Instituto Superior de Engenharia do Porto – Engenharia Informática 5 º ano

Firewalls & Linux

Tutorial de iptables

António Pedro Freitas Fortuna dos Santos – i950059 – pedro.fortuna@netcabo.pt
Orientador do Projecto: Eng.Pedro Alípio

Agradecimentos

Eng.º Pedro Alípio
Sónia Couto
Eng.º Nuno Pereira
Eng.º Rui Ribeiro
João Carvalho

Índice

1 Introdução

1.1 Introdução

O número de sistemas informáticos interligados através da *Internet* tem crescido exponencialmente.

Para as organizações, as redes de computadores e a *Internet* representam um suporte rápido de comunicação (com os funcionários, com parceiros comerciais, com fornecedores, uma meio de contactar clientes beneficiando dos recentes paradigmas de comunicação personalizados que a Internet permite) possibilitando automatização de processos (bases de dados distribuídas, etc.), evitando o uso de papel, reduzindo custos de impressão, reduzindo custos com outros tipos de tecnologias de comunicação (telefone, fax, telemóvel), burocracias várias, entre outros.
Para uma utilização a nível pessoal, as vantagens incidem mais nas vertentes de aprendizagem, lúdica, compras online e serviços variados.

Interligar sistemas informáticos e partilhar informação pessoal numa rede de computadores significa correr certos riscos. São várias as falhas de segurança existentes nas redes de computadores. A maioria dessas falhas devem-se a *bugs*[1] existentes no *software* instalado nos computadores; deficiências de concepção dos protocolos de comunicações utilizados nas redes locais e *Internet*; e o mais importante, configurações incorrectas do *software* e/ou *hardware*. A falta de informação sobre actualizações dos equipamentos e do *software* (*patches*[2]) por parte dos responsáveis pela administração dos sistemas informáticos, constitui também um factor de risco para segurança desses sistemas.

O risco dessas falhas poderem vir a ser exploradas aumenta significativamente quando esses mesmos computadores e redes estão ligadas à *Internet*, estando desta forma, expostos a um número consideravelmente elevado de utilizadores com conhecimentos técnicos capazes e/ou ferramentas de software apropriadas para a elaboração de intrusões, e pelo facto de ocasionalmente fazermos circular para a *Internet* informação sensível, que mesmo protegida por algoritmos de encriptação, está sujeita a ser capturada por *hackers*[3]

[1] Erro na codificação do programa.
[2] Patches são correcções rápidas aplicadas a programas que corrigem bugs encontrados.
[3] Indivíduo que se dedica a quebrar a segurança de sistemas informáticos

engenhosos. Mesmo a informação encriptada corre o risco de ser interceptada e decifrada.

Outros perigos resultantes da interligação de sistemas informáticos são a proliferação cada vez mais rápida de *virus* e *worms*[4].

Nas organizações, as redes são normalmente um conjunto de *LANs*[5] interligadas (distribuídas por várias localizações geográficas), sendo constituída por estações de trabalho e por servidores destinados a servir utilizadores internos e/ou externos à organização.

Actualmente, mesmo um utilizador doméstico necessita de proteger alguma informação crítica que faz circular pela Internet e que armazena no seu computador, como passwords de *home banking*[6], de E-mail, de ligação ao *ISP*[7], entre outras. O facto de normalmente este tipo de utilização ser feita utilizando um sistema operativo comum como o *Microsoft Windows* (que por ser tão comum, as suas falhas são mais publicitadas) justifica atenção permanente utilizando componentes de segurança informática.

A responsabilidade dos dados que circulam nas redes das organizações é demasiado elevada. O sucesso de uma empresa muitas vezes depende da eficiência do sistema de segurança dos sistemas informáticos. Muitas empresas que menosprezaram a importância desta área, foram prejudicas em milhares de contos

Aparentemente menos importante é a segurança informática dos milhões de PC's domésticos, mas são estes mesmos que depois de controlados por *hackers*, servem de base de operações para ataques "anónimos" a servidores de organizações. Evidentemente que dificilmente um indivíduo com reduzidos conhecimentos informáticos, terá meios e tempo para auto-didácticamente aprender como proteger o seu PC doméstico, ou mesmo até a sua LAN doméstica. Poderá investir algum dinheiro num antivírus e mesmo até num pacote integrado de segurança (antivírus, firewall e ferramenta de encriptação de ficheiros), mas são uma minoria os que fazem isto e na verdade, a grande maioria

4 Worms são programas parasitas que se replicam, mas ao contrário dos virus, não infectam outros programas presentes no computador. Os Worms podem criar cópias locais, e/ou podem enviar cópias para outros computadores pela rede.

5 Local Area Network

6 Home Banking é o conceito de gestão, via Internet, a partir de casa (ou outro local que não o próprio banco) de dados das contas bancárias, como efectuar transferências bancárias, pagamentos, etc. Este tipo de operação e a informação trocada é bastante sensível e deve ter protecção máxima.

7 Internet Service Provider

está completamente desprotegida e nem sequer actualiza o *software* que possui com as últimas actualizações existentes.

Tanto organizações como utilizadores domésticos menosprezam a sua aposta em segurança até serem vítimas de ataques informáticos. Para muitas empresas isto revela-se fatal, para outras apenas significa um grande susto e a certeza de que no futuro não faltará dinheiro no orçamento para segurança dos seus meios informáticos. Para os privados, a aposta na segurança sempre será um compromisso de financeiro, de tempo e de conhecimentos.

1.2 Objectivos da tese

Esta tese pretende ser uma introdução ao mundo da segurança informática, introduzindo conceitos fundamentais no sentido de servir de base aos capítulos de introdução às *firewalls*, *firewalls* de filtro de pacotes, e de administração de *firewalls* (utilizando a *firewall iptables*) para o sistema operativo *Linux*.

Para uma compreensão vasta de segurança de redes informáticas e mais especificamente de *firewalls* em *Linux*, é necessário ter bons conhecimentos de protocolos de rede, principalmente (mas não apenas) o *TCP/IP* e *ethernet* (IEEE 802.3), sistemas operativos em geral e bons conhecimentos teóricos e práticos de Linux.

Está fora do objectivo desta tese aprofundar protocolos de comunicação como o *TCP/IP*[8] ou *Ethernet*[9] (802.3), serão feitas referências a características destes protocolos quando necessário. Também fora do objectivo deste documento está o estudo dos vários sistemas operativos e suas arquitecturas Este projecto terá uma abordagem mais prática, com o objectivo de estudar o tema das *firewalls* de software, de filtro de pacotes e particularmente de estudar e configurar a *firewall iptables* (*software* de *firewall* mais comum em ambientes *Linux*).

[8] Para aprofundar conhecimentos de *TCP/IP* consulte o livro vermelho [1], "TCP/IP Tutorial and Technical Overview"
[9] Para aprofundar conhecimentos do protocolo *Ethernet* consulte [2].

No primeiro capítulo, será introduzida a segurança de sistemas informáticos, definição de política de segurança, principais tipos de ataque e principais componentess de defesa de sistemas informáticos.

No segundo capítulo, serão introduzidas as *firewall,* conceito, evolução e classificação.

No terceiro capítulo serão descritas com mais detalhe o funcionamento de uma *firewall* de filtro de pacotes, as suas funções, filtragem de vários tipos de pacotes e por fim serão listadas algumas *firewalls* de filtro de pacotes existentes para o sistema operativo *Linux*.

No quarto capítulo será visto em detalhe a estrutura de funcionamento do *iptables*, a sintaxe dos seus módulos e extensões.

No quinto capítulo será feita a conclusão da tese.

1.3 Segurança Informática

Definir "*Segurança informática*" não é trivial. A dificuldade consiste em conseguir uma definição que seja abrangente o suficiente para ser válida independentemente do sistema a que se refere, mas específica o suficiente para descrever o que segurança é realmente. Genericamente podemos dizer que segurança é a ausência de risco ou perigo. No contexto da informática, segurança é a prevenção de, ou protecção contra,

- acesso a informação por entidades não autorizadas, e
- intencional mas não autorizada destruição ou alteração dessa informação.

Refraseando, segurança informática é a capacidade de um sistema de proteger informação e recursos em respeito à sua confidencialidade e integridade. De notar a inclusão de recursos de sistema que incluem protecção de *CPUs*, discos duros, programas, entre outros.

1.3.1 Classificação de Segurança Informática

A segurança informática é dependente de três propriedades, por vezes representadas por "CIA":

- **Confidencialidade**: garantir que a informação não é acedida por entidades não autorizadas
- **Integridade**: garantir que a informação não é alterada por entidades não autorizadas, de tal forma que utilizadores autorizados sejam enganados.
- **Autenticação**: garantir que os utilizadores são quem dizem ser.

Para garantir essas propriedades é necessário implementar certos mecanismos para:

- **o controlo de acesso**: garantir que os utilizadores acedem unicamente recursos ou serviços a que têm direito e que o acesso a esses mesmos recursos ou serviços não lhes sejam negados.
- **a não-repudiação:** assegurar que o emissor de uma mensagem não possam mais tarde negar que de facto enviou a mensagem.
- **a garantia de disponibilidade:** assegurar que um sistema está operacional e funcional num dado momento. Isto é normalmente conseguido através de redundância de meios. A perda de disponibilidade é normalmente designada como "*denial-of-service*".
- **a privacidade:** assegurar que um indivíduo mantenha o direito de controlar que informação pessoal é acedida por terceiros, como é utilizada, quem a usou, quem a possui e para que motivo é utilizada.

Segundo uma perspectiva teórica, os conceitos de privacidade, confidencialidade, e segurança são bem distintos e possuem diferentes atributos. Privacidade está associada a indivíduos; confidencialidade está associada a dados; e segurança é a propriedade associada ao *hardware* e *software* de sistemas informáticos. Segundo uma perspectiva prática, estes conceitos interelacionam-se. Um sistema que não mantém a confidencialidade ou privacidade individual poderia teoricamente ou até matematicamente ser considerado "seguro", mas não seria sensato utilizá-lo publicamente.

1.3.2 Perspectiva Funcional

A segurança de sistemas informáticos também pode ser analisada funcionalmente segundo os seguintes objectivos:

- **Evitar Riscos**: Fundamental na segurança de sistemas informáticos. É necessário grande reflexão nas necessidades do sistema informático. Será que é mesmo preciso uma ligação á Internet sem restrições? Deveremos optar por standardizar o uso de um sistema operativo do tipo *desktop* sem mecanismo de controlo de acesso a recursos?
- **Dissuasão:** Reduzir as potenciais ameaças através do medo. Pode consistir em estratégias de comunicação desenhadas para impressionar os potenciais atacantes, dando-lhes como provável serem apanhados e devidamente castigados.
- **Prevenção:** A alma da segurança informática. Consiste na implementação de medidas de segurança. A prevenção total é apenas teórica, visto haver um ponto em que medidas de prevenção adicionais não se justificam financeiramente (uma instalação informática certamente será mais segura se for defendida pelos *marines* norte-americanos, mas obviamente essa medida não seria muito rentável!)
- **Detecção:** Funciona melhor quando utilizada em conjunto com medidas de prevenção. Quando a prevenção falha, a detecção deverá actuar rapidamente, preferivelmente a tempo de prevenir dano. Inclui manutenção de registos da actividade do sistema (*logs*) e actividades de auditoria.
- **Recuperação:** Quando tudo o resto falhar, é necessário estar preparado para recorrer a cópias de segurança e reconstruir o sistema do nada, ou recorrer a servidores de prevenção *backup-servers* e a uma ligação à *Internet* de emergência. Evidentemente, esta função deve ser considerada antes das outras.

Analisar segurança funcionalmente poderá ser uma parte valiosa do processo de planeamento de uma política de segurança. Uma política de segurança forte focará todas as cinco áreas, começando pela recuperação. Esta tese concentra-se sobretudo na prevenção.

1.3.3 Domínios de Segurança

A segurança informática é frequentemente definida em termos de vários domínios interdependentes que grosseiramente correspondem a departamentos específicos ou a cargos profissionais:

- **Segurança Física:** Controlar o fluxo de pessoas e materiais; protecção contra os elementos e desastres naturais.

- **Segurança Operacional/Procedimental:** Diz respeito a políticas de gestão e decisões e a hierarquias entre pessoas.
- **Segurança de Pessoal:** Contratação de pessoal, pesquisa de antecedentes e de informação sobre o pessoal, treino, *briefings* de segurança, monitorização, lidar com as saídas de pessoal.
- **Segurança de Sistemas:** Controlo de acesso e autenticação de utilizadores, atribuição de privilégios, manutenção da integridade dos ficheiros e do *filesystem*, *backups*, processos de monitorização, manutenção de *Logs* e auditorias.
- **Segurança de Redes:** Protecção da rede e de todo o equipamento de telecomunicação, protecção dos servidores de rede e das transmissões; combate a escutas alheias; controlar o acesso á rede apartir de redes sem confiança; firewalls e detecção de intrusos.

Esta tese concentra-se primariamente sobretudo na segurança de redes, mais concretamente o controlo do fluxo de informação entre uma rede privada e a uma pública (como a *Internet*) utilizando para isso uma *firewall*. É difícil em sistemas *Linux* separar segurança de sistemas e de redes. Praticamente todas as distribuições de *Linux* nos últimos quinze anos incluiu uma implementação do protocolo *TCP/IP* assim como várias serviços de rede como o *FTP*, *Telnet*, *DNS* e *HTTP*.

1.3.4 Definição

Segundo a definição de Simson Garfinkel e Gene Spafford "Um computador é seguro se o seu software se comportar exactamente como é previsto" [3]. Ou seja, um computador é seguro se podemos confiar nele. A informação introduzida hoje manter-se-á inalterada amanhã. Determinados serviços disponíveis hoje, estarão ainda disponíveis amanhã.

Outra definição de Wilson Oliveira diz que segurança é "a restrição dos recursos de um micro-computador, ou de uma rede, ou de porções desta rede para outros utilizadores ou computadores" [4].

Evidentemente, estas definições assentam numa característica evidente: é suposto ser muito difícil a pessoas não autorizadas entrar num sistema seguro, ou seja, o tempo/valor monetário necessário para uma pessoa não autorizada quebrar a segurança deve exceder o valor da informação protegida. Maximizar o trabalho necessário para uma pessoa não autorizada conseguir quebrar a segurança de um sistema e

aumentar o risco de possíveis intrusos serem detectados, é parte crítica da segurança informática.

A definição que proponho para segurança de um sistema informático é "Implementação contínua de medidas de protecção (com redundância) que garantam tanto a confidencialidade e integridade da informação e dos recursos existentes e que circulam no sistema, de modo a que um utilizador não autorizado tenha que dispender um tempo/valor monetário não aceitável ou correr demasiados riscos, de modo a quebrar a segurança do sistema, com o objectivo final de garantir confiança e total disponibilidade de informação e recursos a utilizadores autorizados no sistema."

1.4 Política de Segurança

Para um sistema informático ser considerado seguro, é necessário definir uma política de segurança. Uma política de segurança define as expectativas da entidade ou organização relativamente a boa conduta e utilização da rede de computadores, que procedimentos devem ser executados para prevenir e responder incidentes de segurança. Uma política de segurança é essencial porque estabelece que recursos devem ser protegidos e que acções ou inacções são potenciais ameaças.

A política de segurança deve levar em conta ameaças á produtividade pessoal e eficiência. Deve também classificar cada alvo de protecção em diferentes níveis de protecção.

Sem uma política de segurança não é possível garantir segurança. Esta deve ser do conhecimento geral e deve ser comunicada a todos os utilizadores do sistema ou rede de computadores.

1.5 Componentes de Segurança de Sistemas Informáticos

Têm se desenvolvido muito os serviços de segurança nas duas últimas décadas, ou seja, sensivelmente desde que começou a haver registos de ataques a redes de computadores.

Estes serviços podem proteger uma rede de computadores a diversos níveis e devem ser utilizados conjuntamente para se criar condições de segurança. Isoladamente cada um deles não constitui uma política de segurança, mas são elementos essenciais que a constituem.

Na Tabela I-1 podemos ter uma visão geral de alguns componentes, serviços e protocolos de segurança e o nível a que actuam dentro do modelo *OSI*[10] e do protocolo *TCP/IP* e que são actualmente usados em segurança de redes informáticas.

Modelo OSI	TCP/IP	Elementos de segurança
Camada Física	Camada Física (*Physical Layer*)	Segurança Física; Controlo de acesso;
Camada de Dados (*Data Link Layer*)	Camada de acesso ao meio (*Medium Acess Layer*)	Controlo de tráfego a nível de acesso ao meio físico.
Camada de Rede	Camada de rede ou Internet (I*nternet/Network Layer*)	*Firewall*, *VPN*, *IPSec*, *ACL* de acesso a rede, *RAS*
Camada de Transporte	Camada de Transporte (*Transport Layer*)	*Firewall* (*Port Filtering*), *SSL*, *TLS*, *SSH*
Camada de Sessão	Camada de Aplicação (*Application Layer*)	*PGP*; *S-HTTP*; Palavras-chave; Mecanismos de encriptação; *IDS*;
Camada de Apresentação		
Camada de Aplicação		

Tabela I-1: Elementos de segurança

1.5.1 Segurança da camada Física

Os mecanismos e políticas de segurança mais evidentes, são por vezes as mais negligenciados: a segurança física, ou o controle de acesso físico ao componentes mais sensíveis da rede, como a sala de administração de rede ou a sala dos servidores. Considerando que a rede e o acesso

[10] OSI (Open Systems Interconnection) é uma descrição standard ou modelo de referência de como informação deve ser transmitida entre dois pontos numa rede de telecomunicações.

aos recursos são geridos nestes locais, é essencial que a segurança física destes locais sejam considerados ao criar uma política de segurança.

Nenhumas medidas de segurança serão eficazes se alguém não autorizado conseguir aceder indevidamente a estes locais. Mesmo numa situação em que um utilizador autorizado não tem qualquer má intenção, poderá sem o desejar ou mesmo ter o conhecimento de que o está a fazer, possibilitar indevidamente o acesso não-autorizado a entidades externas ou poderá passar por cima de certas medidas de segurança que deixam assim de cumprir a sua função.

1.5.2 Segurança da camada de acesso ao meio

Um computador ligado a uma rede local (*LAN*) tem dois endereços. Um é o endereço *MAC* (*Media Access Control*), um endereço único gravado no *hardware* de uma placa de rede, que identifica inequivocamente cada nodo de rede. Cada placa de rede tem apenas um único endereço *MAC*. Este endereço *MAC* é utilizado pelo protocolo *Ethernet* ao construir *frames* para transferir dados entre duas máquinas de uma rede local. O outro endereço é o endereço *IP*, que é utilizado pelos camadas acima da camada de interface de rede.

O cabeçalho de uma *frame Ethernet* utiliza o endereço *MAC* do destino e não o endereço *IP*. Cabe á camada de rede (camada *Internet*) mapear um endereço *IP* ao correspondente endereço *MAC*, visto este ser necessário pelo protocolo de acesso ao meio. A camada de rede inicialmente procura o *IP* para um determinado endereço *MAC* da máquina destino numa tabela, usualmente denominada *ARP cache*. Se o endereço *IP* da máquina destino não se encontrar na lista, o *ARP* (*Adress Resolution Protocolo*) faz o *broadcast* de um *ARP request* com o endereço *MAC* pretendido para todas os computadores da rede. O computador com o endereço em questão responde (*ARP Reply*) ao computador que fez o *ARP Request* devolvendo o seu endereço *IP*. Este endereço *IP* é adicionado ao *ARP cache* do computador que fez o *ARP Request* e é utilizado para todas as comunicações com o computador em questão.

No que diz respeito a segurança é necessário efectuar um controlo dos endereços *MAC* que se ligam á rede, garantindo que só acedem a esta, equipamentos autorizados.

Numa rede *Ethernet* todo o tráfego passa por todos os nodos de rede, e em cada nodo a máquina ou computador verifica nas *frames* que vão

passando, o endereço *MAC* destino e recolhe a *frame* da rede caso o destino seja a própria máquina. Neste tipo de topologia de rede, é possível a alguém que tenha acesso á rede, utilizar um *sniffer* para capturar o tráfego e desse modo aceder a informação privada.
Existem técnicas que permitem reduzir o risco de o tráfego ser capturado ou perceber quando este é capturado identificando assim em que ponto da rede está o infractor.

1.5.3 Segurança da camada de rede/Internet

Firewalls

Uma *firewall* é habitualmente um ponto que interliga duas ou mais redes, publicas (como a *Internet*) e privadas (redes locais). Numa grande maioria dos casos, consiste numa configuração de *hardware* ou *software* que está na fronteira (perímetro) entre uma rede local e uma rede externa e que controla todo o fluxo de comunicações entre estas redes, autorizando apenas algumas delas, baseando-se nos endereços de origem e destino de um pacote IP ou por exemplo das portas destino e opções *TCP* de um pacote.

Embora cada vez mais estejam a aparecer novos métodos de comprometer estes sistemas, se forem bem configurados são eficientes em manter utilizadores não-autorizados fora e em impedir actividades não desejadas na nossa rede privada.

Podem também ser utilizadas para compartimentar diferentes servidores e sub-redes, de modo a controlar o tráfego entre sub-redes internas.

VPN (Virtual Private Network)

Sistemas *VPN* são desenhados para tornar segura uma ligação entre duas ou mais redes privadas remotas entre si, utilizando uma rede publica, como a *Internet*, para as interligar. As *VPN* vieram preencher a necessidade de redes privadas a um custo reduzido, substituindo assim a procura de linhas privadas ou subalugadas cujo custo é exorbitante.

Uma *VPN* pode tomar diferentes formas e configurações, podendo funcionar a vários níveis do modelo *OSI*. Não só se pode classificar uma *VPN* pela camada do modelo *OSI* em que funciona mas também pelo modelo que emprega.

Os modelos existentes são o modelo *Peer-to-Peer*, cujo encaminhamento dos pacotes é decidido a cada nodo ou *hop* (*router*), e o modelo *Overlay*, cujo encaminhamento dos pacotes é fixo e comunicam transparentemente como se estivessem a um nodo ou *hop* de distância.

Em relação à camada do modelo *OSI* onde são implementadas, existem *Link Layer VPNs* (*Layer 2*) que servem de base a diferentes redes serem criadas na camada de rede acima (por exemplo *Frame Relay*, *ATM*), e *Network Layer VPNs* (*Layer 3*), que se consistem na criação de túneis (*tunneling*) na camada de rede entre dois *routers*, entre um *host* e um *router* ou entre dois *hosts*.

Para o mecanismo de tunneling o *VPN* utiliza um protocolo de *tunneling* como o *L2TP* (*Layer 2 Tunneling Protocol*) da *IETF* ou o *PPTP* (*Point-to-Point Tunneling Protocol*) da *Microsoft*. Mais tarde surgiu o *IPSec* (*Internet Protocol Security*) da *IETF* que veio resolver uma série de fraquezas de segurança existentes nos outros protocolos de *tunneling*.

1.5.4 Segurança da camada de transporte

Firewalls

As firewalls também se incluem na segurança da camada de transporte visto a possibilidade de filtragem de pacotes com base na informação dos cabeçalhos *UDP* e *TCP*, como porta origem e destino ou com *flags* específicas do cabeçalho *TCP*.

SSH (Secure Shell)

SSH é um protocolo que encapsula uma sessão de *shell* segura noutro computador remoto, podendo assim executar comandos de um modo seguro. Ou seja, o *SSH* providência mecanismos de autenticação e de comunicação segura em canais (ou redes) inseguros. Surgiu com o objectivo de substituir aplicações reconhecidamente inseguras como o *telnet*, *rlogin*, *rsh* e *rcp*.

Para a comunicação ser segura e autenticada são utilizados vários protocolos de cifragem como *DES*, *3DES*, *IDEA*, *Blowfish*, *Twofish*, *Arcfour*, *Cast128-cbc* para encriptação e *RSA* ou *DSA* para autenticação.

Existem duas versões do *SSH*: o *SSH1* e o *SSH2*. Estas duas versões são completamente diferentes entre si, encriptando partes diferentes dos pacotes, o *SSH1* utiliza chaves de servidor e de *host* para autenticar enquanto *SSH2* utiliza apenas chaves de *host*. O protocolo *SSH2* foi completamente rescrito e tem melhoramentos de segurança, performance e portabilidade não sendo compatível com *SSH1*. Entretanto foram descobertas algumas vulnerabilidades no protocolo *SSH1* e desde então a utilização do *SSH2* foi amplamente recomendada.

SSL (Secure Socket Layer)

O protocolo *SSL* corre acima do *TCP/IP* e abaixo de protocolos de níveis superiores como *HTTP* ou *IMAP*. Constitui assim mais uma camada, uma interface que permite mecanismos de autenticação e de comunicação encriptada (usa *tunneling*).

Autenticação do servidor *SSL*: permite a um utilizador confirmar a identidade do servidor. O cliente *SSL* utiliza técnicas de criptografia de chave-pública para verificar se o certificado e ID do servidor são válidos e foram emitidos por um CA (*Certificate Authority*) presente na lista de CAs de confiança do cliente.

Autenticação do cliente *SSL*: permite ao servidor confirmar a identidade do utilizador (cliente *SSL*). Utiliza os mesmos mecanismos que são utilizados para autenticar o servidor.

Uma ligação encriptada por *SSL* requere que toda a informação enviada entre cliente e servidor seja encriptada pelo *software* do emissor da informação, e decifrada pelo *software* do receptor da informação, garantindo assim um alto grau de confidencialidade que é importante para ambas as partes da transação. Em adição, toda a informação

enviada é protegida por mecanismos que garantem integridade dos dados.

1.5.5 Segurança da camada de sessão, apresentação e aplicação

Mecanismos de palavra-chave

O uso de palavras-chave ou *passwords* são um método de identificar e autenticar utilizadores quando estes acedem a uma rede privada.
A password por si própria não é suficiente para identificar um indivíduo, é preciso um par de *nome de utilizador* (*username)* mais a respectiva palavra-chave (*password)*.

Geralmente podem garantir que um utilizador é quem diz ser, mas infelizmente existem variadíssimas maneiras de uma palavra-chave ficar comprometida:

- Alguém pode "escutar" por um *nome de utilizador* e *palavra-chave* quando um utilizador autorizado acede à rede privada apartir de uma rede pública. Ou seja, alguém entre o emissor e o receptor pode capturar a informação em trânsito e daí retirar o *nome de utilizador* e *palavra-chave*.
- É possível descobrir uma palavra-chave através de ataques do tipo força-bruta (*brute-force attack*)ou ataque tipo-dicionário (*dictionary attack*). Estes ataques consistem em repetidamente tentar diferentes palavras-chave até o sistema alvo do ataque indicar que a palavra-chave correcta foi inserida. O ataque tipo força-bruta consiste em tentar combinações de caracteres alfanuméricos e símbolos para formar palavras-chave. O ataque tipo dicionário consiste em experimentar todas as palavras existentes num dicionário de palavras.
- Os utilizadores muitas vezes "emprestam" a sua palavra-chave a um colega, ou deixam-na escrita num local onde é possível outras pessoas a obterem.

Felizmente, existem técnicas, recomendações e tecnologias para tornar o mecanismo de palavras-chave mais seguro:

- Gerar *one-time passwords* [10], ou seja assumir que uma palavra-chave pode ser sempre comprometida e portanto nenhuma palavra-chave será utilizada duas vezes. É gerada palavra-chave por cada novo acesso. Este processo é útil em situações em acesso remotos esporádicos.

- O sistema operativo em si pode aplicar políticas de segurança de palavras-chave, como prazos-de-validade de palavras-chave, garantir que essas palavras-chave têm um número mínimo de caracteres e um mínimo de letras e números. O sistema operativo não aceita palavras-chave que não cumpram o que está estipulado na sua política de segurança de palavras-chave.
- Os *Smart Cards* são mecanismos de seguros de palavras-chave. São criadas e armazenadas num circuito integrado semelhante a um cartão de crédito, palavras-chave únicas, baseadas num esquema de pergunta-resposta (*challenge-response*). A palavra-chave é introduzida como parte do processo de identificação no sistema e validada num servidor de palavras-chave que regista todos os acessos ao sistema. Como é de esperar, implementar estes sistemas pode ser muito dispendioso.
- Em casos em que um utilizador possui múltiplas palavras-chave para diferentes recursos, estas palavras-chave tornam-se inerentemente menos seguras. O utilizador tende a utilizar a mesma palavra-chave para múltiplos recursos, ou então a apontar as palavras-chave em algum lugar evidentemente não seguro.
  Sistemas de *single sign-on* é a solução para estas situações. Estes sistemas fazem as gestão de palavras-chave de um utilizador. Podem gerar palavras-chave mais fortes, associam cada par *nome de utilizador* e *palavra-chave* ás diferentes áreas/recursos da rede, e fornecem-nas quando necessárias.

Criptografia

Criptografia ou cifragem é o processo de codificação de informação baseado num algoritmo ou tabela de substituição em informação aparentemente não inteligível.
Inicialmente os processos criptográficos eram principalmente baseados em tabelas de substituição. Rapidamente se constatou que estes métodos criptográficos iniciais eram facilmente quebrados e á medida que estes métodos de substituição foram evoluindo, vários métodos de os quebrarem foram aparecendo. Esquemas matemáticos de análise de frequência de letras possibilitaram pegar em qualquer mensagem criptográfica e experimentar as combinações de substituições de símbolos mais prováveis baseadas nas análises de frequências de letras do abecedário previamente conhecidas.

A certo ponto era evidente que o algoritmo por si só não seria garante da confidencialidade de uma mensagem, o que levou ao aparecimento de modelos criptográficos que introduziam as chaves criptográficas.

As chaves eram utilizadas no processo de encriptação mas também no processo de desencriptação. Deste modo para alguém decifrar uma mensagem encriptada, teria que saber não apenas o algoritmo criptográfico, mas também a chave que lhe permitia desencriptar a mensagem. Estes algoritmos são apelidados de algoritmos de criptografia simétrica pois existe uma única chave utilizada para encriptar e para desencriptar.

Outro modelos matemáticos mais ou menos sofisticados foram introduzidos como cifragem de blocos, em que a mensagem é dividida por blocos e cada bloco é encriptado ou cifrado individualmente e em alguns casos os resultados das cifras eram combinados com os blocos vizinhos (cifragem em cadeia ou *stream cipher)*.

Os algoritmos de criptografia simétrica têm um grande calcanhar de Aquiles, são relativamente simples de quebrar utilizando o método de força-bruta (experimentar todas as combinações de caracteres alfanuméricos como chave até o resultado da desencriptação ser inteligível).
Um dos algoritmos simétricos mais utilizados é o *DES* (*Digital Encrypton Standard*).

Um método muito usado actualmente, a criptografia assimétrica ou de chave pública, é ideal para tornar seguras as comunicações em meios inseguros como a *Internet*. A criptografia assimétrica de chave pública baseia-se num esquema de duas chaves: uma para encriptar e outra para desencriptar.
Estas duas chaves funcionam em conjunto, o que é encriptado por uma, só pode ser desencriptado com a outra.
Assim, cada indivíduo possui duas chaves para comunicar, uma pública que é fornecida a todas as entidades que desejam enviar uma mensagem cifrada a esse mesmo indivíduo; e uma chave privada que serve para desencriptar as mensagens recebidas.
Os problemas de segurança surgem na distribuição destas chaves. Existem alguns tipos de ataques que permitem a um terceiro fazer-se passar por um dos interlocutores de uma comunicação (ataques *Man-in-the-middle*).

Um dos algoritmos assimétricos mais utilizados é o *RSA*. (*Digital Encrypton Standard*). Baseado no sistema de chave pública, foi desenvolvido o *PGP* (*Pretty Good Privacy*), um software de segurança que é muito popular hoje em dia.

Para ultrapassar o problema da distribuição de chaves, surgiram as assinaturas digitais que permitem garantir que um interlocutor de uma comunicação é quem diz ser.

Uma outras funções criptográfica muito usada é as funções de *hash*. Esta técnica consiste em transformar um mensagem de comprimento variável, numa mensagem de tamanho fixo. Para ser usado em criptografia, a função de *hash* deve ser tal que seja bastante difícil duas mensagens diferentes produzirem o mesmo resultado (ou *hash*).
As funções *hash*, no entanto, não são reversíveis, ou seja, não permitem uma função inversa. Isso torna estas funções úteis para armazenamento de palavras-chave para fins de autenticação. Um dos algoritmos de *hash* mais utilizado é o *MD5*.

IDS (Intrusion Detection System)

O principal defeito da grande maioria das *firewalls* é o facto de não procederem à verificação do conteúdo dos pacotes, em particular depois de estes terem ultrapassado a *firewall* e viajarem apenas na rede interna. Os *IDS* (*Intrusion Detection System*) ajudam as organizações a identificar os ataques mais cedo, permitindo tomar as medidas necessárias em tempo útil.

Um *IDS* é uma solução complementar à instalação de uma *firewall*. A sua função é analisar permanentemente o tráfego da rede (interno e externo) e compará-lo com padrões conhecidos de comportamento de intrusos. Por estarem situados na rede interna analisam não só o tráfego externo, vindo da *Internet*, como também o tráfego interno. Podem, assim, ser detectados ataques vindos de pessoas internas a uma organização ou que acedem a esta por outros meios.

Um *IDS* pode analisar o tráfego na rede de diferentes perspectivas, cada uma com objectivos e resultados diferentes:

- **Detecção de assinaturas**: Consiste na procura de padrões específicos de tráfego, correspondentes a determinado ataque. A desvantagem é que o padrão de ataque tem de ser conhecido previamente e tem de ser programado no *IDS*. Para além disto, em situações de alto débito, o *IDS* poderá não escalar, eliminando pacotes do sistema de análise.

- **Detecção de comportamentos**: Consiste na análise e na procura de padrões de comportamento, através da identificação de

anomalias estatísticas. A ideia é que uma rede de computadores segue determinados padrões de comportamento que resultam em determinadas estatísticas. Alterações dessas estatísticas (maior tráfego a horas pouco usuais, aumento do número de pacotes de determinado tipo de protocolo, etc.) resultam na identificação de um possível ataque.

- Detecção de anomalias de protocolo: Consiste na análise de conformidade com o standard de pacotes de determinado protocolo. A título de exemplo, os recentes ataques do *worm* Code *Red* são facilmente detectados por este tipo de *IDS*, dado que os pedidos *HTTP* feitos ao servidor não estão conformes com o standard, usando caracteres inválidos para conseguirem subverter o funcionamento do *Web server*.

Ferramentas disponíveis e eficácia

Existem no mercado diversas soluções de *IDS*, não só em termos de plataformas e funcionalidades, como em preço. Como por exemplo, os comerciais *ISS BlackICE*, o *Cisco IDS* ou o *open source Snort*.

Mas a eficácia de uma solução de segurança de redes dependente da instalação de componentes que, de forma complementar, actuem de forma a detectar os vários níveis de ameaças nos vários níveis da rede.

Não sendo, por si só, uma solução única para a segurança de uma rede, a instalação de um *IDS* é uma peça fundamental da solução. A detecção rápida de ataques e a sua protecção ou resolução exige sistemas mais eficazes.

1.6 Conclusão

Este capítulo introduziu a temática da segurança de sistemas informáticos. Foram identificadas as propriedades necessárias para garantir a segurança dos sistemas: a confidencialidade, a autenticação, integridade sendo implementadas por mecanismos de controlo de acesso, não-repudiação, garantia de disponibilidade e privacidade.

Foi descrita a segurança informática segundo uma perspectiva funcional, recorrendo a várias medidas tais como: evitar possíveis riscos,

dissuasão de possíveis infractores, prevenção de possíveis quebras de segurança, detecção de falhas e recuperação das mesmas.

Foi descrita a segurança informática por domínios físico, operacional/ procedimental, pessoal, de sistemas e de redes.

2 Introdução às Firewalls

2.1 Introdução

Neste capítulo analisaremos o conceito de *firewall*, a sua evolução e os diferentes tipos de *firewalls* existentes.

2.2 Conceito de Firewall

Segundo o *RFC 2647*, "Benchmarking Terminology for Firewall performance" [5], uma *firewall* é "um dispositivo, serviço de sistema operativo, ou aplicação que estabelece uma política de controlo de acesso entre redes."

Mais especificamente, o termo *firewall* possui um número de diferentes significados consoante os mecanismos utilizados para implementar a *firewall*, a camada da pilha *TCP/IP* em que a *firewall* opera, e as arquitecturas de rede e *routing* utilizadas.

Uma firewall pode ser de vários tipos, *firewall* estática, dinâmica de filtro-de-pacotes, pode ser um *gateway* de aplicação (ALG) ou servidor *proxy*, e pode ser ou de *software* ou de *hardware*. Mais adiante veremos as principais diferenças entre estes tipos. De qualquer modo, o propósito de todas elas é obrigar ao cumprimento de uma política de segurança previamente definida.

2.3 Evolução das Firewalls

O termo original "*firewall*" refere-se a uma barreira física usada para impedir que incêndios se alastrassem entre muros em casas ou apartamentos. Analogamente, e segundo a definição dada no ponto anterior, uma *firewall* no sentido informático impede que comunicações indesejadas ultrapassem o perímetro de uma rede.

As *firewalls* surgiram e desenvolveram-se rapidamente nos anos 80 em paralelo com o desenvolvimento da *Internet*. Inicialmente apenas o Departamento de Defesa dos Estados Unidos (DoD) investiu no desenvolvimento de *firewalls*, mas rapidamente o sector privado se tornou na maior fonte do seu desenvolvimento.

À medida que a *Internet* evoluiu de uma rede científica fechada para uma rede pública, a necessidade de medidas de segurança para

prevenir ataques contra os sistemas informáticos tornou-se cada inevitável. Desde o aparecimento do primeiro virus da Internet, o *worm Morris*, foram desenvolvidas cinco gerações de *firewalls* para combater ataques a informação sensível das corporações.
A primeira geração de *firewalls* apareceu em meados de 1985, através de *routers IP* com regras de filtragem de pacotes que podiam ser modificadas manualmente pelo administrador. Estas *firewalls* eram fáceis de manter porque existiam poucos serviços que disponibilizados para o exterior: acesso seguro a *telnet*, *FTP*, *E-mail* e *news*. Uma vez que os *routers IP* deixam passar o tráfego directamente, os utilizadores apenas precisavam ter um *IP* válido ou acesso á *intranet*.

A segunda geração de *firewalls*, desenvolvida entre 1989-1990, foram as *firewalls/gateways* ao nível do circuito, impedindo ligações directas entre redes, autorizando-as baseadas no endereço.

Durante a terceira geração de *firewalls*, apareceram as *application layer firewalls* (*firewalls* da camada de aplicação), sendo os casos mais especiais *bastion hosts* a correrem serviços *proxy*. Durante este período, a primeira *firewall* comercial, a *DEC SEAL* (*Secure External Access Link*), foi desenvolvida pela *Digital Equipment Corporation* (*DEC*) em 1991 e usava filtros e *gateway* de aplicação ou *proxys*. O *DEC SEAL* consistia num sistema externo denominado *Gatekeeper*, o único ponto contactável pela *Internet*, uma filtering *gateway*, denominada *Gate*, e um *Mailhub* interno (ver Imagem II-1).

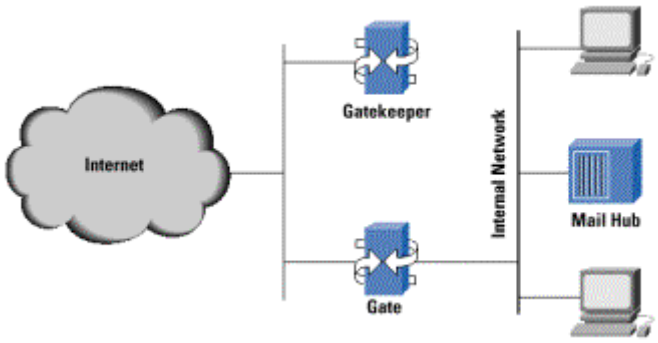

Imagem II-1: DEC SEAL

Por volta de 1991 deu-se o início da quarta geração de *firewalls* que consistiu no desenvolvimento de *firewall* dinâmicas de filtro de pacotes que modificavam dinamicamente as regras de filtragem. Em 1994, a *firewall* desenvolvida pela *Check Point Technologies*, a *Firewall-1*, utilizava

a tecnologia de filtragem dinâmica de pacotes e, ao contrário de *firewalls* anteriores, foi a primeira com instalação e administração simplificada, não sendo necessário editar ficheiros *ASCII* como nas outras *firewalls* comerciais.

Hoje em dia, as *firewalls* encontram-se na quinta geração e distinguem-se por terem uma arquitectura de *proxy* a nível do *Kernel* (*Kernel Proxy architecture*), que monitoriza o tráfego em múltiplas camadas da *stack* protocolar de comunicação.

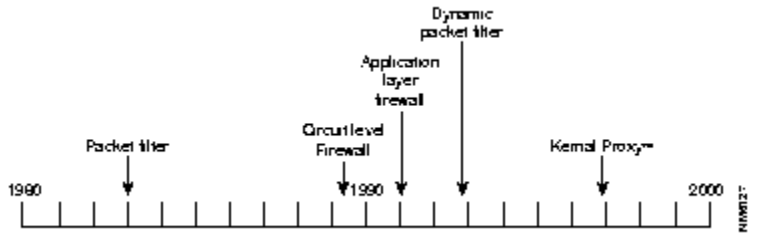

Imagem II-2: Evolução das cinco gerações de arquitecturas de firewall

2.4 Classificação das firewalls

Firewall é o termo vulgarmente atribuído a diferentes sistemas tais como *firewalls* estáticas de filtro de pacotes ou *routers* de filtro de pacotes, *firewalls* dinâmicas ou *stateful firewalls*, *gateways* de aplicação, ou a servidores *proxy*.

2.4.1. Firewalls estáticas de filtro-de-pacotes "stateless"

Uma *firewall* estática de filtro de pacotes é normalmente implementada dentro do sistema operativo ou *hardware* do *router* e opera na camada de rede (*IP network layer*), e muitas vezes também na camada de transporte (*transport layer*).
Estas protegem o sistema fazendo decisões de *routing* após a filtragem dos pacotes baseada na informação presente nos cabeçalhos dos pacotes. Estas decisões são tomadas pacote a pacote.
Em sistemas maiores, a um *filtering router* (também designado *screening router*) é muitas vezes colocado antes da *firewall* para fazer filtragem

inicial com o objectivo de reduzir o esforço de processamento requerido pela *firewall*.

2.4.2. Firewalls dinâmicas, ou Firewalls de inspecção "stateful"

As *firewalls* dinâmicas operam nas camadas de rede e de transporte, podendo filtrar pacotes com base em certas informações de protocolos de acesso ao meio e mesmo de protocolos de camadas superiores à camada de transporte.

Este tipo de *firewalls* de filtro de pacotes mantêm informação sobre sessões *TCP* ou troca de pacotes *UDP*. Os pacotes são filtrados no contexto de uma sessão e não isoladamente.

Por exemplo, este tipo de *firewalls* sabe que um pacote *TCP* com a *flag ACK* deve ser descartado caso não tenha recebido um pacote inicial com a *flag SYN*. Sabe se um determinado pacote *UDP* é resposta esperada a um pacote anteriormente enviado, ou pelo menos se o pacote diz ser de um *host* para o qual foi enviado recentemente um pacote. A *firewall* consegue dizer se uma mensagem de erro *ICMP* chegou em resposta a uma sessão corrente.

É possível reconhecer que um determinado pacote pertence a uma sessão previamente autorizada, e desse modo passar á frente as regras de filtragem a que normalmente deveria ser sujeito esse pacote. Esta função, chamada de *stateful inspection* permite optimizar o funcionamento da *firewall*.

As *firewalls* dinâmicas normalmente têm algum conhecimento específico de aplicações ou protocolos. Por exemplo, uma *firewall* pode estar a fechar as portas não privilegiadas e especificamente abrir uma porta para deixar passar uma sessão de *FTP data*.

2.4.3. Gateways de Aplicação (ALG) e Proxys

O termo *Gateway de Aplicação* (*Application Level Gateway* ou *ALG*) refere-se a um *proxy* de aplicação que funciona como ponto terminal para ambos os lados de uma conexão. Numa *proxy-firewall* são implementadas diferentes aplicações para cada serviço que utilize o *proxy*. A gateway de aplicação conhece a aplicação específica para qual está a fazer de *gateway* e pode fazer inspecção ao nível da aplicação.

Aplicações *Proxy* podem garantir integridade dos dados, ou seja, que a informação trocada é apropriada á aplicação, que é sondada por possíveis virus, e sujeita à política de controlo de acesso ao nível da aplicação.

A *gateway* mantém duas ligações, uma com o cliente e outra com o servidor. Cada aplicação de *proxy* aparenta ser o servidor para o *software* cliente, e aparenta ser o *software* cliente para o servidor. A *gateway* inicia a ligação ao servidor remoto em nome do *software* cliente e responde ao *software* cliente em nome do servidor. Na prática o tráfego local nunca deixa a *LAN* e o tráfego remoto nunca entra na *LAN*.

Também se designa por *ALG* aos módulos de suporte a aplicações para uma *firewall*. Muitas *firewalls* possuem um *FTP ALG* para suportar o canal de dados do *FTP*, onde o cliente de *FTP* indica ao servidor a que porta local se pode ligar de modo a abrir o canal de dados. Portanto é o servidor que inicia a ligação do canal de dados, quando normalmente é o cliente a iniciar todas as ligações. As *ALGs* são normalmente necessárias para fazer passar protocolos multimédia por uma *firewall*, devido á complexidade destes protocolos que normalmente utilizam várias ligações simultâneas iniciadas por ambas partes e muitas vezes utilizam uma combinação de *TCP* e *UDP*.

Uma *ALG* é um *proxy*. Outra forma de *proxy* é um *proxy* de nível de circuito (*circuit-level proxy*). Os *proxys* de nível de circuito não possuem conhecimento específico de aplicações, mas obrigam ao cumprimento de políticas de acesso e autorização, e servem de intermediários na ligação entre dois pontos. *SOCKS* é um exemplo de um *proxy* de nível de circuito.

2.4.4. Firewalls Hardware Vs Firewalls Software

Existem dois tipos de suporte físico para uma *firewall*: *firewalls hardware* e *firewalls software*. A designação indica a principal diferença, a primeira é uma *firewall* embebida em *hardware*, muitas vezes tendo acumulando funções de *router* ou *switch*, a segunda é uma *firewall* de *software* que instalamos num *hardware* e sistema operativo que pretendemos, desde que suportados pela *firewall software* em questão.

As *firewalls hardware* são suportam maiores quantidades de tráfego relativamente as *firewalls* de *software*, são mais caras e são tipicamente

sistemas integrados numa caixa, quase sólidos, poucas peças se podem mover. Tipicamente as ventoinhas são as únicas coisas que podem eventualmente necessitar reparação ou substituição, enquanto instalar uma *firewall* de *software* num *PC* significa que temos preocupações adicionais com o(s) disco(s) duros e outro *hardware*. Para aplicar as actualizações do *software* apenas é necessário fazer um *flash* do sistema operativo, ou seja, tudo é substituído, enquanto numa *firewall* de *software* temos que aplicar os correctivos ao sistema operativo em que esta corre, aplicar as actualizações de *BIOS* do *hardware* em que corre, e outras actualizações.

As *firewalls* de *software* actualmente tendem a suportar mais camadas do modelo *OSI*, tipicamente incluem a camada de rede, de transporte , podendo efectuar alguma filtragem baseado no endereço *MAC* origem dum pacote (quando dentro da mesma rede) normalmente contém módulos *ALG*, ou *proxy*, envolvendo assim camadas acima da camada de rede.

As *firewalls* de *software* não suportam tanto tráfego, mas são soluções bastante económicas que se adequam a cenários onde se pretende gastar o mínimo possível, como redes caseiras ou pequenas e médias empresas com um parque informático pequeno.

As *firewalls* de *software* são necessariamente mais inseguras por ser mais complicado configura-las correctamente. E é neste ponto que se centrará as atenções mais adiante neste documento.

No entanto, existe alguma polémica em definir o que *firewalls* são de facto *firewalls* de *hardware*. Existem soluções integradas que alegam ser *firewalls* de *hardware* mas não passam de *hardware* dedicado onde se instalam *firewalls* de *software*, muitas vezes com sistemas operativos proprietários. Por exemplo, adquire-se uma *Check Point Firewall-1* e teremos uma *firewall* de *software* que se pode instalar no *hardware* existente. A série *Nokia IP* são *firewalls* de "*hardware*" prontas a ligar na rede. Mas se olharmos mais de perto veremos que a série Nokia IP correm a Checkpoint Firewall-1 sendo que a única diferença é que o *software* vem pré-instalado na unidade.

Imagem II-3: Nokia IP Series

Vantagens das *firewalls* de *software*:

- Permite escolher o *hardware* em que a *firewall* vai correr baseado nas necessidades.
- É possível fazer actualizações ao *hardware* caso seja necessário adicionando mais *RAM*, um *CPU* mais rápido, mais interfaces de rede, outras actualizações.
- O custo pode ser dramaticamente menor do que *firewall* de *hardware*.

Desvantagens das *firewalls* de *software*:

- Demoram mais tempo a instalar. Algumas *firewalls* para sistemas operativos com interfaces mais amigáveis como o *Microsoft Windows*, tem progredido, sendo cada vez mais rápidas de instalar (exemplos: *Zone Alarm*, *Norton Internet Security*).

Vantagens das *firewalls* de *hardware*:

- Instalação rápida visto o *software* vir pré-instalado (excepto nos casos em que a *firewall* tem que ser altamente configurada de acordo com uma política de segurança mais rígida e precisa)
- Garantia de compatibilidade entre o *hardware* e o *software*.

Desvantagens das *firewalls* de *hardware*:

- Se o *hardware* falhar houver uma avaria, não é possível a sua substituição imediata, salvo exista meios financeiros que permitam ter alta redundância no equipamento informático. Nestes casos ajuda fazer contractos de suporte que garantam a reposição de equipamento num prazo relativamente curto (24 horas).
- Não permite fazer actualizações ao *hardware*. Em algumas *firewall* é possível instalar interfaces de rede novas.

2.6 Conclusão

Foi referido que uma *firewall* é um componente de segurança cuja principal função é monitorizar e filtrar um fluxo de pacotes. Pode funcionar em níveis diferentes do modelo *OSI*. Desde a camada de acesso ao meio até a camada de apresentação.

O número de camadas em que uma *firewall* actua corresponde sensivelmente ao tipo de *firewall*. Existem dois grupos, as *firewalls* de filtro de pacotes (estáticas ou dinâmicas) e as *gateways* de aplicação ou *proxys* (*proxy* de aplicação ou *proxy* de nível de circuito).

As firewalls podem ser integradas (hardware e software proprietário) ou apenas de *software*, podendo ser instaladas em determinadas plataformas de hardware e software (sistemas operativos).

3 Firewalls de filtro de pacotes

3.1 Introdução

Neste capítulo será analisado de perto o funcionamento e estrutura das firewalls de filtro de pacotes (*packet-filtering firewalls*). Serão descritos os principais tipos de pacotes a filtrar pela *firewall*.

3.2 Funcionamento

Uma *firewall* de filtro de pacotes utiliza uma listas de regras de aceitação e/ou negação. Estas regras explicitamente definem que pacotes é que serão admitidos pela interface de rede, e para isso examinam informação presente nos cabeçalhos dos pacotes. Após esta análise o pacote ou é enviado para o seu destino, ou silenciosamente descartado (rejeitado), ou bloqueado e enviado um pacote com condição de erro para a máquina remetente (negado).

Estas regras são baseadas em diversas informações como a interface de rede e respectivo endereço *IP*, o endereço *IP* origem e destino (camada de rede), as portas origem e destino, *TCP* ou *UDP* (camada de transporte), *flags TCP*, tipo de mensagem *ICMP* e se o pacote está a sair ou a entrar na rede.

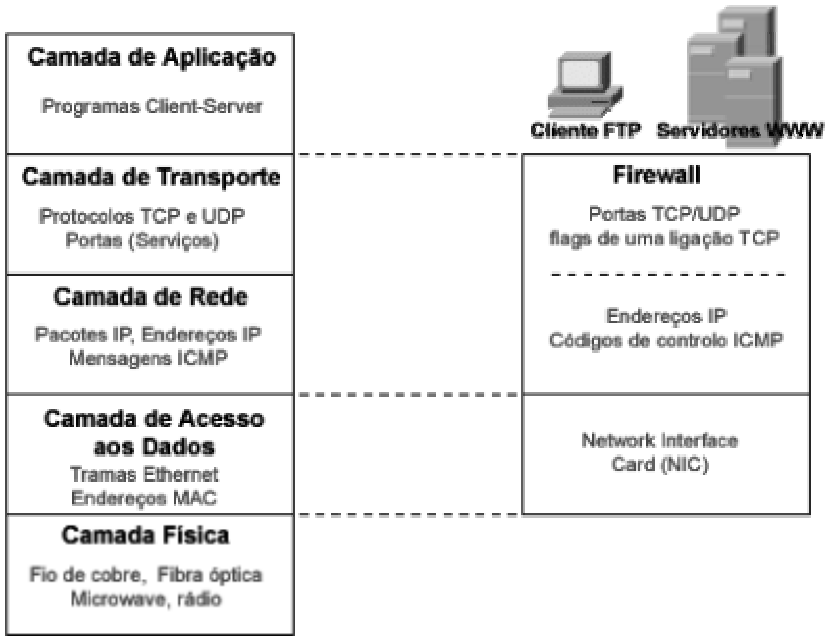

Imagem III-4: Camadas de funcionamento de uma packet-filtering firewall

Na Imagem III-4 podemos ver a que níveis do modelo *TCP/IP* funciona uma *firewall* de filtro de pacotes.

É necessário controlar cuidadosamente todo o tráfego que entra e sai da máquina que está ligada directamente á Internet (a *gateway*) através duma interface de rede. A interface de rede que interliga a *gateway* á Internet é muitas vezes denominada de interface externa.

Para uma rede de apenas uma máquina (apenas uma interface na *firewall*), apenas existe um par de I/O, a interface externa. No caso de haver uma rede privada, existem duas interfaces na *firewall*, dois pares de I/O. Outro cenário frequente é a firewall ter três interfaces de rede, a interface externa, a interface da rede privada e a interface da rede *DMZ*[11].

O filtro de *input* e o filtro de *output* de uma interface tem regras separadas e independentes. As listas de regras que definem que tráfego é permitido e que tráfego não é permitido são chamadas de cadeias de regras (*chains*). Este nome explica-se pelo facto de cada pacote ser comparado com cada regra da lista, percorrendo a cadeia de regras até encontrar uma regra que seja activada pelo pacote, ou até a lista se acabar, como podemos ver na imagem Imagem III-5.

A firewall filtra o que entra e o que sai de uma interface de um modo independente. As regras de filtragem da *firewall* estão agrupadas em várias sub-listas com o propósito de separar tráfego de naturezas diferentes. As sub-listas dividem o tráfego pela sua natureza: tráfego que se destina à *firewall* (cadeia INPUT), tráfego gerado pela *firewall* (cadeia OUTPUT) e tráfego que passa pela *firewall*, destinado a uma máquina interna, ou a um destino externo (cadeia FORWARD).

[11] Demilitarized zone: Zona desmilitarizada, segmento de rede utilizado para isolar servidores que oferecem serviços considerados inseguros a máquinas externas à rede.

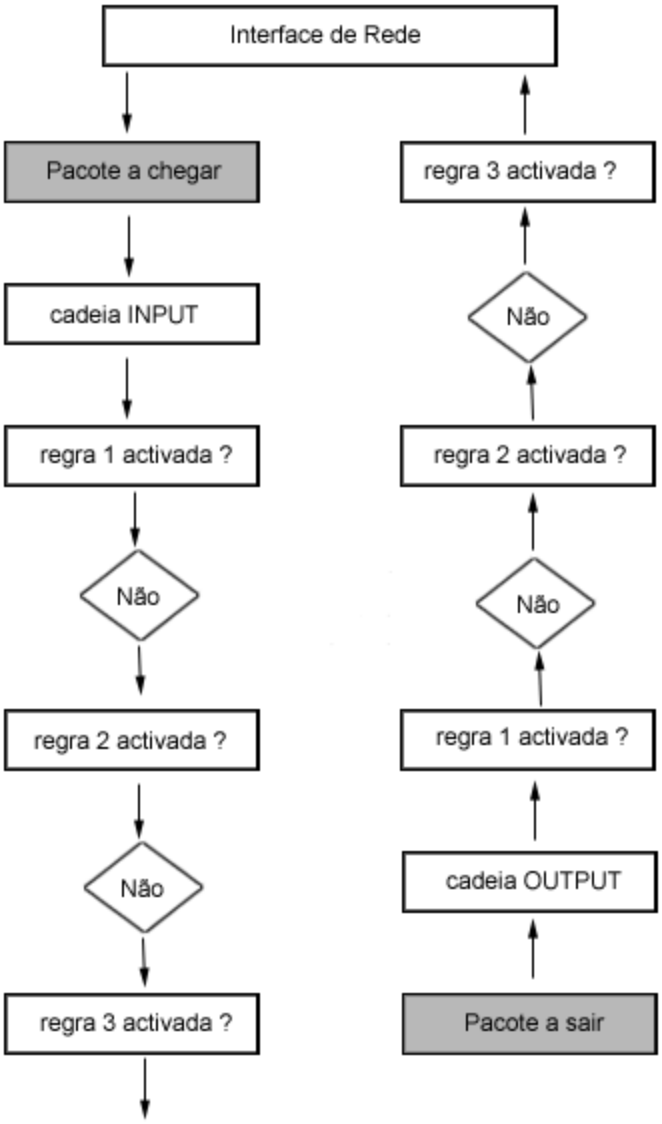

Imagem III-5: Cadeias de regras *INPUT* e *OUPUT*

Ao contrário do que poderá parecer, este mecanismo por si só não garante a segurança desejada. É apenas uma parte do conjunto, um componente do esquema global de segurança. Nem todas as aplicações ou protocolos se adequam bem ao mecanismo de packet-filtering. Este tipo de filtragem é demasiado de baixo nível para permitir autenticação e controlo de acesso. Estes serviços de segurança devem ser garantidos em camadas superiores. O *IP* não tem a capacidade de garantir que o emissor é quem diz ser. A única informação que poderia identificar alguém a este nível é o endereço *IP* origem no cabeçalho do pacote *IP*, mas esta informação pode ser facilmente modificada e não é

possível nem á camada de rede nem á camada de transporte verificar dados da camada de aplicação. De qualquer modo, ao nível do pacote podemos fazer uma filtragem maior e mais simples relativamente ao acesso às portas, ao conteúdo dos pacotes, e obrigar ao funcionamento correcto dos protocolos.

Não faz muito sentido ter aplicações de segurança de camadas superiores, como *proxys* ou *gateways* de aplicações, se não tivermos mecanismos de filtro de pacotes como filtro inicial. As aplicações de níveis superiores não estão preparadas para fazerem filtro de pacotes ou não o fazem de um modo eficiente. Cada camada da pilha protocolar de segurança fornece uma funcionalidade que as outras camadas dificilmente conseguem fornecer.

3.3 Escolher a política por omissão de uma cadeia de regras

Cada cadeia de regras possui uma política por omissão e um conjunto de acções ou regras para seguir em resposta a tipos de mensagens específicas que entram na cadeia de regras. Quando um pacote entra numa cadeia de regras é submetido a uma lista de regras até encontrar uma regra que se aplique ao pacote. Se o pacote não encontrar nenhuma regra que se aplique a ele, a política por omissão é aplicada ao pacote.

Existem duas filosofias básicas, ou políticas por omissão que podem ser utilizadas:

- Negar tudo por omissão e explicitamente permitir certos pacotes.
- Aceitar tudo por omissão e explicitamente negar determinados pacotes.

A filosofia recomendada é a negar tudo por defeito. Deste modo é mais fácil configurar uma *firewall* segura. Esta filosofia implica que é necessário explicitamente activar serviços, permitindo os protocolos utilizados nesses serviços. Isto implica que é necessário conhecer os protocolos de comunicação de cada serviço que é activado. Este filosofia requer mais trabalho inicial na configuração mas é mais segura. Alguns produtos de *firewalls* comerciais só suportam esta filosofia.

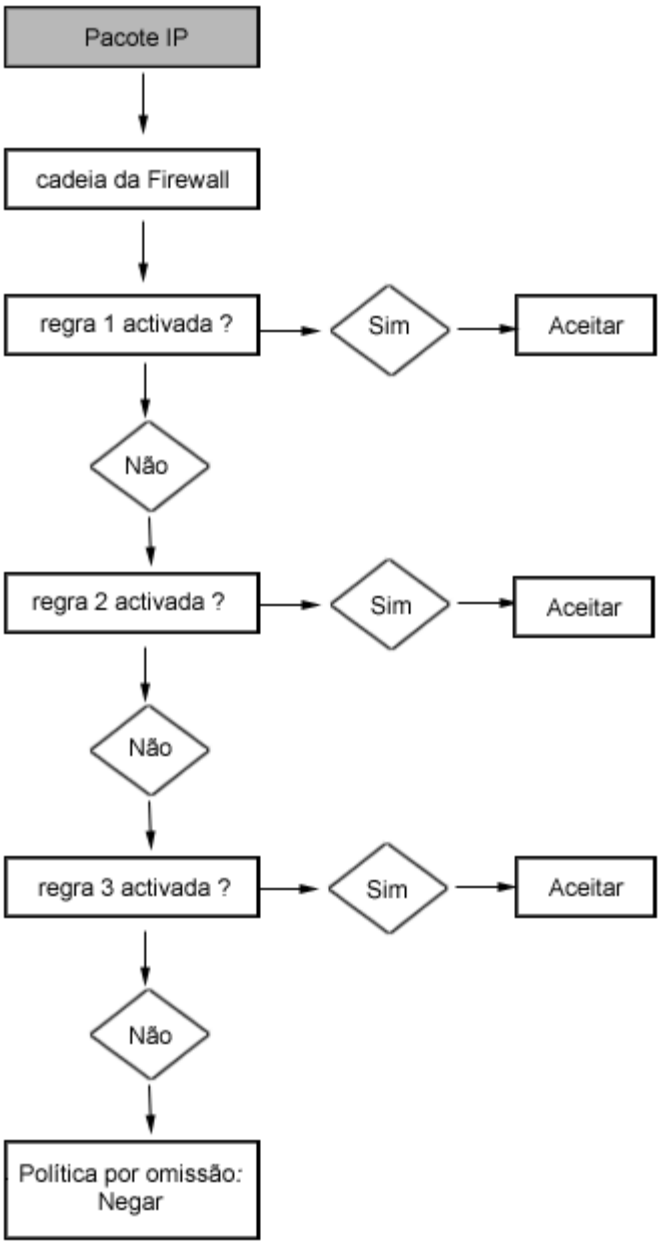

Imagem III-6: Política de negar por omissão

A filosofia de aceitação por defeito é muito mais fácil para configurar e rapidamente a *firewall* está online e a funcionar. Mas força o administrador a prever, antecipar todo o tipo de acessos (por exemplo ataques) ou serviços que se querem filtrar. O perigo é que normalmente só é detectado um tipo de acesso não desejado quando é tarde demais, ou corre-se o risco de mais tarde activar serviços considerados inseguros, sem antes bloquear explicitamente o acesso externo ao serviço. Analisadas bem as coisas, no fim acaba por dar muito mais trabalho, acaba por ser muito mais complicado, e mais susceptível a

erros, configurar uma *firewall* utilizando a política de aceitação por omissão.

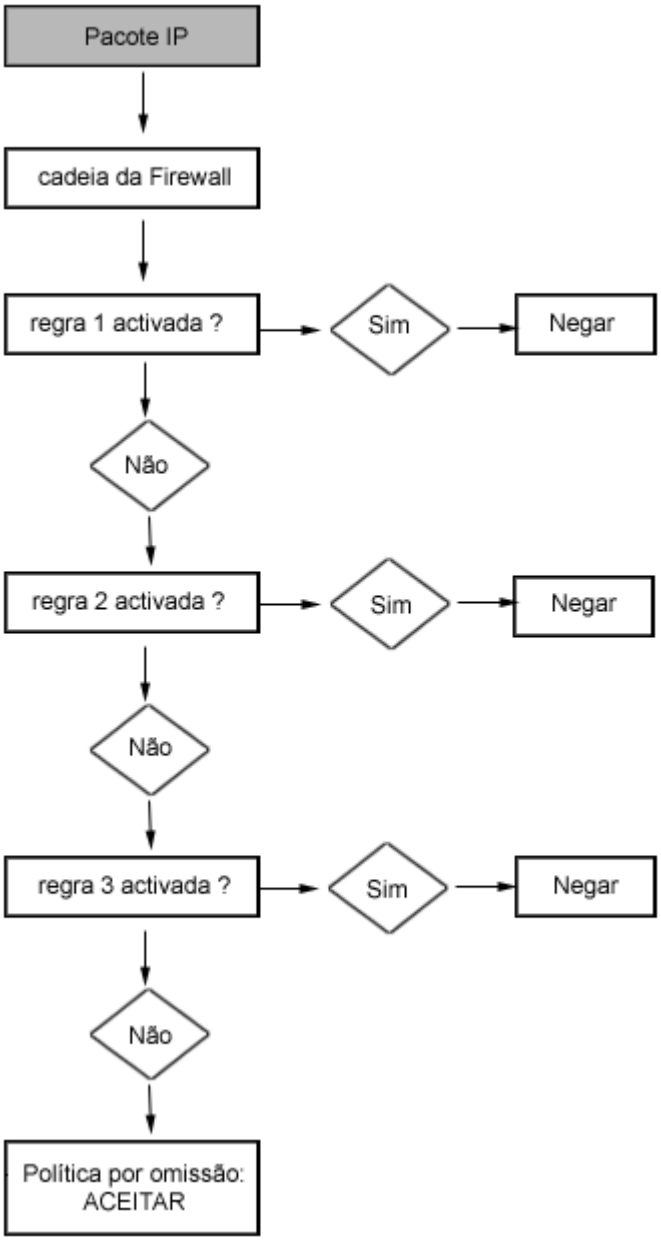

Imagem III-7: Política de aceitar por omissão

Certas firewalls como o *iptables* (que utiliza o mecanismo de *Netfilter* presente no *Kernel Linux* e normalmente é incluído em todas as distribuições de Linux) permitem descartar um pacote de dois modos diferentes:

- Descartar o pacote normalmente (rejeitar).
- Descartar o pacote silenciosamente (negar).

A diferença está que quando um pacote é rejeitado, o pacote além de ser descartado, é enviada uma mensagem de erro *ICMP* ao emissor do pacote. Quando o pacote é descartado silenciosamente, nenhuma mensagem de erro é enviada.

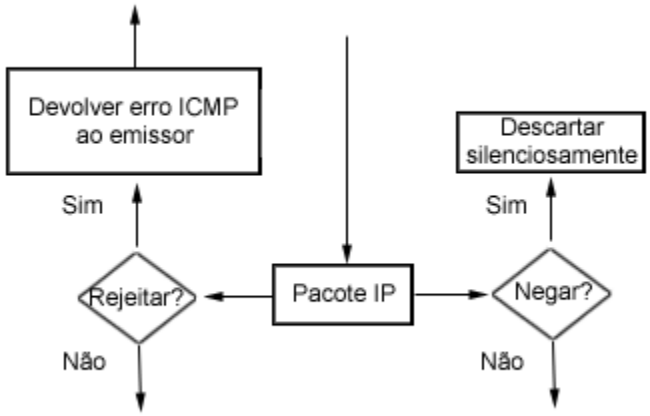

Imagem III-8: Rejeitar (Descartar) Vs Negar (Descartar silenciosamente)

Descartar silenciosamente um pacote é quase sempre a melhor escolha, por três razões. Primeiro enviar uma mensagem de erro aumenta o tráfico. A maioria dos pacotes descartados, são descartados porque eram mal intencionados, e não por significarem tentativas inocentes de aceder a um serviço que nem sequer é oferecido pela *firewall* ou rede interna. Segundo, qualquer pacote que se responda pode ser utilizado num ataque do tipo *denial-of-service* (DoS). Terceiro, qualquer resposta, mesmo uma mensagem de erro, pode dar ao atacante informação que pode ser preciosa.

3.4 Principais funcionalidades uma firewall de filtro de pacotes

O que pode impedir uma *firewall* de filtro-de-pacotes?

- Alguns casos de *spoof* de endereço origem de um pacote.
- Alguma informação útil ser revelada em resposta a *scans* de portas.

- Pacotes *broadcast* intencionalmente mal-formados de modo a identificar sistemas *UNIX*.
- Algumas formas de mapeamento de redes.
- Algumas formas de ataque do tipo *denial-of-service* (DoS)
- Pacotes com *source routing*.
- Algumas formas de *fragmentation bombs*.
- Erros locais que possam afectar máquinas remotas.
- Acesso a serviços privados da *LAN*.
- Alguma protecção contra más configurações do servidor.

As *firewalls* podem possuir serviços de segurança adicionais. Existem soluções de firewalls são constituídas por vários servidores dedicados, filtros, tudo a funcionar conjuntamente. Nestes casos, o termo *firewall* é atribuído e utilizado para referir toda a solução.

Os serviços adicionais podem incluir:

- *Network Address Translation* (*NAT*)
- Port Address Translation (*PAT*)
- Antivírus
- Notificação de eventos
- Filtragem de URLs
- Autenticação de utilizadores
- Encriptação de tráfego (por *VPN* por exemplo)

3.5 Filtragem de pacotes que chegam à firewall

Os pacotes que chegam á rede, chegam à cadeia INPUT. Como foi mencionado antes, podemos filtrar pacotes com base no endereço origem, endereço destino, porta origem, porta destino e flags de ligação *TCP*.

3.5.1 Filtragem com base no endereço origem

Ao nível de um pacote *IP*, o único meio de tentar identificar o emissor de um pacote é verificar o endereço origem do pacote. Isto permite a possibilidade de fazer *spoof* ao endereço origem, em que o emissor, possivelmente com más intenções, coloca no pacote um endereço de origem falso em vez de colocar o seu endereço. O endereço falso

colocado poderá não existir, como pode ser um endereço legítimo pertencendo a outra máquina.

Esta vulnerabilidade pode resultar em:

- num ataque ao sistema local utilizando pacotes aparentando ser locais, e por isso de confiança, podendo ser desse modo ser aceitados no sistema;
- num ataque ao sistema local utilizando pacotes aparentando ser de um terceiro;
- permitir ataques a terceiros aparentando ser originários do sistema local;
- manter o sistema local atarefado a responder a endereços que não existem;
- simplesmente esconder a origem dos pacotes.

Normalmente é impossível detectar se um endereço origem de um pacote foi manipulado (*spoofed*). O endereço em causa pode ser legítimo e existir a possibilidade de chegar ao endereço, mas o pacote pode não pertencer ao endereço origem presente no pacote.

Existem 10 classes de endereços que se devem sempre descartar na interface externa da firewall:

- **O endereço IP da firewall:** é impossível chegar um pacote não spoofed, (com endereço verdadeiro) à firewall alegando que é originário da própria firewall. A única maneira de isto acontecer é se o endereço origem foi ilegalmente alterado (*spoofed*). É impossível detectar pacotes com o endereço origem ilegalmente alterados com o endereço *IP* de terceiros, mas se estiverem spoofed com o endereço local da firewall ou endereços da rede privada, torna-se uma evidência que os pacotes tinham o endereço origem spoofed.

  Nota: Alguns sistemas operativos deixam de responder caso recebam pacotes em que tanto o endereço origem com o endereço destino é o endereço local.

- **Endereços pertencentes a rede interna:** caso nunca se verá pacotes não spoofed chegar à interface externa da *firewall* alegando ser originários da rede interna privada. Isso só será possível se a *LAN* tiver múltiplos acesso à Internet, mas será muito provavelmente um sinal de configuração incorrecta da *LAN*. É possível que isto aconteça num cenário de tentativa por parte de um atacante de explorar relações de confiança locais.

- **Endereços privados Classe A,B e C:** Estes três conjuntos de endereços nas gamas de endereços Classe A,B e C são reservados para uso em *LANs* privadas. Não é previsto o seu uso na Internet. Qualquer máquina de uma rede interna pode usar um endereço destes sem necessitar comprar endereços *IP* registados (públicos). Não é suposto chegarem pacotes com endereços destas classes à *firewall* pelo que devem ser descartados. Estes endereços não são encaminháveis, ou seja, não é possível encontrar um caminho para estes endereços. Assim são muitas vezes utilizados em alguns tipos de ataques do *tipo denial-of-service*.

Classe de Endereços Privados	**Gama de Endereços**
Classe A	10.0.0.0 até 10.255.255.255
Classe B	172.16.0.0 até 172.31.255.255
Classe C	192.168.0.0 até 192.168.255.255

Tabela III-1: Gamas de Endereços Privados

- **Endereços multicast de Classe D:** estes endereços são reservados para utilizarem redes *multicast* de *broadcast*, como *streaming* de *audio* ou *video*. Variam de 224.0.0.0 até 239.255.255.255. Pacotes que cheguem á *firewall* com estes endereços devem ser descartados.

- **Endereços reservados de Classe E:** estes endereços foram reservados para o futuro e para uso experimental e não podem ser utilizados publicamente. Inicialmente a gama de endereços variava de 240.0.0.0 até 247.255.255.255, mas mais tarde foram adicionados endereços o que resultou na gama de 240.0.0.0 até 255.255.255.255. Pacotes que cheguem á *firewall* com estes endereços devem ser descartados.

- **Endereço de Loopback interface:** O endereço de *loopback* interface é um endereço utilizado por vários sistemas operativos para referirem-se a serviços locais. Em vez de enviar o tráfico pelo driver interface da placa de rede, o sistema operativo envia os pacotes pela interface *loopback*, sendo que deste modo, os pacotes não passam pela rede. A gama de endereços loopback é de 127.0.0.0 até 127.255.255.255. Normalmente é utilizado o endereço 127.0.0.1, a palavra-chave *localhost* ou *lo*.

- **Endereços de broadcast mal formatados:** os endereços *broadcast* são endereços especiais que se aplicam a todas as máquinas de uma rede. Qualquer endereço IP normal e o endereço 0.0.0.0 são endereços *broadcast* legítimos, no entanto o endereço 0.0.0.0 é especial pois é usado por servidores e clientes *DHCP*. Em alguns ataques o emissor pode fazer spoof dos pacotes que envia com o endereço 0.0.0.0, pacotes esses que são *unicast*, não *broadcast*, ou seja mal formatados. Pacotes que cheguem á *firewall* com o endereço 0.0.0.0 endereços devem ser descartados.

- **Endereços 0 de Classe A:** Como foi visto antes qualquer endereço entre 0.0.0.0 e 0.255.255.255 é ilegal se forem utilizados em *unicast*.

- **Endereços locais de link:** Por vezes os clientes *DHCP* atribuem a si próprios endereços locais de *link* quando não obtém um endereço do servidor *DHCP*. Estes endereços variam entre 169.254.0.0 até 169.254.255.255. Pacotes que cheguem á *firewall* com estes endereços devem ser descartados.

- **Endereços TEST-NET:** a gama de endereços de 192.0.2.0 até 192.0.2.255 está reservada para redes de teste. Pacotes que cheguem á *firewall* com estes endereços devem ser descartados.

Existem outros endereços que estão reservados pelo *IANA*[12] e que não estão em uso. Estes endereços adicionais podem ser filtrados para benefício da segurança, no entanto existe um perigo: pontualmente alguns desses endereços vão sendo atribuídos pelo *IANA*, tornando-se endereços legítimos, que não devendo ser bloqueados. Para muitos *web servers* comerciais, que até eventualmente foram alvo de ataques de *denial-of-service*, poderá ser do seu melhor interesse reduzir o número de possíveis endereços ilegalmente alterados (*spoofed*) a que os seus *web servers* podem respondem.

Bloquear endereços considerados problema

Outra filtragem possível com base no endereço origem, mas que é menos frequente, é a filtragem de endereços de uma determinada máquina que seja considerada hostil por qualquer motivo, ou mesmo uma gama de endereços inteira, no caso de por exemplo um *ISP* não policiar os seus utilizadores.

[12] IANA – Internet Assigned Numbers Authority

Limitar a recepção de pacotes a um grupo de máquinas remotas

Pode ser necessário limitar o acesso a um grupo restrito de máquinas remotas. Neste caso, as regras de filtragem serão definidas para *IPs* específicos ou para gamas limitadas de endereços *IP* de onde estes pacotes podem ser aceites.

Um primeiro grupo de pacotes recebidos é originário de respostas a pedidos efectuados da *firewall* ou da rede interna. Embora alguns serviços como *HTTP* ou *FTP* possam originar de qualquer máquina remota, outros serviços legitimamente serão originários do *ISP*, ou de outro local específico de confiança. Exemplos de serviços que normalmente são oferecidos apenas pelo *ISP* é o *POP E-mail*, atribuição de endereço *IP* através do *DHCP* e possivelmente resoluções de domínios (*DNS*).

Um segundo grupo de pacotes recebidos é originário de clientes remotos a acederem a serviços locais. Tal e qual o primeiro grupo, alguns serviços, como *HTTP* e *FTP* provavelmente serão esperados de qualquer lado, outros serviços serão oferecidos unicamente a alguns poucos utilizadores remotos. Exemplos de serviços locais mais restritos serão o *telnet*, *SSH*, *portmap* para serviços *RPC* e o *ping*.

3.5.2 Filtragem com base no endereço destino

Filtrar pacotes baseado no endereço destino não tem tanta ciência como filtrar baseado no endereço origem. Normalmente a placa-de-rede ignora pacotes que não lhe são destinados (cujo endereço destino não é o IP da própria interface de rede) excepto se forem pacotes *broadcast*, que serão enviados a todos as máquinas da rede.

O endereço 255.255.255.255 é o endereço destino normalmente utilizado para *broadcast*. Refere-se a todos os hosts no segmento físico de rede imediato, e é apelidado de *broadcast* limitado. Um endereço de *broadcast* pode ser definido mais explicitamente como um endereço de uma rede seguido de 255 no último octecto[13]. Estes endereços de *broadcast* são apelidados de *broadcast* directos de *subnet*.

Caso a gama de endereços esteja segmentada, então é possível haver endereços *broadcast* cujo último número não é 255.

[13] Caso a gama de endereços esteja segmentada, existem endereços broadcast cujo último número não é 255 para cada uma das subredes.

Pacotes *broadcast* enviados para o endereço 0.0.0.0, além de não serem legítimos, são normalmente mal intencionados com o objectivo de tentar verificar se o sistema em causa é uma máquina *UNIX*. Isto acontece por razões históricas, o código de *networking* derivado do *BSD UNIX* retorna uma mensagem de erro *ICMP* tipo 3 em resposta a o endereço 0.0.0.0 ser utilizado como endereço destino de *broadcast*. Outros sistemas operativos normalmente descartam silenciosamente o pacote. Neste caso a mensagem de erro seria o que estaria a ser sondado.

3.5.3 Filtragem com base na porta origem

A porta origem, no computador remoto, identifica o programa que está a enviar o pacote.

Normalmente todos os pacotes de máquinas remotas para um serviço local seguem um padrão, utilizam uma porta na gama de portas não privilegiadas, da porta 1024 á 65.535. Se houver um servidor *WWW* local, os pedidos deverão provir de uma porta remota entre a 1024 e 65.535.

As respostas de servidores remotos a clientes locais seguem outro padrão, utilizam uma porta na gama de portas privilegiadas, abaixo do número 1024. Por exemplo a resposta de um servidor *WWW* vem normalmente da porta 80 remota (embora possa ser configurado para utilizar outra porta), a porta normalmente associada ao serviço *HTTP*.

3.5.4 Filtragem com base na porta local

A porta local, no computador local, identifica o programa ou serviço a que se destina o pacote.

Todos os pacotes e ligações de clientes remotos para um serviço local terão como porta destino, a porta associada ao serviço particular. Por exemplo se um pacote for destinado a um *webserver* local, então terá como porta destino, a porta 80.

As respostas a clientes locais, chegadas de servidores remotos, terão como porta destino uma porta não privilegiada entre 1024 e 65.535.

3.5.5 Filtragem com base nas flags de ligação TCP

As regras da chain que processa pacotes *TCP* podem utilizar as flags de estados de ligação *TCP* para filtrar. Todas as ligações *TCP* utilizam estas flags. Estas flags de estados diferem particularmente no estabelecimento duma nova ligação *TCP* num processo que é denominado *three-way-handshake*. Estas diferenças permitem distinguir o tráfego que chega de clientes remotos e tráfico que chega de servidores remotos.

A *three-way-handshake* funciona utilizando as flags *SYN* e *ACK*. O emissor envia um pacote de inicio de ligação com a flag *SYN* ligada e a flag *ACK* desligada, o receptor confirma a recepção do primeiro pacote respondendo com um pacote apenas com a *flag ACK* ligada. O emissor inicia assim a ligação enviando um segundo pacote também apenas com a flag *ACK* ligada.

Os pacotes *TCP* que chegam de clientes remotos têm a *flag SYN* ligada e a *flag ACK* desligada. Todos os pacotes originários do cliente depois deste primeiro pacote apenas terão a *flag ACK* ligada.

Os pacotes *TCP* que chegam de servidores remotos serão sempre em resposta a ligações iniciadas por um cliente local. Todos os pacotes recebidos de servidores remotos terão apenas a *flag ACK* ligada. As regras da *firewall* podem então exigir que todos os pacotes originários de um servidor remoto tenham a *flag SYN* ligada e a *flag ACK* desligada. Servidores remotos legítimos nunca tentaram iniciar uma ligação para a máquina de um cliente.

3.5.6 Filtragem de probes e scans

Um *probe* é uma tentativa de ligação ou de obter uma resposta de uma porta específica, verificando se um determinado serviço está presente na máquina sondada. Um *scan* é um *probe* a um conjunto de portas e muitas vezes são utilizadas ferramentas para automatizar este processo, muitas vezes denominados *port scanners*.

Um *probe* ou um *scan* por si só é inofensivo. Na *Internet* muitas vezes um probe é a única maneira de saber se uma determinada máquina oferece um particular serviço. Por exemplo, para saber se uma

determinada máquina aloja um *website*, coloca-se o endereço *IP* da máquina num *browser*. Isto pode ser considerado um *probe* à porta *http*.

Infelizmente raramente os *probes* e *scans* são inocentes. Normalmente são o primeiro passo para reunir informação sobre uma máquina, para tentar descobrir vulnerabilidades interessantes antes de lançar um ataque.

Principalmente apartir de 1998, houve um crescimento exponencial dos *scans* pelo mundo inteiro [6]. As ferramentas automáticas estão cada vez mais desenvolvidas e os esforços conjuntos de *hackers* são cada vez mais comuns.

Port Scans

Os *port scans* mais comuns são probes indiscriminados a um grande bloco de portas, possivelmente até ao conjunto completo de portas (ver Imagem III-9). Estes scans estão a tornar-se cada vez menos frequentes, ou pelo menos óbvios, à medida que se desenvolvem ferramentas de *scan* mais sofisticadas, com características *stealth*.

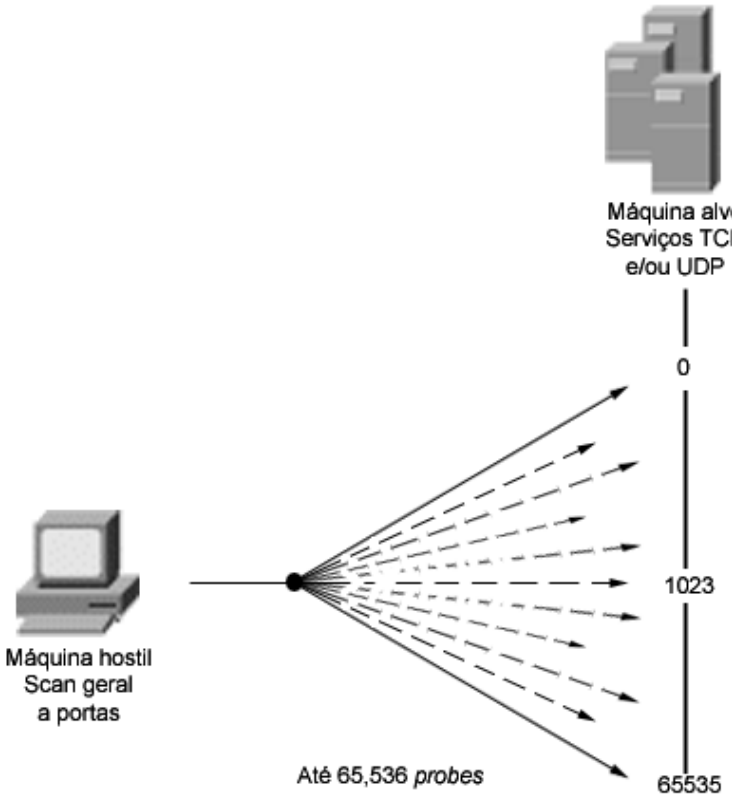

Imagem III-9: *Port Scan* geral

Scans a portas específicas

Os scans a portas específicas, procurando vulnerabilidades específicas (ver Imagem III-10). As ferramentas mais sofisticadas de port scanning apenas tentam verificar se a máquina possui vulnerabilidades específicas, verificando as portas necessárias para saber informações como o *hardware*, sistema operativo e versões de *software*.

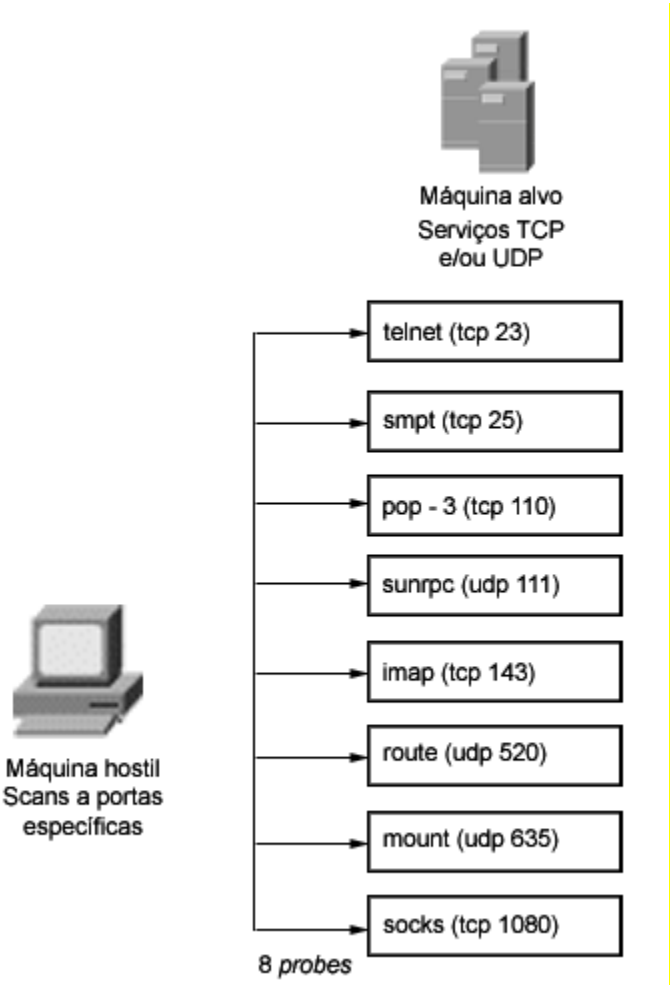

Imagem III-10: *Port Scan* a portas específicas

Portas mais procuradas

Algumas portas são alvos mais procurados por terem vulnerabilidades associadas, como servidores de mail inseguros ou um *daemon portmap* de *RCP* ao descoberto, ou especialmente nos últimos tempos, um servidor de *DNS* vulnerável.

- Pacotes que cheguem a porta 0 são sempre falsos, esta porta é reservada e não pode ser utilizada legitimamente.
- Pacotes *TCP* às portas de 0 à 5 são normalmente sinónimo de um *sscan*[14] [7].
- *Telnet* (porta 23/TCP), *SMTP* (porta 25/TCP), *DNS* (porta 53/TCP/UDP), *POP 3* (porta 110/TCP), *SUN RPC* (porta 111/UDP/TCP), *IMAP* (porta 143/TCP), *SNMP* (porta 161/UDP), *route* (520/UDP), e *mount* (635/UDP) são os serviços e respectivas portas mais famosos pelas suas reconhecidas vulnerabilidades, ou devido a erros de configuração comuns, ou devido a falhas conhecidas no *software*. Por causa destes serviços serem tão comuns é recomendável que o acesso externo a estes serviços seja rigoroso ou mesmo não oferecer estes serviços para fora da rede interna.
- *Netbios* (137/UDP, 138/UDP, 139/UDP), *SMB* no *Microsoft Windows 2000* (445/tcp), Netbus (12345/TCP), *Back Orifice* (31337/UDP) são *probes* bastante comuns. Não são ameaça para sistemas *UNIX/LINUX*, o alvo destas sondas limitam-se a sistemas operativos *Windows*.

Scans do tipo Stealth

Por definição estes scans não são detectáveis. Baseiam-se no comportamento no modo como o protocolo *TCP* reage a pacotes inesperados, ou pacotes com um conjunto de flags de estado de ligação ilegais. Por exemplo, um pacote chega com a *flag ACK* ligada mas não foi pré estabelecida qualquer ligação. Se o referido pacote fosse enviado para uma porta utilizada por um servidor, o *TCP* ao não encontrar nenhuma ligação estabelecida devolveria um pacote *TCP RST* para indicar ao emissor para reiniciar a ligação. Se o referido pacote fosse enviado para uma porta não utilizada, seria devolvido um pacote *TCP RST* para indicar que ocorreu um erro, tal e qual a *firewall* que por defeito devolveria uma mensagem de erro *ICMP*.

O problema torna-se mais complexo visto muitas *firewalls* apenas testarem se a *flag SYN* ou a *flag ACK* estão presentes. Caso surjam pacotes com outras combinações de *flags*, o pacote pode mesmo ser aceite, passando assim para o código do protocolo *TCP*. Assim, e dependendo das combinações das *flags*, o sistema operativo retornará uma mensagem *TCP RST* ou com silêncio. Este mecanismo pode ser utilizado para ajudar a identificar o sistemas operativo de uma máquina. Em qualquer um destes casos estas ocorrências não serão provavelmente detectadas ou registadas em *logs*.

[14] Uma ferramenta de scan popular que surgiu em meados de 1998.

Induzir uma determinado máquina a gerar pacotes *TCP RST* desta modo, pode também ser utilizado para mapear uma rede, determinando que endereços *IP* estão presentes na rede. Isto é especialmente útil se o sistema alvo não é um servidor e a *firewall* que o protege foi configurada para silenciosamente descartar pacotes não desejados.

3.5.7 Filtragem de ataques do tipo Denial-of-service (DoS)

Ataques do tipo *denial-of-service* são sempre baseados no conceito de *flooding*[15] do sistema alvo com o objectivo de tornar indisponível um determinado serviço ou degradar seriamente a ligação à *Internet* do servidor atacado, consumindo recursos de rede tal que pedidos legítimos não possam ser atendidos e no pior dos casos torna o sistema indisponível. Os dois resultados mais comuns é o sistema ficar demasiado ocupado para realizar algo útil e empatar recursos críticos do sistema.

Não é possível protecção total contra este tipo de ataques. Eles podem ocorrer de inúmeras formas, tantas quanto a imaginação do atacante permitir. Qualquer coisa que resulte numa resposta do sistema alvo, qualquer coisa que resulte em o sistema alvo alocar recursos, etc, tudo isto pode ser utilizado num ataque do tipo DoS.

Estes ataques podem ser de diversos tipos como *TCP SYN flooding*, *ping flooding*, *UDP flooding*, *fragmentation bombs*, *buffer overflows* e *ICMP routing redirect bombs*.

TCP SYN Flooding

Este tipo de ataque consome os recurso de sistema do sistema alvo até impossibilitar que novas ligações TCP sejam estabelecidas. O ataque utiliza o protocolo básico de *three-way-handshake* em conjunto com alteração ilegal (*spoof*) de endereço origem.

O atacante faz altera ilegalmente do endereço origem utilizando endereços não encaminháveis (roteáveis) e inicia ligações para um dos serviços *TCP* presentes no sistema alvo. Aparentando ser um cliente legítimo iniciando uma ligação *TCP*, o atacante envia um pacto artificialmente fabricado com a flag *SYN* ligada. A máquina alvo responde aos pedidos com pacotes *SYN-ACK*, confirmando a recepção

[15] Enviar pacotes repetidamente e rapidamente de modo a "inundar" de pacotes um host.

dos pacotes, mas devido ao facto de os endereços origem serem endereços não encaminháveis, não será devolvida nenhuma mensagem *TCP RST* para cancelar a ligação semi aberta.

A última fase do estabelecimento da ligação, a recepção de um pacote com *ACK*, nunca acontecerá. A ligação mantêm-se semi aberta até que a tentativa de ligação resulte em *timeout*. O atacante invade o sistema alvo de pacotes muito mais rapidamente do que os *timeouts* libertam as ligações semi abertas. Eventualmente, todos os recursos se esgotam e deixa de ser possível aceitar novas ligações.

Ping Flooding

Qualquer pacote enviado para um sistema alvo que resulte em numa resposta por parte do sistema pode ser usado para degradar a ligação à Internet desse sistema alvo, forçando o sistema a gastar tempo a responder aos pacotes. Um pacote de *ICMP echo request* (*ping*) é um exemplo de tais pacotes.

Um ataque chamado *smurf*, e as suas variantes, forçam o sistema a gastar os seus recursos a processar *ICMP echo replies*. Isto é conseguido alterando ilegalmente (*spoof*) do endereço origem com o endereço da máquina vítima e fazendo *broadcast* de um *ping* para uma rede inteira de máquinas. Apenas um pacote pode resultar no envio de centenas de milhares de *ICMP echo replies* à máquina vítima. Outro método possível é conseguir instalar *trojans* em sistemas comprometidos pela toda a Internet, e sincronizá-los para enviarem ao mesmo tempo *ICMP echo requests* à máquina vítima. (ver Imagem III-11)

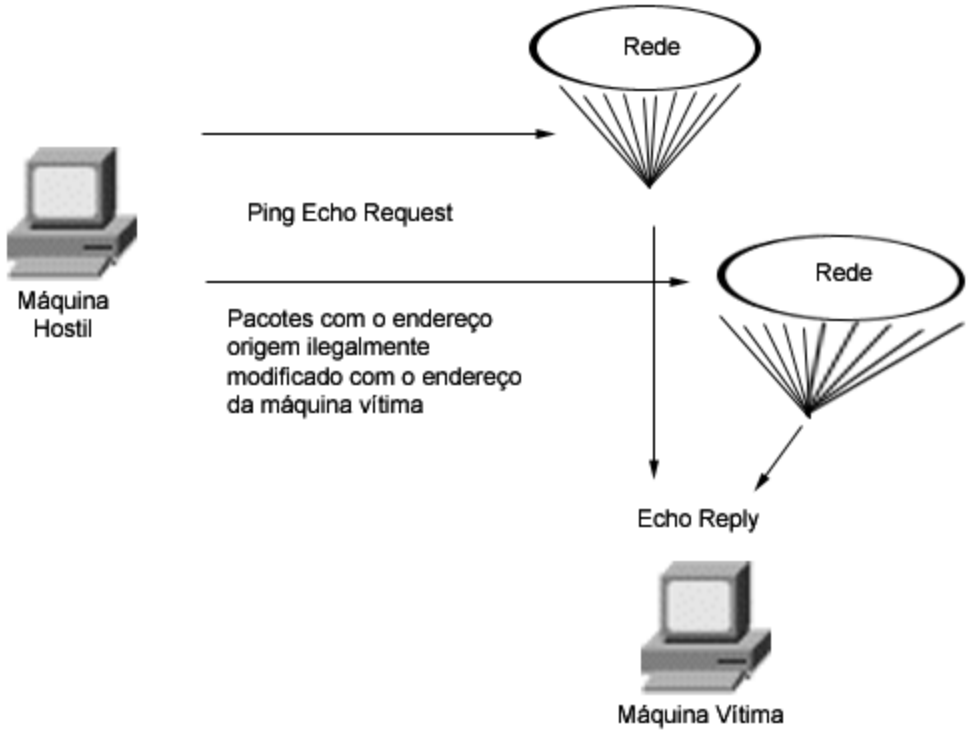

Imagem III-11: *ping flood*

Ping of Death

Um *exploit* [16] mais antigo é denominado *Ping of Death* que consiste em enviar um pacote *ping* particularmente grande. Sistemas operativos *UNIX* e *Linux* não são vulneráveis a este *exploit*, mas muitos outros sistemas, principalmente os mais antigos, são vulneráveis.

O *Ping of Death* pode ser utilizado para bloquear a máquina vítima, de modo a esta ter que fazer um *reboot*. E poderá ser o *reboot* o objectivo do atacante, como em casos em que o atacante tenha previamente instalado um *trojan* na máquina vítima que necessite de uma máquina para ficar activado.

O *Ping* é uma ferramenta básica e útil de *networking*, mas tem um historial de envolvimento em ataques do tipo *denial-of-service* e por esse motivo muitos hosts na Internet deixaram de responder a *Pings* externos ou apenas para um conjunto limitado de máquinas.

[16] Ataque a uma máquina vítima que faz uso de uma vulnerabilidade que a máquina possui.

UDP Flooding

O protocolo *UDP* é especialmente útil num ataque do tipo *denial-of-service*. Ao contrário do protocolo *TCP*, o *UDP* não tem estado de ligação. Não possui mecanismos de controlo de fluxo. Não existem *flags* de ligação. Não são utilizados números de sequência dos datagramas. Nenhuma informação é armazenada de que pacote é esperado a seguir.

Nem sempre é possível identificar se um pacote veio de um servidor ou se veio um de um cliente apenas baseado nas portas origem e destino. Além disso, não havendo informação sobre o estado de uma ligação *UDP*, não é possível distinguir se um pacote que chegou a uma máquina era esperado ou se pelo contrário foi um pacote não solicitado. Perante isto é fácil manter ocupado um sistema a responder a *probes UDP* e desse modo consumir toda a largura de banda existente para a máquina vítima.

Visto os serviços *UDP* serem considerados menos seguros do que serviços *TCP*, é prática geral desactivar todos os serviços *UDP* que não são absolutamente necessários.

O ataque *UDP flood* clássico pode envolver duas máquinas vitimas ou funcionar do mesmo modo que um *smurf Ping flood* (ver Imagem III-12). O atacante envia um pacote originário da porta *UDP echo*, spoofed com o endereço IP da vítima 1, destinado à porta *UDP chargen* da vítima 2. Isto resultará num ciclo infinito de tráfego. Os serviços *echo* e *chargen* são serviços de teste em ambiente de redes. O *chargen* gera uma *string ASCII* e envia ao cliente. O *echo* retorna o que recebeu.

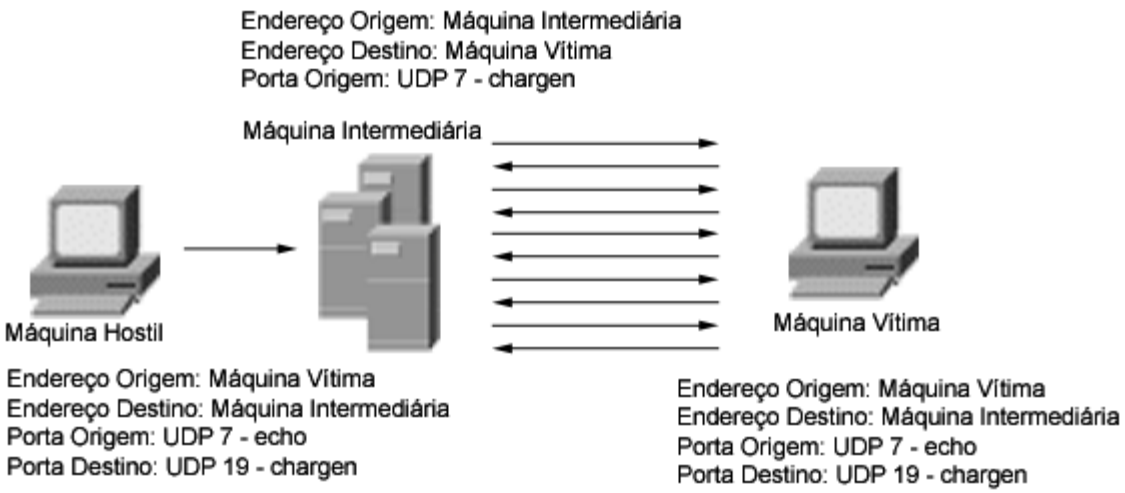

Imagem III-12: UDP *flood*

Fragmentation Bombs

As diferentes infra-estruturas da camada de acesso à rede, como *Ethernet*, *ATM* ou *Token Ring* definem diferentes limites de uma *frame* de *Layer 2*. Como o pacote é passado de um Router para outro ao longo do caminho entre a máquina origem e a máquina destino, os *routers/gateways* podem ter que cortar o pacote em pedaços mais pequenos, chamados fragmentos, antes de os passarem à próxima rede. Na fragmentação legítima, o primeiro fragmento contém as habituais portas origem e destino contidas nos cabeçalhos *UDP* ou *TCP*, mas os fragmentos subsequentes já não.

Apesar do tamanho máximo teórico para um pacote ser de 65.535 bytes, o tamanho máximo para um pacote *Ethernet* (MTU[17]) é de 1500 bytes.

Quanto um pacote é fragmentado, os *routers* intermediários não reconstituem os pacotes. Os pacotes são reconstituídos ou na máquina destino ou no seu *router* adjacente.

Devido ao facto da fragmentação intermediária acabar por ser demasiado custosa em termos de comunicação, os sistemas actuais normalmente tentam encontrar um valor de *MTU* ideal para enviar pacotes para um host remoto específico. Isto é conseguido enviando um pacote com a flag *Don't Fragment* ligada no campo options do

[17] Maximum transmission unit

cabeçalho *IP* do pacote. Se o um router intermediário tiver necessariamente que fragmentar o pacote, descartará o pacote e retornará ao emissor um pacote de erro *ICMP* tipo 3, *fragmentation-required*.

Um tipo de ataques de fragmentação envolve a construção artificial de pacotes muito pequenos. Pacotes com 1 *byte* podem bloquear alguns sistemas. Actualmente os sistemas mais recentes testam esta condição.

Outra técnica em usar fragmentos pequenos é construir um fragmento inicial tal que as portas *UDP* ou *TCP* origem estejam contidas apenas no segundo fragmento. Isto muitas vezes resulta em que as *firewalls* deixem passar o pacote porque a informação que usam para filtrá-lo não se encontra presente. Esta forma de ataque é útil para fazer passar pacotes pela *firewall* que doutro modo não passariam.

O ataque *Ping of Death* referido antes é outro exemplo da utilização de fragmentação. Consiste em enviar um pacote *ICMP* fragmentado, cujo tamanho total é maior que o tamanho permitido. Quando o pacote é reconstituído o tamanho total do pacote excede os 65.535 bytes, levando alguns sistemas a ficarem bloqueados.

Um exemplo clássico de um *exploit* que explora técnicas de fragmentação é o ataque *Teardrop*. Este método pode ser utilizado para fazer passar indevidamente pacotes pela firewall ou simplesmente para empancar um sistema. O primeiro pacote fragmentado é construído de modo a ser aceite pela *firewall*[18]. Se for aceite, os fragmentos subsequentes serão também aceites e reconstruídos no host vítima. Se o primeiro pacote for descartado, todos os pacotes subsequentes serão aceites mas visto o host vítima ficar com um pacote parcial, este será descartado.

O *offset* de dados nos fragmentos subsequentes podem ser alterados para rescrever a informação sobre as portas no primeiro fragmento para aceder a serviços não disponíveis apartir do exterior da máquina. O *offset* pode ser também alterado tal que os *offsets* usados na reconstituição dos fragmentos acabem por resultar em números negativos, e visto que normalmente as rotinas de cópia dos *Kernels* usam números *unsigned*[19], os números negativos são tragados como números inteiros positivos muito grandes, resultando assim em cópias de memória fora do espaço de

18 Muitas firewalls não inspeccionam fragmentos após o primeiro pacote
19 Sem sinal, todos os números representados são positivos.

alocamento normal, rescrevendo dados por cima de dados do Kernel, acabando por resultar no bloqueio do sistema vítima.

As firewalls e as máquinas que façam *NAT*[20] para outras máquinas locais devem ser configurados para reconstituir os pacotes fragmentados antes de os entregarem aos destinatários finais.

Buffer Overflows

Não é possível impedir *exploits* do tipo buffer *overflow* com uma packet-filtering firewall. Este tipo de exploits divide-se em duas categorias principais.

A primeira com o objectivo de bloquear o sistema rescrevendo dados na memória ou na stack de *runtime*.

A segunda categoria requere conhecimentos técnicos especializados de *hardware* e *software* do sistema alvo do ataque. O propósito deste *overflow* é rescrever a runtime *stack* de uma aplicação de modo a *call return stack* contenha um programa e um *jump* para ele, que normalmente abre uma *shell* com privilégios de *root*.

Os *scripts CGI*, utilizados em muitas aplicações de web, são especialmente vulneráveis a *buffer overflows*, excepto se forem tomadas as devidas precauções. Actualmente os *buffer overflows* são das vulnerabilidades mais comuns. Para combater isto, é importante instalar e manter actualizado o software existente nos servidores, instalando todas as actualizações do *software* no momento em que estejam disponíveis para *download*.

ICMP Redirect Bombs

As mensagens de *redirect ICMP* tipo 5 permitem dizer a uma máquina para modificar as suas tabelas de *routing* em memória, adicionando uma rota mais curta. Os *redirects* são enviados por *routers* para as máquinas adjacentes. O seu propósito é informar uma máquina que existe um caminho mais curto.

É suposto as máquinas respeitarem todos as mensagens de *redirect*, adicionando o novo *gateway* à sua *cache de routing*. A excepção a

[20] Network Adress Translation

esta regra está indicada no RFC 1122 [8], "Requirements for Internet Hosts-Communication Layers", Secção 3.2.2.2 "Uma mensagem de *Redirect* DEVE ser descartada silenciosamente caso o endereço da nova *gateway* não esteja ligada na mesma (sub-)rede ou se o emissor do *Redirect* não estiver a um hop de distância da máquina destino."

Se uma máquina utilizar *routing* estático e respeitar as mensagens de *Redirect*, é possível alguém enganar esse sistema levando-o a assumir uma determinada máquina remota como sendo uma máquina local, ou mesmo levando-o a reencaminhar todo o tráfico pela máquina remota.

Outros problemas relacionados com Denial-of-service

Outros problemas são de considerar quando se fala de ataques do tipo *denial-of-service*.

O sistema de ficheiros pode ficar cheio quando por exemplo o host necessita de registar uma grande quantidade de mensagens de erro nos logs ou se receber uma grande quantidade de cópias de um *E-mail* (E-mail *denial-of-service exploits* [9]). É aconselhável configurar o sistema para limitar o uso de recursos e preparar partições separadas para sistemas de ficheiros que possam crescer rapidamente parque haja um limite para esse crescimento e que não prejudique as restantes funções da máquina.

A memória, processador e outros recursos podem ser consumidos por rápidas e repetidas invocações de serviços de rede. Não é possível fazer muito para impedir isto, excepto definir limites em cada serviço individual, utilizar *SYN cookies*, e descartar silenciosamente em vez de devolver mensagens de erro para serviços que não são oferecidos.

3.5.8 Pacotes Source-Routed

Os pacotes *source-routed*[21] utilizam uma opção *IP* raramente utilizada que permite ao emissor de um pacote definir a rota seguida pelo pacote, entre o emissor e a máquina receptora do pacote, ao invés de deixar as decisões de *routing* para os routers intermediários.

[21] Pacotes pré-roteados

Tal como nos pacotes *ICMP Redirect*, esta opção pode ser utilizada para enganar um sistema vítima, fazendo-o pensar que está a comunicar com uma máquina local ou outro host de confiança ou criar o fluxo de pacotes necessário num ataque do tipo *man-in-the-middle*.

Actualmente o *source routing* tem poucas utilizações legítimas e normalmente é ignorado pelos *routers* e os pacotes com essa opção , são descartados pelas *firewalls*.

3.6 Filtragem de pacotes enviados pela firewall

A filtragem de pacotes enviados pela *firewall* pode ser considerada uma tarefa simples caso a rede interna à *firewall* seja considerada um ambiente de confiança. Em redes pequenas ou caseiras, normalmente é adoptada esta filosofia de confiança, mas mesmo para estas redes é importante haver uma filtragem simétrica relativamente aos pacotes que entrem e saiam da *firewall*, principalmente quando a *firewall* protege máquinas com sistemas operativos *Microsoft Windows*. Para redes comerciais ou maiores, a filtragem dos pacotes que saiam da *firewall* é indubitavelmente importante.

Se a *firewall* protege uma *LAN* de sistemas operativos *Microsoft Windows*, controlar o tráfico de saída torna-se muito mais importante. Isto não é uma crítica à *Microsoft*. Devido à grande percentagem de máquinas que utilizam o *Windows*, foram desenvolvidos muitos mais *trojans* para este sistema operativo. Recentemente dezenas de milhares de máquinas com *Windows* foram comprometidas e usadas para coordenar ataques *denial-of-service* distribuídos (DDoS). Por esse motivo especial, é importante filtrar o que sai da rede.

Filtrar os pacotes que saiam também permite correr serviços apenas na *LAN* sem que haja fuga de pacotes para a *Internet*. Não é uma questão de apenas bloquear o acesso externo a serviços privados da *LAN* mas também uma questão de não deixar circular informação sensível pela *Internet*.

Outra razão muito importante para controlar do tráfego que sai da rede local, é travar pacotes de utilizadores mal intencionados. Todas as ilegalidades que possam ser feitas apartir de uma rede interna são cota parte da responsabilidade dos administradores da rede.

O desenvolvimento de *software* em fase de testes também pode resultar em que por acidente alguns pacotes acabem por ir sair para a *Internet*. O mesmo pode acontecer com tráfego local que inadvertidamente possa sair para a *Internet* (como por exemplo tráfego *SNMP*).

3.6.1 Filtragem com base no endereço origem

Filtrar pacotes enviados baseado nos seus endereços de origem é bastante fácil. Para uma rede pequena ou apenas um computador ligado à *Internet*, o endereço origem é sempre o endereço normal *IP* da *firewall*. Não há razão para permitir a saída de pacotes com outro endereço origem, e a *firewall* deve forçar isto.

Para máquinas cujo *IP* seja atribuído dinamicamente pelo seu *ISP*, existe uma excepção que é específica ao *DHCP*, o *broadcast* de pacotes com o endereço origem 0.0.0.0. Nestes casos, o endereço origem 0.0.0.0 deve ser aceite pela *firewall* para o protocolo *DHCP*.

Para *LANs* cujo endereço *IP* da *gateway/firewall* seja atribuído dinamicamente, limitar o envio de pacotes ao endereço *IP* da *firewall* protege a rede de erros comuns de configuração, que se podem confundir com tentativas de enviar pacotes com o endereço origem spoofed ou enviar pacotes com endereços ilegais.

Se os utilizadores da rede ou o *software* que estes usam não 100% de confiança, é importante garantir que o tráfego local contem apenas endereços legítimos e roteáveis para evitar participar em *ataques denial-of-service* que utilizaram spoof do endereço origem.

Este último ponto é particularmente importante. O *RFC* 2827 [10] "Network Ingress Filtering: Defeating Denial Of Service Attacks Which Employ IP Source Adress Spoofing," refere a importância deste ponto. Idealmente todos os *routers* deveriam filtrar todos os pacotes cujos endereços origem são obviamente ilegais e assegurar que o tráfego que é enviado da rede local contém apenas endereços origem roteáveis pertencentes a rede local.

3.6.2 Filtragem com base no endereço destino

Tal e qual os pacotes que chegam à rede, poderá ser mais seguro garantir que certos pacotes que são enviados da rede só podem ser enviados para destinos pré determinados, sejam redes completas, ou máquinas individuais.

A primeira classe de pacotes a filtrar com base no endereço destino são os pacotes enviados a servidores remotos. Embora alguns pacotes como os que são enviados a servidores *web* ou servidores *FTP*, não tenham destino fixo, podem ser enviados a qualquer ponto na *Internet*, outros serviços remotos serão legitimamente apenas oferecidos pelo *ISP* ou hosts específicos com quem há uma relação de confiança. Exemplos de serviços normalmente oferecidos pelo *ISP* são o *POP-3 E-mail*, *DHCP* para atribuição dinâmica de *IP* e o serviço de *Usenet newsgroups*.

A segunda classe de pacotes a filtra com base no endereço destino são os pacotes destinados a clientes remotos que estão a aceder a um serviço oferecido localmente. De novo, alguns dos pacotes em resposta a por exemplo um servidor web local, podem ser enviados para qualquer ponto na *Internet*, mas existem serviços que apenas são permitidos a um grupo restrito de hosts externos. Exemplos de tais serviços são o *telnet*, *SSH*, e serviços *RPC* acedidos via portmap. A *firewall* deve garantir que estes serviços são restritos a um grupo reduzido de máquinas de confiança.

3.6.3 Filtragem com base na porta origem

Definir explicitamente que portas podem ser usadas por um serviço aplica-se a clientes e a servidores. Especificar que portas origem podem ser utilizadas nos pacotes enviados assegura que os programas estão a trabalhar como se espera, e protege outras máquinas externas de tráfego local que poderia escapar para a *Internet*.

As ligações que partam de clientes locais utilizam quase sempre portas não privilegiadas. Ao limitar os clientes a essas portas não privilegiadas e desse modo obrigando os programas clientes a se comportarem como é esperado, assegura-se que máquinas externas são protegidas de possíveis erros originados da rede local.

As ligações que partam de servidores locais utilizam a porta associada ao serviço em questão. Ao limitar que os servidores locais utilizam a porta associado ao serviço, assegura-se que os protocolos dos servidores locais funcionam correctamente. Mais importante, ajuda a proteger serviços locais privados que estejam disponíveis apenas para uso interno. Também ajuda a proteger máquinas externas de serem incomodados com tráfego que deveria estar confinado à rede local.

3.6.4 Filtragem com base na porta destino

Normalmente, os clientes locais ligam-se a servidores remotos utilizando as portas associadas a esses serviços. Utilizando esta perspectiva, pode-se obrigar os clientes locais a ligarem-se a determinados serviços via a porta associada a esse serviço, assegurando assim que o protocolo é correctamente utilizado. Com isto se consegue salvaguardar que clientes locais para uso interno, de inadvertidamente aceder a servidores externos. Segundo impede tentativas de ligação erróneas de sairem da rede local, *port scans* a máquinas externas e outras ligações mal intencionadas.

Os programas servidores locais participarão quase sempre em ligações que foram originadas em portas não privilegiadas. A *firewall* deve garantir isto mesmo.

3.6.5 Filtragem com base nas flags de estado TCP

As regras de filtragem dos pacotes enviados da rede local podem utilizar as flags de estado de ligação *TCP* do mesmo modo que se podem utilizar na filtragem dos pacotes que chegam à rede local. Todas as ligações *TCP* utilizam o mesmo conjunto de estados de ligação.

Os pacotes enviados por clientes locais terão a *flag SYN* ligada no primeiro pacote enviado como parte do esquema *three-way-handshake*. A *flag ACK* no primeiro pacote está desligada. Depois do primeiro pacote terão apenas a *flag ACK* ligada. A *firewall* deve ser configurada para permitir que os clientes locais apenas enviam pacotes em que as *flags SYN* ou *ACK* estejam ligadas.

Os pacotes enviados por servidores locais serão sempre em resposta a ligações iniciadas por um cliente numa máquina remota. Todos os pacotes enviados terão apenas a *flag ACK* ligada. A *firewall* deve ser

configurada para permitir que os servidores locais apenas enviam pacotes em que a *flags ACK* esteja ligada.

3.7 Firewalls de filtro de pacotes em Linux

3.7.1 Iptables

O *iptables* pertence a uma *framework* existente no *Kernel Linux* (desde a versões 2.4.x) que fornece mecanismos de filtro-de-pacotes (*stateful*), *NAT* e *packet-mangling*.

Essa framework assenta no *netfilter*, um mecanismo presente na implementação na pilha de rede (*network stack*) do *Kernel* do *Linux*, que permite a módulos do *Kernel* registarem funções *callback* que são executadas cada vez que um pacote de rede passa por uma série de fases.

O *iptables* é uma estrutura de tabelas que definem conjuntos de regras. Cada regra dentro de uma tabela contém um conjunto de correspondências (*matches*) e um destino de pacote (*target*).

O *netfilter,* o *iptables*, o mecanismo de *connection tracking* e o mecanismo de *NAT* constituem a *framework*.

Esta framework está embebida no próprio *Kernel*, mas a *firewall* incluiu ainda um módulo exterior ao *Kernel*, uma aplicação de configuração com o mesmo nome da firewall, *iptables*. Este módulo de configuração é normalmente fornecido em todas as distribuições do *Linux*.

O *iptables* resulta de significativas evoluções e reestruturações dos seus antecessores *ipchains* (presente nos *Kernel* versão 2.2.x) e *ipfwadm* (presente nos *Kernel* versão 2.0.x).

As principais características incluem:

- Filtro dinâmico de pacotes (*stateful*)
- Todos os tipos de *NAT*
- É possível configurar *proxys* transparentes.

- Manipulação de pacotes (*packet mangling*)
- Estrutura flexível e extensível.
- Existem muitas extensões disponíveis.

O mecanismo de *packet-mangling* permite definir o valor de *TOS* (*type of service*) de um pacote.

Actualmente, esta *firewall* é a mais utilizada em ambiente *Linux*, e a diferença de performance para muitas outras firewalls é significativa. Isto deve-se ao facto de a maior parte da firewall *iptables* correr em modo privilegiado do *Kernel*, evitando assim que os pacotes tenham que "subir" até às camadas superiores da pilha protocolar *TCP-IP* para serem filtrados.

As ultima versão da aplicação de configuração (actualmente a versão 1.2.7a) pode ser obtida em http://www.netfilter.org.

3.7.2 ipchains

O *ipchains* é o precedente do *iptables* no *Linux*. Actualmente, ainda é bastante utilizado mas está a ser progressivamente substituído pelo *iptables* que é mais evoluído e permite um maior grau de controlo do fluxo de pacotes, utilizando regras mais simples.

As principais diferenças entre o *ipchains* e o *iptables* são:

- O *iptables* é uma firewall dinâmica de filtro de pacotes. Mantêm tabelas de estado para as ligações o que aumenta bastante a performance, e permite activar regras com base no estado da ligação.
- O *iptables* separa modularmente as suas funcionalidades, ao contrário do *ipchains* que tem uma estrutura pouco organizada.
  Os módulos principais do *iptables* são o *filter*, o *NAT* e o *packet-mangling*.
- O *iptables* tem capacidades melhoradas de *Logging* em relação ao *ipchains.*
- O *iptables* pode redireccionar o processamento de um pacote para um programa a correr em modo de utilizador.
- O *ipchains* utilizava a aplicação *ipmasqadm* para configurar o redirecionamento de portas. No *iptables* possui essa funcionalidade de base no módulo *NAT*.
- No *ipchains* os pacotes percorrem várias cadeias de regras, enquanto no *iptables* percorrem apenas uma cadeia de regras.

- O *iptables* é mais rigoroso na sintaxe de configuração.

O *ipchains* resultou da evolução do seu antecessor *ipfwadm*.

3.7.3 Outras Firewalls

De seguida são listadas algumas alternativas ao *iptables* e suas principais características.

Check Point Firewall-1 para Linux

O Check Point Firewall-1 tem liderado o mercado das firewall desde a sua introdução no mercado em 1994. Um reconhecida vantagem desta firewall é sua interface de administração, bastante intuitiva o que acabou por a destacar em relação a outros produtos da mesma área.

Por ser bastante dispendiosa, é principalmente procurada por organizações enquanto utilizadores domésticos optam principalmente por outras firewalls *open-source* como o *iptables*.

Num recente sondagem do *IDC*, estimou-se que a Check Point detia cerca de 41% do mercado de *software* de segurança. Esta margem inclui versões para todos os sistemas operativos, plataformas de *hardware* e acessórios de rede.

Oficialmente, a *Firewall-1*para *Linux* apenas suporta as versões 6.0 à 6.2 da distribuição do *Red Hat Linux*, precisamente as que utilizam a versão 2.2.x do *Kernel*. Também existem relatos [11] de instalações de sucesso no *Red Hat Linux* 7.0 e mesmo noutras distribuições como o *Debian* (com o *Kernel* 2.2.12), SuSE 6.2 ou Mandrake 7.0.

SINUS

O SINUS é uma *firewall* de filtro de pacotes *TCP/IP* distribuída livremente para o *Linux* ao abrigo do licenciamento *GNU GPL*. Resultou de uma tentativa de resolver os principais problemas presentes no *ipchains*: endereços *IP* estáticos e uma administração complexa.

A principal vantagem é a facilidade de configuração, através de um interface baseado em *Java* de nome *sfControl,* que remotamente

administra a *firewall*. A SINUS permite alternar entre diferentes configurações sem quebrar as ligações *TCP* existentes.

TIS Firewall Toolkit (FWTK)

O *TIS Firewall Toolkit* (também conhecido como FWTK) foi desenvolvido pela *Trusted Information Systems*, que mais tarde em 1998 se fundiu com *Network Associates*. O *FWTK*, disponível com o código fonte ao abrigo da *TIS Firewall Toolkit Licence*, uma licença do tipo *BSD*, é um conjunto de servidores proxy e ferramentas para auxiliar a construção de firewalls. Contém proxy servers para *FTP*, *Telnet*, *SMTP*, *HTTP*, *Gopher e rlogin*. O *FWTK* contém uma ferramenta, o *netacl* (*Network Access Control List*) que simplifica a gestão do controlo de acesso a serviços da rede. O acesso é baseado nos endereços IP. O *FWTK* inclui ainda o *login-sh*, um programa que substitui a shell inicial dos utilizadores. Esta shell suporta os serviços de autenticação *Security Dynamics SecurID, SecureNet Key* e *Racal Watchword*.

floppyfw

O *floopyfw* é um router/firewall estático para *Linux* que cabe numa disquete. Não possui todas as funcionalidades que actualmente encontramos nas melhores firewalls. Basicamente é um *screening router*.

Suporta listas de acesso, *IP-masquerading* (*NAT*), filtragem de pacotes com acesso a estado das ligações, *routing* avançado. Contém um sistema simples de extensões (*packages*) para o qual existem vários módulos disponíveis para download, com funcionalidades diferentes como manipulação de tráfego (*traffic shaping*).

A distribuição possui ainda um servidor *DHCP* e capacidades de *cache* de endereços *DNS*.

T.Rex Firewall

O *T.Rex Firewall* ajusta-se a sistemas críticos, tolerantes a falha. Existe versões para *Linux*, *AIX* e *Solaris*. Entre as suas características, as mais distintas são: grande disponibilidade, balanceamento de carga, cache de *web*, filtro de conteúdos, *NAT*, suporte *VPN*, uma aplicação avançada de *Proxy* e a possibilidade de produzir até cinquenta e dois tipos de relatórios.

3.8 Conclusão

Neste capítulo foi descrito o funcionamento de uma *firewall* de filtro de pacotes. Uma *firewall* de filtro de pacotes utiliza cadeias de regras associadas aos diferentes sentidos dos pacotes que passam na *firewall*. Cada uma dessas cadeias de regras possui uma política por omissão que pode ser: aceitar por omissão, negar por omissão.

Foram descritas as principais funcionalidades das *firewalls* de filtro de pacotes, nomeadamente tipos de ataques que podem impedir. Foram descritos as principais medidas para configurar uma *firewall* de filtro de pacotes, como impedir os principais tipos de ataques e como filtra-los com uma *firewall* de filtro de pacotes.

Foi também visto a *firewall* de filtro de pacotes actualmente mais utilizada no sistema operativo *Linux*, o *iptables*. As principais diferenças do *iptables* em relação ao seu antecessor, o *ipchains*. Foram referidas outros *firewalls* existentes para o *Linux*.

O próximo capítulo será inteiramente dedicado ao *iptables* e a sua sintaxe e configuração.

4 iptables

4.1 Introdução

Neste capítulo será descrita a sintaxe do programa de administração do *iptables*.

No capítulo 3 foi visto ainda, os conceitos por trás de uma firewall de filtro de pacotes. Cada cadeia de regras tem a sua política por omissão. Cada regra pode ser aplicada não a uma cadeia de regras individual, mas também a uma interface de rede específica, protocolo (TCP, UDP ou ICMP), porta ou tipo de mensagem ICMP. As regras para aceitar, rejeitar (descartar) ou negar (descartar silenciosamente) um pacote são definidas para as chains INPUT, OUTPUT e FORWARD.

No capítulo 3 foram ainda descritas as principais características do *iptables*. Como vimos o *iptables* é uma *firewall* de filtro de pacotes (*stateless* ou *stateful*), que suporta todos os tipos de *NAT* (*Network Address Translation*) e *packet mangling*.

4.2 Fluxo dos pacotes

No iptables, são utilizadas três chains: INPUT, OUTPUT e FORWARD. Os pacotes que chegam passam por um módulo de pré encaminhamento dos pacotes (*routing*), que determina se o pacote será entregue à cadeia de regras INPUT ou à cadeia de regras FORWARD. O fluxo dos pacotes no *iptables* pode ser observado na figura Imagem IV-13.

Se um pacote que chegou à *firewall* for aceite pela cadeia INPUT, o pacote é entregue localmente. Se o um pacote destinado remotamente é aceite pela cadeia FORWARD, o pacote é enviado para a interface apropriada.

Os pacotes de processos locais são passados á cadeia OUTPUT. Se o pacote for aceite, é enviado para a *interface* apropriada. Cada pacote é apenas filtrado uma vez (excepto os pacotes *loopback* que são filtrados duas vezes).

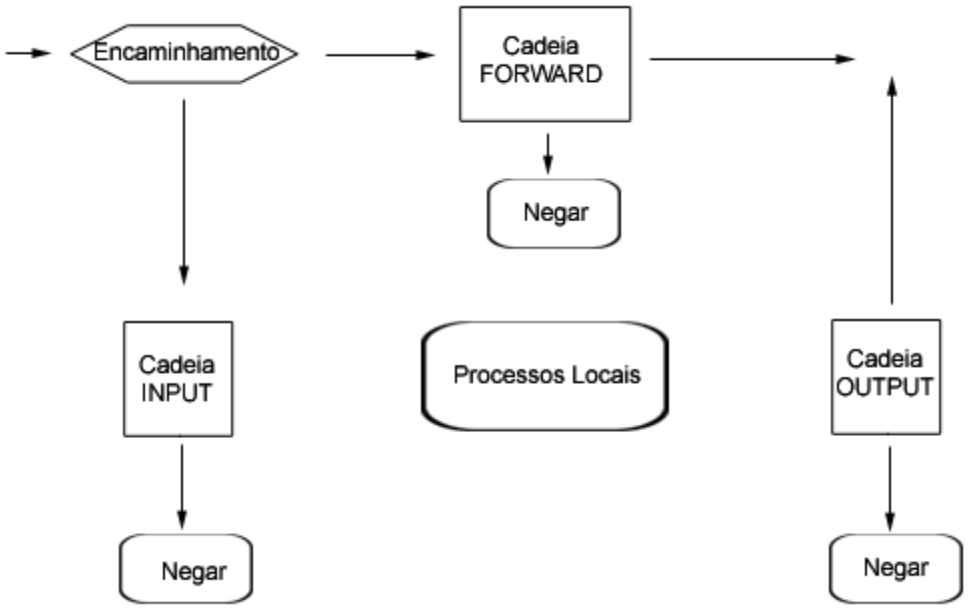

Imagem IV-13: Fluxo dos pacotes no *iptables/Netfilter*

4.3 Tabelas de regras

O *iptables* possui tabelas de regras para diferentes tipos de pacotes processados. Estas tabelas de regras são implementadas como módulos de funcionalidades separadas. Os três módulos primários são a tabela de regras *filter*, a tabela para Network Adress Translation *nat* e a tabela para manipulação de pacotes *mangle*. Cada um destes três módulos tem extensões que são dinamicamente carregadas assim que são referenciadas.

4.3.1 Tabela filter

A tabela filter é a tabela utilizada por omissão. Para se especificar outras tabelas, é necessário utilizar um parâmetro na linha de comandos.

As características básicas da tabela filter incluem:

- Operações relacionadas com as três cadeias de regras base (INPUT, OUTPUT e FORWARD) e as cadeias definidas pelos utilizadores (*user defined chains*).
- Aceitar ou negar (descartar silenciosamente) um pacote (ACCEPT e DROP)

- Operações para encontrar *matchs* no cabeçalho IP, do protocolo, dos endereços origem e destino, das interfaces de entrada e de saída e da manipulação de fragmentos.
- Operações para fazer *match* de informação presentes nos campos dos cabeçalhos TCP, UDP e ICMP.

A tabela filter possui dois módulos de extensão, o *target* e o *match*. A extensão *target* inclui a opção de fazer um REJECT a um pacote, descartar um pacote enviando uma mensagem de erro *ICMP* a notificar o emissor, e possui ainda a funcionalidade LOG que permite que a chegada de determinados pacotes sejam registados no *log* do *Linux*.

Módulo de extensão *match*

O módulo de extensão *match* permite identificar pacotes usando as seguintes informações ou funcionalidades:

- **O estado da ligação TCP**: O estado da ligação actual através das *flags* de estado TCP. Todas as flags podem ser inspeccionadas, e as decisões de filtragem podem ser tomadas com base nos resultados. O *iptables* pode por exemplo tentar detectar *stealth scans*.

  Através destas *flags* é possível perceber se um pacote faz parte de uma ligação previamente estabelecida e aceite pela *firewall*, podendo aceitar o pacote sem que este passe necessariamente por todas as regras que o pacote inicial passou.

- **Campo *options* do TCP**: um pacote TCP pode especificar qual o tamanho máximo de um segmento que o emissor aceita em retorno. A filtragem com base neste valor é possível.
- **Listas de portas**: Listas de portas origem e destino (suportado pelo módulo *multiport*), que pode suportar até 15 portas. Listas de portas (exemplo: 21,22,23,80)e intervalos de portas (exemplo: 21-23) não podem ser combinadas.
- **Endereço ethernet MAC**: Os pacotes que chegam à firewall podem ser filtrados com base no endereço *MAC* do emissor. Isto só funciona como autenticação local visto os endereços *MAC* só estarem presentes nos pacotes entre máquinas e *routers* adjacentes.
- **Utilizador, grupo, processo ou ID do grupo de processos**: Os pacotes gerados localmente podem ser filtrados com base no utilizador, grupo, processo ou ID do grupo de processos do

programa que gerou o pacote. Deste modo, o acesso a determinados máquinas remotas pode ser autorizado a nível da *firewall* baseado no utilizador. Isto é uma opção especial para hosts multifacetados, com multiutilizadores, visto que uma *firewall* não deve, por motivos de segurança, ter contas de utilizadores.

- **Campo TOS (*type of service*)**: O campo *TOS*, presente no cabeçalho do pacote *IP*, tem apenas interesse histórico. Este campo é normalmente ignorado ou usado como diferenciação de serviço por *routers* intermediários, como por exemplo para certo tráfego ter um encaminhamento específico.
- **O campo iptables *mark*:** definido pela tabela *mangle*.
- **Limitar o *matching* de pacotes**: A aceitação de tipos específicos de pacotes pode ser limitada a uma taxa inicial de pacotes por segundo, se essa taxa for alcançada, uma nova taxa, mais restritiva será utilizada nos pacotes subsequentes.

  Se a limitação de *match* de pacotes estiver activada, o comportamento por omissão tem como limite inicial, 5 pacotes por segundo, e se esse limite for alcançado, um limite de 3 pacotes por hora é imposto. Por exemplo, se o sistema fosse invadido por pacotes ping, os primeiros 5 pacotes seriam *matched*, a seguir disso, só seria *matched* um ping passado vinte minutos, e outro *matching* passado outros vinte minutos, independentemente de quantos pings chegassem a *firewall*. O resultado dos pacotes que não fizessem *match*, seriam passados ás regras subsequentes da chain em questão.

Módulo de extensão *target*

- **Target LOG**: O módulo de extensão *target* permite fazer log de certos pacotes filtrados. As mensagens de log podem ter um prefixo definido pelo administrador. As mensagens podem ser associadas a diferentes níveis de *Kernel Logging*, definido no /etc/syslog.conf. Isto permite que o log dos pacotes seja ligado ou desligado, e permite definir os ficheiros de output para os *logs*.
- **Target REJECT**: O target REJECT pode ser opcionalmente especificar que tipo de erro *ICMP* (ou RST para TCP) devolver ao descartar um pacote. O standard IPv4 define que o *TCP* aceita tanto um pacote *RST* ou um pacote *ICMP* como indicação de erro, embora o comportamento por omissão seja um pacote *RST*. O comportamento por omissão é não devolver nada (DROP) ou então devolver um erro *ICMP* (REJECT).

- **Target QUEUE**: O QUEUE é um *target* especial que permite passar o pacote via *netlink* o tratamento do pacote a um programa específico. Se o programa não se encontrar à escuta, o pacote é descartado silenciosamente.
- **Target RETURN**: O *target* RETURN permite retornar imediatamente de uma cadeia de utilizador, sem que o processamento do pacote nessa cadeia tenha acabado.

4.3.2 Tabela nat

Existem três funções gerais de utilização do NAT:

- **NAT tradicional, unidireccional aplicado a pacotes enviados**: Utilizado em redes que utilizam endereços privados.
  - **NAT básico**: apenas tradução de endereço. Normalmente utilizado para mapear endereços privados com um conjunto de endereços públicos.
  - **NAPT ou PAT (Network Adress Port Translation)**: Normalmente utilizado para mapear endereços privados para um único endereço público (exemplo: *Masquerading* do *Linux*).
- **NAT bidireccional**: tradução bidireccional de endereços para pacotes que chegam ou que saiam da *firewall*. Uma possível utilização é conversão bidireccional entre endereços IPv4 e IPv6.
- **Nat duplo**: permite tradução bidireccional dos endereços origem e destino para traduzir os endereços origem e destino. Pode ser utilizado quando os endereços de rede origem e destino colidem, o ser resultado de por um lapso, alguém estar a utilizar um endereço *IP* pertencente a outra máquina. Também pode ser utilizado quando por conveniência os endereços de uma rede foram alterados e o administrador não quer alterar os endereços individualmente em cada máquina da rede.

A tabela *nat* possui módulos de extensão para tradução de endereços origem e destino e para tradução de portas (*PAT*). Estes módulos suportam as seguintes formas de *NAT*:

- SNAT: *Source NAT*
- DNAT: *Destination NAT*
- MASQUERADE: Uma forma especializada de *NAT* utilizado em ligações à *Internet* cujo *IP* é atribuído temporariamente de uma forma dinâmica. O estado de uma ligação é esquecido logo após a ligação se perder.

- REDIRECT: Uma forma especializada de *NAT* que redirecciona um pacote para a máquina local, independentemente do endereço *IP* que este possui no campo destino do cabeçalho *IP*.

Tal como a tabela filter, a tabela *NAT* também possui 3 cadeias de regras base, no caso PREROUTING, OUTPUT e POSTROUTING.

- A cadeia PREROUTING permite alterar o endereço destino dos pacotes antes de os passar a função de encaminhamento (*routing*) (Destination NAT). A alteração do destino pode ser para a própria *firewall* (*proxy* transparente, redirecionamento de portas) ou para outra máquina (*masquerading*, *port forwarding* em terminologia utilizada no *Linux*).
- A cadeia OUTPUT permite alterar o destino de um pacote fabricado na *firewall*, antes das decisões de encaminhamento (DNAT, REDIRECT). Isto é utilizado para transparentemente redireccionar um pacote de saída da *firewall*, para um *proxy* local, mas pode também ser utilizado para fazer *port forwarding* para uma máquina diferente.
- A cadeia POSTROUTING permite modificar o endereço origem de pacotes de saída da *firewall* (SNAT, Masquerade). As alterações são feitas depois da função de encaminhamento (routing).

A diferença entre SNAT regular e MASQUERADE é que em SNAT o estado da ligação é mantido até um período de *timeout* se esgotar. Se uma ligação cair e for restabelecida rapidamente, os programas de rede podem continuar a funcionar sem perturbações visto o seu endereço *IP* não ter mudado, e qualquer tráfico *TCP* que possa ter sido interrompido, é retransmitido.

A distinção entre MASQUERADE e SNAT é uma tentativa de evitar uma situação que ocorria em anteriores implementações do NAT/MASQUERADE, quando uma ligação telefónica caía e o utilizar restabelecia a ligação imediatamente, um novo IP era designado para o a ligação. O novo *IP* não podia ser utilizado porque o antigo endereço IP e a informação mantida pelo *NAT* mantinham-se em memória até passar o tempo de timeout.

A figura Imagem IV-14 ilustra as chains *nat* em relação à função de routing e às chains INPUT,OUTPUT e FORWARD.

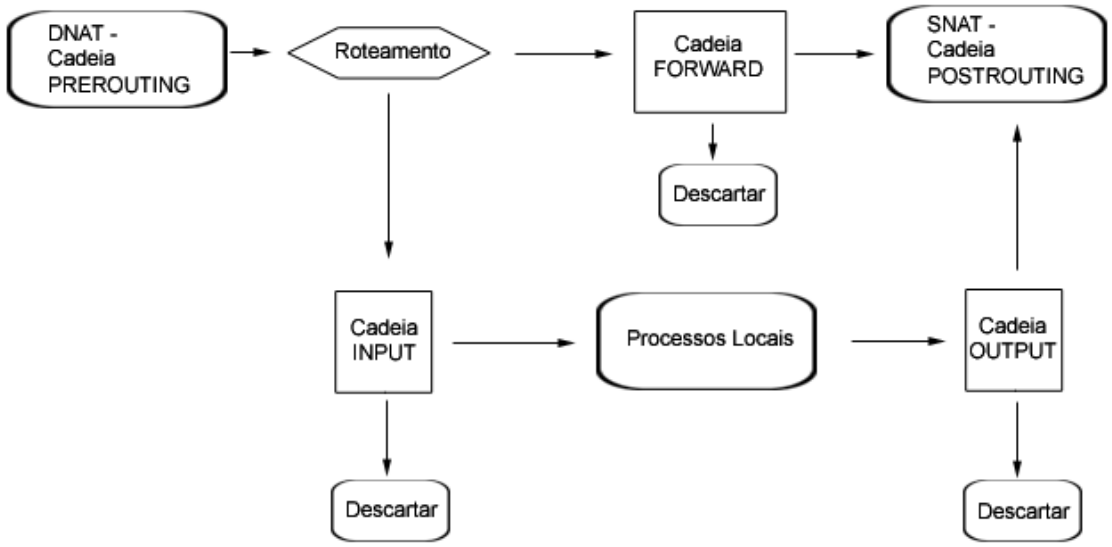

Imagem IV-14: Fluxo de um pacote no *iptables* (inclui *NAT*)

4.3.3 Tabela mangle

A tabela *mangle* permite marcar ou associar um valor (mantido pelo Netfilter/iptables) a um pacote, e modificar o campo TOS (*type of service*) do pacote para casos particulares antes de ser feita a decisão de routing. A tabela *mangle* tem duas cadeias de regras base: PREROUTING e OUTPUT.

- A cadeia PREROUTING permite modificar os pacotes à medida que cheguem à *firewall*, antes de passarem pela função de encaminhamento (routing).
- A cadeia OUTPUT permite modificar os pacotes gerados localmente.

A *firewall/router Linux* local pode ser configurado para respeitar a *flag* TOS definida pela tabela *mangle* ou por uma máquina local.

Existe pouca informação sobre a marcação de pacotes na documentação do *iptables*. Essa marcação é utilizada na implementação do *Linux QoS*[22] e que é utilizada como *flag* de comunicação entre módulos do *iptables*.

[22] Quality Of Service

Esta secção forneceu uma visão geral das características do *iptables*, da sua estrutura geral e das funcionalidades de cada módulo individual. A secção seguinte apresentará a sintaxe utilizada para invocar estas funcionalidades.

4.4 Sintaxe

Como foi referido anteriormente, o *iptables* utiliza o conceito de tabelas de regras separadas para diferentes funções de processamento de pacotes. As tabelas não *default* são especificadas por um opção na linha de comandos.

Existem três tabelas:

- Tabela *filter*: É a tabela default. É a tabela que contém a função de filtragem de pacotes. Inclui três cadeias de regras base:
  - INPUT
  - OUTPUT
  - FORWARD
- Tabela *nat*: A tabela *nat* contém as regras para tradução de endereços origem e destino e de portas. Estas regras funcionam de um modo diferente das regras de filtragem. Inclui três cadeias de regras base:
  - PREROUTING (DNAT/REDIRECT)
  - OUTPUT (DNAT/REDIRECT)
  - POSTROUTING (SNAT/MASQUERADE)
- Tabela *mangle*: A tabela *mangle* contêm as regras para manipular as *flags* especializadas de *routing* de pacotes. Estas *flags* podem depois ser inspeccionadas por regras na tabela *filter*. Inclui duas cadeias de regras base:
  - PREROUTING (pacotes gerados externamente à firewall)
  - OUTPUT (pacotes gerados localmente)

4.4.1 Scripts iptables

O *iptables* é configurado por intermédio de comandos que serão descritos nas secções seguintes. Para cada regra de cada cadeia de regras a introduzir, é necessário um comando. Tudo no *iptables* é configurado via comandos. Esses comandos são normalmente

agrupados num ficheiro *script* que é executado quando o servidor *Linux* reinicia de modo a reconfigurar a *firewall*.

As regras utilizam muitos dados em comum como o endereço *IP* local, o nome da interface externa, etc. É prática comum definir esses dados como variáveis no início do *script* da firewall. As variáveis em *shell scripts* tem o prefixo *$*. Em alguns exemplos nas secções seguintes encontraremos algumas variáveis nos comandos tais como:

- $ENDIP – endereço IP local
- $PORTNPRIV – gama de portas não privilegiadas 1024-65535
- $INTEXT – interface externa da firewall
- $INTLOCAL – interface da rede local

O próprio nome da variável já será um indicativo do que representa.

4.4.2 Comandos da tabela filter

Os comandos da tabela *filter* são fornecidos pelo módulo *ip_tables*. Esta funcionalidade é activada quando o módulo é carregado, o que acontece automaticamente na primeira invocação do comando *iptables*.

Operações em chains

A tabela Tabela V-2 apresenta as operações com chains.

Comando	**Descrição**
-N | --new-chain <cadeia>	Cria uma cadeia de regras de utilizador.
-F | --flush [<cadeia>]	Apaga o conteúdo de uma cadeia, ou o conteúdo de todas as cadeias, caso nenhuma tenha sido especificada.
-X | --delete-chain [<cadeia>]	Apaga uma cadeia de utilizador, ou todas as cadeias caso nenhuma tenha sido especificada.
-P | --policy <cadeia> <policy>	Define a política default para uma das cadeias base,

	INPUT, OUTPUT e FORWARD. As políticas possíveis são ACCEPT ou DROP.
-L | --list [<cadeia>]	Listas as regras de uma cadeia, ou todas as cadeias caso nenhuma tenha sido especificada.
-Z | --zero	Reinicia as contagens de bytes e pacotes associadas com cada cadeia.
-h | <comando> -h	Lista os comandos iptables e as suas opções, ou se for especificado algum comando, lista a sintaxe e opções desse comando.

Tabela V-2: Operações em cadeias de regras

Opções do comando de Listar chains (-L)

O comando *List* tem opções adicionais. (Tabela V-3).

Comando	**Descrição**
-L –n | --numeric	Lista os endereços IP e portas numericamente, em vez de listar por nome.
-L –v | --verbose	Lista informação adicional acerca de cada regra, como as contagens de bytes e pacotes, opções da regra, interfaces de rede relevantes, etc.
-L –x | --exact	Lista os valores exactos das contagens, em vez de contagens arredondadas.
-L –line-numbers	Lista a posição da regra dentro da cadeia a que pertence.

Tabela V-3: Opções do comando List

Operações aplicadas a uma regra

Na Tabela V-4 podemos ver os comandos mais frequentes para criar e apagar regras dentro de uma cadeia.

Comando	Descrição
-A | --append <cadeia>	Adiciona uma regra ao fim de uma cadeia.
-I | --insert <cadeia>	Adiciona uma regra ao início de uma cadeia.
-D | --delete <cadeia> <número da regra>	Apaga um regra na posição indicada por parâmetro de uma cadeia.

Tabela V-4: Operações aplicadas a uma regra

Operações básicas de *match*

Na Tabela V-5 estão listadas as operações básicas de *match* da tabela *filter*.

Comando	Descrição
-i | --in-interface [!] [<interface>]	Para pacotes nas cadeias de INPUT ou FORWARD, ou nas suas sub-cadeias definidas pelo utilizador. Específica a interface a que a regra se aplica. Se não for especificada uma interface, a regra é aplicada a todas as interfaces.
-o | --out-interface [!] [<interface>]	Para pacotes nas chains OUTPUT ou FORWARD, ou nas suas sub-cadeias definidas pelo utilizador. Especifica a interface a que a regra se aplica. Se não especificada uma interface, a regra é aplicada a todas as interfaces.
-p | --protocol [!] [<protocolo>]	Específica o protocolo IP a que a regra se aplica. As

	hipóteses permitidas são *tcp*, *udp*, *icmp* e *all*. O valor do parâmetro *protocolo* pode ser o nome ou o valor numérico, listado em /etc/protocols
-S | --source | --src [!] <endereço>[/<máscara>]	Específica o host ou rede do endereço origem do pacote, no cabeçalho IP.
-d | --destination | --dst [!] <endereço>[/<mascara>]	Específica o host ou rede do endereço destino do pacote, no cabeçalho IP.
-j | --jump <target>	Específica o *target* (destino de regras) para o pacote caso haja um *match* da regra. Os *targets* por defeito são o ACCEPT e o DROP, ou uma cadeia definida pelo utilizador.

Tabela V-5: Operações básicas de *match*

Nota: Os *targets* (destinos de regras) são opcionais. Se um pacote fizer um *match* com uma regra que não possua um *target*, os contadores de pacotes são actualizados e o pacote segue para a próxima regra da cadeia.

Operações de *match* TCP

Na tabela Tabela V-6 estão listadas as operações de *match* no cabeçalho TCP da tabela *filter*.

–p tcp	**Descrição**
--source-port | --sport [!] <porta>[:<porta>]	Este parâmetro específica a porta origem.
--destination-port | --dport [!] <porta>[:<porta>]	Este parâmetro específica a porta destino.
--tcp-flags [!] <máscara>[,<máscara>] <activados>[,<activados>]	Este parâmetro testa os bits de uma lista de máscaras, e que bits é que devem estar activados para haver um *match*.
[!] --syn	A flag SYN deve estar ligada (início de uma

	ligação).
--tcp-option [!] <número>	A única opção legal do TCP é o tamanho máximo do pacote que um host está disposto a aceitar.

Tabela V-6: Operações de *match* TCP

Operações de *match* UDP

Na Tabela V-7 estão listadas as operações de *match* no cabeçalho UDP da tabela *filter*.

–p udp	**Descrição**
--source-port | --sport [!] <porta>[:<porta>]	Este parâmetro específica a porta origem.
--destination-port | --dport [!] <porta>[:<porta>]	Este parâmetro específica a porta destino.

Tabela V-7: Operações de *match* UDP

Operações de *match* ICMP

Na Tabela V-8 estão listadas as operações de *match* no cabeçalho UDP da tabela *filter*.

–p icmp	**Descrição**
--icmp-type [!] <tipo>	Especifica o tipo de pacote ICMP, sendo que *tipo* pode ser um nome ou um número. O tipo de pacote ICMP é usado em detrimento da porta origem.

Tabela V-8: Operações de *match* ICMP

Os nomes dos tipos de pacotes ICMP mais utilizados e seus valores numéricos são:

- echo-reply (0)
- destination-unreachable (3)
    - network-unreachable
    - host-unreachable

  - protocol-unreachable
  - port-unreachable
  - fragmentation-needed
  - network-unknown
  - host-unknown
  - network-prohibited
  - host-prohibited
- source-quench (4)
- redirect (5)
- echo-request (8)
- time-exceeded (10)
- parameter-problem (11)

O *iptables* suporta outros tipos e *subtipos* de mensagens menos comuns ou específicas de *routing*.
Para se ver a lista inteira existe o comando: iptables –p icmp –h.

4.4.3 Comandos do módulo de extensões target da tabela filter

O módulo de extensão *target* da tabela *filter* permite fazer log dos pacotes, descartar um pacote enviando um pacote de erro ICMP.

Operações com a extensão *target* LOG

Na Tabela V-9 estão listadas as opções disponíveis para a *target* LOG.

-j LOG	**Descrição**
--log-level <nível de prioridade do syslog>	O <nível de prioridade syslog> pode ser o número ou *keyword* correspondente das prioridades syslog (ver /usr/include/sys/syslog.h). Os níveis são emerg (0), alert (1), crit (2), err (3), warn (4), notice (5), info (6) e debug (7)
--log-prefix <"string descriptiva">	O prefixo é uma string entre aspas que constará no inicio da mensagem de log da regra em questão.
--log-ip-options	Este parâmetro inclui no log as opções presentes no

	cabeçalho IP.
--log-tcp-sequence	Este parâmetro inclui no log o número de sequência de pacotes TCP.
--log-tcp-options	Este parâmetro inclui no log as opções presentes no cabeçalho TCP.

Tabela V-9: Operações com a *target* LOG

Operações com a extensão *target* REJECT

Na Tabela V-10 estão listadas as opções disponíveis para a *target* REJECT.

-j REJECT	**Descrição**
--reject-with <ICMP type 3>	Por defeito, quando um pacote é rejeitado, é enviado um pacote ICMP type 3 *icmp-port-unreachable*. Outros tipos de mensagens ICMP type 3 podem ser retornados, incluindo *icmp-net-unreachable, icmp-host-unreachable*, *icmp-proto-unreachable, icmp-net-prohibited* e *icmp-host-prohibited*.
--reject-with tcp-reset	Os pacotes TCP podem ser rejeitados com um pacote TCP RST, em detrimento de uma mensagem de erro ICMP.
--reject-with echo-reply	Os pacotes ping *echo-request* podem ser rejeitados com um falso *echo-reply*. Ou seja, a firewall responde ao *echo-request* mas sem passar o pedido ao host local.

Tabela V-10: Operações com a *target* REJECT

4.4.4 Comandos do módulo de extensões match da tabela filter

O módulo de extensão *match* da tabela *filter* permite fazer *match* de pacotes baseado em informação dos cabeçalhos *TCP*,*UDP* e *ICMP*, estado de ligação, listas de portas, acesso ao endereço *MAC* e ao campo *TOS* do cabeçalho *IP*.

Para carregar o módulo da extensão *match* é necessário utilizar o parâmetro –m ou --match, seguido de opções de *match*.

Operações com a extensão *match multiport*

A extensão *match multiport* permite especificar listas de até 15 portas separadas por vírgulas. Não é permitido espaços em branco entre as vírgulas e os números das portas. Não é permitido conjugar na lista números de portas com gamas de portas.

Para carregar o módulo da extensão *match multiport* é necessário utilizar o parâmetro *–m multiport* e este tem obrigatoriamente que ser precedido pelo parâmetro *–p <protocolo>*.

Na Tabela V-11 estão listadas as opções disponíveis para a extensão *match multiport*.

-m | --match multiport	**Descrição**
--source-port <port> [,<port>]	Especifica a porta origem
--destination-port <port> [,<port>]	Especifica a porta destino
--port <port> [,<port>]	Para casos em que a porta origem e destino são a mesma.

Tabela V-11: Operações com a extensão *match multiport*

Exemplos:

A regra seguinte descarta pacotes *UDP* que chegam a portas associadas ao serviço *NFS* e *lockd*:

```
iptables –A INPUT –i eth0 –p udp\
```

```
        -m multiport –-destination-port 2049,4045 –j DROP
```

A regra seguinte rejeita (descarta silenciosamente) ligações a máquinas remotas, às portas associadas aos serviços TCP *NFS*, *socks* e *squid*:

```
iptables –A OUTPUT –o eth0 –p tcp\
        -m multiport –-destination-port 2049,1080,3128 --syn –j REJECT
```

A regra seguinte aceita ligações a máquinas remotas, às portas associadas aos serviços HTTP e S-HTTP:

```
iptables –A OUTPUT –o eth0 –p tcp\
        -m multiport –-destination-port 80,443 --syn –j ACCEPT
```

Operações com a extensão *match limit*

A extensão *match limit* permite fazer *match* limitado de pacotes. Esta função é util para prevenir situações em que pode a *firewall* pode ser inundada de pacotes de certo tipo. Existem várias aplicações para esta função tais como a limitação das mensagens de *log* ou a limitação do número de pacotes aceites.

A Tabela V-12 lista as opções disponíveis para a extensão *match limit*.

-m | --match limit	**Descrição**
--limit <tempo>	Número máximo de pacotes que são *matched* pela regra no período de tempo especificado. O valor por omissão é 3 *matches* por hora (3/hour). Os especificadores horários são */second*, */minute*, */hour* e */day*.
--limit-burst <número>	Especifica o número (*burst)* de pacotes que podem ser *matched* dentro de um limite temporal antes de activar o limite no *matching*. O valor por omissão é de 5 *matches*.

Tabela V-12: Operações com a extensão *match limit*

O número (*burst)* inicial de pacotes que podem ser *matched* antes de aplicar o limite no *matching* é incrementado sempre que um pacote é *matched*. Se não chegar nenhum pacote dentro do limite de tempo específicado, o esse número é decrementado (caso seja superior a zero).

A regra seguinte limita o *logging* de *matches* de pacotes *ping* a um por segundo depois de cinco pacotes *ping* (*equo-requests*) serem *matched* num segundo:

```
iptables –A INPUT –i eth0 \
        -p icmp –icmp-type echo-request \
        -m limit ––limit 1/second –j LOG
```

As conjugação das duas regras seguintes limitam a aceitação de pacotes *ping* a um por segundo depois de cinco pacotes *ping* (*equo-requests*) serem *matched*:

```
iptables –A INPUT –i eth0 \
        -p icmp –icmp-type echo-request \
        -m limit ––limit 1/second –j ACCEPT

iptables –A INPUT –i eth0 \
        -p icmp –icmp-type echo-request –j DROP
```

Operações com a extensão *match state*

As *firewalls* estáticas de filtro-de-pacotes examinam o tráfego pacote a pacote sem estabelecer qualquer relação entre eles. Não existe qualquer contextualização das ligações estabelecidas e respectivas trocas de pacotes.

A extensão *match state* adiciona ao *iptables* a capacidade de contextualizar os pacotes, associando-os pelo facto de pertencerem à mesma ligação. A informação de estado é gravada quando uma ligação *TCP* ou uma troca de pacotes *UDP* é inicializada. Os pacotes subsequentes de uma ligação são identificados como sendo parte de uma ligação preestabelecida e podem ser examinados com base em não apenas a informação *estática* (endereços origem e destino, portas origem e destino, etc.), mas também pelo contexto da sua ligação. Pacotes *ICMP* podem ser associados a uma determinada ligação.
Deste modo algum agrupamento dos pacotes existente em camadas superiores é reconhecido ao nível da filtragem de pacotes.

As vantagens de manter informação sobre o estado das ligações são menos evidentes em ligações *TCP*, devido ao facto de estas nativamente

manterem alguma informação sobre o seu estado. Em ligações UDP, esta função permite identificar respostas a datagramas previamente enviados. Por exemplo, se a firewall enviar um pacote *UDP* com um *DNS request*, isto constituirá uma nova sessão de troca de pacotes *UDP*, o que permitirá à *firewall* reconhecer e aceitar dentro de uma determinada janela temporal, uma resposta *UDP* originada da porta da máquina a quem foi enviado inicialmente o pacote *UDP* com o *DNS request*.

O reconhecimento do contexto de uma ligação permite reconhecer um fluxo de pacotes. Isto permite que um pacote seja processado pela *firewall* sem percorrer necessariamente o caminho normal das cadeias. O processamento de ligações *TCP* é particularmente optimizado. Devido à complexidade de certos protocolos *TCP*, muitas vezes o número de regras necessárias para aceitar a passagem desses protocolos é grande. Ao manter tabelas de estado de ligações, é possível comprimir quase as regras relativas a um protocolo *TCP* numa única regra de aceitação inicial. O número de filtros é assim menor.

A principal desvantagem de manter tabelas de estado de ligações deve-se ao facto de ser necessária mais memória do que as regras standard de *firewalls*. Por exemplo, um router com 70.000 ligações simultâneas requer uma grande quantidade de memória para manter uma tabela de estado de ligação para cada ligação. A manutenção de informação do estado de ligações, por motivos de performance, é normalmente efectuada em *hardware*, podendo assim pesquisar as tabelas simultaneamente. Caso não haja memória disponível, eventualmente para novas ligações não existirá informação sobre o estado da ligação, e nesse cenário, uma regra que aceite um pacote por este ser reconhecido por fazer parte de uma ligação preestabelecida falhará. Nestes casos, a *firewall* deve passar à próxima regra na cadeia que se encontra em processamento.

Criar, pesquisar e descartar uma tabela de estado de uma ligação significa algum tempo de processamento, um *overhead*. Em certos protocolos tais como por exemplo o *FTP* ou certos protocolos multimédia em que existe uma grande troca de pacotes, o custo deste *overhead* é nulo. Mas em ligações simples como *DNS* ou *NTP*, o custo deste *overhead* é pesado. Nestes casos, manter o e estado da ligação pode significar tanto tempo de processamento, como simplesmente atravessar todas as regras da dadeia processada.

A Tabela V-13 lista as opções disponíveis para a extensão *match state*.

-m | --match state	**Descrição**
--state <estado> [,<estado>]	Existe um *match* caso o estado da ligação esteja presente na lista. Os valores possíveis de <estado> são NEW,ESTABLISHED,RELATED e INVALID.

Tabela V-13: Operações com a extensão *match state*

O estado de uma ligação *TCP* ou o estado de uma troca de pacotes *UDP* pode ser mantido, permitindo que novas trocas de pacotes sejam filtradas como sendo novas ligações (NEW), parte de ligações previamente estabelecidas (ESTABLISHED), relacionadas com ligações previamente estabelecidas (RELATED), ou ligações inválidas (INVALID).

- NEW é equivalente ao inicial pedido de inicio de ligação *TCP*, com a *flag SYN* activada, ou ao primeiro pacote de uma troca de pacotes *UDP*.
- ESTABLISHED refere-se a pacotes TCP em transito com a *flag ACK* activada, a respostas UDP trocadas entre os mesmos *hosts* e as mesmas portas e a mensagens *ICMP echo-reply* enviadas em resposta a *ICMP echo-request*.
- RELATED refere-se apenas a mensagens de erro *ICMP*. Para outros casos semelhantes como o canal de dados do protocolo *FTP*, existem módulos ou ALGs que permitem reconhecer as ligações relacionadas. Com estes módulo instalados, o RELATED pode ser utilizado para os protocolos em questão.
- INVALID refere-se a pacotes inválidos como por exemplo um pacote de erro *ICMP* que não surgiu em resposta a um pacote previamente enviado, ou um *ICMP echo-reply* que não surgiu em resposta a um *ICMP echo-request*.

A utilização do *match state* ESTABLISHED permite que as regras de filtragem correspondentes a um dado serviço sejam reduzidas a uma única regra. Por exemplo, uma ligação de um cliente remoto a um web server local necessita apenas que seja aceite o primeiro pacote *TCP SYN*. Um cliente de *DNS* necessita apenas que seja aceite o primeiro pacote UDP com o DNS request.

Nos casos em que uma ligação se estende por um longo período temporal, como uma sessão *FTP* ou *telnet*, existe a possibilidade de uma tabela de estado de ligação ser prematuramente descartada antes da sessão ser finalizada. Nesses casos certos pacotes podem não ser reconhecidos como fazendo parte de uma ligação anteriormente aceite

pela *firewall*. Por esse motivo é prudente introduzir alguma redundância quando se utiliza regras *stateful* com a extensão *match state*.

As seguintes regras exemplificam a utilização da extensão *match state* para uma ligação telnet, introduzindo alguma redundância para o caso da tabela de estado de ligação ser prematuramente descartada:

```
# utiliza uma política de negar por omissão
iptables –A INPUT –m state \
        --state ESTABLISHED,RELATED – j ACCEPT

iptables –A OUTPUT –m state \
        --state ESTABLISHED,RELATED – j ACCEPT

iptables –A OUTPUT ––out-interface $INTEXT –p tcp \
        -s $ENDIP --source-port $PORTNPRIV \
        -d $SERVIDOR_TELNET_REMOTO –destination-port 23 \
        --state NEW –j ACCEPT

iptables –A OUPUT ––out-interface $INTEXT –p tcp ! – syn \
        -s $ENDIP --source-port $PORTNPRIV \
        -d $SERVIDOR_TELNET_REMOTO –destination-port 23 \
        -j ACCEPT

iptables –A INPUT ––in-interface $INTEXT –p tcp ! – syn \
        -s $SERVIDOR_TELNET_REMOTO –source-port 23 \
        -d $ENDIP –destination-port $$PORTNPRIV \
        -j ACCEPT
```

Operações com a extensão *match mac*

A Tabela V-14 lista as opções disponíveis para a extensão *match mac*.

-m | --match mac	**Descrição**
--mac-source [!] <endereço MAC>	Faz o *match* de pacotes com o endereço *ethernet* do emissor. O endereço é especificado na forma xx:xx:xx:xx:xx:xx

Tabela V-14: Operações com a extensão *match state*

Os endereços *MAC* só estão presentes em pacotes enviados por hosts locais, no mesmo segmento ou *LAN* da *firewall*. A extensão *match mac* pode ser utilizada com o parâmetro *in-interface,* nas cadeias INPUT, PREROUTING e FORWARD.

O seguinte exemplo permite ligações ssh de um host específico e local:

```
iptables –A INPUT ––in-interface $INTLOCAL –p tcp ! – syn \
        -s $ENDIP --source-port $PORTNPRIV \
        -d $SERVIDOR_TELNET_REMOTO –destination-port 23 \
        -j ACCEPT
```

Operações com a extensão *match owner*

A Tabela V-15 lista as opções disponíveis para a extensão *match owner*.

-m | --match owner	**Descrição**
--uid-owner <id-utilizador>	Faz o *match* de pacotes enviados pelo utilizador específicado.
--gid-owner <id-grupo>	Faz o *match* de pacotes enviados pelo grupo de utilizadores específicado.
--pid-owner <id-processo>	Faz o *match* de pacotes enviados pelo número do processo específicado.
--sid-owner <id-sessão>	Faz o *match* de pacotes enviados pelo número da sessão específicado.

Tabela V-15: Operações com a extensão *match state*

Esta extensão refere-se ao criador e emissor de um pacote, e só pode ser utilizada com a cadeia OUTPUT.

O exemplo seguinte demonstra como aceitar ligações *SSH* de um determinado utilizador, para uma máquina local:

```
iptables –A OUTPUT –o $INTLOCAL –p tcp \
        -s $ENDIP --sport $PORTNPRIV \
        -d $SERVIDOR_SSH_LOCAL –dport 22 \
        -m owner ––uid-owner $USERID_ADMIN \
        --gid-owner $GROUPID_ADMIN -j ACCEPT
```

Operações com a extensão *match mark*

A Tabela V-16 lista as opções disponíveis para a extensão *match mark*.

-m | --match mark	Descrição
--mark <valor> [/<máscara>]	Faz o *match* de pacotes aos quais foram atribuidos determinados valores atribuídos por outros módulos do *iptables*.

Tabela V-16: Operações com a extensão *match state*

O valor a máscara são variáveis *unsigned long*. Se for específicada uma máscara, o valor e a máscara são sujeitos a uma operação binária de *AND*.

No exemplo seguinte, a regra é activada caso um pacote de uma ligação *telnet* tenha sido previamente marcado por outro módulo da *firewall*:

```
iptables –A FORWARD –i eth0 –o eth1 –p tcp \
        -s $UM_ENDERECO_IP --sport $PORTNPRIV \
        -d $HOST_LOCAL –dport 23 \
        -m mark –mark 0x00010070
        -j ACCEPT
```

Pacotes marcados podem ser diferenciados para serem tratados de modo diferente do que pacotes em tudo idênticos, excepto estarem marcados.

Operações com a extensão *match tos*

A Tabela V-17 lista as opções disponíveis para a extensão *match tos*.

-m | --match tos	Descrição
--tos <valor>	Activa a regra caso o valor indicado de TOS (*type of service*) coincida com o valor TOS do pacote em processamento.

Tabela V-17: Operações com a extensão *match tos*

O parâmetro <valor> pode ser uma das seguintes *strings* ou um números:

- minimize-delay, 16, 0x10
- maximize-throughtput, 8, 0x08
- maximize-reliability, 4, 0x04
- minimize-cost, 2, 0x02
- normal-service, 0, 0x00

Os *bits* do campo *tos* são apenas de interesse histórico e normalmente são ignorados pelos sistemas informáticos. O *Linux* suporta a sua utilização.

Recentemente, o campo *tos* foi redefinido como sendo o campo *DS* (*Differentiated Services*) utilizado no protocolo Differentiated Services Control Protocol (DSCP) [12] [13] [14].

Operações com a extensão *match unclean*

A extensão *match unclean* não possui nenhuma opção como podemos ver na Tabela V-18.

-m | --match unclean	A regra é activada caso o pacote possua ilegalidades na sua construção.

Tabela V-18: a extensão *match unclean*

As verificações de validade de pacotes efectuadas pela extensão *match unclean* não estão bem documentadas. O módulo é considerado experimental e os autores do *iptables* desaconselham a sua utilização.

No exemplo seguinte os pacotes *TCP* ilegais e outros pacotes ilegais são registado em *Log* e de seguida descartados silenciosamente:

```
iptables –A INPUT –p ! tcp –m unclean \
        -j LOG –-log-prefix "PACOTE ILEGAL: " \
        --log-ip-options

iptables –A INPUT –p tcp –m unclean \
        -j LOG –-log-prefix "PACOTE TCP ILEGAL: " \
        --log-ip-options \
       --log-tcp-sequence –l-og-tcp-options

iptables –A INPUT –m unclean –j DROP
```

4.4.5 Comandos da tabela nat

Tal como foi referido anteriormente, o *iptables* suporta quatro formas de *Network Adress Translation* (NAT): *source NAT* (SNAT), *destination NAT* (DNAT), *masquerading* (MASQUERADE) e redirecionamento de portas local (REDIRECT). Qualquer um destes destinos de regra estão disponíveis quando a tabela *nat* é especificada através do parâmetro ***–t nat***.

Operações com a extensão target SNAT

A tradução de endereços e portas origem (*NAPT*), é o tipo de *NAT* mais comum. Como se pode ver na Imagem IV-15, a tradução de endereços origem só pode ser efectuado depois da fase de encaminhamento (*routing*). O destino de regra SNAT só poder ser utilizado na cadeia *POSTROUTING*. Devido ao facto de a tradução ser efectuada imediatamente antes de um pacote ser enviado, só pode ser especificada uma interface para a saída do pacote.

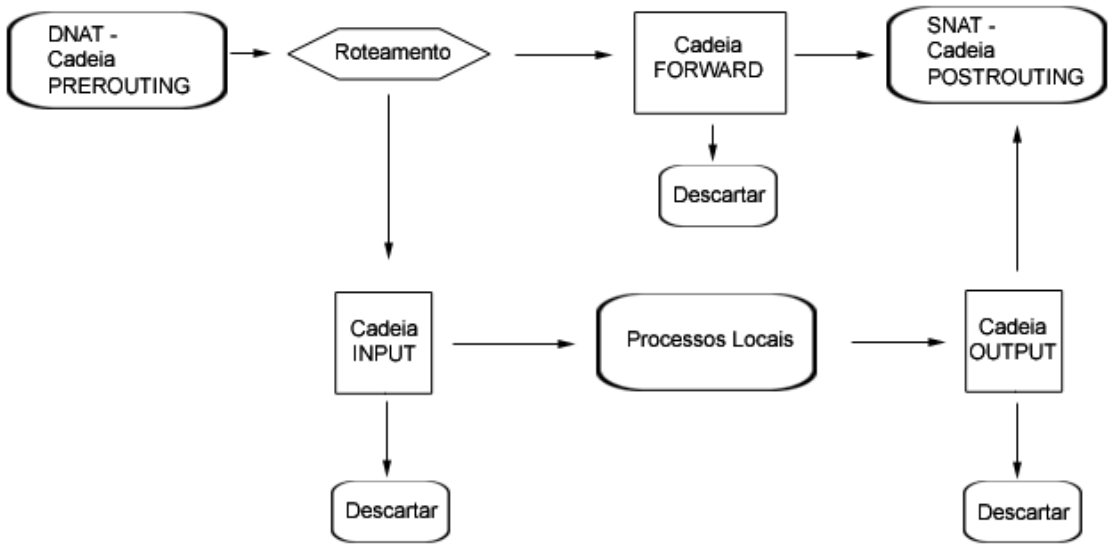

Imagem IV-15: Travessia de um pacote através do iptables com o módulo *NAT*.

Esta forma de *NAT* (a forma mais comum) é por vezes referida como *NAPT*, tornando-se assim mais fácil lembrar que a porta origem também é traduzida. A outra forma de *NAT* tradicional, unidireccional é o *NAT* básico, cuja diferença é que não altera o endereço origem. Essa forma é utilizada em situações onde se pretende traduzir entre endereços *IP* privados de uma *LAN* e um conjunto de endereços *IP* públicos.

O *NAPT* é utilizado em situações em que há apenas um endereço IP público. A porta origem é traduzida para uma porta livre na *firewall* devido ao facto de a *firewall* traduzir vários endereços da rede interna, e a porta que uma dada máquina interna utiliza, pode já estar ocupado na *firewall*. Quando as respostas chegam à *firewall*, a porta destino é a única informação disponível para a *firewall* determinar se o pacote se destina à própria máquina ou a uma máquina da rede interna.

A sintaxe geral para o SNAT é:

```
iptables –t nat –A POSTROUTING ––out-interface <interface de saída> … \
        -j SNAT ––to-source <endereço> [-<endereço>][:<porta>-<porta>]
```

O endereço origem pode ser mapeado a um conjunto de possíveis endereços IP, caso exista mais que um endereço *IP* disponível.
A porta origem pode ser mapeada para uma gama específica de portas na *firewall*.

Operações com a extensão target MASQUERADE

A tradução de endereço origem foi implementada de dois modos diferentes no *iptables*, como *SNAT* e como *MASQUERADE*. A diferença é que o destino de regra *MASQUERADE* se destina a interfaces cujas ligações possuem endereços IP atribuídos dinamicamente, especialmente em casos em que a ligação é temporária e o endereço IP ser diferente em cada ligação, como ligações *dial-up*.

Devido ao facto de o destino de regra *MASQUERADE* ser uma forma especializada de *SNAT*, também é apenas possível utilizá-la na cadeia *POSTROUTING*, e a regra só pode referir uma interface para a saída do pacote. Ao contrário do destino de pacotes *SNAT*, que é mais generalizado, o *MASQUERADE* não possui um parâmetro para especificar para que endereço *IP* será traduzido o endereço *IP* origem do pacote. O endereço *IP* da interface de saída do pacote é utilizado automaticamente.

A sintaxe geral para o MASQUERADE é:

```
iptables –t nat –A POSTROUTING ––out-interface <interface de saída> … \
        -j MASQUERADE [––to-ports <port>[-<port>]]
```

A porta origem pode ser mapeada para uma gama específica de portas na *firewall*.

Operações com a extensão target DNAT

A tradução de endereços destino e porta destino (DNAT) é uma forma muito especializada de *NAT*. Esta característica é especialmente útil em ambientes em que apenas a firewall possui IP público, e se torna assim a única máquina visível na *Internet*. Nesta situação é muitas vezes necessário encaminhar ligações a servidores internos não visíveis na *Internet*. Esta característica pode ser utilizada para substituir software de terceiros para redirecionamento de portas como o *ipmasqadm*.

Como podemos verificar ainda na Imagem IV-15 a tradução de endereços e portas destino só é possível antes da decisão de *routing*. A utilização do DNAT apenas é possível nas cadeias PREROUTING e OUTPUT. Na cadeia PREROUTING, o DNAT pode ser o destino de regra quando uma interface de entrada de pacotes é especificada. Na cadeia OUTPUT, o DNAT pode ser o destino de regra quando uma interface de saída de pacotes é especificada.
A sintaxe geral para o DNAT é:

```
iptables –t nat –A PREROUTING ––in-interface <interface de entrada> … \
     -j DNAT ––to-destination <endereço> [-<endereço>][:<porta>-<porta>]

iptables –t nat –A OUPUT ––out-interface <interface de saída> … \
      -j DNAT ––to-destination <endereço>[-<endereço>][:<porta>-<porta>]
```

O endereço destino pode ser mapeado a um conjunto de possíveis endereços IP, caso exista mais que um endereço IP disponível.
A porta destino pode ser mapeada para uma gama específica de portas na máquina destino.

Operações com a extensão target REDIRECT

O redirecionamento de portas é um caso específico de DNAT. Um pacote é redireccionado para uma porta na *firewall*. Os pacotes de que entram na firewall e que de outro modo seriam reencaminhados para outras máquinas, são redireccionados para a cadeia *INPUT*. Os pacotes enviados pela *firewall* são redireccionados para uma porta na interface *loopback* da *firewall*.

O *REDIRECT* é uma simples conveniência para um caso especializado que poderia facilmente ser solucionado com o destino de regra *DNAT*.

Tal como o destino de regra *DNAT*, o *REDIRECT* apenas é válido nas cadeias *PREROUTING* e *OUTPUT*. Na cadeia *PREROUTING*, o *REDIRECT* pode ser um destino de regra quando uma interface de entrada de pacotes é especificada. Na cadeia *OUTPUT*, o *REDIRECT* pode ser um destino de regra quando uma interface de saída de pacotes é especificada.

A sintaxe geral para o REDIRECT é:

```
iptables –t nat –A POSTROUTING ––in-interface <interface de entrada> … \
        -j REDIRECT [––to-ports <porta>[-<porta]]

iptables –t nat –A OUPUT ––out-interface <interface de saída> ... \
        -j REDIRECT [––to-ports <porta>[-<porta]]
```

A porta destino pode ser mapeada para uma porta diferente ou para uma gama específica de portas na firewall.

4.4.6 Comandos da tabela mangle

A tabela *mangle* e as suas extensões aplicam-se as cadeias de *OUTPUT* e *PREROUTING*. Para utilizar as funcionalidades da tabela *mangle* é necessário utilizar o parâmetro ***–t mangle***.

Operações com a extensão target mark

A Tabela V-19 lista os destinos de regras existentes na tabela *mangle*.

-t mangle	**Descrição**
-j MARK ––set-mark <valor>	Define o valor com que o pacote é marcado.
-j TOS ––set-tos <valor>	Define o valor de *TOS* (*type of service*) no cabeçalho IP do pacote.

Tabela V-19: Operações com as extensões de destino de regra da tabela *mangle*

Existem duas extensões de destino de regra na tabela *mangle*: MARK e TOS. O MARK permite definir um valor *unsigned long* com que pacote é marcado pela tabela *mangle*.

Um exemplo da sua utilização é:

```
iptables –t mangle  –A PREROUTING ––in-interface eth0 –p tcp \
        -s <endereço origem> --sport 1024:65535 \
        -d <endereço destino> --dport 23 \
        -j MARK ––set-mark 0x00010070
```

O TOS permite definir os *bits* de TOS no cabeçalho IP do pacote.

Um exemplo da sua utilização é:

```
iptables –t mangle  –A OUTPUT … \
        -j TOS ––set-tos <valor de tos>
```

Os valores possíveis de *tos* são os mesmos disponíveis na extensão *match TOS* da tabela *filter*.

4.5 Conclusão

Este capítulo descreveu as características do *iptables*, sua estrutura e sintaxe dos comandos de configuração

.

O *iptables* possui uma estrutura modular com três tabelas principais: *filter*, *nat* e *mangle*.

Por omissão a tabela *filter* é utilizada, para se utilizar qualquer uma das outras tabelas é necessário especificar através do parâmetro –t.

A tabela *filter* possui funcionalidades que permitem activar uma regra com base em informação presente nos cabeçalhos IP, TCP, UDP, ICMP e *ethernet* de um pacote, e na informação sobre o estado das ligações.

A tabela *nat* possui funcionalidades completas de *NAT* que permitem fazer tradução de endereços origem e destino, e tradução de portas origem e destino.

A tabela *mangle* possui funcionalidades para definir o valor de *TOS* de um pacote.

5 Conclusão

5.1 Conclusão

Para se dominar a área de segurança de sistemas informáticos é necessária muita pesquisa e muito estudo. Existem muitas disciplinas envolvidas desde a matemática, à programação, redes de computadores, administração de sistemas informáticos, sistemas operativos, hardware, entre outros.

O mundo da segurança informática e conceitos fundamentais que a constituem foram introduzidos.

Esta tese centrou-se no estudo das *firewalls*, um componente essencial de segurança de sistemas informáticos, tendo incidido primariamente no estudo de firewalls de filtros de pacotes para *Linux*.

No último capítulo este projecto teve uma abordagem mais prática, com o objectivo de estudar e configurar a *firewall iptables* (*software* de *firewall* mais comum em ambientes *Linux*).

Os objectivos a que esta tese se propôs foram assim cumpridos.

Glossário

ACCEPT Uma decisão de uma firewall de filtro de pacotes que aceita um pacote.

ACK *Flag* TCP que reconhece a recepção de um segmento TCP.

Application Level Gateway Também referido como ALG, refere-se a módulos de suporte específicos a aplicações para inspeccionar pormenores relacionados com a aplicação e respectivos protocolos utilizados.

Autênticação Processo de determinação de que uma entidade é quem diz ser.

Autorização Processo de determinar que serviços e recursos uma entidade pode utilizar.

Broadcast Envio de um pacote IP para todas as interfaces ligadas a uma rede local ou subnet.

Cadeia de regras (chains) Lista de regras que definem que pacotes podem e que pacotes não podem passar por uma interface.

Camada de acesso aos dados (Data Link Layer) A segunda camada do modelo OSI.

DARPA Defence Advanced Research Agency

Daemon Um serviço básico de sistema que corre em *background*

Datagrama IP Um pacote de nível de rede.

Denial-of-service, Ataque Um tipo de ataque que se baseia em enviar dados inesperados, ou inundar um sistema com pacotes danifiquem seriamente a ligação à Internet da máquina vítima de modo a pedidos legítimos serem recusados por falta de recursos. Também podem resultar em bloquei da máquina vítima.

DHCP Dynamic Host Configuration Protocol, é utilizado para dinamicamente atribuir endereços IP e fornecer informação de *routing* a clientes com endereços IP registados. Foi desenvolvido para substituir o BOOTP.

DMZ Zona desmilitarizada, uma zona do perímetro da rede que contém máquinas que contém serviços públicos. Esses seguros são separados da rede privada por questões de segurança.

DNS Domain Name Service

DROP Uma decisão de uma firewall de filtro de pacotes que nega um pacote.

Firewall de filtro de pacotes Uma firewall implementada nas camadas de rede e transporte que filtra o tráfego de rede pacote-a-pacote, fazendo decisões de encaminhamento baseadas em informação presente nos cabeçalhos IP e TCP.

Flooding Um tipo de ataque *denial-of-service* em que é enviado à

vítima mais pacotes de um determinado tipo que a vítima pode receber.

Fragmento Um pacote IP contendo uma fatia de um pacote TCP

Gateway de Circuito ver *proxy*

HTTP Hypertext Transfer Protocol, um protocolo utilizado pelos servidores de WWW e *browsers*

IANA Internet Assigned Numbers Authority. A entidade que regula a atribuição de endereços IP.

ICMP Internet Control Message Protocol. Um protocolo de nivel de rede para assinalar o estado e controlar o fluxo de pacotes.

IP Internet Protocol. Protocolo de nível de rede do TCP/IP.

Ipchains aplicação de firewall antecessora do *iptables*

IPFW Mecanismo de firewalling que foi substituído pelo *Netfilter*

ISP Internet Service Provider

LAN Local Area Network

Masquerading o processo de substituição do endereço origem de um pacote de saída da *firewall* com o endereço da firewall ou gateway.

MTU Maximum transmissium unit, o tamanho máximo de um pacote numa determinada infraestrutura de rede.

NAT Network Address Translation, o processo de substituição do endereço origem ou destino de um pacote, com o endereço de outra interface.

Netfilter O mecanismo de *firewalling* do *Linux* a partir de versões 2.4.x do *Kernel*

OSI Open Systems Interconnection

Ping Uma ferramenta simples de análise de rede utilizada para determinara se uma máquina remota é alcançável e responde a pedidos

Política por omissão: define o destino de um pacote caso este não active nenhuma regra da cadeia que está a ser processada.

Política de Aceitar por omissão Uma política que aceita todos os pacotes que não activaram nenhuma regra na cadeia. Por esse motivo muitas *firewalls* seguem uma política de negar por omissão.

Política de Negar por omissão Uma política que nega todos os pacotes que não activarem nenhuma regra na cadeia. Assim, a maior parte das regras são regras de aceitação de pacotes, constituindo excepções à política por omissão.

Port Scan Probes a uma ou a um conjunto das portas de uma máquina, normalmente procurando vulnerabilidades

Probe Sondagem de uma vulnerabilidade, um serviço numa determinada porta de uma máquina remota.

Proxy Um programa que cria e mantém uma ligação com um

terceiro em nome de um cliente do proxy.

Tramas Ethernet Numa rede ethernet, os datagramas IP são encapsulados em tramas *Eterhnet*.

Bibliografia

[1] IBM's Redbook Abstract Series, "TCP/IP Tutorial and Technical Overview", http://www.redbooks.ibm.com/pubs/pdfs/redbooks/gg243376.pdf

[2] Internetworking Technologies Handbook, Chapter 7, "Ethernet", http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ethernet.pdf

[3] GARFINKEL Simson, SPAFFORD Gene, Pratical UNIX & Internet Security, 2nd Edition (Sebastopol, CA: O'Reilly, 1996)

[4] OLIVEIRA Wilson, Técnicas para Hackers – Soluções para Segurança (Edições Centro Atlântico 2000)

[5] RFC 2647, "Benchmarking Terminology for Firewall Performance", http://www.ietf.org/rfc/rfc2647.txt

[6] Hacking Lexicon, http://www.robertgraham.com/pubs/hacking-dict.html

[7] CERT® Incident Note IN-99-01, "sscan Scanning Tool", http://www.cert.org/incident_notes/IN-99-01.html

[8] RFC 1122, "Requirements for Internet Hosts-Communication Layers", Secção 3.2.2.2 http://www.ietf.org/rfc/rfc1123.txt

[9] CERT® Coordination Center – "Email Bombing and spamming" http://www.cert.org/tech_tips/email_bombing_spamming.html

[10] RFC 2827, "Network Ingress Filtering: Defeating Denial Of Service Attacks Which Employ IP Source Adress Spoofing" http://www.ietf.org/rfc/rfc2827.txt

[11] Security Focus - Check Point Firewall-1 on Linux http://online.securityfocus.com/infocus/1401

[12] RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", http://www.ietf.org/rfc/rfc2474.txt

[13] RFC 2475, "An Architecture for Differentiated Services", http://www.ietf.org/rfc/rfc2475.txt

[14] RFC 2990, "Next Steps for the IP QoS Architecture", http://www.ietf.org/rfc/rfc2990.txt

[15] KEISER Gerd E., Local Area Networks (McGraw-Hill Education 1989)

[16] RFC 1938, "A one-time password system" , http://www.ietf.org/rfc/rfc1938.txt

[17] RFC 2289, "A one-time password system" , http://www.ietf.org/rfc/rfc2289.txt

[18] Lusignan R., Steudler O., Allison J. - Managing CISCO Network Security

Sites relacionados com o tema

Security Focus Corporate Site
http://www.securityfocus.com

Security Focus Online
http://online.securityfocus.com/

The Sans Institute
http://www.sans.org

CERT
http://www.cert.org

NewOrder
http://neworder.box.sk

SearchSecurity
http://searchsecurity.techtarget.com/

The WWW Security FAQ
http://www.w3.org/Security/Faq/

Firewall.com
http://www.firewall.com/

Páginas Relevantes

Arstechnica Forum
http://arstechnica.infopop.net

SearchSecurity – Firewall definition
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci212125,00.html

SearchSecurity – Personal Firewall definition
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci331881,00.html

TechWeb Encyclopedia
http://www.techweb.com/encyclopedia/

McAfee - Virus Glossary of Terms
http://www.mcafee.com/anti-virus/virus_glossary.asp

searchSystemsManagement – ADSL definition

http://searchsystemsmanagement.techtarget.com/sDefinition/0,,sid20_gci213764,00.html

SearchSecurity – SSL definition
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci343029,00.html

SSL 3.0 Specification
http://wp.netscape.com/eng/ssl3/index.html
Business Standard - A inteligência dos negócios Internet
"As intranets também morrem"
http://www.businessstandard.com.br/bs/tecnologia/2002/03/0002

searchEBusiness – e-commerce definition
http://searchebusiness.techtarget.com/sDefinition/0,,sid19_gci212029,00.html

CERT: Home Network Security
http://www.cert.org/tech_tips/home_networks.html

A Firewall White Paper
http://firewall.esoft.com/

IBM's Online TCP/IP Redbook (7.69 MB)
http://www.redbooks.ibm.com/pubs/pdfs/redbooks/gg243376.pdf

NSA Glossary of Terms Used in Security and Intrusion Detection
http://www.sans.org/newlook/resources/glossary.htm

Computer Security publications at Chalmers
http://www.ce.chalmers.se/staff/ulfl/group-pubs.html

Internet Engineering Task Force, Site Security Policy Handbook Working Group. Site Security Handbook, (RFC 2196, FYI 8). ftp://ftp.isi.edu/in-notes/rfc2196.txt (1997)

Manual de criptografia básica
http://unsekurity.virtualave.net/texto1/manual-crypto.txt

RSA – a searchSecurity Definition
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214273,00.html

OSI – a searchNetworking Definition
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212725,00.html

Cisco - SAFE: A Security Blueprint for Enterprise Networks
http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/safe_wp.htm

SwitchSniff
http://www.awot.biz/sf/sf/files/verkko/txt/switchsniff.htm

Introduction do SSL
http://developer.netscape.com/docs/manuals/security/sslin/contents.htm

Evolution of the Firewall Industry
http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ch3.htm

The History Of Worms
http://www.software.com.pl/newarchive/misc/Worm/darbyt/pages/history.html

Por: ***Pedro Fortuna – Setembro 2002***