois nas técnicas de varredura efetuadas pelas ferramentas convencionais, os resultados se baseiam principalmente em respostas de mensagens de retorno ICMP, e em alguns casos estas mensagens nem sempre ocorrem.

O mesmo não ocorre com o sistema *WHATWALL*, pois o sistema observa também os pacotes que passaram através do *firewall*, utilizando-se de *probes* estrategicamente posicionados nas interfaces de rede do *firewall*.

CAPÍTULO 5

ARQUITETURA E MÉTODOS DE ANÁLISE

Neste capítulo são apresentadas a arquitetura do sistema e a descrição dos métodos de análise utilizados. O sistema foi batizado de *WHATWALL*. Como já mencionado, possui como principal objetivo realizar a verificação da viabilidade de utilização de técnicas de análise ativa baseadas no método de injeção de pacotes e observação dos resultados para descoberta do comportamento de *firewalls* de rede.

Para aplicar esta abordagem, as bases de regras dos *firewalls* serão consideradas “caixas pretas”. Os *firewalls* submetidos a esta metodologia podem ser avaliados principalmente quanto as seguintes características:

- Verificar se existe restrição para passagem de pacotes com características específicas (endereço IP de origem, endereço IP de destino, porta de origem, porta de destino);
- Verificar se as regras de filtragem do firewall operam no modo *statefull* ou *stateless*;
- Verificar se o *firewall* implementa mecanismos de proteção contra ataques de inundação que podem ocasionar exaustão de recursos do próprio *firewall*. Exemplos de proteções são: SYN *Cookies* ou SYN *Protection*.

A seguir, primeiramente serão descritas a arquitetura do sistema proposto, as partes que o compõem, suas funcionalidades e posteriormente a metodologia de utilização do sistema.

5.1 ARQUITETURA DO SISTEMA *WHATWALL*

A arquitetura do sistema proposto neste trabalho é representada na figura 5.1. Conforme pode ser observado, o sistema é composto por um controlador e diversos agentes. Cada agente é composto por módulos dos tipos: injetor e *probe*.

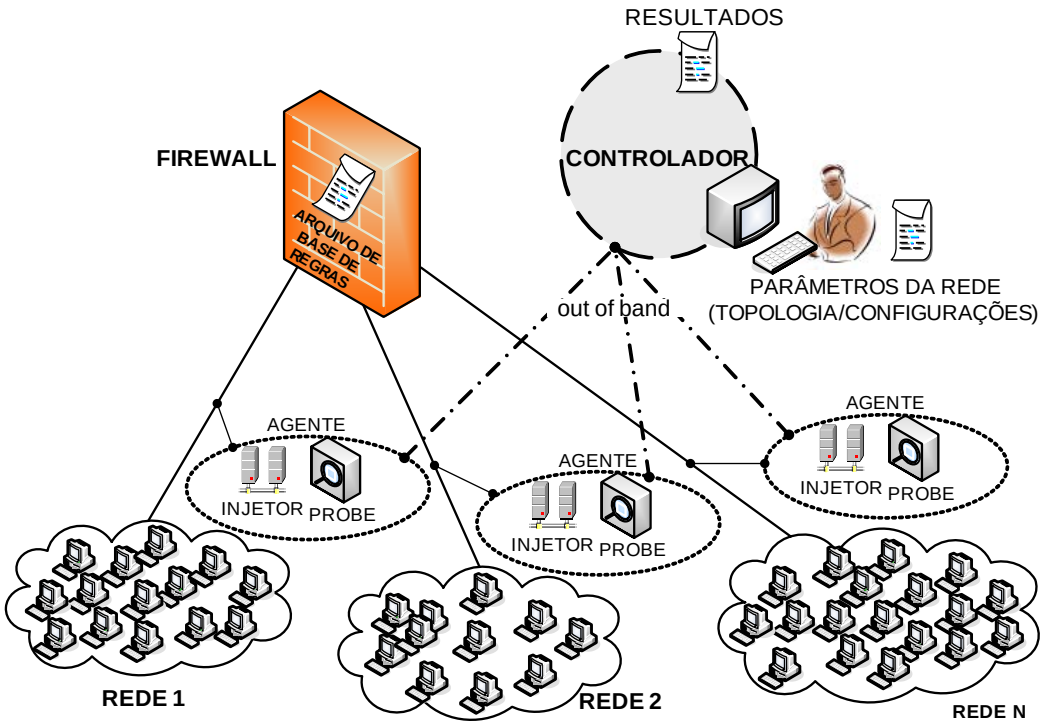

Figura 5.1 – Arquitetura do sistema de análise proposto

5.1.1 CONTROLADOR

O controlador é considerado o núcleo do sistema, sendo responsável principalmente por:

- Interação do usuário com o sistema;
    - Recebimento de informações a respeito da topologia de rede e informações sobre o *firewall*;
    - Controle dos módulos Injetores;
    - Controle dos módulos *Probes*;
- Recebimento dos registros gerados pelos agentes;
- Análise dos resultados;
- Exibição dos resultados para o usuário;

É por meio do controlador que ocorre a interação do usuário com o sistema, como a entrada de informações a respeito da topologia da rede, por exemplo, endereços IP e “dicas” a respeito das configurações do *firewall* ou da rede, estas “dicas” têm por objetivo identificar faixas de endereços IP e portas que possuam tratamento específico de forma que seja possível reduzir o número de pacotes necessários em algumas análises.

Também pelo controlador o usuário aciona os módulos injetores e *probes* para início, interrupção ou finalização dos testes para as diversas análises.

Ao final do processo, isto é, após o recebimento dos registros dos diversos agentes envolvidos na análise, cabe ao controlador comparar estes registros recebidos e apresentar os resultados ao usuário.

5.1.2 AGENTES

Os agentes, compostos por módulos injetores e módulos *probes*, são responsáveis pela condução das análises, seja injetando pacotes a partir do módulo injetor, seja observando os resultados nas interfaces do *firewall* através dos módulos *probes*. Podem existir vários tipos de agentes.

Para cada análise podem haver diversos agentes, sendo cada um associado a uma interface de rede do *firewall*, conforme será visto adiante. Para cada teste, dependendo de seu posicionamento e do sentido em que os testes serão realizados os agentes recebem denominações específicas,

conforme será visto no item 5.2.

5.1.2.1 MÓDULOS INJETORES

O módulo injetor é responsável pela injeção dos pacotes específicos para cada análise, como por exemplo: UDP, IP, ICMP, TCP para o *firewall*. Exemplos de módulos injetores são: Injetor UDP, Injetor SYN *Flood*, Injetor ICMP, Injetor *Source Routing*, Injetor ICMP *Directed Broadcast*, etc. As funcionalidades específicas de cada um são respectivamente:

- Injetor UDP: injetar datagramas UDP para observar se os mesmos serão filtrados pelo *firewall*;
- Injetor TCP: injetar datagramas TCP para observar se os mesmos serão filtrados pelo *firewall*;
- Injetor ICMP: injetar datagramas ICMP para observar se os mesmos serão filtrados pelo *firewall*;
- Injetor SYN *Flood*: realizar análises de resistência do *firewall* a ataques de inundação de segmentos TCP SYN. Este módulo é responsável por duas análises: (a) SYN *Flood Protection*: utilizada para verificar se o *firewall* implementa alguma técnica de proteção contra ataques de SYN *Flood*, como por exemplo, SYN *Protection* ou SYN *Cookies*;
  (b) SYN *Flood Resistance*: utilizada para verificar se a implementação de mecanismos de proteção contra ataques de SYN *Flood* não causam degradação de funcionalidades ou exaustão de recursos do próprio *firewall* em uma situação de ataque.
- *Injetor Source Routing*: verificar se existe a filtragem de datagramas IP *source routing*;
- Injetor de Fragmentação: verificar se existe controle e resistência a ataques de fragmentação;
- Injetor ICMP *Directed Broadcast*: verificar filtragem de datagramas ICMP direcionados para endereços *broadcast*;

Os pacotes injetados na rede podem trafegar juntamente com outros

tipos de tráfegos existentes (tráfego de fundo), de forma que os pacotes enviados pelo módulo injetor deverão ser identificados posteriormente.

A identificação dos pacotes é efetuada pelo conteúdo do campo "*identification*" do cabeçalho IP.

5.1.2.2 MÓDULOS *PROBES*

Os módulos *probes* são responsáveis pela monitoração, coleta e registros de informações do tráfego da rede. Estes registros são armazenados em um arquivo e posteriormente repassados ao controlador.

5.2 PROCESSO DE DESCOBERTA DE REGRAS DE FILTRAGEM

Considere a topologia de rede do cenário genérico representado na figura 5.2. Denominações foram dadas às redes, com a finalidade de simplificar a identificação das redes envolvidas na análise e o sentido em que os testes serão efetuados.

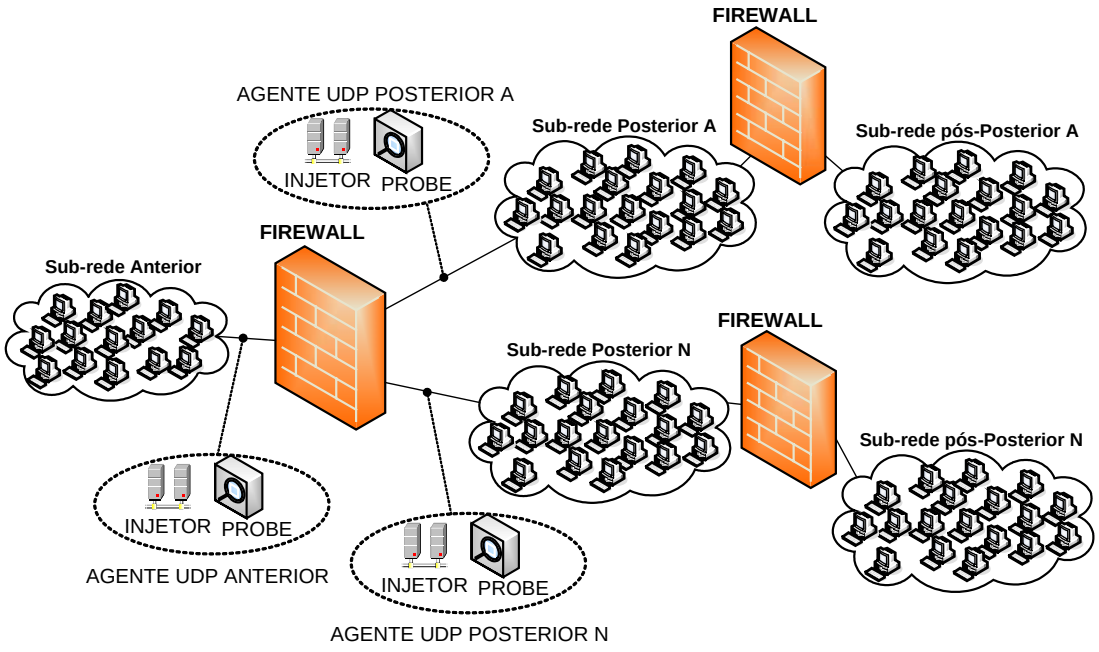

Figura 5.2 – Cenário genérico

Para uma determinada análise, a rede onde é originada a injeção dos pacotes é denominada sub-rede anterior. Já as redes para onde os pacotes serão encaminhados são denominadas sub-redes posteriores; Cada rede abaixo destas, sub-rede pós-posterior.

Analogamente, o mesmo ocorre com os diversos agentes existentes. Os agentes são classificados como: agente anterior e agente posterior.

O agente anterior, como o próprio nome sugere, é posicionado em uma interface pertencente a uma sub-rede anterior ao *firewall*, no lado onde os pacotes são gerados para serem injetados. Enquanto o agente posterior é posicionado nas interfaces pertencentes às sub-redes posteriores ao *firewall* para onde os pacotes são direcionados.

Conforme já comentado, pacotes com determinadas características para a análise a ser efetuada são gerados e enviados pelo módulo injetor do agente anterior em direção a uma ou mais sub-redes isoladas pelo *firewall* cujo comportamento se deseja analisar ou, em certas análises diretamente para o próprio *firewall*.

Posteriormente são verificados os registros dos módulos *probes* nos agentes anteriores e posteriores para constatar a passagem ou não dos pacotes previamente enviados. Dessa forma é possível concluir se o *firewall* permitiu ou não a passagem dos pacotes enviados, efetuando um rastreamento dos pacotes pelos registros dos módulos *probes* anterior e posteriores.

No cenário apresentado na figura 5.2, é possível a ocorrência de diferentes situações, por exemplo:

- O agente anterior através de seu módulo injetor, envia pacotes para um computador na sub-rede posterior, e o *firewall* permite a passagem dos pacotes.
- O agente anterior através de seu módulo injetor, envia pacotes para um computador na sub-rede posterior, o *firewall* efetua uma conversão de endereço de rede (NAT) e redirecionamento de portas e repassa os pacotes para o computador da sub-rede posterior.
- O agente anterior através de seu módulo injetor, envia pacotes para

um computador de uma sub-rede posterior, o *firewall* bloqueia a passagem dos pacotes e descarta silenciosamente os pacotes enviados, sem retornar nenhuma mensagem de erro para o agente anterior.

- O agente anterior através de seu módulo injetor, envia pacotes para um computador de uma sub-rede posterior, o *firewall* filtra a passagem dos pacotes e retorna mensagens ICMP para o agente anterior, como por exemplo, ICMP *port unreachable* ou ICMP *host unreachable*.

Estes exemplos serão descritos em maiores detalhes na Tabela 5.1. Cabe ressaltar que a técnica aqui utilizada não consiste na utilização de varredura de portas, cujos comentários e considerações são mencionadas posteriormente no item 5.8.

5.3 SEQÜÊNCIA DE ETAPAS PARA ANÁLISE

A seqüência de etapas para analises a ser seguida pelo sistema proposto, para efetuar os diversos tipos de análise, aplicada em cada interface de rede do *firewall* é composta por 5 etapas principais, conforme ilustrado no diagrama apresentado na figura 5.3.

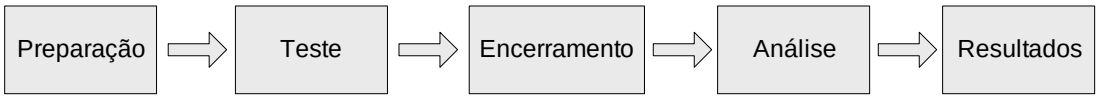

Figura 5.3 – Seqüência de etapas para análise

5.3.1 ETAPA DE PREPARAÇÃO

Antes do início de cada teste existe a etapa de preparação. A etapa de preparação é composta dos seguintes passos:

- Definição do sentido do teste;

- Identificação dos agentes;
- Calibração;

5.3.1.1 DEFINIÇÃO DO SENTIDO

Neste passo é definido o sentido do teste, ou seja, de qual sub-rede serão injetados os pacotes e para qual sub-rede os pacotes serão enviados, identificando-se então a sub-rede anterior e a sub-rede posterior ou sub-redes posteriores no caso da análise ser efetuada em mais de uma sub-rede posterior.

5.3.1.2 IDENTIFICAÇÃO DOS AGENTES

Neste passo é identificado o agente anterior na sub-rede anterior e também são identificados os agentes posteriores na sub-redes posteriores.

Cada agente deve ser adequadamente posicionado de forma que possa observar o fluxo de comunicação na respectiva interface de rede do *firewall*.

5.3.1.3 CALIBRAÇÃO

Excepcionalmente, alguns testes, antes de serem realizados, necessitam de uma fase de calibração. O objetivo da calibração é determinar o intervalo de tempo máximo entre envio de pacotes para que não ocorram perdas dos mesmos devido à exaustão de recursos, observando que este intervalo pode variar conforme o ambiente onde a análise é efetuada.

Para determinar qual o intervalo de tempo ideal que os pacotes devem ser enviados sem que ocorram perdas é adotado o seguinte procedimento:

Primeiramente, o usuário seleciona uma lista de tuplas[6] que não

[6] Tupla é aqui definida como sendo o conjunto de parâmetros dos pacotes: $<IP_O; P_O; IP_D; P_D>$, sendo: IP_O: Endereço IP de origem; IP_O: Porta de origem; IP_D: Endereço IP de destino; P_D: Porta de destino;

estejam sendo filtradas pelo *firewall*. Os parâmetros dos pacotes a serem utilizados na calibração pertencem à lista de tuplas selecionadas para a calibração.

Os parâmetros das tuplas e o protocolo a ser utilizado são passados para o controlador pelo usuário. Então o usuário, através do controlador inicializa o módulo injetor.

Os pacotes são injetados seqüencialmente e, posteriormente é efetuada uma análise dos registros dos módulos *probes* para observar se ocorreram perdas de pacotes. Se não for observada a ocorrência de perdas de pacotes significa que não é necessário interpor intervalo entre os envios. Caso seja observada a ocorrência de perdas de pacotes, significa que existe a necessidade de interpor intervalos entre os envios dos pacotes. Então, um intervalo da ordem de nano segundos é adicionado, e a injeção de pacotes é repetida.

Se for novamente observada a ocorrência de perda de pacotes, significa que o intervalo entre o envio de pacotes precisa ser aumentado. Dessa forma, o intervalo de tempo ideal entre o envio de pacotes a ser utilizado no teste deverá ficar abaixo do limiar no qual foi observada a ocorrência de perdas.

Para testes nos quais a fase de calibração não se aplica, esta etapa não é efetuada, passando diretamente para as etapas subseqüentes.

5.3.1.4 ATIVAÇÃO DOS MÓDULOS *PROBES*

Os módulos *probes* são ativados pelo usuário através do controlador. E iniciam a monitoração e coleta do tráfego da rede. Para cada agente *probe* posterior deve ser ativado o método de descoberta de endereços IP ativos.

5.3.1.5 ATIVAÇÃO DO MÓDULO INJETOR

O módulo injetor é ativado pelo usuário através do controlador, então

recebe os parâmetros referentes as tuplas que serão analisadas e o protocolo que será utilizado. Observando que as tuplas utilizadas podem ou não fazerem parte da lista de tuplas usadas na etapa de calibração.

5.3.2 Etapa de Teste

Para cada tupla é montado um pacote. Este pacote e injetado na sub-rede anterior em direção a uma ou mais sub-rede posterior, ou dependendo da análise, diretamente para o *firewall*.

5.3.3 Etapa de Encerramento

A etapa de encerramento do teste ocorre após a finalização do envio dos pacotes e, posteriormente na coleta dos registros efetuados pelos diversos agentes, isto é, os registros gerados pelos módulos injetores e *probes*. Após esta coleta, os registros são enviados ao controlador.

5.3.4 Etapa de Análise

Após o recebimento dos registros dos módulos injetores e *probes*, o controlador efetua a comparação de todos de registros recebidos.

Para cada pacote injetado é analisado seu trânsito pelas diferentes sub-redes através dos registros nos módulos *probes*.

5.3.5 Etapa de Apresentação dos Resultados

A apresentação dos resultados pelo controlador ocorre após a comparação e análise dos registros recebidos dos módulos injetores e *probes* de cada agente, resultando em um registro das diferenças entre os registros enviados pelos módulos injetores e *probes*.

Dessa forma é possível identificar quais pacotes enviados pelo módulo

injetor passaram ou não pelo *firewall* ou ainda, verificar mensagens ICMP que eventualmente foram retornadas.

A seqüência de análise descrita anteriormente foi desenvolvida para ser aplicada para a análise da grande maioria dos protocolos da pilha TCP/IP. A seguir será utilizada para exemplificar como efetuar a análise para os protocolos TCP e UDP. Estes dois protocolos foram escolhidos devido à importância que os mesmos representam para a pilha de protocolos TCP/IP, bem como para os estudos das técnicas de filtragem de pacotes.

5.4 MÉTODO PARA ANÁLISE DE FILTRAGEM UDP

O método para análise de filtragem UDP é utilizado para verificação do tratamento dado pelo *firewall* aos datagramas UDP com características específicas.

A seguir, serão aplicadas as etapas de preparação, teste, encerramento do teste, análise e apresentação dos resultados, conforme descritos anteriormente no item 5.3.

5.4.1 POSSÍVEIS SITUAÇÕES E CENÁRIOS

O *firewall*, no recebimento de um datagrama UDP, pode tratá-lo conforme as situações descritas na tabela 5.1. A coluna “situação” refere-se às funcionalidades do *firewall* em cada caso, a coluna “T” representa o instante de tempo, no qual “t_1” refere-se à observação do datagrama no instante 1, “t_2” refere-se à observação do datagrama no instante posterior e assim sucessivamente. “*Probe* anterior” e “*Probe* posterior” indicam o tráfego observado por cada um dos *probes* utilizados.

Tabela 5.1 – Possíveis situações de tratamento pelo *firewall* no recebimento de um datagrama UDP

Situação	T	*Probe* anterior	*Probe* posterior
Regra sem filtragem	t_1	Observa o datagrama UDP original	
	t_2		Observa o datagrama UDP original
Regra sem filtragem e com NAT	t_1	Observa o datagrama UDP original	
	t_2		Observa o datagrama UDP, porém com endereço IP destino, endereço IP origem, porta UDP destino ou porta UDP de origem alteradas.
Regra com filtragem e descarte silencioso.	t_1	Observa o datagrama UDP original.	
	t_2		Não observa
Regra com filtragem e com retorno de mensagem ICMP.	t_1	Observa o datagrama UDP original.	Não observa
	t_2	Observa um datagrama ICMP com mensagem ICMP associada.	
Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Não há filtragem de retorno de mensagem ICMP associada.	t_1	Observa o datagrama UDP original.	
	t_2		Observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	Observa mensagem ICMP associada.	
Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Existe filtragem de retorno de mensagem ICMP associada.	t_1	Observa o datagrama UDP original.	
	t_2		Observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	Não observa.	

Como observado na tabela 5.1 não é possível identificar precisamente o comportamento do *firewall* quando o endereço IP de destino do pacote não está ativo na sub-rede posterior. Por este motivo é conveniente, nesta situação, que o *probe* posterior responda à requisição ARP *Request* a fim de possibilitar o encaminhamento do pacote pelo *firewall*.

5.4.2 SEQÜÊNCIA DE ETAPAS PARA ANÁLISE

A seguir são descritas as etapas para análise de filtragem UDP.

5.4.2.1 ETAPA DE PREPARAÇÃO

A etapa de preparação do método para análise de filtragem UDP consiste nos seguintes passos:

A-) Definição do sentido

A etapa de definição do sentido do teste é realizada para a identificação da sub-rede anterior e das sub-redes posteriores.

B-) Posicionamento dos agentes

Nesta etapa é posicionado o agente UDP anterior na sub-rede anterior e também são posicionados os agentes posteriores na sub-redes posteriores. Os agentes são posicionados de forma que possam observar o fluxo de comunicação nas interfaces de redes do *firewall*.

C-) Calibração

Para o teste de análise UDP é necessário efetuar a calibração antes da execução dos testes para determinar o intervalo de tempo ideal entre envio dos datagramas UDP para que não ocorram perdas de pacotes devido à exaustão de recursos, observando que estes fatores variam de acordo com o ambiente de testes.

A etapa de calibração é efetuada seguindo os procedimentos já descritos na seção 5.3.1.3.

D-) Ativação dos módulos *Probes*

Os módulos *probes* são ativados. Para cada agente *probe* posterior deve ser ativado o método de descoberta de endereços IP ativos.

E-) Ativação do módulo Injetor

O módulo injetor é ativado após o recebimento dos parâmetros referentes as tuplas que serão analisadas.

5.4.3 ETAPA DE TESTE

Para cada tupla UDP é montado um pacote IP/UDP, e enviado N[7] vezes na sub-rede anterior em direção ao *firewall*, com uma taxa inferior à obtida na etapa de calibração.

5.4.4 ETAPA DE ENCERRAMENTO

A etapa de encerramento do teste consiste na coleta dos registros gerados pelos módulos injetores e *probes*. Após esta coleta, os registros são enviados ao controlador. O conteúdo dos registros são informações semelhantes às apresentadas na tabela 5.2.

5.4.5 ETAPA DE ANÁLISE

Após o recebimento dos arquivos contendo os registros dos módulos injetores e *probes*, o controlador efetua a comparação de todos de registros recebidos. Cada pacote UDP injetado tem seu trânsito rastreado pelos diversos registros de cada agente, semelhante à tabela 5.2, que foi obtida pela análise das considerações apresentadas na tabela 5.1.

Tabela 5.2 – Análise de resultados de filtragem UDP

Injetor	T	*Probe* anterior	*Probe* posterior	Conclusão
$<IP_O; P_O; IP_D; P_D>$	t_1	Não observa		Falha no teste
$<IP_O; P_O; IP_D; P_D>$	t_1	$<IP_O; P_O; IP_D; P_D>$		Regra sem filtragem
	t_2		$<IP_O; P_O; IP_D; P_D>$	
$<IP_O; P_O; IP_D; P_D>$	t_1	$<IP_O; P_O; IP_D; P_D>$		Regra sem filtragem e com NAT.
	t_2		$<IP'_O; P'_O; IP'_D; P'_D>$	
$<IP_O; P_O; IP_D; P_D>$	t_1	$<IP_O; P_O; IP_D; P_D>$		Regra com filtragem e com descarte silencioso.
	t_2		Não observa.	
$<IP_O; P_O; IP_D; P_D>$	t_1	$<IP_O; P_O; IP_D; P_D>$		Regra com filtragem e com retorno de mensagem ICMP associada.
	t_2	Observa um datagrama ICMP.	Não observa.	

[7] N é o número de vezes que o pacote é transmitido, tipicamente N é igual a 3.

5.4.6 Etapa de Apresentação dos Resultados

A exibição dos resultados pelo controlador ocorre após a etapa de análise descrita no item 5.4.5, resultando em um registro das diferenças entre os registros enviados pelos módulos injetores e *probes*.

Dessa forma é possível identificar quais pacotes enviados pelo módulo injetor passaram ou não pelo *firewall* ou ainda, verificar mensagens ICMP que eventualmente foram retornadas.

Na tabela 5.2, é possível notar o pacote que foi injetado conforme as características de uma determinada tupla $<IP_O; P_O; IP_D; P_D>$, o que foi observado por cada módulo *probe* em cada instante, e as possíveis conclusões.

A coluna “conclusão” mostra os resultados do comportamento do *firewall* mediante o datagrama enviado.

5.5 Método Para Análise de Filtragem TCP

Pelo motivo do protocolo TCP ser orientado a conexão, foram elaboradas duas tabelas para análise:

(a) As possíveis situações de tratamento para um segmento TCP pelo *firewall* no estabelecimento de conexão TCP;

(b) As possíveis situações de tratamento pelo *firewall* no envio de um segmento TCP de dados não precedido de estabelecimento de conexão TCP, representadas respectivamente pelas tabelas 5.3 e 5.4.

A seguir serão abordados os principais cenários para a análise de filtragem TCP.

5.5.1 Possíveis Situações e Cenários

Para análise TCP as possíveis situações e cenários mudam em relação ao protocolo UDP, como será visto adiante.

5.5.1.1 ESTABELECIMENTO DE CONEXÃO TCP

O *firewall*, no recebimento de uma solicitação de conexão TCP, pode tratá-la conforme as situações descritas na tabela 5.3.

Tabela 5.3 – Possíveis situações de tratamento para um segmento TCP pelo *firewall* no estabelecimento de conexão TCP

Situação	T	***Probe* anterior**	***Probe* posterior**
Regra sem filtragem e sem NAT.	t_1	Observa o segmento TCP original, com o *flag* SYN ativo.	
	t_2		Observa o segmento TCP original, com o *flag* SYN ativo.
	t_3		Observa a resposta com os *flags* SYN/ACK.
	t_4	Observa a resposta com os *flags* SYN/ACK ativos.	
	t_5	Observa um segmento TCP com o *flag* ACK ativo.	
	t_6		Observa o segmento TCP com *flag* ACK.
Regra sem filtragem e com NAT.	t_1	Observa o segmento TCP original, com o *flag* SYN ativo.	
	t_2		Observa o segmento TCP original, com o *flag* SYN ativo, com portas e/ou endereço IP de origem e/ou destino alterados.
	t_3		Observa um segmento TCP com os *flags* SYN/ACK ativos.
	t_4	Observa o segmento TCP com os *flags* SYN/ACK.	
	t_5	Observa o datagrama TCP original, com o *flag* ACK.	
	t_6		Observa o segmento TCP original, com o *flag* ACK.
Regra com filtragem e com descarte silencioso.	t_1	Observa o segmento TCP original, com o *flag* SYN, porta e/ou endereço IP de origem e/ou porta/endereço IP de destino originais.	
	t_2	Não observa	Não observa.
Regra com filtragem e com retorno de mensagem ICMP.	t_1	Observa o segmento TCP original, com o *flag* SYN, porta e ou endereço IP de origem originais.	
	t_2	Observa um datagrama ICMP associado.	Não observa.

Regra com configuração SYN *proxy*	t_1	Observa o datagrama TCP original, com o *flag* SYN.	
	t_2	Observa o datagrama TCP com os *flags* SYN/ACK.	
	t_3	Observa o datagrama TCP com os *flags* ACK.	
	t_4		Observa o datagrama TCP original, com o *flag* SYN.
	t_5		Observa o datagrama TCP com os *flags* SYN/ACK.
	t_6		Observa o datagrama TCP com os *flags* ACK.
Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Não há filtragem de retorno de mensagem ICMP.	t_1	Observa o datagrama TCP original, com o *flag* SYN, porta/endereço IP de origem originais.	
	t_2		Observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	Observa mensagem ICMP *host unreachable*.	
Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Existe filtragem de retorno de mensagem ICMP presente.	t_1	Observa o datagrama TCP original, com o *flag* SYN, porta e ou endereço IP de origem originais.	
	t_2		Observa requisição ARP Request para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	Não observa.	

5.5.1.2 ENVIO DE UM SEGMENTO DE DADOS TCP NÃO PRECEDIDO DE ESTABELECIMENTO DE CONEXÃO TCP

Na tabela 5.4 é uma situação no qual ocorre o envio de um segmento TCP de dados não precedido de estabelecimento de conexão. O principal objetivo é apresentar o comportamento quando a regra de filtragem do *firewall* opera no modo *statefull* ou *stateless*.

Tabela 5.4 – Possíveis situações de tratamento pelo *firewall* no envio de um segmento de dados TCP não precedido de estabelecimento de conexão TCP

Situação	T	*Probe* anterior	*Probe* posterior
Regra *statefull*	t_1	Observa o segmento TCP de dados	
	t_2	Não observa pacotes relacionados	Não observa pacotes relacionados
Regra *stateless* sem filtragem e sem NAT	t_1	Observa o segmento TCP de dados	
	t_2		Observa o segmento TCP de dados
Regra *stateless* sem filtragem e com NAT	t_1	Observa o segmento TCP de dados, original.	
	t_2		Observa o segmento TCP de dados, porta de origem, endereço IP de origem, porta de destino ou endereço IP de destino alterado.
Regra *stateless* com filtragem e com descarte silencioso.	t_1	Observa o segmento TCP de dados, original.	
	t_2	Não observa.	Não observa.
Regra com filtragem e com retorno de mensagem ICMP.	t_1	Observa o segmento TCP de dados, original.	Não observa.
	t_2	Observa um segmento ICMP associado.	
Regra *stateless* e endereço IP de destino pertencente a uma faixa da sub-rede posterior sem existir endereço IP ativo. Não há filtragem de retorno de mensagem ICMP associada.	t_1	Observa o segmento TCP de dados, original.	
	t_2		No instante t_2 observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	No instante t_3 observa mensagem ICMP associada.	
Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Existe filtragem de retorno de mensagem ICMP.	t_1	Observa o segmento TCP de dados, porta/endereço IP de origem originais.	
	t_2		No instante t_2 observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.
	t_3	Não observa.	

5.5.2 SEQÜÊNCIA DE ETAPAS PARA ANÁLISE

A seguir são descritas as etapas para análise de filtragem TCP.

5.5.2.1 ETAPA DE PREPARAÇÃO

A etapa de preparação do método para análise de filtragem TCP consiste nos seguintes passos:

A-) Definição do sentido do teste

Neste momento é definido o sentido do teste e realizada a identificação da sub-rede anterior e das sub-redes posteriores.

B-) Posicionamento dos agentes

Nesta etapa é posicionado o agente TCP anterior na sub-rede anterior e também são posicionados os agentes posteriores na sub-redes posteriores. Os agentes são posicionados de forma que possam observar o fluxo de comunicação nas interfaces de redes do *firewall*.

C-) Calibração

Para o teste de análise TCP é necessária uma fase de calibração antes da execução dos testes. O objetivo da calibração é determinar o intervalo de tempo ideal entre envio de pacotes TCP para que não ocorram perdas de pacotes devido à exaustão de recursos, observando que estes fatores variam de acordo com o ambiente de testes.

A etapa de calibração é efetuada seguindo os procedimentos já descritos na seção 5.3.1.3.

D-) Ativação dos módulos *probes*

Os módulos *probes* são ativados. Para cada agente *probe* posterior deve ser ativado o método de descoberta de endereços IP ativos.

E-) Ativação do módulo injetor

O módulo injetor é ativado após o recebimento dos parâmetros referentes as tuplas que serão analisadas.

5.5.2.2 ETAPA DE TESTE

Para cada tupla TCP deve ser criado um segmento de dados e datagramas IP e posteriormente, enviado “N” vezes na sub-rede anterior em direção ao *firewall*, com uma taxa inferior à obtida na etapa de calibração.

Neste teste também deve ser criado um segmento TCP SYN e datagrama IP e posteriormente enviado N vezes o pacote.

5.5.2.3 ETAPA DE ENCERRAMENTO

A etapa de encerramento do teste consiste na coleta dos registros efetuados pelos dos agentes. Após esta coleta, os registros são enviados ao controlador.

5.5.2.4 ETAPA DE ANÁLISE

Após o recebimento dos registros dos módulos injetores e *probes*, o controlador efetua a comparação de todos os arquivos de registros recebidos. Cada pacote tem seu trânsito monitorado através dos módulos *probes* anterior e posteriores.

Na tabela 5.5 são apresentadas as conclusões (na coluna conclusões) para os cenários analisados na tabela 5.3.

Tabela 5.5 – Análise de resultados das possíveis situações da tabela 5.3

Injetor	T	*Probe* anterior	*Probe* posterior	Conclusão
<IP_O; P_O; IP_D; P_D>	t_1	Não observa		Falha no teste
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN.		Regra sem filtragem e sem NAT.
	t_2		<IP_O; P_O; IP_D; P_D> SYN.	
	t_3		<IP_D; P_D; IP_O; P_O> SYN/ACK.	
	t_4	<IP_D; P_D; IP_O; P_O> SYN/ACK.		
	t_5	<IP_O; P_O; IP_D; P_D> ACK.		
	t_6		<IP_O; P_O; IP_D; P_D> ACK.	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Regra sem filtragem e com NAT.
	t_2		<IP'_O; P'_O; IP'_D; P'_D> SYN	
	t_3		<IP'_O; P'_O; IP'_D; P'_D> SYN/ACK	
	t_4	<IP_O; P_O; IP_D; P_D> SYN/ACK		
	t_5	<IP_O; P_O; IP_D; P_D> ACK		
	t_6		<IP'_O; P'_O; IP'_D; P'_D> ACK	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Regra com filtragem e com descarte silencioso.
	t_2		Não observa.	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Regra com filtragem e com retorno de mensagem ICMP.
	t_2	"mensagem ICMP"		
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Regra com SYN *Protection* implementado
	t_2	<IP_O; P_O; IP_D; P_D> SYN/ACK		
	t_3	<IP_O; P_O; IP_D; P_D> ACK		
	t_4		<IP_O; P_O; IP_D; P_D> SYN	
	t_5		<IP_O; P_O; IP_D; P_D> SYN/ACK	
	t_6		<IP_O; P_O; IP_D; P_D> ACK	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Não há filtragem de retorno de mensagem ICMP *host unreachable*.
	t_2		ARP *Request*	
	t_3	ICMP *host unreachable*		
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> SYN		Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Existe filtragem de retorno de mensagem ICMP *host unreachable*.
	t_2		ARP *Request*	
	t_3	ICMP *host unreachable*		

Na tabela 5.6 são apresentadas as conclusões para os cenários analisados na tabela 5.4.

Tabela 5.6 – Análise dos resultados de envio de um segmento TCP não precedido de estabelecimento de conexão

Injetor	T	*Probe* anterior	*Probe* posterior	Conclusão
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D>		*Não conclusivo*: a regra *statefull* ou regras *stateless* com descarte silencioso.
	t_2	Não observa.	Não observa.	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> + dados		Regra *stateless* sem filtragem e sem NAT
	t_2		<IP_O; P_O; IP_D; P_D> + dados	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> + dados		Regra *stateless* sem filtragem e com NAT
	t_2		<$IP´_O$; $P´_O$; $IP´_D$; $P´_D$> + dados.	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> + dados		Regra *stateless* com filtragem e com retorno de mensagem ICMP.
	t_2	Observa um segmento ICMP com mensagem "ICMP *port unreachable*".	Não observa.	
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> + dados		Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Não há filtragem de retorno de mensagem ICMP *host unreachable*.
	t_2		Requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.	
	t_3	No instante t_3 observa mensagem ICMP *host unreachable*.		
<IP_O; P_O; IP_D; P_D>	t_1	<IP_O; P_O; IP_D; P_D> + dados		Endereço IP de destino pertence a uma faixa da sub-rede posterior sem existir endereço IP ativo. Existe filtragem de retorno de mensagem ICMP *host unreachable* presente.
	t_2		No instante t_2 observa requisição ARP *Request* para resolver endereço de destino ou, no caso de NAT para o endereço de destino traduzido.	
	t_3	Não observa.		

5.5.2.5 APRESENTAÇÃO DOS RESULTADOS

A exibição dos resultados pelo controlador ocorre após a comparação e

análise dos registros recebidos dos módulos injetores e *probes*, resultando em um registro das diferenças entre os arquivos enviados pelos módulos injetores e *probes*.

Dessa forma é possível identificar quais pacotes enviados pelo módulo injetor passaram ou não pelo *firewall* ou ainda, verificar mensagens ICMP que eventualmente foram retornadas e ainda concluir se as regras do *firewall* analisado operam no modo *statefull* ou *stateless*.

5.6 MÉTODO PARA ANÁLISE SYN *FLOOD*

5.6.1 ANÁLISE SYN *FLOOD PROTECTION*

A análise SYN *Flood Protection* é utilizada para verificar se o *firewall* implementa alguma técnica de proteção contra ataques de SYN *Flood*, como por exemplo, SYN *Cookies* ou SYN *Protection*.

5.6.1.1 ETAPA DE PREPARAÇÃO

A etapa de preparação do método para análise de filtragem SYN *Flood Protection* consiste nos seguintes passos:

A-) Definição do sentido do teste

Neste momento é definido o sentido do teste e realizada a identificação da sub-rede anterior e das sub-redes posteriores.

B-) Posicionamento dos agentes

Nesta etapa é posicionado o agente SYN *Flood Protection* anterior na sub-rede anterior e também são posicionados os agentes posteriores na sub-redes posteriores. Os agentes são posicionados de forma que possam observar o fluxo de comunicação nas interfaces de redes do *firewall*.

Neste teste não é necessário efetuar a fase de calibração.

C-) Ativação dos módulos *probes*

Os módulos *probes* são ativados. Para cada agente *probe* posterior deve ser ativado o método de descoberta de endereços IP ativos.

D-) Ativação do módulo injetor

O módulo injetor é ativado após o recebimento dos parâmetros referentes as tuplas que serão analisadas.

5.6.1.2 ETAPA DE TESTE

O módulo injetor obtém do controlador a lista de tuplas a serem analisadas. Então envia seqüencialmente 3 segmentos TCP com o *flag* SYN ativo para uma porta TCP aberta no *firewall*. Estes segmentos devem ser registrados por um *probe* na sub-rede anterior.

5.6.1.3 ETAPA DE ENCERRAMENTO

A etapa de encerramento do teste consiste na coleta dos registros efetuados pelos dos agentes incluindo os registros gerados pelos módulos injetores e *probes*. Após esta coleta, os registros são enviados ao controlador.

5.6.1.4 ETAPA DE ANÁLISE

Após o recebimento dos arquivos contendo os registros dos módulos injetores e *probes*, o controlador efetua a comparação de todos de registros recebidos.

Se o *firewall* responder aos 3 segmentos SYN enviados, ou seja, intermediar a solicitação de abertura de conexão e só repassá-la a máquina de destino após a conexão estiver efetuada, é possível que o *firewall* possua

mecanismos de proteção contra ataques de SYN *Flood* implementado.

Porém, caso o *firewall* permita a passagem do 3 segmentos TCP enviados, os *probes* da sub-rede posterior deverão registrar o recebimento dos segmentos enviados. Caso isso ocorra, pode indicar a inexistência de proteção contra ataque SYN *Flood*.

5.6.1.5 Etapa de Apresentação dos Resultados

A exibição dos resultados pelo controlador ocorre após a comparação e análise dos registros recebidos dos módulos injetores e *probes*, resultando em um registro das diferenças entre os registros enviados pelos módulos injetores e *probes*.

Dessa forma é possível identificar quais segmentos enviados pelo módulo injetor passaram ou não pelo *firewall* ou ainda, verificar mensagens ICMP que eventualmente foram retornadas (mensagem ICMP associada).

5.7 Método De Análise SYN *Flood Resistance*

O método para análise de resistência a SYN *Flood* é efetuado pelo Injetor SYN *Flood*. A principal função do injetor SYN *Flood* é realizar análises de resistência do *firewall* a ataques de inundação de segmentos TCP SYN.

No método para análise de SYN *Flood Resistance* o objetivo é verificar se a implementação de mecanismos de proteção contra ataques de SYN *Flood* não causam degradação de funcionalidades ou exaustão de recursos do próprio *firewall* em uma situação de ataque.

5.7.1 Etapa de Preparação

A etapa de preparação do método para análise de filtragem Análise SYN *Flood Resistance* consiste nos seguintes passos:

A-) Definição do sentido do teste, analogamente aos testes anteriores.

B-) Posicionamento dos agentes

Nesta etapa é posicionado o agente SYN *Flood Resistance* anterior na sub-rede anterior e também são posicionados os agentes posteriores na sub-redes posteriores. Os agentes são posicionados de forma que possam observar o fluxo de comunicação nas interfaces de redes do *firewall*.

C-) Calibração, conforme já descrito anteriormente.

D-) Ativação dos módulos *probes*

Os módulos *probes* são ativados. Para cada agente *probe* posterior deve ser ativado o método de descoberta de endereços IP ativos.

E-) Ativação do módulo injetor

O módulo injetor é ativado após o recebimento dos parâmetros referentes as tuplas que serão analisadas.

5.7.2 ETAPA DE TESTE

O módulo injetor obtém do controlador a lista de tuplas para análise SYN *Flood Resistance*. Posteriormente envia repetidamente uma seqüência de segmentos SYN com endereço de origem falsificado ou de uma máquina que não possa responder ao SYN/ACK ou RST para o *firewall*. Simultaneamente são enviados seqüencialmente 3 segmentos TCP com o flag SYN ativo e com endereço IP que não estejam sendo filtrados pelo *firewall*.

5.7.3 ETAPA DE ENCERRAMENTO

A etapa de encerramento do teste consiste na coleta dos registros efetuados pelos dos agentes incluindo os registros gerados pelos módulos injetores e *probes*. Após esta coleta, os registros são enviados ao controlador.

5.7.4 Etapa de Análise

Os segmentos enviados devem ser registrados pelo módulo *probe* da sub-rede anterior e posteriormente pelos módulos *probes* da sub-rede posterior.

Após o recebimento dos arquivos contendo os registros dos módulos injetores e *probes*, o controlador efetua a comparação de todos de registros recebidos.

Se na exibição dos resultados, forem observados os registros dos 3 segmentos TCP enviados, pode ser um indicativo de que o *firewall* não sofreu exaustão de recursos ou degradação de suas funcionalidades mediante condição de ataque.

Se os pacotes não forem observados, indica que o *firewall* descartou os pacotes, indicando que não foi capaz de tratar os segmentos válidos recebidos sob uma situação de ataque, ou seja, sofreu exaustão de recursos.

5.7.5 Etapa de Apresentação dos Resultados

A exibição dos resultados pelo controlador ocorre após a comparação e análise dos registros recebidos dos módulos injetores e *probes*, resultando em um registro das diferenças entre os registros enviados pelos módulos injetores e *probes*.

5.8 Considerações dos Métodos Para Análise

Conforme pôde ser observado, cada método de análise possui suas peculiaridades. Nos casos de análises descritos, é possível notar que os procedimentos a serem seguidos nas etapas iniciais diferem ligeiramente devido às características intrínsecas de cada análise ou protocolo.

Em ambos os casos de análises UDP e TCP, houve a necessidade das etapas de calibração. Também foi possível notar diferenças no caso do protocolo ser orientado à conexão em comparação ao protocolo não orientado

à conexão, existe a necessidade de considerações diferentes para os dois casos. Isso foi mostrado nas tabelas de estados utilizadas para cada protocolo analisado.

CAPÍTULO 6

IMPLEMENTAÇÃO E TESTES

Neste capítulo são descritos os procedimentos adotados para a escolha das ferramentas de software utilizadas para compor a arquitetura do sistema *WHATWALL*, apresentada no Capítulo 5. Também aborda aspectos da implementação de um protótipo do sistema. Essa implementação tem por objetivo validar o modelo proposto. Inicialmente é apresentado o escopo da prova de conceito, depois as ferramentas selecionadas para compor os módulos que fazem parte do sistema e posteriormente a descrição dos testes realizados para avaliação de prova de conceito do sistema proposto.

6.1 ESCOPO DA PROVA DE CONCEITO

A abrangência da prova de conceito está em verificar o funcionamento do método e do sistema proposto. Não se pretende efetuar todas as possibilidades de testes, mas sim alguns testes que permitam analisar o funcionamento do método, possibilitando avaliar se o sistema funciona de acordo com as expectativas.

Os testes selecionados para prova de conceito, cujos resultados serão descritos nas próximas seções foram:

- Análise de filtragem UDP;

- Análise de filtragem TCP;
- Análise de resistência a SYN *Flood*;
- Análise SYN *Flood Protection*;

6.2 SELEÇÃO DE FERRAMENTAS

Diversas ferramentas de código aberto encontradas na Internet e, outras obtidas diretamente dos autores dos trabalhos relacionados no Capítulo 4 foram testadas e analisadas no decorrer deste trabalho, com o intuito de serem adaptadas para reproduzir o protótipo da arquitetura do sistema *WHATWALL*, descrito no capítulo anterior.

Após os testes destas ferramentas, foi constatado, que nenhuma delas possuía completa aderência às funcionalidades desejadas para a arquitetura do sistema *WHATWALL*, de modo a satisfazer completamente a reprodução do protótipo da arquitetura almejada, principalmente por apresentarem limitações referentes a características desejadas no sistema proposto, tais como:

- Ausência de flexibilidade para preenchimento dos campos dos cabeçalhos dos protocolos utilizados;
- Limitação da quantidade de pacotes enviados em cada análise;
- Configuração de intervalos de endereços de rede e portas de origem e destino;
- Tempo gasto no envio de pacotes;
- Controle de tempo entre envio de pacotes;

A flexibilidade para preenchimento dos campos dos cabeçalhos dos protocolos utilizados em cada teste é importante, pois possibilita a criação de pacotes que atendam as características do teste a ser efetuado.

A maioria das ferramentas analisadas limitavam a quantidade de pacotes enviados e, não possuíam controle de tempo de intervalo entre envio de pacotes. Estas flexibilidades também são de extrema importância, pois o intervalo de tempo entre envio de pacotes pode variar de acordo com o

ambiente de testes, ainda na fase de calibração.

A possibilidade para configuração de intervalos de endereços IP e portas, tanto de origem quanto de destino, é importante, pois na maioria dos casos de testes se deseja testar mais de um endereço IP ou porta de origem e destino simultaneamente.

A informação do tempo gasto no envio de pacotes é necessária principalmente para a fase de calibração e também para uma estimativa de tempo gasto para a realização de cada teste.

Pelo motivo da arquitetura ser modularizada, o que favoreceu a implementação, optou-se então, por integrar algumas destas ferramentas analisadas que mais se aproximassem dos objetivos do trabalho de modo a atender as funcionalidades para implementação do protótipo, formando um sistema composto por um conjunto de ferramentas.

Foram necessárias adaptações para que as ferramentas pudessem satisfazer as funcionalidades do sistema *WHATWALL* e serem utilizadas de forma complementar, ou seja, o resultado de uma ferramenta pudesse ser utilizado como entrada de dados para outras ferramentas.

6.3 IMPLEMENTAÇÃO

Algumas considerações foram feitas antes da implementação. Foi constatado, por testes efetuados, que o formato mais conveniente para os registros que seriam utilizados como base para comparação e análise dos resultados deveria ser em formato de texto, facilitando a integração e compatibilidade entre os diversos módulos, e ainda com outras ferramentas.

Por se tratar de uma prova de conceito optou-se pela utilização de *scripts* em linguagens *Bash* e *Perl* para casos onde é necessária a entrada de dados e comparação de resultados, devido a sua praticidade.

Os *scripts* permitiram a integração das novas funcionalidades às ferramentas utilizadas, a integração das ferramentas entre si e ainda, a concatenação dos diversos registros gerados para obtenção dos resultados.

6.3.1 CONTROLADOR

Para desempenhar as funcionalidades exigidas para o módulo controlador, não foi encontrada nenhuma ferramenta que pudesse ser modificada ou adaptada para atender as características desejadas.

O controlador foi desenvolvido em linguagem C para a plataforma GNU/Linux, em modo de linha de comando.

Através do controlador é possível fazer o gerenciamento das várias partes que compõe o sistema. Esse módulo permite a chamada do módulo injetor para injeção de pacotes: *Ethernet*, ARP, IP, UDP, TCP e ICMP.

O controlador foi desenvolvido de modo a permitir o máximo de mobilidade e praticidade de uso ao usuário do sistema. Através de seu *help*, é possível o usuário obter informações de utilização, sintaxe de parâmetros e exemplos de utilização.

Todos os campos devem ser preenchidos em decimal, com exceção de alguns campos como o de *checksum* e *payload* que devem ser em hexadecimal.

Em certos casos são efetuadas verificações de consistências de dados, por exemplo: o usuário não pode entrar com um valor para o número de porta acima de 65535, os campos para entrada dos octetos de endereços IP não aceitam valores acima de 255.

6.3.2 MÓDULO INJETOR

O usuário interage com o módulo injetor através do controlador. O módulo injetor conforme descrito no capítulo anterior possui várias funções na arquitetura do sistema, principalmente, injeção dos pacotes com as características das tuplas para cada teste, de uma sub-rede anterior para uma sub-rede posterior.

A ferramenta selecionada e utilizada para desempenhar a função de módulo injetor foi a Nemesis (NATHAN, 2005). Originalmente esta ferramenta, permite criar e enviar um único pacote de cada vez e suporta os protocolos: *Ethernet*, ARP, ICMP, UDP, IP e TCP.

Devido às limitações já descritas, foi necessário o desenvolvimento de novos programas para adaptar novas funcionalidades e assim atender as funcionalidades do módulo injetor.

Na ferramenta Nemesis originalmente, os valores a serem passados através da linha de comando para preenchimento de alguns campos do protocolo no momento da criação do pacote são fixos, não sendo possível estabelecer faixas de valores para estes campos como: endereço IP de origem, porta de origem, endereço IP de destino, porta de destino. Dependendo da análise a ser efetuada, existe a necessidade que estes parâmetros sejam variáveis (intervalos de valores) como, por exemplo, intervalo de portas ou de endereços de rede.

As principais funcionalidades agregadas à ferramenta Nemesis foram:

- Possibilidade de configuração para intervalos de valores: adequação para que seja possível informar à ferramenta pela linha de comandos intervalos de valores para portas (origem e destino) e endereços de rede (origem destino).

- Modificação na quantidade de pacotes enviados de cada vez: antes a ferramenta era limitada a enviar um pacote de cada vez, após a adaptação a quantidade de pacotes passou a ser variável, possibilitando enviar quantos pacotes forem necessários para a os diferentes testes a serem efetuados.

- Tempo de envio de pacotes: após as adaptações também é possível conhecer o tempo gasto para o envio dos pacotes em cada teste efetuado, funcionalidade que não existia antes da modificação da ferramenta.

Em certos campos com exceção dos citados no item 6.3.1, não é efetuado nenhum tipo de consistência de dados. Essa possibilidade permite ao usuário criar diversos tipos de pacotes, de acordos com as necessidades dos testes a serem realizados. Isso permite ao usuário efetuar diferentes

testes para a exploração de vulnerabilidades em sistemas e protocolos.

Todos os pacotes criados podem ser salvos no formato da Libpcap (uma biblioteca padrão para captura de pacotes), permitindo uma compatibilidade com outras ferramentas, facilitando assim a integração do sistema.

6.3.3 MÓDULO *PROBE*

Para a função de módulo *probe*, entre as diversas ferramentas analisadas e que merecem destaque é o Tcpdump (TCPDUMP, 2005) e o Tethereal (TETHEREAL, 2005), que operam de formas semelhantes e utilizam as mesmas bibliotecas para captura de pacotes. Porém o Tcpdump mostrou-se menos amigável no aspecto de configuração de filtros de pacotes específicos para serem utilizados com os *scripts* desenvolvidos, característica importante e desejável para atender as funcionalidades do módulo *probe*, principalmente pelo motivo da existência de uma elevada quantidade de tráfego de fundo existente nos testes.

Assim a ferramenta selecionada foi a Tethereal. Esta ferramenta apresentou-se satisfatória, para a função de observar o tráfego na rede. Também utiliza a biblioteca padrão (*Libpcap*), isto facilita a integração com outras ferramentas do sistema. Também permite a configuração ou adaptação de filtros para capturar pacotes específicos em uma determinada análise, sendo esta característica de extrema importância quando se analisa um ambiente real onde existe tráfego de fundo, possibilitando a minimização da análise apenas a pacotes relevantes.

6.4 TESTES IMPLEMENTADOS E RESULTADOS

Os testes implementados possibilitaram obter informações que permitiram avaliar a viabilidade e funcionalidades da técnica e da arquitetura do sistema proposto.

As etapas de métodos foram omitidas, pois já foram descritas no capítulo anterior, repetidas vezes.

6.4.1 AMBIENTE DE TESTES

Para efetuar os testes de prova de conceito foi montado um ambiente de testes, representado na figura 6.2.

Os *firewalls* utilizados nos testes foram: Netfilter, Iptables e *Firewall*1.

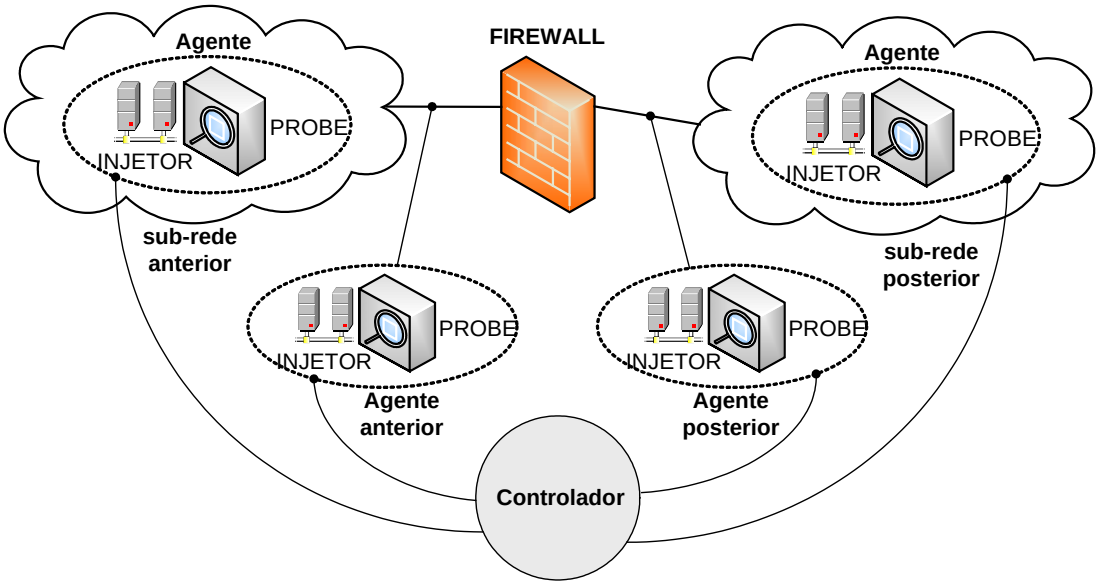

Figura 6.2 – Ambiente Utilizado Para Testes de Prova de Conceito

Os recursos utilizados no ambiente de teste foram:

- Hardware:
  - 06 servidores Intel Xeon *dual processor* 2.4 GHz, 1.2 GB de memória RAM, 03 interfaces de rede Gigabit *ethernet*.

- Softwares:
  - *Check Point Firewall*-1.
  - Flex, bison, gcc, make, libc, libc-dev, nemesis, ethereal.
  - Bibliotecas: libpcap-0.7.2, libnet-1.1.2.1, libdnet-1.8, libc e libdnet.
  - Netfilter.

- Iptables.

O sistema *WHATWALL* versão 1.0 foi escrito para ser executado no ambiente Linux, distribuição Debian 3.1 com o kernel 2.4.27 e Iptables. O programa é implementado em linguagem C e alguns *scripts* na linguagem *bash* e *Perl*. É utilizado o compilador GNU C gcc versão 2.95.4 e o GNU *debugger* gdb 5.3.

Para a entrada dos parâmetros referentes às características do teste, como lista de tuplas e informações de topologia da rede são utilizados os *scripts bash* e *Perl* que interagem com a ferramenta Nemesis.

6.4.2 ANÁLISE DE FILTRAGEM UDP

O objetivo desta análise é testar e verificar o funcionamento do método e da implementação para determinação dos datagramas UDP filtrados pelo *firewall*, ou seja, verificar possíveis portas UDP abertas e fechadas no *firewall*.

Para esta análise, foi selecionada uma lista de tuplas de uma rede classe C no intervalo 192.168.25.1–254. Para as portas de origem e destino foi selecionado o intervalo de 1 a 65535. Isto representa 2^{25} pacotes (2^{8} endereços IP de destino, 2^{16} portas UDP).

Foram seguidas todas as etapas descritas para o método UDP, já apresentadas no Capítulo 5.

O sentido do teste foi da sub-rede anterior para sub-rede posterior, ou seja, os pacotes foram injetados a partir do agente UDP na sub-rede anterior.

Foi posicionado um agente UDP na sub-rede anterior e um agente UDP na sub-rede posterior.

Devido ao fato da taxa de transmissão de pacotes variar de acordo com o ambiente testado, efetuou-se a calibração antes da execução dos testes UDP.

Foi selecionada uma lista de tuplas pertencentes ao intervalo anterior e não filtradas pelo *firewall*.

A identificação dos pacotes pelos módulos *probes* é efetuada pelo

número do campo de identificação do cabeçalho do protocolo IP.

Os pacotes UDP gerados para os testes de calibração não continham dados, sendo compostos apenas pelo cabeçalho IP e UDP, com um tamanho total de 28 bytes. As portas UDP de origem e destino, endereços IP de origem e destino permaneceram constantes variando o número de pacotes enviados, conforme a tabela 6.1.

A Tabela 6.1 apresenta os resultados dos testes de calibração. Neste teste foi obtida uma taxa média de transmissão (média aritmética) de 576,5 pacotes/s com desvio padrão da taxa de transmissão de 0,81 pacotes/s, o que corresponde a aproximadamente 0,14 % da taxa média de transmissão.

Tabela 6.1 - Resultados dos testes de calibração

PACOTES TRANSMITIDOS	DURAÇÃO (S)	TAXA DE TRANSMISSÃO (PACOTES/S)
10.000	17,38	575,37
50.000	86,68	576,83
100.000	173,17	577,46
200.000	346,55	577,11
300.000	521,06	575,74

Pelas taxas de transmissões obtidas, é possível constatar que as mesmas permaneceram praticamente constantes mediante as variações da quantidade de pacotes transmitidos.

Não foram observadas pelos módulos *probes* a ocorrência de perdas de pacotes devido à exaustão de recursos, não sendo necessário impor intervalo de tempo entre o envio dos pacotes.

Após a finalização da etapa de calibração, é passada uma lista de tuplas que serão analisadas para o injetor, observando que estas tuplas não estavam sendo filtradas pelo *firewall*.

O injetor de acordo com as características recebidas das listas de tuplas envia os pacotes em direção a uma sub-rede posterior. O resultado esperado era que o *probe* anterior registrasse a passagem dos datagramas

UDP enviados e posteriormente o *probe* posterior também efetuasse o registro destes datagramas.

Os registros dos pacotes enviados e observados pelos módulos *probes* anterior e posterior foram salvos e enviados ao controlador.

O controlador efetua a comparação dos registros recebidos, resultando em um arquivo “vazio”, ou seja, não existe diferença entre os arquivos do *probe* anterior e posterior.

Os resultados obtidos constataram que os pacotes enviados pelo módulo injetor do agente UDP na sub-rede anterior passaram pelo *firewall* e foram registrados pelo *probe* posterior, ratificando a inexistência de regras no *firewall* bloqueando os pacotes enviados.

Outro teste foi realizado analogamente ao teste anterior, exceto que nesta análise também foram enviadas tuplas que previamente foram configuradas para serem bloqueadas no *firewall*.

Posteriormente, o resultado obtido foi a diferença dos registros dos pacotes enviados e recebidos indicando quais tuplas o *firewall* não permitiu a passagem, bem como as mensagens ICMP *port unreachable* e ICMP *host unreachable*, no *probe* anterior, indicando portas fechadas ou computadores que não puderam ser alcançados, concluindo-se que as respectivas portas estavam fechadas e os endereços IP estavam bloqueados.

Os testes também serviram para comprovar que a taxa de envio utilizada na calibração não ocasionou perda de pacotes.

Os firewalls utilizados neste teste se comportaram de forma semelhante.

6.4.3 Análise de Filtragem TCP

O objetivo desta análise é testar e verificar o funcionamento do método e da implementação para determinação dos datagramas TCP filtrados pelo *firewall*, ou seja, verificar possíveis portas TCP abertas e fechadas no *firewall*. O teste foi aplicado em dois *firewalls* distintos, sendo que um deles as regras de filtragem operam no modo *statefull* e no outro as regras de filtragem

operam no modo *stateless*.

Para esta análise, foi selecionada uma lista de tuplas de uma rede classe C no intervalo 192.168.25.1–254. Para as portas de origem e destino foi selecionado o intervalo de 1 a 65535.

Na etapa de calibração não foram observadas perdas de pacotes, não sendo necessário intervalo de tempo entre envios de pacotes.

No primeiro teste foram selecionadas tuplas do intervalo mencionado anteriormente, para as quais existiam regras de filtragem configuradas no *firewall*. Os resultados obtidos, realmente serviram para comprovar que as tuplas estavam sendo bloqueadas, sem retorno de mensagens ICMP.

No segundo teste foram enviadas tuplas que estavam sendo bloqueadas juntamente com tuplas que não estavam sendo bloqueadas. Neste teste os *firewalls* foram configurados para retornar mensagens ICMP *port unreachable* e ICMP *host unreachable*. Após o encerramento dos testes o controlador apresentou conforme era esperado os resultados das tuplas para as quais o *firewall* não permitiu a passagem e ainda as mensagens ICMP retornadas.

Outro teste realizado foi o envio de datagramas que não eram os primeiros, simulando o envio de pacotes de uma conexão já existente, observando que as tuplas utilizadas não estavam sendo filtradas pelos *firewalls*.

O *firewall* que implementa regras *statefull* não permitiu a passagem dos datagramas enviados, enquanto que o *firewall* que implementava regras *stateless* permitiu a passagem dos datagramas enviados com as mesmas características, comprovando assim a funcionalidade da técnica.

6.4.4 ANÁLISE SYN *FLOOD PROTECTION*

Pelo módulo Injetor TCP foram enviados seqüencialmente 3 segmentos TCP com o *flag* SYN ativo, previamente sabido que não estavam sendo bloqueados pelo *firewall*, para o agente *probe* na sub-rede posterior. Inicialmente os segmentos foram registrados pelo módulo *probe* da sub-rede

anterior e em seguida pelo módulo *probe* da sub-rede posterior.

Posteriormente foram observados 3 segmentos SYN/ACK, no *probe* da sub-rede anterior, ou seja, agente posterior respondeu aos 3 segmentos TCP SYN. Isto indica que o *firewall* não estava implementando mecanismos de defesas contra ataque de SYN *Flood*.

O procedimento foi repetido para um segundo *firewall*, mantida a mesma arquitetura da rede.

Os segmentos foram registrados pelo módulo *probe* anterior e posteriormente também foram registrados 3 segmentos SYN/ACK no *probe* anterior e nada foi registrado no *probe* posterior. Ou seja, ao invés de permitir a passagem dos segmentos injetados para o agente *probe* posterior, o *firewall* intermediou o estabelecimento da conexão, indicando que o mesmo implementa mecanismos de proteção contra ataques de SYN *Flood*.

6.4.5 ANÁLISE SYN *FLOOD RESISTANCE*

A seguir será descrito o teste efetuado para análise da resistência do *firewall* a ataques de inundação SYN *Flood* e verificar se o mesmo apresenta algum mecanismo de defesa contra esse tipo de ataque.

Antes do início dos testes de análise foi efetuada a fase de calibração, seguindo os procedimentos já descritos no Capítulo 5. A taxa utilizada para este teste foi 576 pacotes/s, coincidentemente a mesma obtida para o teste UDP.

Foi enviado repetidamente pelo módulo Injetor TCP uma quantidade aleatória de 10^9 segmentos SYN cujo endereços de origem não estavam respondendo propositalmente, com a intenção de que a várias aberturas de conexões permanecessem pendentes no *kernel* do *firewall*. Em seguida foram enviados seqüencialmente 3 segmentos TCP com o *flag* SYN ativo e com endereço IP e portas válidas, que não estavam sendo filtradas pelo *firewall*.

Foi observado pelos *probes* na sub-rede posterior o recebimento dos pacotes enviados, ou seja, mesmo sob condição de uma inundação por

pacotes inválidos o *firewall* utilizado na análise não sofreu exaustão de recursos e foi capaz de estabelecer conexões válidas.

Pode-se concluir que o *firewall* implementa mecanismos de resistência a ataques de SYN *Flood*.

O mesmo teste foi repetido, duplicando-se a quantidade de segmentos SYN (10^{18}), acreditando ser suficiente para inundar o *firewall*. O resultado obtido foi o mesmo, ratificando o que foi obtido no teste anterior.

Para testar um outro *firewall*, mantivemos as mesmas condições dos testes anteriores. Neste teste foi contatado que os 3 datagramas enviados não foram registrados pelo *probe* posterior, indicando que o *firewall* não foi capaz de distinguir os datagramas válidos dos inválidos, ou seja indicando que o mesmo não possui resistência a ataque de SYN *Flood*.

6.5 CONSIDERAÇÕES

Os testes serviram para comprovar o funcionamento do sistema e a viabilidade da técnica.

Os resultados obtidos nos testes ficaram dentro das expectativas. A partir destes resultados é possível partir para testes mais complexos. Algumas características adicionais já estão especificadas como a criação de bibliotecas de testes, porém não implementadas.

CAPÍTULO 7

CONCLUSÃO

O principal objetivo deste trabalho consiste em um estudo para verificar a viabilidade da utilização de técnicas de análise ativa, por injeção de pacotes e observação dos resultados, para descoberta do comportamento de *firewalls* da camada de rede.

Para comprovação da viabilidade da técnica, e verificação de seu correto funcionamento foi implementado um protótipo de arquitetura.

7.1 VIABILIDADE DA TÉCNICA PROPOSTA

A técnica apresentou-se viável, desde que sejam adotadas técnicas para a minimização do espaço de análise. Para isto, é necessário um conhecimento prévio de informações a respeito das características da arquitetura da rede onde o *firewall* está instalado.

Existem diversos desafios para o uso efetivo da técnica de análise ativa, principalmente o problema da explosão combinatória na análise de filtragem de datagramas IP.

7.2 CONTRIBUIÇÕES DO TRABALHO

A principal contribuição deste trabalho é a possibilidade desta técnica poder ser utilizada para analisar diversos tipos de *firewalls* de rede sem que seja necessária a implementação de módulos adaptadores para *firewalls* específicos.

Esta característica de ser genérico, tida como importante para o sistema desenvolvido, é o principal diferencial em relação aos demais sistemas de análise de *firewalls*, os quais são desenvolvidos para analisar uma solução específica.

Também pode ser destacada a possibilidade de testes do próprio *firewall*, verificando se o mesmo possui determinadas proteções, como por exemplo, proteção contra ataques de exaustão de recursos, característica esta, inexistente na maioria das ferramentas analisadas e nos trabalhos relacionados.

Ainda podemos ressaltar que o modelo de arquitetura proposto é modularizado, consistindo em um sistema expansível, possibilitando ainda aos usuários a capacidade de desenvolver seus próprios testes, não sendo obrigados a depender de bibliotecas de testes fornecidos juntamente com o sistema.

7.4 CONTINUIDADE DO TRABALHO

Entende-se que este trabalho seja o ponto de partida para pesquisas mais abrangentes a respeito de ferramentas para análise de comportamento de *firewalls* e estudo de técnicas alternativas para o problema da explosão combinatória. Existem alguns trabalhos que podem ser desencadeados a partir desta pesquisa e serem citados como continuidade a este trabalho. Conforme pode ser observado na figura 7.1, alguns módulos externos ao sistema auxiliariam os usuários em outras tarefas como, por exemplo, o módulo de comparação, que recebe os resultados do sistema *WHATWALL* e os compara com a política de segurança definida pela corporação.

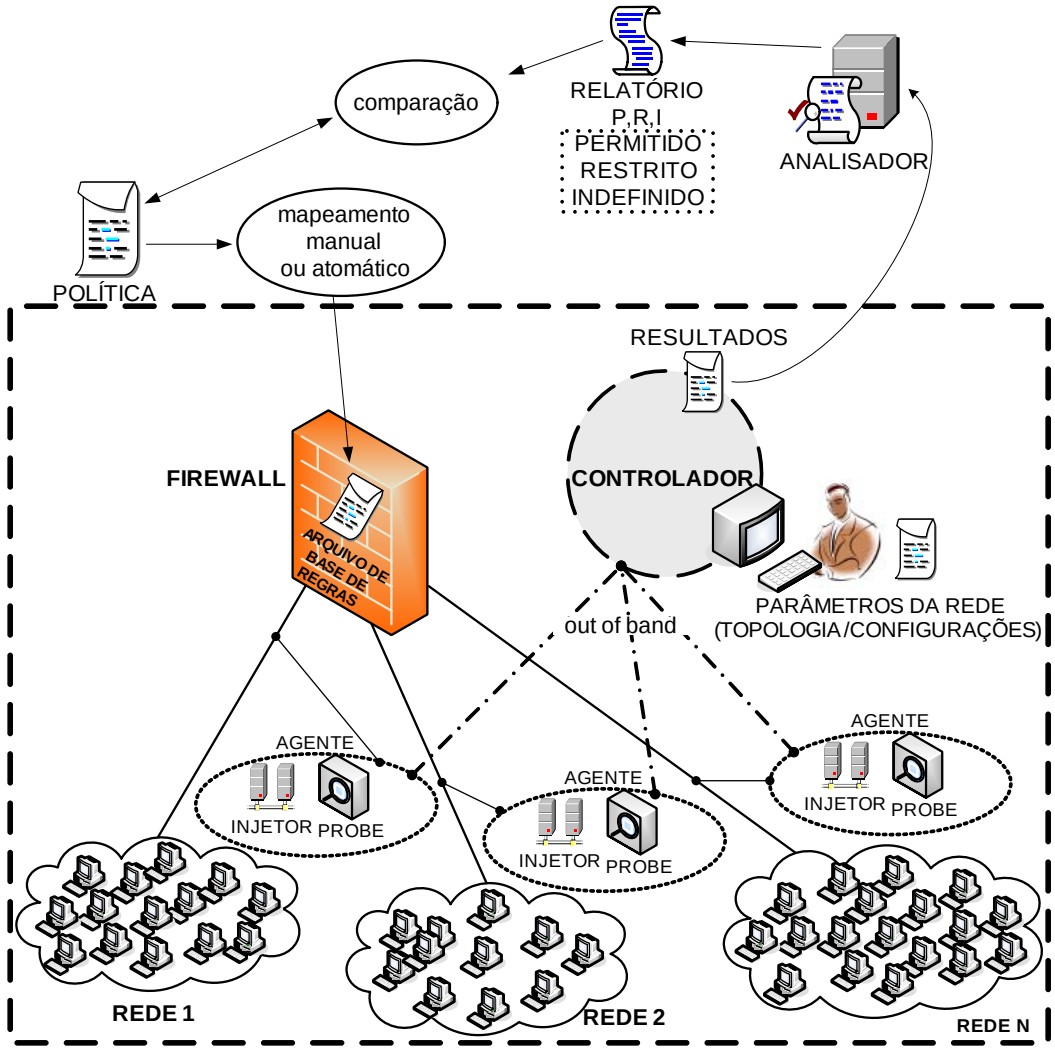

Figura 7.1 – Módulos de Extensão do Sistema *WHATWALL*

Os resultados obtidos desta verificação poderão ainda ser utilizados para confrontar a política de segurança definida e as regras aplicadas atualmente em funcionamento no *firewall*, permitindo identificar se as regras em vigor seguem as especificações previamente estabelecidas na política de segurança, possibilitando identificar a existência de possíveis brechas ou erros de configuração, ou ainda, auxiliar na definição de regras que irão compor as bases-de-regras.

7.5 CONSIDERAÇÕES FINAIS

Conforme foi visto, existem trabalhos relacionados que auxiliam a elaboração de regras de *firewall* a partir de uma política de *firewall* definida. Também existem trabalhos de pesquisa que tratam de geração automática de regras para *firewalls*. Mas nenhum dos trabalhos foca especificamente no estudo do comportamento de *firewalls* com a utilização de injeção de pacotes e observação dos resultados.

Este estudo ofereceu a proposta de um método complementar aos trabalhos existentes para análise de comportamento de *firewalls* baseado na injeção de pacotes e observação dos resultados. Assim, serviu para comprovar que a técnica é viável, e auxilia a mitigar algumas dificuldades encontradas na análise do comportamento efetivo das funcionalidades de proteção ativas em um *firewall*, permitindo testar *firewalls* da camada de rede, independentemente do fabricante ou modelo do equipamento.

A partir dos resultados obtidos neste trabalho, e das proposições apresentas no item anterior, modificações podem ser efetuadas no sentido de aprimorar a implementação, visto a importância e necessidade de ferramentas que apresentem tais características e funcionalidades.

REFERÊNCIAS BIBLIOGRÁFICAS

ADI, K. et al. **Evaluation of current research in firewall analysis.** Disponível em: <http://lotos.site.uottawa.ca/ftp/pub/Lotos/TechRep/Department of National Defense.2003>. Acesso em: 3 ago 2003.

AL-SHAER, E.S.; HAMED, H.H. Firewall policy advisor for anomaly discovery and rule editing. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 8., 2003, Colorado Springs. **Integrated network management VIII:** managing it all. Boston: Kluwer Academic Publishers, 2003. p. 17-30.

AL-SHAER, E.S.; HAMED, H.H. Discovery of policy anomalies in distributed firewalls. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 23., 2004, Hong Kong. **Proceedings:** IEEE INFOCOM 2004. Hong Kong: IEEE Computer Society, 2004a. p. 2605-2616.

AL-SHAER, E.S.; HAMED, H.H. Modeling and management of firewall policies. **eTransactions on Network and Service Management**, New York, v. 1, n. 1, Apr. 2004b. Disponível em: <http://www.comsoc.org/livepubs/etn/public/2004/apr/index.html>. Acesso em: 2 set 2004.

AL-TAWIL, K.; AL-KALTHAM, I.A. Evaluation and testing of internet firewalls. **International Journal of Network Management,** New York, v. 9, n. 3, p. 135-149, May/June 1999.

BARTAL, Y. et al. Firmato: a novel firewall management toolkit. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1999, Oakland. **Proceedings**. Los Alamitos: IEEE Computer Society, 1999. p. 17-31.

BELLOVIN, S. M. Security problems in the tcp/ip protocol suite. **Computer Communication Review,** New York, v. 19, n. 2, p. 32-48, Apr. 1989.

BELLOVIN, S. M. Using the domain name system for break-ins. In: USENIX UNIX SECURITY SYMPOSIUM, 5., 1995, Salt Lake City, Utah. **Proceedings..** Berkeley, CA: USENIX Assoc, 1995. p. 199-208.

BELLOVIN, S. M. A look back at "Security problems in the TCP/IP protocol suite". In: ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 20., 2004, Tucson. **Proceedings:** ACSAC 20. Los Alamitos: IEEE, 2004. p. 229-249.

BERNSTEIN, D. J. **Syn cookies**. Disponível em : <http://cr.yp.to/syncookies.html>. Acesso em: 22 ago 2004.

BURCK, H.; SONG, D. **A security study of the Internet:** an analysis of firewall behavior and anonymous DNS. Disponível em: <http://reports-archive.adm.cs.cmu.edu/cs2004.html>. Acesso em: 5 ago 2004.

CERT Coordination Center. **CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks.** Rev. 2000. Disponível em: <http://www.cert.org/advisories/CA-1996-21.html>. Acesso em: 20 jan 2005.

CERT Computer Emergency Response Team. **CERT FTP Archive.** Disponível em: <ftp://ftp.cert.org/pub/cert_advisories/>. Acesso em: 22 dez 2004.

CHAMBERS, C.; DOLSKE, J.; IYER, J. **TCP/IP security**. Disponível em: <http://www.linuxsecurity.com/resource_files/documentation/tcpip-security.html>. Acesso em: 22 dez 2005.

CHAPMAN, D. B. Network (in) security through IP packet filtering. In: UNIX SECURITY SYMPOSIUM, 3., 1992, Baltimore, MD. **Proceedings.** Berkeley, CA: USENIX Assoc, 1992. p. 63-76.

CHECK POINT Software Technologies Ltd. **Check point firewall-1:** guide. 2001a. Disponível em: <http://www.checkpoint.com/support/technical/documents/docs-5.0/firewall_ng_sp0.pdf>. Acesso em: 02 jan 2004.

CHECK POINT Software Technologies Ltd. **Check point firewall-1 architecture and administration version 4.0**. . 2001b. Disponível em: <http://www.checkpoint.com/support/technical/online_ug/firewall-14.0/miscsec.htm#4005>. Acesso em: 22 mar 2005.

CHECK POINT Software Technologies Ltd. **Stateful inspection firewall technology.** 1998. (Tech Note). Disponível em: <http://www.checkpoint.com/support/technical/online_ug/sift.htm>. Acesso em: 03 jan 2005.

CHESWICK, W.R.; BELOVIN, S.M.; RUBIN, A.D. **Firewalls e segurança na Internet:** repelindo o hacker ardiloso. 2.ed. Porto Alegre: Bookman, 2005.

CISCO SYSTEMS. **IOS firewall**. [2002]. Disponível em: <http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_white_paper0900aecd8029d0a6.shtml>. Acesso em: 10 out 2004.

COMER, D. E. **Internetworking with TCP/IP:** principles, protocols and architecture. 5.ed. Englewood Cliffs: Prentice-Hall, 2005. v. 1.

CRONJE, G. Choosing the best firewall. In: ____. **GIAC level one security essentials practical assignment - version 1.2b.** Bethesda: SANS Institute, 2003. Disponível em: <http://www.securitytechnet.com/resource/security/firewall/951.pdf>. Acesso em: 15 jul. 2004.

EDDY, W. TCP SYN Flooding Attacks and Common Mitigations. **Draft-Eddy-Syn-Flood-02**. Verizon Federal Network Systems, 2006. Disponível em: <http://www.ietf.org/internet-drafts/draft-eddy-syn-flood-02.txt>. Acesso em: 27 mai 2006.

ERONEN, P.; ZITTING, J. An expert system for analyzing firewall rules. In: NORDIC WORKSHOP ON SECURE IT SYSTEMS, 6., 2001, Copenhagen. **Proceedings:** NordSec 2001. Helsinki: Nixu, 2001. p. 100–107. (Technical University of Denmark. Technical Report, IMM-TR-2001-14). Disponível em: <http:\\citeseer.ist.psu.edu/eronen01expert.html>. Acesso em: 3 Jan. 2003.

FARROW, R. **Sequence number attacks**. Disponível em: <http://www.networkcomputing.com/unixworld/security/001.txt.html>. Acesso em: 7 Jun. 2003.

FARROW, R. **Distributed denial of service attacks**. Disponível em: <http://www.networkmagazine.com/shared/printableArticle.jhtml?articleID=8702755>. Acesso em: 01 Jun. 2005.

FLYNT, C. **Validator:** an agent-based firewall validation application. Disponível em: <http://www.tcl.tk/community/tcl2004/Tcl2003papers/flynt.pdf>. Acesso em: 7 jan. 2004.

GODDARD, S.; KIECKHAFER, R.; ZHANG, Y. An unavailability analysis of firewall sandwich configurations. In: IEEE SYMPOSIUM ON HIGH ASSURANCE SYSTEMS ENGINEERING, 6., 2001, Boca Raton. **Proceedings:** HASE'01. Los Alamitos: IEEE, 2001. p. 139-148.

GONT, F. **ICMP attacks against TCP.** Disponível em: <http://www.gont.com.ar/drafts/draft-gont-tcpm-icmp-attacks-05.txt>. Acesso em: 24 out. 2005.

GOUDA, G.M.; LIU, A.X. Firewall design: consistency, completeness and compactness. In: IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS, 24., 2004, Tokyo. **Proceedings.** Los Alamitos: IEEE Computer Society, 2004. p. 320-327.

GUTMAN, J.D. Filtering postures: local enforcement for global policies. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1997, Oakland. **Proceedings.** Los Alamitos: IEEE Computer Society, 1997. p. 120-129.

HATCH, B.; LEE, J.; KURTZ, G. **Segurança contra hackers.** São Paulo: Futura, 2003.

HEDRICK, C. Routing Information Protocol. **RFC 1058**. Rutgers University, jun 1988. Disponível em: <http://www.faqs.org/rfcs/rfc1058.html>. Acesso em: 08 jan 1988.

JONCHERAY, L. **A simple active attack agaist TCP**. In: USENIX UNIX SECURITY SYMPOSIUM, 5., 1995, Salt Lake City. **Proceedings.** Berkeley: USENIX Assoc., 1995. p. 7-19.

JÜRJENS, J.; WIMMEL, G. Specification-based testing of firewalls (extended abstract/regular talk). In: INTERNATIONAL ANDREI ERSHOV MEMORIAL CONFERENCE, 4., 2001, Novosibirsk, Russia. **Perspectives of system informatics:** revised papers. Berlin: Springer-Verlag, 2001. p. 308-316. (Lecture Notes In Computer Science, v. 2244).

KAMARA S. et al. Analysis of vulnerabilities in internet firewalls. **Computers and Security,** Amsterdam, v. 22, n. 3, p. 214-232, Apr. 2003.

KRISHNAN, P.; HARTLEY, D. Using model checking to test a firewall: a case study. In: EUROMICRO CONFERENCE, 28., 2002, Dortmund, Germany. **Proceedings.** Los Alamitos: IEEE Computer Society, 2001. p. 284-291.

LEE, J.-S.; JEON J.-C.; YOO K.-Y. A security scheme for protecting security policies in firewall. **ACM SIGOPS Operating Systems Review,** New York, v. 38, n. 2, p. 69-72, Apr. 2004.

LEE, T. K. et. al. Compiling policy descriptions into reconfigurable firewall processors. In: ANNUAL IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES, 11., 2003, Napa. **Proceedings:** FCCM 2003. Los Alamitos: IEEE, 2001. p. 39-50.

LIMA, M. B. **Provisão de serviços inseguros usando filtros de pacotes com estados**. 2000. 128 p. Dissertação (Mestrado) - Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2000.

LIU, A.X.; GOUDA M.G. Diverse firewall design. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2004, Florence, Italy. **Proceedings:** DSN 2004. Piscataway: IEEE Computer Society, 2004. p. 595-604.

LUCENT TECHNOLOGIES. **Managed firewall**. 1998. Disponível em: <http://www.lucent.com/press/0398/980316.nsb.html>. Acesso em 03 jan 2003.

MAYER, A.; WOOL, A.; ZISKIND, E. Fang: a firewall analysis engine In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 2000, Berkeley, CA. **Proceedings:** S&P 2000. Los Alamitos: IEEE Computer Society, 2000. p. 177-187.

MOCKAPETRIS, P. **Domain names -** implementation and specification, STD 13, RFC 1035. USC/Information Sciences Institute, November 1987. Disponível em: <http://www.iana.org/assignments/dns-parameters>. Acesso em: 10 mar. 2004.

MORIS, R. T. A Weakness in the 4.2BSD Unix TCP/IP Software. AT & T Bell Laboratories, February 1985.

NATHAN, J. **NEMESIS 1.4.** 2003. Disponível em: <http://nemesis.sourceforge.net>. Acesso em: 2005.

NEONSURGE, R. P. SYN Floods and SYN Cookies. The Cause and Cure. 1996.

NMAP, **NMAP 2.5 Reference Guide 2005**. Insecure.Com LLC. Disponível em: <http://nemesis.sourceforge.net>. Acesso em: 08 mai. 2005.

NOURELDIEN, N.A.; OSMAN, I.M. On firewalls evaluation criteria. In: TENCON, 2000, Kuala Lumpur, Malaysia. **Proceedings.** Piscataway: IEEE, 2000. v. 3, p. 104–110.

ONE, A. DNS ID hacking. **Phrack Magazine**, Vol. 8, n. 52, p. 21-28, Jan 1998.

PERMPOONTANALARP, Y.; RUJIMETHABHAS, C. A graph theoretic model for hardware-based firewalls networks. In: IEEE CONFERENCE ON NETWORKS, 9., 2001, Bangkok. **Proceedings:** ICON 2001. Los Alamitos: IEEE, 2003. p. 228-233.

PERMPOONTANALARP, Y.; RUJIMETHABHAS, C. A unified methodology for verification and synthesis of firewall configurations. In: INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATIONS SECURITY, 3., 2001, Xian, China. **Proceedings:** ICICS 2001. Berlin: Springer, 2001b. p. 328-339. (Lecture Notes in Computer Science, 2229).

PLUMMER, D.C. **An ethernet address resolution protocol or converting network protocol addresses to 48 bit Ethernet address for transmission on ethernet hardware - RFC 826.** Network Working Group, 1982. Disponível em: <http://www.rfc-editor.org/rfc/rfc826.txt>. Acesso em: 19 mar. 2003.

POSTEL, J. **User datagram protocol - RFC 768.** Marina del Rey: Information Sciences Institute/Univ. Southern California, Aug. 1980. Disponível em: <http://www.faqs.org/rfcs/rfc768.html>. Acesso em: 02 jan 2004.

POSTEL, J. **Transmission control protocol - RFC 793**. Marina del Rey: Information Sciences Institute/Univ. Southern California, Sept. 1981a. Disponível em: <http://www.rfc-editor.org/rfc/rfc793.txt>. Acesso em: 08 jan 2004.

POSTEL, J. **Internet protocol - RFC 791**. Marina del Rey: Information Sciences Institute/Univ. Southern California, Sept. 1981b. Disponível em: <http://www.rfc-editor.org/rfc/rfc791.txt>. Acesso em: 08 jan 2004.

POSTEL, J. **Internet control message protocol** - **RFC 792**. Marina del Rey: Information Sciences Institute/Univ. Southern California, Sept. 1981c. (Network Working Group). Disponível em: <http://www.faqs.org/rfcs/rfc792.html>. Acesso em: 08 jan 2004.

POUW, K. D.; GEUS, P.L. Uma análise das vulnerabilidades dos firewalls. In: WORKSHOP SOBRE ADMINISTRAÇÃO E INTEGRAÇÃO DE SISTEMAS , 1996, Fortaleza. **WAIS´96**. Fortaleza, 1996. Disponível em: <http://www.las.ic.unicamp.br/paulo/wais96/papers-tech/duarte-rev.pdf>. Acesso em: 9 jun. 2003.

RANUN, M.J.; AVOLIO, M.F. A toolkit and methods for internet firewalls. **International Journal of Network Management,** New York, v. 9, n. 3, p. 135-149, May-June 1999.

REKHTER, Y.; LI, T. (Ed.). **A boarder gateway protocol 4(BGP-4) - RFC 1771**. [S.l.]: Internet Engineering Task Force, Mar. 1995. Disponivel em: < ftp://ftp.rfc-editor.org/in-notes/rfc1771.txt> Acesso em: 2 jul. 2003.

SCHUBA, C.L. **Addressing weaknesses in the domain name system protocol.** 1993. Thesis (Master' s) - Purdue University, West Lafayette, 1993.

SCHUBA, C.L.; SPAFFORD, E.H. A reference model for firewall technology. In: ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 13., 1997, San Diego. **Proceedings**. Los Alamitos: IEEE, 1997. p. 133-145.

SILANDER, M. **Denial of service attacks**. Helsinki: Departament of Computer Science, University of Technology, 1999.

STEVENS, W.R. **TCP/IP ilustred.** Reading, Addison-Wesley, 1994. (Addison-Wesley Professional Computing Series, v. 1).

STREBE, M.; PERKINS, C. **Firewalls 24 seven.** São Paulo: Makron Books, 2002.

SUN MICROSYSTEMS. **SunScreen3.1 Lite.** [2002]. Disponível em: <http://www.sun.com/software/securenet/lite/>. Acesso em: 4 jan. 2004.

TCPDUMP 3.9.4. In: TCPDUMP.ORG. 2005. Disponível em: <http://www.tcpdump.org>. Acesso em: out. 2005.

TETHEREAL 0.10.11. In: **tethereal**.1.html. 2005. Disponível em: www.ethereal.com/docs/man-pages/**tethereal**.1.html. Acesso em: out. 2005.

URIBE, T.E.; CHEUNG S. Automatic analysis of firewall and network intrusion detection system configurations. In: ACM WORKSHOP ON FORMAL METHODS IN SECURITY ENGINEERING, 2004, Washington. **Proceedings:** FMSE'04. New York: ACM, 2004. p. 66-74.

VERMA, P.; PRAKASH, A. FACE: a firewall analysis and configuration engine. In: 2005 SYMPOSIUM ON APPLICATIONS AND THE INTERNET, 2005, Trento. **Proceedings:** SAINT 2005. Los Alamitos: IEEE Computer Society, 2005. p. 74-81.

VERWOERD, T.; HUNT, R. Policy and implementation of an adaptive firewall. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 10., 2002, Singapore. **Proceedings:** ICON 2002. Piscataway: IEEE, 2002. p. 434–439.

VIVO, M. et al. **Internet vulnerabilities related to TCP/IP and T/TCP.** Disponível em: < http://portal.acm.org/citation.cfm?id=505754.505760 >. Acesso em: 4 abr. 2004.

VIXIE, P. DNS and BIND security issues. In: USENIX UNIX SECURITY SYMPOSIUM, 5., 1995, Salt Lake City. **Proceedings.** Berkeley: USENIX Assoc., 1995. p. 209-219.

YUE, W.T.; BAGCHI, A. Tuning the quality parameters of a firewall to maximize net benefit. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED COMPUTING, 5., 2003, Kolkata. **Proceedings:** IWDC 2003. Berlin: Springer-Verlag, 2003. p. 321-329. (Lecture Notes In Computer Science, v. 2918).

WOLL, A. Architecting the Lumeta firewall analyzer. In: USENIX SECURITY SYMPOSIUM, 10., 2001, Washington, DC. **Proceedings:** Berkeley: USENIX, 2001. Disponível em: <www.usenix.org/events/sec01/full_papers/wool/wool.pdf>. Acesso em: 14 set. 2004.

WOLL, A. A quantitative study of firewall configuration errors. **Computer,** New York, v. 37, n. 6, p. 62-67, June 2004a.

WOLL, A. The use and usability of direction-based filtering in firewalls. **ACM Transactions on Computer Systems**, Baltimore, v. 22, n. 4, p. 381-420, Nov. 2004b.

ZWICKY, E.D.; COOPER, S.; CHAPMAN, D.B. **Construindo firewalls para a Internet.** Rio de Janeiro: Campus, 2000.