Universidade Federal Fluminense - UFF

Centro de Estudos de Pessoal - CEP

Curso de Criptografia e Segurança em Redes

Segurança em Redes 2

Marcos Tadeu von Lützow Vidal

Rio de Janeiro
2006

Créditos

Capa: Maria Rachel Barbosa

Redação pedagógica e revisão: Ana Maria Andrade Araujo
Heloisa Cardoso de Castro

V649s Vidal, Marcos Tadeu von Lützow
Segurança em redes 2 / Marcos Tadeu von Lützow Vidal. – Rio de Janeiro: UFF / CEP - EB, 2006.
XXXp. – (Curso de Criptografia e Segurança em Redes).

ISBN 00-00000-00-0

1. Criptografia. 2. Redes de computadores.
CDD – 652.8

Centro de Estudos de Pessoal (CEP)
Praça Almte. Júlio de Noronha S/N
Leme - Rio de Janeiro - RJ
22010-020
Tel 21 2295-1140

Informações sobre a disciplina

Ementa

O conceito “Firewall” – filtragem de pacotes/NAT/PAT. Monitoração e auditoria. Gerenciamento de chaves - PKI. Utilização de criptografia.

Carga horária

60 horas

Objetivos

Reconhecer conceitos teóricos e, principalmente, práticos sobre segurança em redes de computadores.

Distinguir os métodos básicos de configuração e procedimentos conscientes com a segurança da informação.

Metodologia

O conteúdo programático será apresentado sob a forma de textos e exemplos, com atividades a serem realizadas.

Avaliação

Prova escrita ao fim da disciplina e atividades.

Sumário

Unidade 1 - Firewall 8
Texto 1 - Tecnologia – tipos de filtros 10
Packet Filters 10
Stateless Inspection 11
Stateful Inspection 12
Application Gateways 13
Texto 2 - Tecnologia – arquitetura 15
Roteador filtrante 15
Máquina com dupla base 17
Sub-rede filtrada 18
Firewall com três placas de rede 20
Unidade 2 - Filtragem de pacotes por IP/serviço 22
Texto 1 - Roteador X Firewall 22
Por que “pacotes”? 22
Por que filtrar pacotes 24
Vantagens de filtragem 24
Desvantagens de filtragem 25
Texto 2 - Processamento em um filtro 25
Texto 3 - Criando filtros 32
Serviços bidirecionais, tráfego idem 32
Postura padrão 34
Texto 4 - Lista de Acesso Estendida (ACL) 35
A primeira ACL 36
Traduzindo para ACL 37
Netfilter e iptables 39
Tradução das regras – iptables 45
Texto 5 - Filtragem por serviço 45
Exemplo de filtragem – SMTP 45
Exercício 48
Tabela SMTP 48
Exercício 48
ACL para o exemplo SMTP 49
Exercício 50
Netfilter para o exemplo SMTP 50
Exercício 52
Texto 6 - Filtros clássicos obrigatórios 52
Exercício 53
Texto 7 - Ligação de Intranet na Internet 53
NAT 54
Uso obrigatório de NAT ou Masquerade 56
PAT 58
Unidade 3 - Ferramentas de monitoração e auditoria 59
Texto 1 - Ping 59
Texto 2 - Traceroute 61
Texto 3 - Tcpdump 64

Texto 4 - NMAP..... 67
Texto 5 - Ethereal..... 68
Texto 6 - IDS - Intrusion Detector System..... 70
Texto 7 - Detector de vulnerabilidades..... 72
Outros softwares..... 75
Ettercap..... 75
Unidade 4 - Protocolos de criptografia & PKI..... 77
Texto 1 - Gerência de chaves..... 77
Geração das chaves..... 77
Armazenagem da chave..... 78
Distribuição das chaves..... 78
Texto 2 - Força de um sistema de criptografia..... 79
Ataque de força bruta..... 79
Importância do tamanho da chave..... 80
Tamanho mínimo de chaves..... 81
Texto 3 - Introdução a PKI 82
O que é o ICP-Brasil..... 83
Texto 4 - Padrões para criptografia de chave pública..... 84
PKCS #1: Criptografia RSA..... 85
PKCS #7: Sintaxe de mensagens criptografadas..... 85
PKCS #10: Requisição de certificados..... 85
PKCS #11: (Criptoki) Cryptographic Token Interface..... 85
PKCS #15: Formato de arquivos para dados criptografados..... 86
PKCS: Outros Documentos..... 86
Texto 5 - Certificados Digitais..... 87
Credibilidade do certificado digital..... 88
Validação de um certificado..... 88
Certificados – revogação..... 89
Implementação de infra-estruturas de chaves públicas..... 89
Procedimentos e políticas..... 91
Custódia ou Armazenamento de Chaves..... 92
Unidade 5 - Usando a criptografia..... 94
Texto 1 - PGP..... 94
Texto 2 - SSL..... 96
Negociação da encriptação..... 97
Texto 3 - SSH..... 98
Texto 4 - OpenSSL..... 99
O comando openssl..... 100
Texto 5 - Gerando certificados..... 101
Chave RSA de 2048 bits..... 102
Certificado auto-assinado..... 105
Chave do servidor..... 107
Criptografar ou não a chave..... 107
Requisição de certificado..... 108
Assinando a requisição com o certificado da CA..... 109
Scripts prontos..... 112
Unidade 6 - VPN – Rede Virtual Privada..... 115

VPN – A idéia básica..... 115
Texto 1 - Protocolos antigos..... 116
GRE (Generic Routing Protocol)..... 116
SOCKS..... 116
L2F-Layer-2 Forwarding..... 117
PPtP..... 119
MS-PPtP – Análise..... 121
L2TP-Layer 2 Tunneling Protocol..... 123
Texto 2 - Túnel SSL..... 124
Texto 3 - IPSec..... 126
IPSec: Cabeçalho de Autenticação (AH)..... 127
IPSec: Cabeçalho de Encapsulamento de Dados de Segurança (ESP)..127

Sumário de Figuras

Figura 1 - Firewall do tipo Filtro de Pacotes.....10
Figura 2 - Firewall Stateful Inspection.....13
Figura 3 - Firewall do tipo Application Gateway.....14
Figura 4 - Roteador Filtrante.....16
Figura 5 - Firewall com máquina com duas placas de rede.....17
Figura 6 - Firewall com sub rede filtrada.....18
Figura 7 - Firewall com DMZ - três placas de rede.....20
Figura 8 - Comunicação entre o servidor interno e uma estação externa.....27
Figura 9 - Comunicação com duas estações externas.....30
Figura 10 - Serviço interno e externo: cada um com dois fluxos.....33
Figura 11 - Arquitetura do sistema de filtros no Netfilter.....40
Figura 12 - Fluxo de serviços entre um servidor interno, um na DMZ, e a Internet.....46
Figura 13 - Firewall com NAT, ligado à Internet.....55
Figura 14 - Arquitetura do Netfilter: a passagem dos pacotes pelas tabelas nat e filter.....56
Figura 15 - Ethereal: um poderoso analisador de protocolos.....68
Figura 16 - Conteúdo de um certificado digital.....88
Figura 17 - Relações entre as entidades de um PKI.....90
Figura 18 - GPA - o gerenciador de chaves.....95
Figura 19 - A sub camada SSL na pilha TCP/IP.....96
Figura 20 - Estabelecimento de uma sessão SSL.....97
Figura 21 - Certificado assinado por uma CA não reconhecida.....111
Figura 22 - Certificado assinado por uma CA reconhecida.....112
Figura 23- Detalhes do certificado e do CA.....113
Figura 24 - VPN: a idéia básica.....115
Figura 25 - Túnel GRE.....116
Figura 26 - Túnel L2F.....118
Figura 27 - Túnel L2TP.....124
Figura 28 - Criando túneis seguros.....125
Figura 29 - IPSec - Os novos cabeçalhos.....127

Unidade 1 - Firewall

O termo Firewall tem sua origem na analogia feita com as paredes (muros) de alvenaria, levantadas entre as casas de alguns bairros americanos, onde as construções são feitas com madeira. Um incêndio em uma localidade dessas se alastra com velocidade. A parede corta fogo isola uma casa, protegendo-a do incêndio do terreno vizinho, e impedindo que ele se alastre com facilidade.

A analogia da idéia se aplica em redes de computadores, já que a Internet, ou qualquer rede fora do alcance de sua administração, pode ser considerada um incêndio onde ataques acontecem a todo instante. O Firewall pode isolar uma rede inteira de outra, de modo a que o administrador tenha somente um ponto para se preocupar: o que está aberto ou não. Em um Firewall, porta aberta significa dar passagem para os pacotes de dados, normalmente destinados para a porta em questão.

É como se esta parede de fogo, tradução do nome, fosse um segurança na porta de um local, verificando a identidade de quem deseja passar pela porta, negando o acesso a quem não está autorizado a entrar e liberando os demais, baseado numa política de segurança que alguém especificou.

Todo Firewall está associado a um conjunto de regras, cuja ação bloqueia, nega, rejeita ou aceita um tráfego específico que passe por ele. Este tráfego, aceito ou não, pode ser registrado num banco de dados, para análise futura, ou em tempo real. Há no mercado muitos tipos de Firewall, com várias funcionalidades agregadas, conforme o preço de cada um. Existem Firewalls baseados em software e/ou hardware.

> Firewall: um componente ou um conjunto de componentes que restringe o acesso entre uma rede protegida e a Internet ou quaisquer outras redes.

Um Firewall corporativo é o primeiro passo na proteção de uma rede. Não que seja obrigatório, ou que o Firewall resolverá todos os problemas de segurança. Na verdade, é apenas mais um componente do conjunto, mas o risco envolvido em colocar uma rede interna numa rede pública, como a Internet,

sem um Firewall, é muito grande. É como ter uma porta que foi deixada aberta para algumas pessoas passarem, mas sem um guarda ou qualquer outro agente de controle, para fazer o controle de acesso. Temos uma situação de risco muito alto. Mesmo se tivermos um vigia (cara-crachá-cara-crachá...) para controlar quem pode ou não entrar, alguém mal intencionado poderia confundi-lo, suborná-lo ou eliminá-lo usando engenharia social, tornando a porta frágil para o acesso de qualquer um. Se colocarmos uma máquina para realizar tal função, teremos mais garantias de segurança, mas ficaremos dependentes do fabricante da máquina e, talvez, da junção máquina/porta.

Podemos voltar à solução com um ser humano e colocar dois seguranças na porta, e um ficaria com a função de auxiliar o outro, caso precisasse se ausentar. Logicamente eles não podem ser amiguinhos, ficar papeando etc.

Na solução com Firewall, descobrimos que podemos usar o mesmo princípio: colocando dois Firewalls em “paralelo”, aumentamos nossa segurança e garantimos uma certa continuidade no tráfego mesmo que um deles pare de funcionar por falha. Essa solução se chama Alta Disponibilidade (High Availability). Outra opção seria um Firewall com fontes redundantes, portas redundantes etc. Cada modelo e solução depende da necessidade diretamente ligada ao valor da informação e quanto se está disposto a pagar pela segurança dos dados.

A função de um Firewall é proteger uma rede de outra(s). Então, é natural que seja posicionado entre a rede interna (LAN) e a rede externa (Internet). Pode-se colocar mais um Firewall protegendo um determinado servidor mais crítico. A maioria das implementações de Firewall do mercado usa o conceito de área desmilitarizada. Essa área é uma terceira rede, também chamada de DMZ e serve para se colocar servidores de domínio público, como servidores Web, correio eletrônico, sem perder a segurança, porque o Firewall só deixará passar tráfego do serviço(s) entre este(s) servidor(es) e a origem do pedido.

Às vezes, o termo Firewall é empregado tanto para referenciar um complexo de equipamentos que faz a interligação entre a rede interna e a rede externa, quanto a cada um dos equipamentos que realizam a tarefa de filtrar o tráfego. Nesta unidade, o termo também será usado das duas formas.

Texto 1 - Tecnologia – tipos de filtros

Em relação à evolução dos sistemas de Firewall, nós a apresentaremos a partir da capacidade de seus filtros. Inicialmente, eles só analisavam endereços IP. Depois, passaram a levar em consideração também também os protocolos acima da rede (ICMP, TCP, UDP) e as portas que estavam em uso. São os *Packet Filters.*

Packet Filters

Existem Firewalls que funcionam como simples filtro de pacotes, ou seja, só permitem que o pacote entre dentro da rede interna se o endereço de destino for de um servidor da rede interna. Podem validar se o endereço da origem pertence a um conjunto de endereços previamente cadastrados no Firewall como endereço válido de origem. Este tipo de Firewall é muito eficiente em matéria de velocidade, uma vez que os dados carregados, que representam a maior parte do pacote, não serão analisados, mas pode deixar a desejar pelo mesmo motivo, ao não analisar a porção mais importante, o dado. Além disso, faz a análise como se cada pacote fosse isolado e independente dos outros. No TCP, por exemplo, não conhece o conceito de conexão, não sabe o que é um pacote SYN, um ACK etc. Você sabe, não? (hora de recordar!)

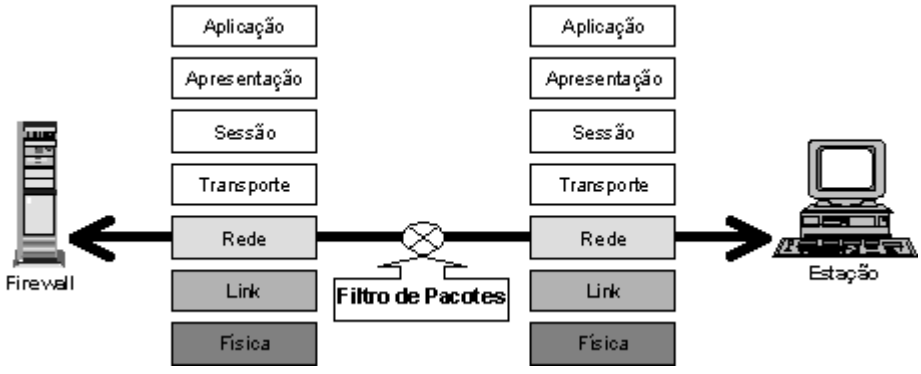

Figura 1 - Firewall do tipo Filtro de Pacotes.

Stateless Inspection

Na verdade, um Firewall do tipo filtro de pacotes analisa o pacote no equivalente à camada de Rede e de Transporte do Modelo OSI. Então, no modelo TCP/IP, analisa os endereços IP, o tipo de protocolo (TCP/UDP/ICMP) e as portas origem e destino. Atualmente, esta filtragem de pacotes pode ser implementada na maioria dos roteadores e é transparente aos usuários. Os Firewalls baseados em filtros de pacote são os mais simples e baratos, mas oferecem proteção limitada e são considerados obsoletos. Não conhecem conexão e nem o conceito de fluxo de dados. Não atuam no nível de aplicação, ou seja, não "entendem" o protocolo de aplicação.

Deve-se ter em mente que um acesso qualquer entre um cliente e um servidor sempre possui tráfego indo e vindo. Em um filtro *stateless,* para cada serviço é necessário criar uma regra para os pacotes que vão e outra para os que vêm!

Outro problema: quando o Firewall é um simples filtro de pacotes e não entende nada das necessidades da aplicação, é que existem serviços que usam portas não fixas.

Por exemplo, o serviço FTP é uma destas exceções e, por ser muito conhecido e usado, sempre foi um problema para os sistemas simples de segurança. Como complicativo, o FTP pode operar de dois modos distintos, os chamados modo passivo ou modo ativo. Usa sempre duas conexões: uma do cliente para o servidor na porta 21, que é a conexão de controle, por onde passam os comandos do FTP; a outra conexão depende do modo de operação: ativo ou passivo.

No modo ativo, a conexão é iniciada pelo servidor (!) com porta origem 20 e destino para uma porta alta no cliente, acima de 1.023, conexão esta usada para a transferência do arquivo. Simplificando, podemos dizer que essa porta alta é escolhida pelo cliente (aplicação) de FTP e enviada pela conexão de comando, já estabelecida na porta 21. Então, para o FTP funcionar em um Firewall que não conhece o protocolo FTP, todas as conexões TCP vindas de porta 20 para portas acima de 1.023 teriam que ser liberadas, expondo muito a rede interna. É possível usar uma configuração para modo passivo, que utiliza portas aleatórias, altas, o que inicia a conexão do cliente para o

servidor. Assim, temos que manter todas as portas altas abertas mas, pelo menos tudo fica aberto de dentro para fora (rede interna para Internet). No entanto, nem todos os clientes de FTP suportam este modo passivo.

No Linux, o antigo kernel da série 2.0 e superiores já possuía a funcionalidade necessária para se criar um Firewall *Stateless*. O FTP era um problema de tal prioridade na segurança, que em julho de 1999, um colaborador do software livre se empenhou em criar a primeira inspeção quase Stateful (ainda sem este nome), para o sistema de Firewall do kernel 2.2.10 do Linux, apenas para atender ao FTP.

Posteriormente, outros colaboradores foram atualizando o patch para novas versões do kernel 2.2.x, mas o restante dos serviços ainda continuou *stateless*, até a publicação do kernel 2.4 e seu sistema netfilter. Depois de aplicado esse patch no kernel, existia um módulo que analisava os comandos do protocolo FTP. Quando passava o pacote de comando que continha a especificação da porta que seria usada no cliente para receber a conexão para a transferência do arquivo, o módulo criava automaticamente uma regra no Firewall, somente para deixar passar essa nova conexão. Leia a explicação do autor do patch, e guarde os exemplos de comandos que estudaremos mais tarde, em:

http://hasenstein.com/README.patch.ftp-data-2

Felizmente, Firewalls com *Stateful Inspection* eliminam este e outros problemas.

Stateful Inspection

Trata-se de um outro grupo de Firewall hoje considerados uma das melhores opções de mercado. Controlam toda sessão aberta entre a origem e o destino, mantendo uma tabela de controle de *status*, garantindo, com isso, que as conexões abertas entre dois pontos sejam mais rápidas e garantindo maior segurança entre os pontos.

Um Firewall do tipo *Stateful Inspection* analisa o pacote no mínimo até a camada de transporte, examinando o fluxo de dados e prevendo a existência

de pacotes de resposta (ACK) e de controle (ICMP) relacionados a uma conexão TCP ou pacotes de resposta de uma sessão UDP. Este é o caso da maioria dos sistemas de Firewall, e é o que está implementado desde o kernel 2.4 do Linux.

Para tanto, não é suficiente examinar pacotes isoladamente. A informação de estado – derivada de comunicações passadas e de outras aplicações – é um fator essencial para a tomada de decisão de controle sobre novas tentativas de comunicação. Dependendo da tentativa de comunicação, tanto o estado da comunicação (derivado de comunicações passadas) quanto o estado da aplicação (derivado de outras aplicações) podem ser críticos para a decisão de controle. Logo, para garantir o alto nível de segurança, entende-se que o Firewall deva ser capaz de acessar, analisar e utilizar tais informações. A Figura 2 mostra a idéia do analisador deste tipo de Firewall.

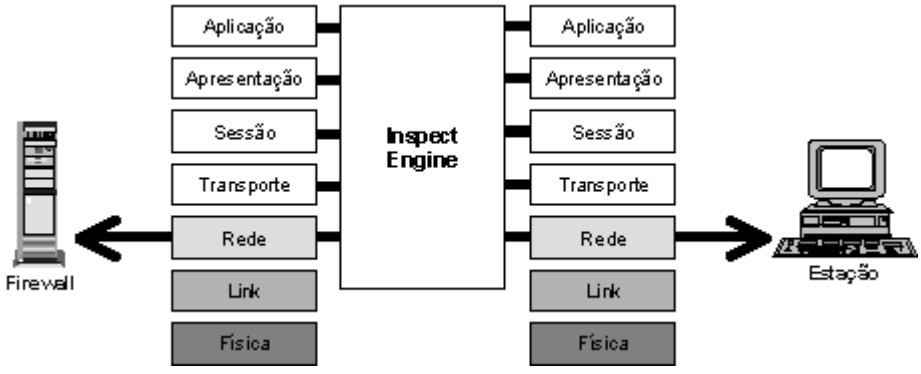

Figura 2 - Firewall Stateful Inspection.

Application Gateways

Há Firewalls que incluem a funcionalidade de ser um procurador[1] (Proxy) com um analisador e distribuidor de serviços da aplicação, verificando o tipo de serviço do pacote, como HTTP, FTP, Telnet etc. e redirecionando o pacote para o servidor que pode responder ao pedido.

1 Procurador: alguém que tem uma autorização para agir no lugar de quem assinou a procuração.

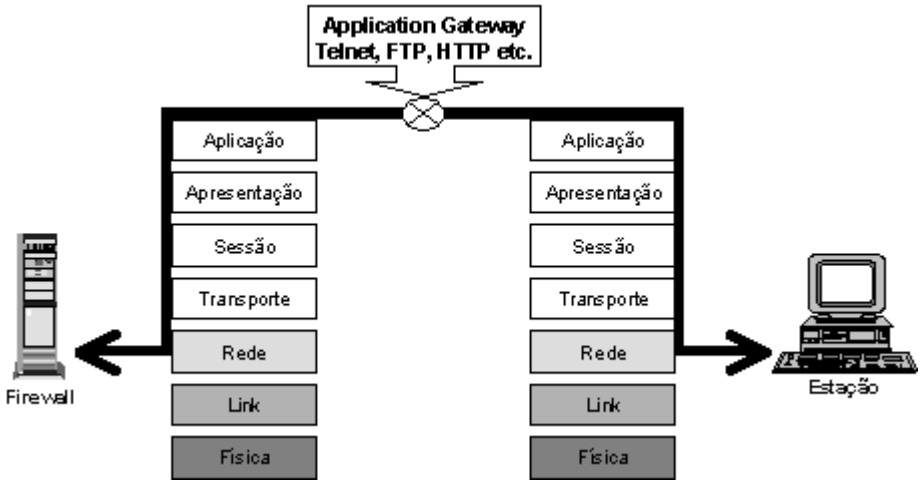

Figura 3 - Firewall do tipo Application Gateway.

Um Firewall que contém funcionalidade de *Application Gateway* analisa o pacote até a camada de Aplicação do modelo OSI, ou seja, conhece todo o protocolo de comunicação até a aplicação. Tal tipo de solução é bem mais segura, mas exige hardware mais poderoso em termos de processamento, uma vez que o analisador de pacote percorre todas as camadas. Outra limitação da solução é que há analisadores somente para serviços específicos. Assim, é necessário a um projetista conhecer a fundo cada protocolo de aplicação e escrever um analisador para cada aplicação que queira atender. Então, normalmente esta é uma solução limitada a determinados serviços.

Como exemplo de software que implementa um *Application/Gateway* orientado a determinados serviços, temos o squid, um proxy/cache especializado no protocolos HTTP, HTTPS, FTP e outros. Também faz cache das consultas DNS e possui uma série de funcionalidades para conversar com outros proxys, filtros de URL, autenticação com possibilidade de trabalhar integrada com sistemas centrais de SSO etc.

Um possível problema nesta solução é que o Firewall quebra a comunicação direta entre o cliente e o servidor, entrando no meio, ou seja, o cliente faz uma conexão com o Firewall que irá comunicar-se com o servidor através de outra conexão iniciada entre o proxy e o servidor. Em aplicações cliente/servidor, este nó no caminho às vezes traz algumas dores de cabeça aos desenvolvedores.

Por exemplo: no passado, alguns desenvolvedores de aplicativos seguros para bancos chegaram a usar o IP origem da conexão como parte da "segurança". O programa cliente codificava o IP origem e o mandava como dado para o servidor. O servidor comparava o IP informado e o IP origem de onde estava chegando o pacote. Se não coincidiam, a sessão não continuava e o serviço não funcionava. O método deu muita dor de cabeça para administradores de rede entre 1998 e 2000. No entanto, com a massificação do uso dos IPs designados para uso interno pela RFC1918 aliado ao uso de NAT (*Network Address Translation*), tal prática de "segurança" foi abandonada.

Texto 2 - Tecnologia – arquitetura

Veremos as arquiteturas de rede com Firewall mais empregadas. O livro *Building Internet Firewalls*, de Brent Chapman e Elizabeth D. Zwicky, usado como referência, é antigo mas trata-se de um clássico no assunto. No entanto, devido a sua idade, algumas "verdades" nele escritas já poderiam ser atualizadas.

Estaremos usando a Internet como rede externa "perigosa", mas tenha em mente que o conceito se aplica a qualquer outra rede que não esteja sob sua administração direta, sendo, portanto, não confiável.

Roteador filtrante

É o mais simples método para conexão à Internet (Figura 4). Colocam-se regras para filtragem de pacotes em um roteador que filtra o tráfego que vai ou vem da Internet para uma máquina presente na rede interna. Na Figura 4, apenas a "máquina filtrada" pode fazer e receber acessos da Internet (linhas cheias). Os demais pacotes que tentam entrar ou sair batem na parede (filtro) e são bloqueados (linhas pontilhadas).

Se houver necessidade de outras máquinas internas realizarem acessos à Internet, elas o farão a partir de pedidos encaminhados à "máquina filtrada",

que, por isso, também terá que rodar algum software de proxy e/ou *Application/Gateway*. O tráfego interno é mostrado por linhas cheias na Figura 4.

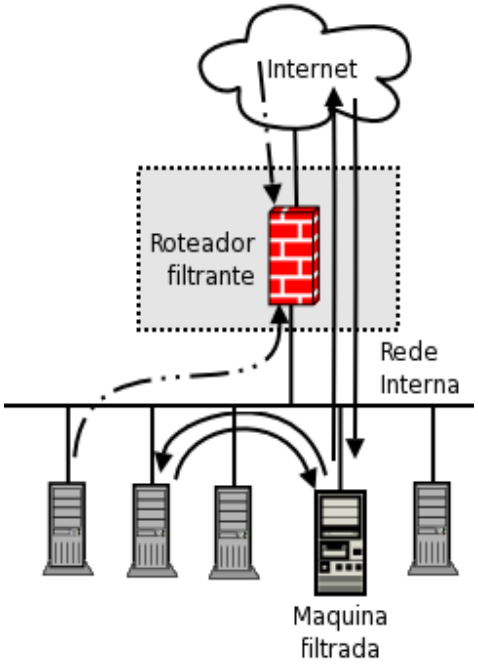

Figura 4 - Roteador Filtrante.

O problema com esta solução simples é que os mecanismos internos precisam ser seguros. A “máquina filtrada” tem que ter seu uso muito bem controlado, pois ela está dentro da rede interna e qualquer falha nela irá comprometer toda a segurança da rede.

Em algumas soluções deste tipo, é comum verificarmos a existência de várias “máquinas filtradas”, ou seja, que podem fazer acessos e/ou serem acessadas da Internet. Tal fato só aumenta a possibilidade de problemas, pois o número de máquinas e usuários a controlar é maior. Basta uma delas ser invadida ou pegar um vírus e toda a rede estará comprometida.

Outro problema é que as regras de filtragem utilizadas nos roteadores de mercado são *stateless,* o que dificulta a manutenção da segurança.

Máquina com dupla base

Nesta arquitetura, mostrada na Figura 5, o roteador é substituído por um computador de uso genérico que roda um sistema operacional otimizado para operar como Firewall. O computador possui duas placas de rede: uma ligada na rede interna e outra ligada à Internet[2]. Pode ter como sistema operacional, por exemplo, um Linux com kernel 2.4 ou 2.6. Neste caso, o sistema de filtragem é *stateful* e há a possibilidade de rodar alguns softwares que atuem como *Application/Gateway* para serviços muito utilizados, tais como: acesso web (HTTP, HTTPS) e correio eletrônico com sistemas especialistas (anti-spam/anti-vírus).

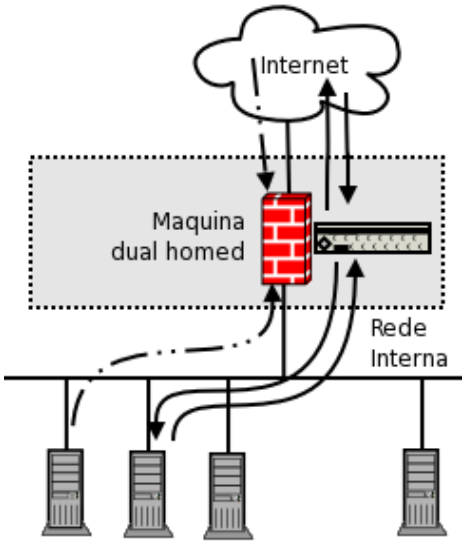

Figura 5 - Firewall com máquina com duas placas de rede.

Esta máquina sempre será dedicada, ou seja, não pode ter usuários em seu console. É considerado um servidor que deve ser cuidadosamente escondido na sala de servidores!

Novamente, a segurança da rede interna fica dependente de uma única máquina: quanto mais aplicativos rodarem nela, maior a possibilidade de uma

2 Placa normalmente ligada a um roteador da operadora do link, não mostrado na ilustração.

vulnerabilidade comprometer a rede interna. Mesmo os aplicativos que prestam serviço de *Application/Gateway* devem ser considerados como potenciais fontes de vulnerabilidades.

Sub-rede filtrada

Trata-se de uma arquitetura das mais seguras (Figura 6). O Firewall deixa de ser uma única máquina e se transforma em um complexo de equipamentos que, juntos, formam o conceito de Firewall. Utiliza pelo menos dois roteadores filtrantes, um externo e um interno, e mais uma máquina que roda serviços de proxy e/ou *Application/Gateway,* por exemplo, para HTTP e SMTP. Esta máquina também pode conter aplicativos de serviços públicos, tais como DNS, HTTP, FTP.

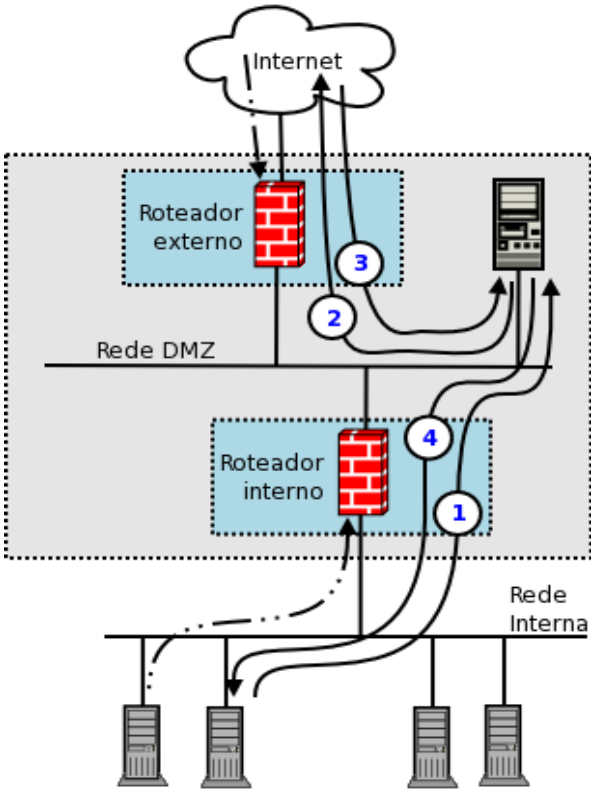

Figura 6 - Firewall com sub rede filtrada.

Nesta arquitetura, existe uma rede dentro do “complexo” do Firewall. Como a rede se encontra em uma zona neutra, ou zona desmilitarizada, é chamada de

DMZ (*De-Militarized Zone*[3]).

Tanto o roteador externo quanto o roteador interno possuem regras de filtragem de pacotes. Mas é importante que possuam sistemas de filtragem com *Stateful Inspection*. Da mesma forma que vimos anteriormente, tais roteadores podem ser substituídos por computadores rodando sistemas otimizados para operar como Firewall. Só que, no caso, nada mais roda nas máquinas além do mínimo para se ter o sistema de filtragem de pacotes. Então, o perigo de existir alguma vulnerabilidade futura em um software rodando é praticamente zero. No lugar desses roteadores é comum encontrarmos os equipamentos que, no mercado, são denominados Firewall.

Na rede DMZ é possível colocar diversos servidores para prestar os mais variados serviços públicos ou sistemas especializados em proxy para serviços específicos.

A Figura 6 mostra um exemplo de acesso com fluxo de dados. Um cliente na rede interna que necessita fazer algum acesso à Internet:

- inicia fazendo um pedido para o proxy (fluxo 1);
- este envia o pedido para a Internet (fluxo 2);
- ao receber a resposta (fluxo 3), o proxy a envia para o cliente (fluxo 4).

Nas regras de filtragem nunca existirá um caminho direto entre a Internet e a rede interna. Da mesma forma, todo esforço deve ser feito para evitar qualquer caminho direto entre a rede interna e a Internet. Agindo assim, a segurança será a melhor possível.

Como desvantagem desta arquitetura, pode-se citar sua maior complexidade em manutenção das regras, já que um mesmo serviço deverá ser configurado nos dois sistemas com filtros de pacotes. Criar uma documentação com figuras e fluxos de dados, similar à Figura 6, para cada serviço e/ou servidor na DMZ facilita a manutenção.

3 Denominação utilizada após a criação da zona que separa a Coréia do Norte e do Sul.

Firewall com três placas de rede

Embora o sistema da Figura 6 seja o mais seguro, além da complexidade, ele pode ter um outro problema: o preço! Se o projetista de rede quiser comprar sistemas comerciais de marcas famosas para tomar o lugar dos dois roteadores, o custo tomará vultos impraticáveis para a maioria dos casos.

Uma melhor relação custo benefício pode ser alcançada fazendo-se uma união das funcionalidades dos dois roteadores em um único equipamento, conforme mostra a Figura 7.

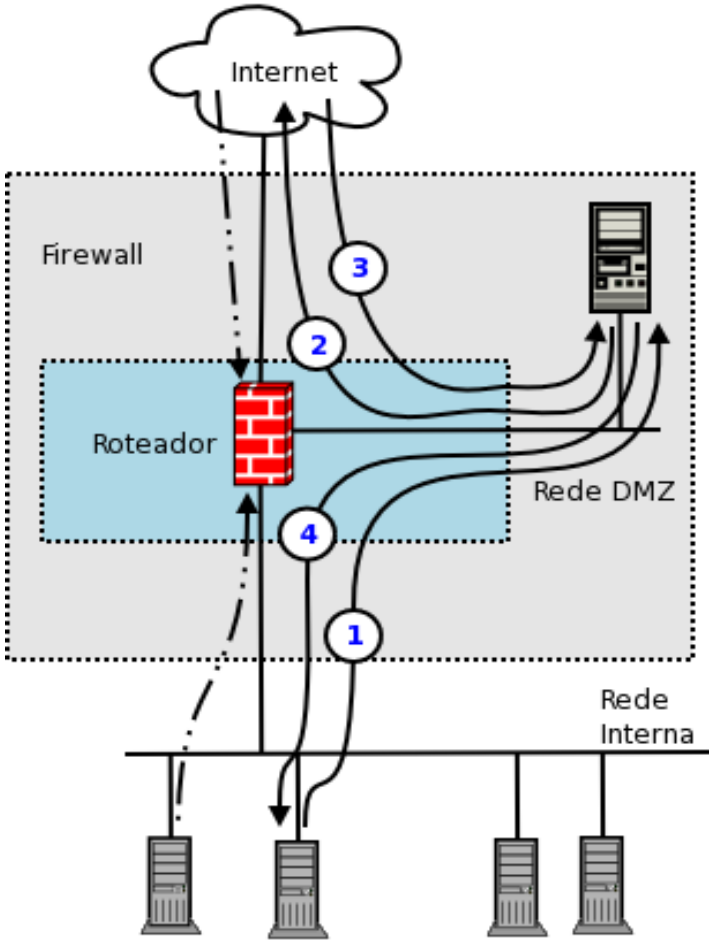

Figura 7 - Firewall com DMZ - três placas de rede.

Neste tipo de arquitetura o roteador é normalmente denominado “O Firewall” propriamente dito. É essencial que ele seja bastante confiável, pois a ocorrência de qualquer problema nele comprometerá a rede interna. Um simples erro de configuração, por exemplo, pode abrir o caminho.

Outra questão a considerar no momento do projeto: se a rede DMZ contiver serviços importantes a serem usados pelos clientes na rede interna e “O Firewall” sofrer um ataque DoS vindo da Internet e sucumbir, então também a rede interna ficará sem comunicação com os servidores na DMZ. Isto não acontece no caso anterior, com o sistema de filtragem interno separado do externo. Se o externo cair, a comunicação entre a rede interna e DMZ não é afetada.

A despeito de ter a segurança minimizada, tal arquitetura dominou o mercado, impulsionada pelas empresas que vendem caros sistemas de marca. Com o advento do NAT e melhoria na confiabilidade do equipamento, até mesmo o proxy na DMZ tem sido abandonado. No caso, existe uma regra que permite fluxo direto da rede interna para a Internet para alguns serviços (portas), e o Firewall *stateful* se encarrega de deixar passar só as respostas da Internet para a rede interna.

Uma solução de compromisso, se alguma ordem de cunho não técnico obrigar a utilizar os caros sistemas de marca, é manter a arquitetura anterior (Figura 6), usando um sistema de marca no Firewall externo e outro, tão seguro quanto mais econômico, no Firewall interno!

Unidade 2 - Filtragem de pacotes por IP/serviço

Faremos aqui uma introdução aos mecanismos usados nas construções do mais importante componente para a segurança de uma rede: o filtro de pacotes do Firewall. Vamos excursionar desde alguns conceitos necessários até a prática para implementação de filtragem simples em roteadores Cisco e GNU/Linux com kernel 2.4/2.6 (comando iptables). A partir daí, cabe a você, interessado, buscar informações e se especializar para conseguir criar conjuntos de regras mais complexas.

Texto 1 - Roteador X Firewall

Por que "pacotes"?

A unidade de transmissão em redes TCP/IP é o pacote-IP. Então, tipicamente uma mensagem é dividida em vários pedaços, cada um contido em um pacote-IP. Por isso, os sistemas de roteamento e filtragem tratam pacote por pacote.

Nos roteadores, os pacotes-IP são encaminhados até seu destino através de uma seqüência de roteadores, ou *rota*. Cada roteador pode ser um sistema genérico (UNIX, DOS etc.) ou um sistema dedicado. Em uma rede, cada roteador determina qual será o próximo passo na rota do pacote.

Todo Firewall é um roteador!

Tomando o termo Firewall como sendo o equipamento que faz a inspeção do fluxo de dados (pacotes), vamos compará-lo com um roteador. O que tem um roteador que um Firewall não tem? A princípio, nada!

Obrigatoriamente, o Firewall tem que tratar os cabeçalhos dos pacotes IP e decidir para onde o pacote será enviado, se sua passagem for permitida. Esta

decisão de permitir ou não a passagem é a única coisa que um simples roteador não faz. Por essa razão, ao se implementar regras de filtragem em um roteador, é comum passar a chamá-lo de “Firewall”.

Um consenso em termos de segurança é que um Firewall não deve rodar nenhum protocolo de roteamento dinâmico, tais como RIP, OSPF, BGP etc. Um protocolo de roteamento dinâmico pode ser usado para um ataque de DoS pois, se for possível a um atacante burlar o sistema de roteamento dinâmico, ele pode criar uma rota falsa para a rede interna ou mesmo para a Internet, parando o sistema todo.

Em sistemas simples, um pacote é verificado inicialmente pelo filtro. Se passar, é encaminhado pelo sistema de roteamento. Um administrador, conhecendo sua rede e o roteamento, pode criar regras associadas a uma determinada interface de rede e, assim, decidir se um pacote passa ou não, baseado na rede de onde veio o pacote (ou para onde vai). Você, atentamente, poderia perguntar: mas sabendo o IP origem já não sei de que rede veio? Se o IP origem é o da minha rede interna, então ele veio de dentro, não? Resposta: _____

_____

_____

Em sistemas mais completos, o sistema de filtragem “conhece” e conversa com o sistema de roteamento. Logo, é possível criar regras em que a interface de entrada ou de saída ou ambas é citada. Note que, para que o filtro fique sabendo qual a interface de saída, o pacote já tem que ter passado pela etapa de roteamento. Nestes sistemas, a filtragem e o roteamento operam cooperativamente.

Um roteador fiscal (*screening router*) também decide SE o pacote deverá seguir sua rota, de acordo com a política de segurança ditada pelas regras de filtragem.

A filtragem de pacotes permite controlar a passagem de pacotes baseada em:

- o endereço de onde (supostamente) vem o pacote;
- o endereço de destino;

- o protocolo de aplicação (TCP/UDP/ICMP) sendo usado;
- porta origem TCP/UDP;
- porta destino TCP/UDP;
- detalhes dependentes do protocolo. No TCP, por exemplo, analisa se é um pedido de conexão ou se o pacote pertence a uma conexão já estabelecida. No ICMP, pode analisar o tipo de mensagem e a mensagem propriamente dita, além de verificar se este ICMP diz respeito a alguma conexão já estabelecida ou em fase de estabelecimento.

Por que filtrar pacotes

Em relação a decisões de filtragem, observe que:

- são livres de contexto;
- não utilizam os dados da aplicação; não entendem de “usuário” nem de “arquivo”;
- permitem limitar todo acesso (ou a algumas aplicações) a partir de determinados locais;
- permitem negar acesso a todos os sistemas numa rede protegida.

Mas, atenção: o endereço de origem pode ser forjado! Não se deve depender somente dele. Saber qual a interface de chegada pode permitir determinar se o IP origem foi forjado. Por exemplo: se vier um pacote tentando entrar pela interface que está ligada na Internet com o IP origem fazendo parte do bloco de endereçamento de sua rede interna, certamente este IP origem foi forjado.

Vantagens de filtragem

Lembre-se que:

- um roteador fiscal pode proteger uma rede inteira;
- filtragem não depende da cooperação dos usuários;
- filtragem está amplamente disponível em produtos comerciais e em

pacotes de software.

Desvantagens de filtragem

Já a lista de desvantagens é mais longa:

- ferramentas imperfeitas;
- regras de filtragem são difíceis de configurar;
- as regras são difíceis de testar;
- certos filtros desejáveis são difíceis ou impossíveis de usar;
- software de filtragem pode apresentar riscos (bugs);
- alguns protocolos são difíceis de tratar só com filtragem. Por exemplo, NIS, comandos “r” de Berkeley;
- algumas regras desejáveis não podem ser usadas pois os filtros não conhecem “usuário”, “aplicação” e “arquivo”.

Um filtro de pacotes implementado em um roteador tem a vantagem de poder proteger uma rede, mas é preciso ter cuidado: a desvantagem está na nova carga de processamento. Deve-se verificar se será suportada pelo hardware do roteador. A criação indiscriminada de dezenas e, às vezes, centenas de regras em roteadores – cuja funcionalidade primária é rotear e não ser Firewall – é capaz de causar retardo ou até perda de pacotes, fazendo o usuário final experimentar lentidão nos serviços, mesmo estando os enlaces de dados (link) com uso abaixo do valor nominal da vazão contratada (bps).

Texto 2 - Processamento em um filtro

Um conjunto de regras de filtragem é sempre aplicado em seqüência, uma após a outra. Assim que a primeira regra for satisfeita, o sistema pára de verificar as regras e toma a ação especificada na regra em questão. Apenas se não houver ação especificada na regra, o sistema continuará a submeter o pacote às regras seguintes.

Pode-se pensar em um conjunto de regras como uma seqüência de testes que pára assim que um teste for satisfeito. Uma regra só executa sua ação e paralisa a busca nas demais regras se **todos** os testes desta regra forem positivos. Se apenas um falhar, imediatamente o sistema passa para a regra seguinte.

Então, mudar a ordem em que as regras estão escritas pode culminar em diferentes resultados.

Vejamos um exemplo fácil, sem nos atermos à sintaxe de nenhum equipamento ou software: suponha que nosso sistema de Firewall é bem simples e que temos que nos comunicar com somente uma máquina na Internet, a partir de um servidor interno, conforme ilustra a Figura 8.

Neste caso, vamos criar regras simples como as da Tabela 1.

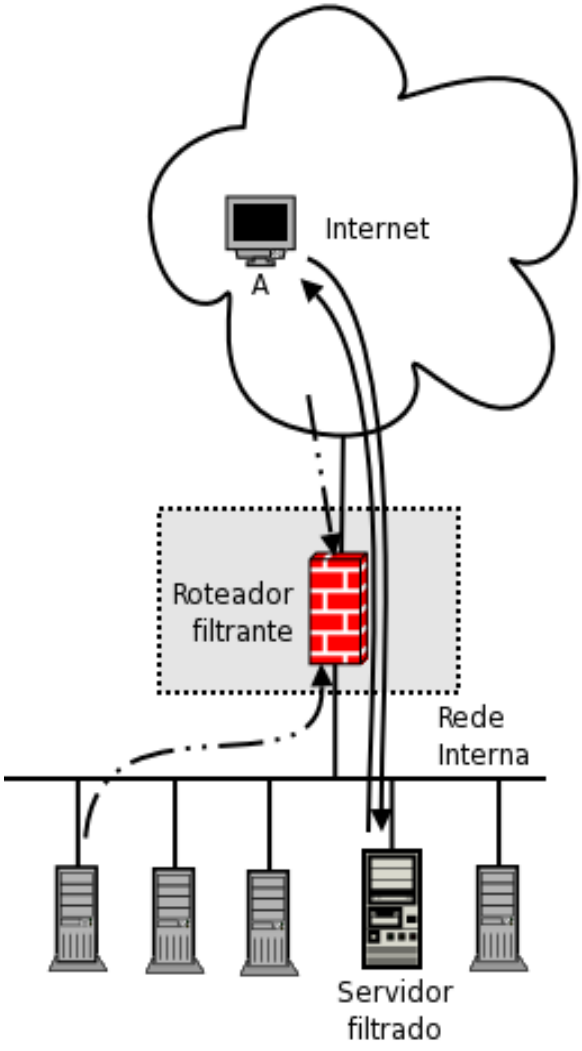

Figura 8 - Comunicação entre o servidor interno e uma estação externa.

Ordem	*Origem*	*Destino*	*Ação*
1	Servidor interno	Estação A	Passa
2	Estação A	Servidor interno	Passa
3	Qualquer	Qualquer	Bloqueia

Tabela 1 - Regras para comunicação entre um servidor interno e uma estação externa.

- Supondo um pacote que tenta ir do **servidor interno** para a estação **A** na Internet.

A seguinte seqüência de processamento acontece:

1. A origem é o **servidor interno E** o destino é a estação **A**?

   SIM -> Executa a ação: PASSA.

Rápido, pois o pacote satisfez a primeira regra.

- Supondo um pacote que tenta ir da estação **A** para o **servidor interno** na Internet.

A seguinte seqüência de processamento acontece:

1. A origem é o **servidor interno E** o destino é a estação **A**?

   NÃO

2. A origem é a estação **A E** o destino é o **servidor interno**?

   SIM -> Executa a ação: PASSA.

O pacote não satisfez a primeira regra, mas satisfez a segunda.

- Supondo uma estação interna qualquer tentando mandar um pacote para qualquer lugar na Internet.

O fato dela não satisfazer a condição de ser “o servidor interno” faz com que as regras 1 e 2 não sejam satisfeitas e que seja verificada a regra 3. Esta regra é um “coringa”, pois qualquer pacote, vindo ou indo de/para qualquer lugar, vai satisfazê-la. Assim sendo, a regra 3 executa a sua ação – Bloqueia – e a busca termina.

E se seu sistema evoluir e for instalada uma nova máquina na Internet (de identificação B) para a qual é necessário mandar dados do mesmo servidor interno? Veja a Figura 9.

Ao receber a notícia, o administrador rapidamente acrescenta uma regra na tabela de regras, que fica como na Tabela 2. O que vai acontecer? Nada!

Vejamos na seqüência a seguir o que ocorre com um pacote que tem origem no “servidor interno” e destino na estação “B”. Atenção: as regras são sempre verificadas uma a uma, a partir da primeira.

1. A origem é o **servidor interno E** o destino é a estação **A**?

   NÃO

2. A origem é a estação **A E** o destino é o **servidor interno**?

   NÃO

3. A origem é “qualquer uma” **E** o destino é “qualquer um”?

   SIM -> Executa a ação: **BLOQUEIA**.

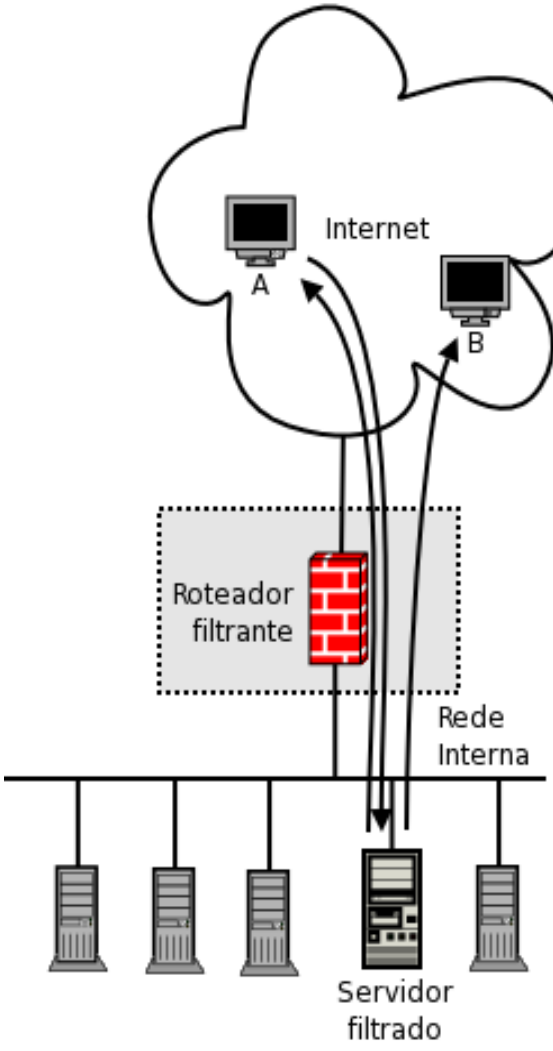

Figura 9 - Comunicação com duas estações externas.

Ordem	*Origem*	*Destino*	*Ação*
1	Servidor interno	Estação A	Passa
2	Estação A	Servidor interno	Passa
3	Qualquer	Qualquer	Bloqueia
4	Servidor interno	Estação B	Passa

Tabela 2 - Adicionando, apressadamente, uma nova estação na lista de regras.

A regra “3” foi satisfeita e o pacote bloqueado. Como a regra “3” é um coringa, é sempre satisfeita e sua ação sempre executada. Neste caso, na

prática, a regra 4 nunca será sequer verificada e nunca chegará nada para a estação B. O que fazer?

Deve-se reescrever as regras de tal forma que as mais abrangentes (os coringas) fiquem sempre mais para o fim da seqüência. Para o exemplo dado, veja as corretas na tabela:

Ordem	*Origem*	*Destino*	*Ação*
1	Servidor interno	Estação A	Passa
2	Estação A	Servidor interno	Passa
3	Servidor interno	Estação B	Passa
4	Qualquer	Qualquer	Bloqueia

Tabela 3 - Seqüência correta de regras para acessar também a estação B.

Então, o processamento de um pacote por um filtro possui como ação final duas opções:

- permitir passar; ou
- bloquear (drop).

Normalmente, existe uma opção que permite registrar (syslog) todos os pacotes, somente os bloqueados ou apenas para algumas regras. Para TCP, só se deve registrar o pacote inicial de uma conexão, a fim de minimizar o tamanho do arquivo de logs.

O processamento de um pacote bloqueado por um filtro pode, ainda, gerar mensagem ICMP de retorno ao IP origem do pacote. Existem três alternativas:

- “destino inalcançável”: pode causar aborto de outras conexões IP da mesma origem;
- “destino administrativamente inalcançável”: revela que existe um filtro de pacotes e que o pacote foi bloqueado por ele;
- silêncio: não revela nada. É a alternativa recomendada, com poucas exceções.

Texto 3 - Criando filtros

A configuração de um roteador para fazer filtragem deve seguir o procedimento:

- determinar a “postura padrão” a ser utilizada (veremos adiante);
- determinar quais serviços deseja permitir/negar por máquina ou rede;
- traduzir suas características em regras de filtragem dos pacotes que implementam estes serviços, lembrando que protocolos e serviços são bidirecionais.

Serviços bidirecionais, tráfego idem

Para implementar um dado serviço, os pacotes geralmente passam em ambos os sentidos. Então, é necessário distinguir entre e ter regras distintas para:

- serviço externo usado por um usuário interno;
- o mesmo serviço, agora interno, usado por um usuário externo.

A Figura 10 ilustra um exemplo para um serviço qualquer:

- as linhas grossas indicam o serviço e sua direção;

  Essa direção também é a do primeiro pacote que trafega.

- as linhas finas mostram os fluxos de dados;

  Para a maioria dos serviços, o fluxo de dados acontece sempre nos dois sentidos. É necessário analisar e ter regras para cada um dos quatro tráfegos mostrados, pois cada um tem características que permitem distingui-lo dos demais.

- a seta cheia indica a direção em que passa o primeiro pacote.

  No TCP, esse primeiro pacote é o pedido de conexão. E no UDP? No UDP, é apenas o primeiro pacote: nada o diferencia dos demais, a não ser o fato de ser o primeiro!

Relembre: o TCP possui em seu cabeçalho uma série de bits de controle. O primeiro pacote, o que indica o pedido de conexão, possui dois bits obrigatoriamente em valores determinados. São eles:

- BIT _____ com valor _____.
- BIT _____ com valor _____.

Nos demais pacotes de uma conexão TCP, o BIT _____ estará sempre _____.

Tais considerações sobre esses bits de controle são muito importantes, pois eles permitem ao filtro de pacotes diferenciar o tratamento para cada um dos fluxo de dados.

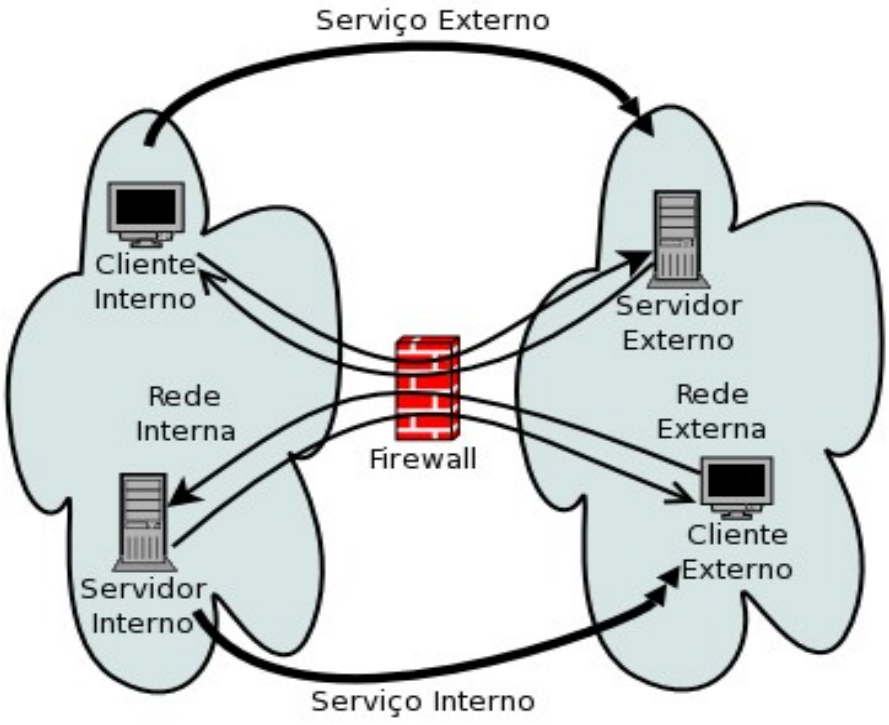

Figura 10 - Serviço interno e externo: cada um com dois fluxos.

Postura padrão

Ao escrever regras para um sistema de filtragem, devemos ter em mente a postura padrão que o sistema usa ou a que você pretende escolher. Ela determina a necessidade ou não de várias regras. Essa postura pode ser:

- liberal ou permissiva: significa permitir tudo que não for explicitamente negado;
- conservadora ou restritiva: significa negar tudo que não for explicitamente permitido.

A escrita de regras é completamente diferente, dependendo da postura padrão.

Na postura permissiva a última regra permite passar tudo. Às vezes, essa última regra é implícita, ou seja, não precisa ser escrita. Então, normalmente as regras são, em sua maioria, de bloqueio, exceto a última (que pode ser implícita). Um pacote que não satisfizer a nenhuma das regras será encaminhado (roteado) ao seu destino (passa). Se um pacote satisfizer a uma regra, será bloqueado.

Na postura restritiva a última regra bloqueia tudo. Esta regra pode ser implícita. Então, normalmente as regras são, em sua maioria, de permissão. Um pacote que não satisfizer a nenhuma das regras, exceto a última, será bloqueado. Se um pacote satisfizer a uma regra, será roteado.

As regras da Tabela 3 são permissivas ou restritivas? R. _____

A intenção deste módulo é mostrar o início do caminho para que os interessados em sistemas de Firewall tenham por onde começar.

Cada sistema de filtragem utiliza sua "linguagem". Utilizaremos nossa própria linguagem genérica, em forma de tabelas, como já apareceu anteriormente, e, depois, traduziremos as regras para as seguintes plataformas:

- Cisco – Lista de acesso estendida (*Extended ACL*);
- GNU/Linux – Kernel 2.2.x: ipchains;
- GNU/Linux – Kernel 2.4.x/2.6.x: iptables.

Texto 4 - Lista de Acesso Estendida (ACL)

> Regras de filtragem criadas com listas de acesso (ACL) são *STATELESS*!

A maioria dos administradores de roteadores com IOS[4] Cisco domina a utilização das listas estendidas. Mas é necessário ter cuidado, pois utilizada de maneira simples, as listas de acesso têm comportamento *stateless*, ou seja, será sempre necessário ter regras explícitas de ida e de volta, e o sistema não associa os pacotes de uma conexão ou não sabe se um pacote UDP é uma pergunta ou resposta. Existem comandos para tornar o comportamento *stateful,* mas sua utilização não tem sido o padrão.

No sistema de edição por terminal com IOS mais antigos, há, ainda, o problema de não existir edição. Não é possível incluir uma regra entre a regra 5 e a 6, por exemplo. Só é possível incluir regras no fim da lista. Assim, a maioria dos administradores faz uma conexão remota com o equipamento, mantendo em seu micro uma janela com o console do roteador. Então, usa um editor de texto no seu próprio micro, inclui ou retira a(s) regra(s) onde deseja no arquivo texto e, voltando ao terminal remoto, apaga toda a atual lista e, depois, cola a nova lista recém-editada (isso mesmo: *copy and paste*).

Nos IOS mais novos é permitido ao administrador identificar cada regra por um número inteiro, de modo que as regras possam ser referenciadas e ordenadas. Agora, a ordem que vale é a da identificação numérica, e não a ordem em que as regras são escritas. Mas ainda não existe edição que permita inserir entre a regra de número 5 e a de número 6. Então, os administradores costumam numerar as regras de 10 em 10, para facilitar uma possível necessidade de inserir alguma regra depois, já que, desta forma, é possível inserir uma regra 15, que ficará entre a regra 10 e a 20! (e voltamos ao basic! Alguém lembra?)

4 Corresponde ao sistema operacional do roteador.

A primeira ACL

Vamos agora usar nossa lista de regras que permitem comunicação com as máquinas A e B na Figura 9, a partir das regras da Tabela 3. Porém, esta tabela de regras precisa ser melhorada. Por isso, substituímos "qualquer" por um "*" (asterisco – o coringa, que pode aparecer em qualquer coluna) e inserimos colunas para melhorar a filtragem do TCP/IP. Temos, então, a Tabela 4.

#	Dir	IP Origem	IP Destino	Prot.	P.O.	P.D.	ACK	Ação
1	S	Servidor interno	Estação A	*	*	*	*	Passa
2	E	Estação A	Servidor interno	*	*	*	*	Passa
3	S	Servidor interno	Estação B	*	*	*	*	Passa
4	*	*	*	*	*	*	*	Bloqueia

Tabela 4 - Regras mais completas para acesso de/para as estações A e B.

Na tabela temos as colunas:

- **#**: número da regra;
- **Dir**: direção do pacote – Entrada, Saída;
- **IP Origem**;
- **IP Destino**;
- **Prot**: Protocolo acima do IP (ICMP, UDP, TCP);
- **P.O.**: Porta Origem;
- **P.D.**: Porta Destino;
- **ACK**: Condição do BIT ACK, se o protocolo for o TCP (Ligado, Desligado);
- **Ação**.

Em um sistema de filtragem com ACL (Cisco), as regras têm que ser aplicadas a uma interface. Perceba que já identificamos os pacotes que entram e que

saem da rede interna na coluna direção. Para que isto possa ser mantido assim, vamos aplicar as regras na interface externa do roteador/Firewall. Por quê? Porque o roteador sempre vê os pacotes em relação a seu próprio umbigo! Então, os pacotes que estão indo da Internet para a rede Interna, entram pela interface externa, passam pelo roteador, e saem dele pela interface interna, entrando na rede interna. Ou seja, se as regras forem feitas para serem aplicadas na interface interna do roteador, a direção dos pacotes nas regras precisa ser invertida.

Outro fator importante: um conjunto de regras (ou ACL) tem que ser aplicado a uma interface para pacotes em uma direção determinada pelo comando que aplica a ACL: ou para pacotes entrando (*in*), ou para pacotes saindo (*out*) da interface.

Na Tabela 5, mostramos as máquinas com endereços IP. São endereços fictícios e, inclusive, não podem ser usados na Internet, por pertencerem à lista de endereços de uso privado estabelecida na RFC-1918.

#	Dir	IP Origem	IP Destino	Prot.	P.O.	P.D.	ACK	Ação
1	S	192.168.5.43	10.12.0.33	*	*	*	*	Passa
2	E	10.12.0.33	192.168.5.43	*	*	*	*	Passa
3	S	192.168.5.43	172.18.1.3	*	*	*	*	Passa
4	*	*	*	*	*	*	*	Bloqueia

Tabela 5 - Regras de acesso com endereços IP.

Traduzindo para ACL

As ACLs possuem uma sintaxe, resumida a seguir. Em cada linha de uma ACL há os campos:

```
access-list Número Ação Protocolo Origem Destino Log
```

Onde:

- access-list: palavra chave do comando;
- Número: maior que 100 e menor que 200;

- Ação: permit ou deny;
- Protocolo: ip, tcp, udp, icmp, número de protocolo;
- Origem e Destino: Número IP ou Rede e máscara (*hostmask*);

Uma novidade: a *netmask*, ou máscara de rede, que já conhecemos, deve ser invertida bit a bit para formar a *hostmask*. Por exemplo, a máscara padrão de uma classe "C" é 255.255.255.0. A *hostmask* equivalente é 0.0.0.255.

Qual é a *netmask* para indicar um único *host*? Resp.: 255.255.255.255.

Então, para indicar uma rede classe "C", por exemplo, o campo origem (ou destino) fica: 192.168.130.0 0.0.0.255

Para indicar o endereço de uma única máquina, fica: 10.1.2.7 0.0.0.0

Você pode pensar na *hostmask* como um número que, somado ao IP, indica o último IP do range. Por exemplo: 192.168.130.0 + 0.0.0.255 = 192.168.130.255. Logo, o primeiro endereço da rede é 192.168.130.0 e o último é 192.168.130.255.

Uma especificação 0.0.0.0 255.255.255.255 aponta qualquer endereço, ou seja, nosso "*".

Se as ACLs são *stateless*, então é necessário estabelecer regras para pacotes entrando e regras para pacotes saindo. Além disso, precisamos pensar separadamente em criar uma ACL para pacotes que saem e outra para pacotes que entram. Nossa tabela já nos auxilia a separar os dois casos.

Desse modo, os comandos a seguir criam suas ACLs, uma denominada 101 e a outra, 102:

```
access-list 101 permit ip 10.12.0.33 0.0.0.0 192.168.5.43 0.0.0.0
access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
access-list 102 permit ip 192.168.5.43 0.0.0.0 10.12.0.33 0.0.0.0
access-list 102 permit ip 192.168.5.43 0.0.0.0 172.18.1.3 0.0.0.0
access-list 102 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

A última regra de nossa tabela, a que bloqueia tudo, tem que ser escrita em cada ACL, tanto na 101 quanto na 102, pois, conforme veremos, o roteador só examina uma das listas para um dado pacote.

Após escrevermos as listas de acesso, nada estará ativo ainda. É preciso aplicar as listas (ACLs) em uma ou mais interfaces para que passem a fazer efeito. Supondo que a interface externa seja chamada “serial 1”, a seguinte seqüência de comandos aplica e torna ativa as regras:

```
interface serial 1
access group 101 in
access group 102 out
```

Assim, após os comandos, a lista de regras 101 passa a examinar todos os pacotes que entrarem pela interface serial 1 e a lista de regras 102 passa a examinar todos os pacotes que saírem pela interface serial 1.

O sistema de filtragem da Cisco pretende ser inteligente e determina a postura padrão de maneira automática: se suas regras são “permit” implicitamente, a última regra será um “deny”. E vice-versa. Mas, para evitar supresas, é preferível escrever a última regra explicitamente, o que facilita, inclusive, o entendimento e a socumentação.

E no Linux?

Netfilter e iptables

O Netfilter é o sistema de filtragem e manipulação de pacotes desenvolvido para o kernel 2.4 e ainda utilizado no kernel 2.6. É integrado ao kernel, mas é um projeto independente (http://www.netfilter.org).

Para configurar o Netfilter, usa-se o comando iptables – quem deseja dominar a técnica de construir Firewall com GNU/Linux deve estudá-lo. Aqui apresentaremos exemplos básicos.

Atenção: em relação ao kernel 2.2.x, a arquitetura interna de passagem de pacotes mudou completamente, (comando ipchains). Esqueça o kernel 2.2 e o ipchains!

Pode ligar o seu SLAX Linux e teclar:

```
man iptables
```

Não se assuste: o iptables, conforme o manual mostra, possui MUITOS parâmetros. Veremos apenas um resumo com o mínimo necessário agora.

O Netfilter possui a noção de *chain*. Traduziremos por “cadeia”, no sentido de cadeia de regras: as regras são encadeadas, uma após a outra, e verificadas em seqüência. (Com certeza a “cadeia” que lhe veio a mente foi outra... Sinal dos tempos em que vivemos).

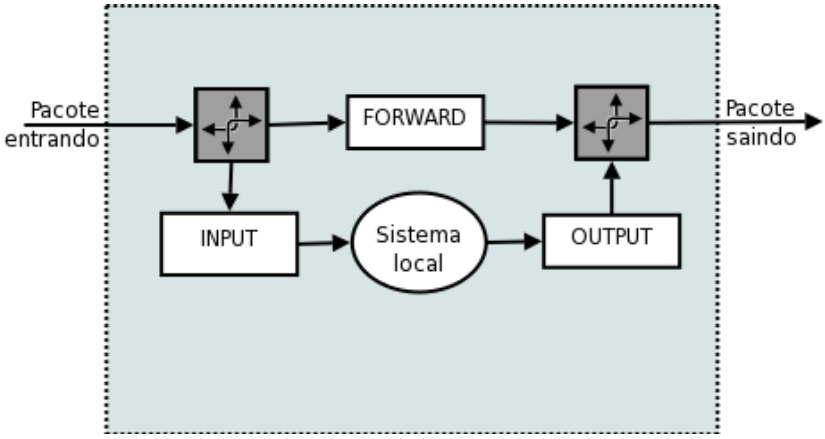

Figura 11 - Arquitetura do sistema de filtros no Netfilter.

No subsistema de filtragem (*filter*), existem três cadeias preexistentes e que não podem ser destruídas: INPUT, FORWARD e OUTPUT. A Figura 11 mostra a arquitetura interna da localização destas cadeias em relação ao sistema de Firewall e em relação ao sistema local (a própria máquina). Note que a cadeia de forward funciona como se estivesse à parte da máquina. Apenas pacotes que entram em direção ao sistema local passam pela cadeia de input e somente pacotes que saem do sistema local passam pela cadeia de output. Pacotes que apenas atravessam a máquina como a um roteador, e que não se destinam ou se originam dela, passam só pela cadeia de forward.

Em um comando iptables, cada parâmetro normalmente começa com um hífen e uma letra. Alguns começam com dois hífens (--). A seguir, apontamos parâmetros necessários agora:

-F [cadeia]

Apaga (*flush*) todas as regras da cadeia especificada. Se nenhuma for especificada, apaga de todas as cadeias.

-A cadeia

Acrescenta a regra no fim da cadeia especificada.

-I [num] chain

Inclui a regra no início (primeira regra), ou como regra número “num”.

-i interface

Verifica se o pacote entrou (in) pela interface. Não pode ser usado na cadeia de OUTPUT. Exemplos de nomes válidos para interfaces: eth0, eth1, ppp0, ipsec0 etc.

-o interface

Verifica se o pacote sairá pela interface. Não pode ser usado na cadeia de INPUT.

-p protocolo

Especifica o protocolo. Exemplos: ICMP, TCP, UDP, número etc. A relação com os protocolos se encontra no arquivo /etc/protocols.

-s origem

-d destino

Especificam os endereços IP no formato IP/MASK. Se a máscara não for explicitada, então será usada /32, que indica somente o IP especificado.

--sport porta

--dport porta

Indicam a porta origem (*source*) e destino. Devem ser usados apenas com protocolo UDP ou TCP. Por isso, para usar estes parâmetros, o parâmetro -p precisa ser especificado também.

-j ação

Especifica a ação a ser tomada, caso todos os testes sejam positivos. No momento, as ações existentes e importantes são: ACCEPT, DROP, REJECT, LOG. No Netfilter, fazer o "log" é uma ação que não causa o término da verificação da seqüência de regras, pois não é uma ação a ser aplicada ao pacote. Apenas é tirada uma foto dele para ser fichado!

Assim, após uma regra com ação LOG, deve existir alguma outra com uma ação sobre o pacote, mesmo que seja a regra padrão da cadeia.

A diferença entre um DROP e um REJECT está na resposta do filtro. No primeiro caso, não retorna nada, ficando em silêncio. No segundo caso é enviado um pacote ICMP para o IP origem do pacote que foi bloqueado, informando o ocorrido.

-m state --state NEW

Esta seqüência está relacionada com o sistema *stateful* do Netfilter. Ela é um teste que irá se satisfazer se o pacote for um pedido de conexão TCP ou um pacote não relacionado a alguma conexão estabelecida, ou seja, armazenada na memória do *stateful*.

-m state --state ESTABLISHED,RELATED

Esta seqüência relaciona-se ao sistema *stateful* do Netfilter. Ela é um teste que irá se satisfazer com todos os pacotes que pertencerem a uma conexão TCP já estabelecida ou pacotes relacionados a estas como, por exemplo, pacotes ICMP. Ela também é um teste que dá positivo para pacotes UDP que

sejam resposta de um serviço. O Netfilter guarda informação de cada pacote UDP que passa e espera 30 segundos pela resposta, antes de apagar a informação da memória. Se a resposta só chegar depois deste tempo, não irá passar.

-N nome_de_cadeia

Como o usuário (administrador) pode impor suas próprias cadeias, com este parâmetro cria uma cadeia vazia. A partir da criação, ela pode ser populada com o -I ou -A. Para uma regra passar por tal cadeia, basta que em uma das cadeias tradicionais seja usado o “-j nome_de_cadeia”. Se, ao passar por uma cadeia definida pelo usuário, nenhuma ação for realizada e o processamento chegar ao fim da cadeia de regras, o processamento continuará na regra seguinte àquela que fez a chamada para a cadeia do usuário. Exatamente: igual a uma chamada de sub-rotina! Há, inclusive, uma ação ainda não apresentada: “-j RETURN”. Esta ação faz parar a verificação na cadeia definida pelo usuário e voltar para a regra seguinte àquela que chamou. Como não estamos em um curso de “linguagem de programação iptables”, não vamos nos estender no assunto.

-L [cadeia] [-n] [-v]

Lista as regras. Se a cadeia for especificada, lista somente as regras desta. Normalmente, ao listar as regras, cada porta tcp/udp é traduzida pelo seu nome equivalente no arquivo /etc/services e os endereços IPs são enviados para uma pesquisa de DNS reverso, a fim de determinar o nome das máquinas e listá-los. O -n previne esta operação e lista tudo em modo numérico. O -v (*verbose*) lista mais colunas de informação por regra.

-P cadeia política

Indica a política (postura padrão) para a cadeia especificada. Assim, mesmo que a última regra que define a postura padrão não tenha sido escrita, ela estará especificada. Se isto não for definido, a postura padrão é _____.

Treino rápido: no Linux, digite e veja o que será listado, completando a linha do parágrafo anterior:

```
iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

- target (alvo): é a ação a ser tomada;
- prot: protocolo;
- opt: opções;
- source: IP origem;
- destination: IP destino.

Digite agora:

```
iptables -L -nv
Chain INPUT (policy ACCEPT 2868 packets, 2272K bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain OUTPUT (policy ACCEPT 2605 packets, 302K bytes)
 pkts bytes target     prot opt in     out     source
destination
```

As novas colunas são:

- pkts e bytes: contam quantos pacotes e bytes já satisfizeram a regra;
- In e out: indicam a interface, se ela foi especificada com -i e/ou -o.

Tradução das regras – iptables

Voltando à Tabela 5 (p. 37), vamos traduzir as regras para o GNU/Linux com comandos iptables. Suponhamos que a interface externa é a ppp0:

```
iptables -o ppp0 -A FORWARD -s 192.168.5.43 -d 10.12.0.33 -j ACCEPT
iptables -i ppp0 -A FORWARD -s 10.12.0.33 -d 192.168.5.43 -j ACCEPT
iptables -o ppp0 -A FORWARD -s 192.168.5.43 -d 172.18.1.3 -j ACCEPT
iptables -A FORWARD -j LOG
iptables -A FORWARD -j DROP
```

Digite os comandos e, depois, liste com ”-L” e com “-L -nv”.

Texto 5 - Filtragem por serviço

O exemplo anterior, filtrando somente por IP origem e destino, tem sua função didática, mas, na vida real, é bem pouco utilizado, pois indica que entre as máquinas relacionadas, todas as portas estão abertas (passam) pelo Firewall. Então, vejamos um caso real, realizando a filtragem para um servidor de correio eletrônico que se encontra em uma rede DMZ.

Exemplo de filtragem – SMTP

Vamos supor uma arquitetura como mostrado na Figura 12. O servidor interno de correio eletrônico deve enviar (fluxo 1) e receber (fluxo 4) emails do servidor que está na DMZ. Este, por sua vez, deve enviar (fluxo 2) e receber (fluxo 3) emails da Internet. O fluxo de dados entre as demais estações e o servidor interno não está representado, pois não passa pelo Firewall e, por isso, não nos interessa.

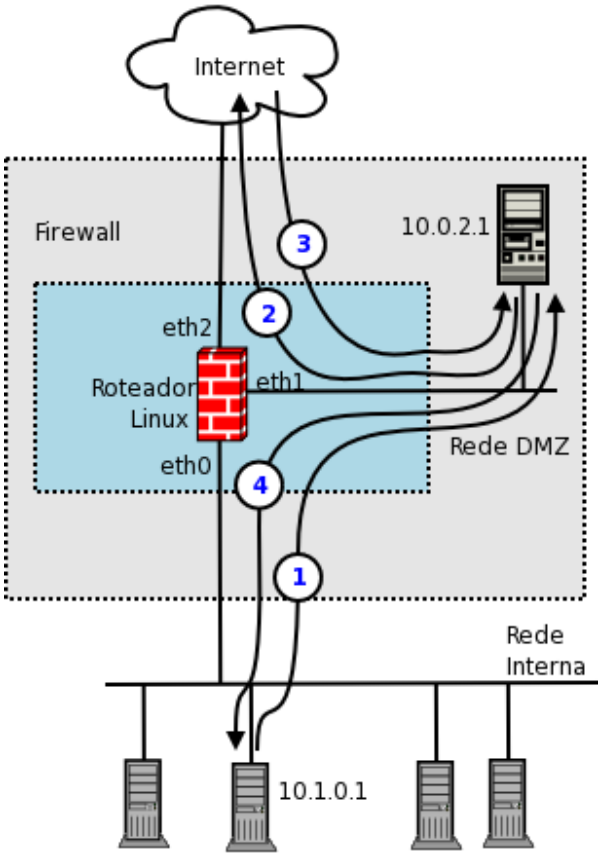

Figura 12 - Fluxo de serviços entre um servidor interno, um na DMZ, e a Internet.

Os endereços IP dos dois servidores envolvidos e os nomes das interfaces do roteador/Firewall estão especificados na figura.

Examinemos as características do serviço provido na DMZ e usado pelo servidor da rede interna (saída – fluxo 1):

1. Para pacotes saindo do servidor interno para o servidor na DMZ (fluxo de pacotes 1):
   - IP-origem é 10.1.0.1;
   - IP-destino é 10.0.2.1;
   - protocolo é TCP;
   - porta-origem é Y, um valor arbitrário acima de 1023;

- porta-destino é 25;
- apenas o primeiro pacote não terá o bit ACK ligado.

2. Para pacotes saindo do servidor na DMZ para o servidor interno (fluxo de pacotes 4)
   - IP-origem é 10.0.2.1 ;
   - IP-destino é 10.1.0.1;
   - protocolo é TCP;
   - porta-origem é 23;
   - porta-destino é Y;
   - todos os pacotes têm o bit ACK ligado.

Agora, vejamos as características do serviço provido na Internet (externo) e usado pelo servidor da DMZ (saída – fluxo de serviço 2)

3. Para pacotes saindo do servidor na DMZ para um servidor na Internet (fluxo de pacotes 2):
   - IP-origem é 10.0.2.1;
   - IP-destino é 0.0.0.0/0 (qualquer – o /0 indica netmask 0.0.0.0);
   - protocolo é TCP;
   - porta-origem é Y, um valor arbitrário acima de 1023;
   - porta-destino é 25;
   - apenas o primeiro pacote não terá o bit ACK ligado.
4. Para pacotes saindo do servidor na Internet para o servidor na DMZ (fluxo de pacotes 3)
   - IP-origem é 0.0.0.0/0;
   - IP-destino é 10.0.2.1;
   - protocolo é TCP;
   - porta-origem é 25;

- porta-destino é Y;
- todos os pacotes têm o bit ACK ligado.

Exercício

Como cada fluxo de serviço possui dois fluxos de dados, ainda faltam os fluxos de dados de números 5 a 8. Liste as características de cada fluxo de dados que está faltando.

Tabela SMTP

A Tabela 6 mostra as regras em nossa linguagem genérica para que o serviço SMTP funcione na arquitetura da Figura 12. Inserimos uma nova coluna "Int." para especificar a interface onde será aplicada a regra. Com isto, definimos por qual interface um determinado pacote vai entrar ou sair, cercando melhor as possibilidades diferentes.

#	Dir	Int.	IP Origem	IP Destino	Prot.	P.O.	P.D.	ACK	Ação
1	S	eth1	10.1.0.1	10.0.2.1	TCP	*	25	*	passa
2	E	eth1	10.0.2.1	10.1.0.1	TCP	25	*	S	passa
3	S	eth2	10.0.2.1	*	TCP	*	25		passa
4	E	eth2	*	10.0.2.1	TCP	25	*	S	passa
5									passa
6									passa
7									passa
8									passa
9	*	*	*	*	*	*	*	*	Bloq.

Tabela 6 - Regras para o serviço SMTP.

Exercício

Complete o restante da Tabela 6 (regras 5 a 8).

ACL para o exemplo SMTP

Agora será necessário especificar o protocolo (TCP) e a porta do serviço SMTP (25). Em uma regra de ACL, isto é feito colocando “eq 25” logo após o IP . A posição na linha é importante, pois, se colocado logo depois do:

- IP origem, será analisado como porta origem;
- IP destino, será analisado como porta destino.

Novidade: para se especificar o IP de uma única máquina, usa-se a palavra “*host*” antes do IP e não é necessário escrever o *hostmask*.

Em nosso roteador, as interfaces interna e com a DMZ se chamam, respectivamente, “ethernet 0” e “ethernet 1”, e a interface com a Internet é a “serial 1”. Como ACLs são stateless, teremos que criar duas listas de acesso para cada fluxo de serviço. Chamaremos de 101 a 108. No exemplo, usaremos as facilidades de “edição” de ACL:

```
ip access-list extended 101
10 permit tcp host 10.1.0.1 host 10.0.2.1 eq 25
20 deny any any log
exit
ip access-list extended 102
10 permit tcp host 10.0.2.1 eq 25 host 10.1.0.1 match-any +ack +rst
20 deny any any log
exit
ip access-list extended 103
10 permit tcp host 10.0.2.1 any eq 25
20 deny any any log
exit
ip access-list extended 104
10 permit tcp any eq 25 host 10.0.2.1 match-any +ack +rst
20 deny any any log
exit
interface ethernet 1
access group 101 out
access group 102 in
exit
interface serial 1
access group 103 out
access group 104 in
```

Lembre-se: a primeira linha da seqüência de comandos faz com que o roteador passe a colocar as próximas regras digitadas na ACL especificada no comando, até que se digite *exit.* Então, não é necessário repetir esta parte inicial do comando a cada regra.

A seqüência *match-any +ack +rst* colocada na primeira regra da ACL 102 indica que alguns flags do cabeçalho TCP devem ser verificados e estão em 1 (ligado) para tornar a regra satisfeita. No caso, o ACK, que somente fica ligado em conexões TCP já estabelecidas, e o RST (reset) que, às vezes, é usado para terminar uma conexão, em caso de erro. O *match-any* indica que, pelo menos, um dos dois tem que estar ligado para satisfazer o teste.

Para quem pretende se especializar: há muito mais sobre ACL na documentação da Cisco. Veja um exemplo da documentação:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_4/gtaclflg.htm

Exercício

Faça o grupo de comandos para criar e aplicar as ACLs 103 e 104, que faltam.

Dicas: na nossa tabela, o "*" em endereço IP pode ser substituído pela palavra "any", em uma ACL.

Netfilter para o exemplo SMTP

Para criar a seqüência de comandos iptables, iremos utilizar nosso conhecimento sobre o fato do *Netfilter* ser *stateful*. Veja como isto irá minimizar o número de comandos.

Vamos usar a facilidade de script shell, ou seja, colocar uma parte do comando em uma variável, para que os comandos posteriormente escritos fiquem menores (e caibam em uma só linha no livro!). Completar as duas últimas linhas é sua tarefa:

```
IPT=iptables -A FORWARD
${IPT} -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
${IPT} -i eth0 -o eth1 -p TCP -s 10.1.0.1 -d 10.0.2.1 --dport 25 -j
ACCEPT
${IPT} -i eth1 -o eth2 -p TCP -s 10.0.2.1 --dport 25 -j ACCEPT
${IPT}
${IPT}
${IPT} -i eth0 -j LOG
${IPT} -j DROP
```

A primeira linha coloca na variável TCP o pedaço do comando depois do "=". O shell substitui ${IPT} por "iptables -A FORWARD", na hora de interpretar as próximas linhas.

As regras no GNU/Linux não são aplicadas a uma interface. Em cada regra, é permitido especificar por qual interface se espera que os pacotes entrem e saiam.

A segunda linha garante que os pacotes de retorno sempre passarão, para qualquer conexão estabelecida.

Especificar a interface de entrada aumenta a segurança, à medida em que previne a falsificação do IP origem.

Na quarta linha, não foi colocado um IP destino, pois pode-se mandar email para qualquer lugar na Internet. Não especificar o destino é equivalente a dizer qualquer destino. Essa observação também vale para especificar a recepção de emails da Internet.

A linha com o LOG só irá registrar os pacotes que forem bloqueados e que tiverem entrado pela interface eth0, vindos da rede interna. O número de ataques e de pacotes bloqueados em sistemas conectados à Internet é tão grande, que não mais se justifica ficar registrando as tentativas já bloqueadas pelo firewall. No entanto, registrar tudo que é bloqueado pode ser válido para pesquisar por que um determinado serviço não funciona, ou seja, verificar se está sendo bloqueado pelo Firewall. Também pode ser válido para fazer longos relatórios e mostrar serviço.

Registrar as tentativas internas de sair para a Internet sem permissão irá permitir ao administrador corrigir uma configuração na estação que está tentando, ou chamar a atenção de algum usuário, que está fazendo o que não

devia.

Exercício

Use o manual do iptables e procure por *multiport*. Com isto, crie regras para que todas as estações internas possam usar HTTP, HTTPS e FTP na Internet.

Texto 6 - Filtros clássicos obrigatórios

Em consequência de existirem endereços IP que não podem ser usados na Internet, toda rede ligada à Internet deve ser protegida por filtros que já se tornaram um padrão.

Esses endereços são reservados para uso em ambientes não públicos, ou seja, fora da Internet – existe uma RFC que lista estes endereços. Assim, a filtragem deve ser baseada no fato de que nenhum pacote vindo da Internet (ou indo) pode ter um desses IPs como IP origem.

A RFC-3330 lista os endereços IPv4 de uso especial. São eles:

- 0.0.0.0/8 – "This" Network;
- 10.0.0.0/8 – Private-Use Networks [RFC-1918];
- 14.0.0.0/8 – Public-Data Networks [RFC1700];
- 24.0.0.0/8 – Cable Television Networks;
- 39.0.0.0/8 – Reserved but subject to allocation [RFC1797];
- 127.0.0.0/8 – Loopback [RFC1700, page 5];
- 128.0.0.0/16 – Reserved but subject to allocation;
- 169.254.0.0/16 – Link Local;
- 172.16.0.0/12 – Private-Use Networks [RFC1918];
- 191.255.0.0/16 – Reserved but subject to allocation;

- 192.0.0.0/24 – Reserved but subject to allocation;
- 192.0.2.0/24 – Test-Net;
- 192.88.99.0/24 – 6to4 Relay Anycast [RFC3068];
- 192.168.0.0/16 – Private-Use Networks [RFC1918];
- 198.18.0.0/15 – Network Interconnect Device Benchmark Testing [RFC2544];
- 223.255.255.0/24 – Reserved but subject to allocation;
- 224.0.0.0/4 – Multicast [RFC3171];
- 240.0.0.0/4 – Reserved for Future Use [RFC1700, page 4].

Além desses, também devemos incluir uma ou mais regras impedindo que pacotes com endereço IP pertencentes às redes internas apareçam como IP origem de pacotes vindos de fora.

Exercício

Crie as regras para a serem aplicadas na eth2, com os filtros obrigatórios. Elas devem estar logo após a regra que estabelece a passagem tendo como base no *stateful*, ou seja, a partir da segunda regra.

Texto 7 - Ligação de Intranet na Internet

Quando um projetista de rede escolhe os endereços IP que serão usados em uma nova rede, sempre usa endereços IP privados, pertencentes à RFC-1918, salvo raras exceções: no passado, as primeiras empresas a entrarem na Internet ganharam numeração IP suficiente para colocar em cada máquina interna. Hoje, ao alugar um enlace

Redes de uso privativo (RFC-1918):

10.0.0.0/8

172.16.0.0/24

192.168.0.0/16

de dados com acesso IP de uma operadora, esta lhe fornecerá um pequeno número de endereços IP. Às vezes, somente 1 (um)!

Toda rede que usa tecnologia da Internet, ou seja, o sistema de endereçamento (IP), os protocolos (TCP/UDP), as aplicações (SMTP, WWW, FTP etc.) em um ambiente privativo (endereços da RFC-1918) e controlado, é chamada de **Intranet**. Atualmente, poucos sistemas usam tecnologia diferente.

Então, se esta rede tiver que ser ligada à Internet, será necessário impedir que os pacotes saiam com IP origem pertencente à RFC-1918. Por isso, foi criado o NAT – *Network Address Translation*. O NAT é uma função que foi incorporada a roteadores e ao Firewall, para traduzir endereços, seja na saída de pacotes para a Internet, seja na chegada de pacotes da Internet para máquinas com IP privado.

Se a função não for incorporada, nada funcionará! Isto acontece porque nenhum roteador na Internet tem rota para estes IPs privados. E nem poderiam existir rotas para eles na Internet, pois um IP privado pode ser (e é) usado repetidamente em várias redes internas do mundo todo.

NAT

O NAT normalmente é colocado na interface externa do Firewall. Quando um pacote vai atravessar esta interface, o sistema de NAT faz as trocas necessárias.

Pode-se dividir o NAT em:

- NAT de saída para a Internet – troca o IP origem por um IP não privado;
- NAT de entrada – troca o IP destino por um IP privado.

Os sistemas de NAT possuem uma tabela interna para desfazer as trocas, quando o pacote de resposta voltar.

O NAT é muito utilizado na saída para a Internet. Mas esta não é sua única aplicação. Imagine duas empresas precisando interligar seus sistemas de dados para agilizar transações comerciais. E se ambas estiverem usando a

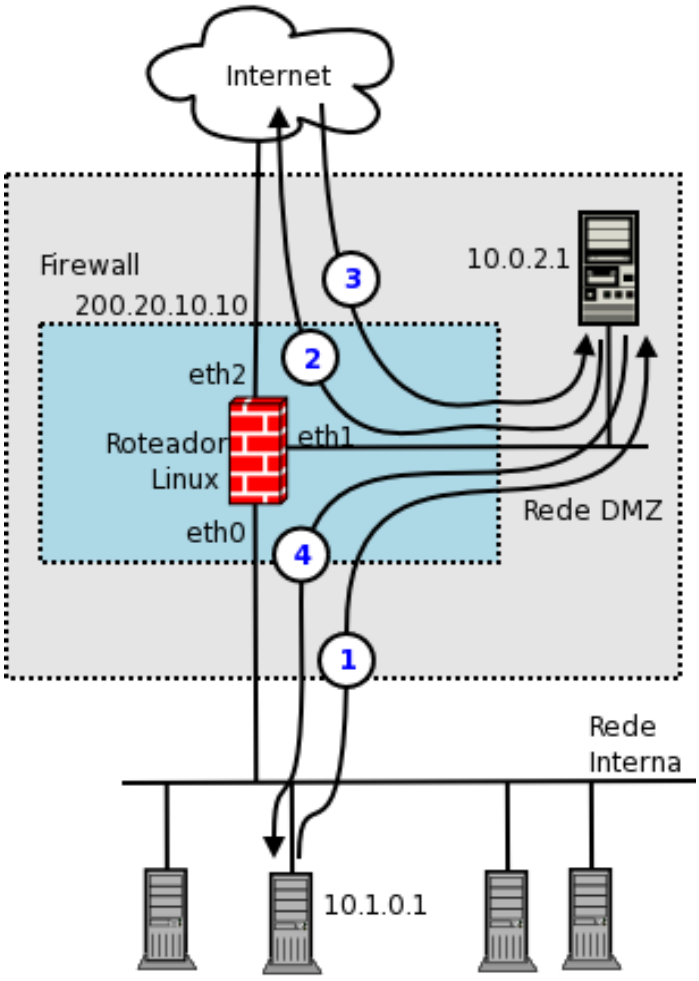

Figura 13 - Firewall com NAT, ligado à Internet.

mesma rede privada de endereçamento? O NAT irá resolver a questão.

O GNU/Linux possui em seu Netfilter todas as facilidades para NAT.

Vejamos a Figura 13. Todas as máquinas possuem endereços IP privados. Porém, o Firewall possui o IP 200.20.10.10 (não privado) na interface externa. Uma forma de fazer com que todos os pacotes que saem da rede para a Internet usem este IP é rodar a seguinte seqüência de comandos:

```
iptables -t nat -A POSTROUTING -o eth2 -j SNAT --to-source 200.20.10.10
```

A linha mostra algumas novidades: “-t nat” indica que será utilizada uma nova tabela de regras, a tabela de NAT e não de filtro. A tabela de filtro é especificada por “-t filter”. Mas como é o padrão, seu uso é implícito e não precisa ser especificado. Por isso, “-t filter” não foi usado nos exemplos

anteriores.

A nova tabela *nat* possui também novas cadeias, visualizadas na Figura 14, em conjunto com as cadeias do já apresentado *filter*.

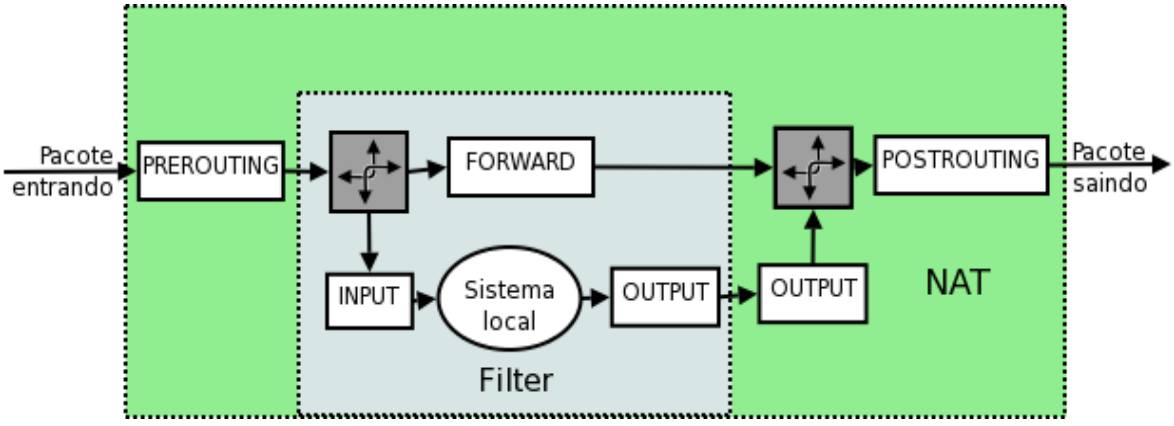

Figura 14 - Arquitetura do Netfilter: a passagem dos pacotes pelas tabelas nat e filter.

Há três novas novas cadeias de regras na tabela *nat*: *prerouting, postrouting* e *output* (sim, mesmo nome da cadeia de saída usada na tabela *filter*). Um pacote que está entrando na máquina passa primeiro pela cadeia *prerouting* do *nat*, para depois ser roteado ou para o sistema local, via cadeia *input*, ou para fora, via cadeia *forward*. Depois da etapa final de roteamento, o pacote passa pela cadeia de *postrouting*.

A essas duas cadeias estão associadas mais duas novas ações:

- DNAT (*Destination NAT*), na *prerouting,* para trocar o IP de destino;
- SNAT (*Source NAT*), na *postrouting,* para trocar o IP de origem.

Veja-as no Linux:

```
iptables -t nat -L
```

Uso obrigatório de NAT ou Masquerade

O comando para criar o NAT de saída em nosso sistema de exemplo é:

```
iptables -t nat -A POSTROUTING -o eth2 -j SNAT --to-source 200.20.10.10
```

Em sistemas que utilizam conexão com linha discada, ou algum outro método

de acesso dinâmico, normalmente os ditos “banda larga”, nem sempre existe um IP fixo – por exemplo: NET (e suas variações), Virtua, Velox, Speed etc. Como permitir que este tipo de conexão seja usado por várias máquinas em um escritório ou residência? No caso, não é necessário olhar o IP que está na interface, o que poderia ser feito com o comando ifconfig. O Netfilter tem a possibilidade de usar IP dinâmico, ou seja, mesmo que haja troca do IP da interface, o NAT continua a trocar o IP origem dos pacotes pelo IP da interface. No GNU/Linux, tal procedimento é chamado de *masquerade*.

Isto é conseguido com o comando:

```
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

E se a interface externa for a ppp0 (linha discada, Velox etc.)? Resposta:

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Em alguns sistemas, o procedimento de usar automaticamente o IP da interface tem o nome de *hide nat*.

O servidor de correio (gateway SMTP) da DMZ (Figura 13) possui um IP privado e não pode ser alcançado por servidores que estejam na Internet. Mas o IP da interface eth2 (200.20.10.10) pode! Então, no sistema de DNS, publicar-se-á o IP 200.20.10.10 como sendo o IP do servidor de correio. Entretando, este IP é do Firewall!!!

Sim! Com o uso do NAT é possível fazer os pacotes que chegarem da Internet para o IP externo serem redirecionados para o servidor na DMZ:

```
iptables -t nat -A PREROUTING -i eth2 -j DNAT --to-destination 10.0.2.1
```

O comando é para ser usado em conjunto com os comandos de filtragem já exemplificados anteriormente. Ele só faz a tradução, ou seja, em pacotes que entrarem pela interface eth2, troca o endereço destino por 10.0.2.1.

Assim, junto com as regras de filtragem, o Firewall irá bloquear qualquer coisa que não seja para a porta 25. Pode-se, ainda, incluir os parâmetros “-p tcp --dport 25” no comando para que o NAT só seja realizado se for um pacote destinado à porta 25. Mas, por enquanto, estamos tratando o assunto de modo mais genérico e aberto.

Pergunta: se o endereço IP destino do pacote que chega é 200.20.10.10, por que a regra de filtragem do Firewall usa o IP 10.0.2.1?

Resposta: note, na Figura 14, que a cadeia de regras onde é colocado o DNAT é a PREROUTING (pré-roteamento). Assim, o endereço de destino é trocado antes do pacote passar pelas regras de filtragem. Por isso, o IP destino visto já será o IP real da máquina destino.

> Todos os comandos de filtragem devem ser criados tendo-se em mente os IPs reais das máquinas envolvidas, independente do uso de NAT.

PAT

O termo PAT (*Port Address Translation*) serve para designar o mesmo conceito do NAT, só que aplicado às portas de serviços TCP/UDP. Pode ser usado em conjunto com o NAT ou de forma independente. Na verdade utiliza os mesmos métodos de configuração (comandos no Linux).

Por exemplo: suponha que o servidor da DMZ (Figura 13) rode também o serviço SSH[5] que, por padrão, atende na porta tcp/22. Suponha que o administrador decidiu que este servidor deve ser acessado da Internet, mas quer publicar o serviço como atendendo na porta 2222. Ele avisa aos usuários que é possível fazer um ssh no ip 200.20.10.10, na porta 2222, para acessar o terminal remoto do servidor da DMZ. Alguns comandos deverão ser incluídos no Firewall, tanto na tabela do *filter* quanto no *nat*, observando que este serviço usa um fluxo de serviço como o fluxo 3 da Figura 13:

```
iptables -t nat -I PREROUTING -i eth2 -p tcp --dport 2222 -j DNAT
     --to-destination 10.0.2.1:22
iptables -I FORWARD 6 -i eth2 -p tcp -d 10.0.2.1 --dport 22 -j ACCEPT
```

Note que foi usada a inserção no segundo comando (-I FORWARD 6), de modo que esta regra será a sexta na seqüência, portanto antes das regras que fazem o LOG e o DROP.

5 Esse serviço permite o uso de terminal remoto, estilo telnet, mas todos os dados são criptografados. Veremos o assunto adiante.

Unidade 3 - Ferramentas de monitoração e auditoria

Nesta unidade apresentaremos algumas ferramentas para verificação, monitoração e auditoria de sistemas de rede, que permitem melhorar a segurança, através do conhecimento de como está a rede e permitindo reações pró-ativas. Com a apresentação das ferramentas, pretendemos possibilitar um posterior estudo com sua utilização. Nosso objetivo é conhecer o que existe e o que é possível fazer com cada uma, conforme aqui apresentado.

Entretanto, note que não se trata de um curso sobre cada ferramenta, pois certamente é possível fazer muito mais do que mostraremos. Mas, este mínimo é o suficiente para introduzir o assunto e permitir avanços em futuras pesquisas. Você deve concentrar-se em conhecer o que é possível fazer e na utilidade das ferramentas, e não como usá-las, bem como a todos os seus recursos etc. Isto virá com o tempo.

Algumas destas ferramentas já estão instaladas no CD Live utilizado no curso. Embora não seja requisito, outras poderão ser baixadas e instaladas.

Texto 1 - Ping

Todo mundo que trabalha com rede, ou próximo a alguém que trabalha, já ouviu ou pronunciou uma frase durante uma fase de testes para verificar se um computador está corretamente ligado na rede:

“–Verifica se está pingando...”

O ping é um aplicativo, um programa, que gera pacotes ICMP com a mensagem de requisição de eco (*Echo Request*). Os sistemas que possuem a

pilha de protocolos TCP/IP, se habilitados em sua configuração, respondem automaticamente a um ICMP *Echo Request* com um pacote ICMP *Echo Response*. Esse pacote de resposta é também chamado de “pong” (de ping-pong), mas o termo não ficou famoso, como o “ping” ou o “pingando”.

O fato de se conseguir resposta a um ping indica que uma máquina está na rede. Mas, não conseguir resposta, não indica muita coisa, pois diversos fatores podem ser culpados pela não resposta a um *Echo Request*. Assim, na ausência de resposta, o administrador deve investigar os motivos. Mesmo o óbvio motivo da não resposta (a máquina realmente não está em rede) pode ter várias explicações: máquina desligada; sem cabo de rede; com cabo de rede ligado na tomada, mas a tomada desligada na sala ou armário de hub/switches; defeito no cabo de rede; defeito na placa de rede; placa de rede sem configuração de IP etc.

Além desses fatores imediatos, pode haver um Firewall entre a máquina de teste e a testada, bloqueando pacotes ICMP *Echo Request*. Alguns administradores menos informados chegam a bloquear todo e qualquer pacote ICMP, trazendo lentidão e transtornos aos usuários da rede pois, sem pacotes de controle (ICMP), o TCP/IP é capaz de perder desempenho e até deixar de funcionar.

Dependendo da versão do programa ping, são fornecidas algumas interessantes informações estatísticas sobre o caminho entre a origem e o destino, tais como: tempo de resposta de cada pacote e, findo o teste, o número de pacotes enviados e respondidos, os tempos das respostas: mínimo, médio, máximo e o desvio em torno da média. Vejamos um exemplo:

```
ping 200.20.10.18
PING 200.20.10.18 (200.20.10.18) 56(84) bytes of data.
64 bytes from 200.20.10.18: icmp_seq=1 ttl=54 time=7.03 ms
64 bytes from 200.20.10.18: icmp_seq=2 ttl=54 time=8.54 ms
64 bytes from 200.20.10.18: icmp_seq=3 ttl=54 time=7.07 ms
64 bytes from 200.20.10.18: icmp_seq=4 ttl=54 time=7.90 ms
64 bytes from 200.20.10.18: icmp_seq=5 ttl=54 time=6.70 ms
64 bytes from 200.20.10.18: icmp_seq=6 ttl=54 time=6.59 ms

--- 200.20.10.18 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5052ms
rtt min/avg/max/mdev = 6.598/7.310/8.541/0.691 ms
```

Se sua máquina não está em rede, teste com o ip 127.0.0.1.

Para sair do ping, pressione Control C.

No Windows, o ping, por padrão, realiza apenas alguns envios e pára sozinho. Para manter o teste indefinidamente, usa-se “ping -t”

Texto 2 - Traceroute

Relembrando o cabeçalho IP, o campo com o tempo de vida de um pacote (TTL - *Time To Live*) indica, na verdade, por quantos roteadores este pacote ainda pode passar, antes que seja descartado. Ao transmitir um pacote IP, uma estação inicia este campo com um número inteiro. O valor inicial depende do sistema operacional. No GNU/Linux, costuma ser 64. Em cada roteador, este valor é decrementado de um, e testado se chegou a zero. Se chegar em zero, o roteador não mais roteia este pacote, que é considerado “morto”. Mas, normalmente, o roteador “devolve o corpo”, ou seja, o pacote “falecido” é colocado em um pacote ICMP *Time Exceeded in-transit* e enviado para o IP originário.

O traceroute usa este mecanismo “obituário” para descobrir a seqüência de endereços IP dos roteadores entre a máquina origem em que roda o traceroute e o destino. Por padrão, o traceroute do GNU/Linux cria pacotes UDP, destinados a uma porta qualquer entre 33434 e 33524, em séries de três pacotes, sendo a primeira série com TTL = 1, a segunda com TTL = 2, a terceira com TTL = 3 e assim por diante.

Para a primeira série, o TTL dos pacotes chegará a zero logo no primeiro roteador, no caminho para o destino. Este roteador envia os pacotes ICMP e o programa traceroute imprime uma linha mostrando quem é a máquina (IP ou nome do roteador) e os tempos entre o pacote UDP sair da máquina origem e o retorno do ICMP.

Já na segunda série, com TTL = 2, será no segundo roteador do caminho que irá enviar os pacotes ICMP. O traceroute imprime na tela uma segunda linha, com a denominação do roteador e os tempos.

Isto ocorre até que o IP origem dos pacotes ICMP seja o IP para onde se disparou o traceroute. Assim, o programa fica sabendo que chegou a seu destino e que pode parar de enviar séries de pacotes.

Para esta versão de traceroute funcionar, é necessário que não haja filtragem para UDP/ 33434 até UDP/33524, nem dos ICMP de retorno.

O comando equivalente no Windows se chama tracert. A diferença é que o tracert usa pacotes ICMP *Echo Request*, como o ping, para criar as séries de pacotes de envio.

Então, em uma determinada instalação, dependendo da configuração dos roteadores e/ou Firewalls do meio do caminho, pode ser que se consiga fazer um trace de um GNU/Linux e não de um Windows, ou vice-versa.

Analise um exemplo com a utilização do parâmetro -n, que mostra tudo numérico. Sem o -n, o traceroute tenta descobrir o nome ao resolver o IP por DNS reverso:

```
traceroute -n 200.20.10.18
traceroute to 200.20.10.18 (200.20.10.18), 30 hops max, 38 byte packets
 1  10.10.0.1  0.868 ms  0.489 ms  0.464 ms
 2  10.102.0.1  27.111 ms *  68.348 ms
 3  10.100.0.2  33.004 ms *  24.327 ms
 4  200.239.245.118  78.158 ms  53.909 ms  40.876 ms
 5  200.239.245.55  27.378 ms  47.129 ms  149.293 ms
 6  200.239.245.62  155.454 ms  62.203 ms  35.558 ms
 7  200.239.194.100  35.521 ms  79.060 ms *
 8  200.239.245.254  24.819 ms  99.092 ms  58.314 ms
 9  200.239.192.253  47.479 ms  23.004 ms  199.598 ms
10  200.159.255.1  50.028 ms  43.246 ms  67.813 ms
11  200.143.254.126  45.296 ms  39.192 ms  40.461 ms
12  * 200.143.254.137  43.200 ms  161.689 ms
13  200.20.94.56  77.998 ms  70.727 ms  107.738 ms
14  200.20.92.146  104.066 ms  77.720 ms  103.059 ms
15  200.20.10.18  100.545 ms  33.509 ms  27.598 ms
```

Como você deve ter percebido, em algumas linhas apareceu um asterisco (*).

Significa que, como não houve resposta em tempo hábil (por padrão, cinco segundos), o traceroute considerou o pacote perdido e colocou um asterisco.

Se todas as respostas começarem a ser o asterisco, quer dizer que o pacote de teste não está passando mais. Normalmente, isto acontece quando um Firewall no meio do caminho está filtrando (bloqueando) os pacotes de traceroute. Outro motivo é a máquina destino não existir. Exemplo com uma máquina inexistente:

```
traceroute -n 200.20.10.11
traceroute to 200.20.10.11 (200.20.10.11), 30 hops max, 38 byte packets
 1  10.10.0.1  0.832 ms  0.490 ms  0.552 ms
 2  10.102.0.1  40.582 ms  20.659 ms  110.288 ms
 3  10.100.0.2  23.770 ms  210.963 ms  50.236 ms
 4  200.239.245.118  88.781 ms  19.888 ms  194.283 ms
 5  200.239.245.55  21.531 ms  48.698 ms  15.545 ms
 6  200.239.245.62  78.391 ms  247.030 ms  53.948 ms
 7  200.239.194.100  77.653 ms  59.800 ms  61.342 ms
 8  200.239.245.254  42.898 ms *  114.158 ms
 9  200.239.192.253  75.061 ms  35.602 ms  48.001 ms
10  200.159.255.1  108.985 ms  131.758 ms  83.022 ms
11  200.143.254.126  59.591 ms  42.805 ms  52.238 ms
12  200.143.254.137  44.631 ms  37.210 ms  58.489 ms
13  * 200.20.94.56  182.843 ms  35.154 ms
14  200.20.92.146  58.885 ms  113.009 ms  99.803 ms
15  200.20.0.27  44.148 ms  69.488 ms  58.062 ms
16  * * *
17  * * *
18  * * *
```

O traceroute continua, por padrão, até completar trinta linhas. Porém, pode ser encerrado antes com um Control C no teclado.

Texto 3 - Tcpdump

Esta ferramenta possibilita capturar e mostrar todos os pacotes que chegam ou aparecem em uma determinada interface da máquina em que é rodado.

Se for rodado sem nenhum parâmetro, o comando irá usar a interface eth0, que é a primeira interface de rede no GNU/Linux, para fazer a captura.

Por padrão, o tcpdump mostra a hora em que o pacote chegou, com precisão de microssegundo, o endereço IP origem, o endereço IP destino, uma descrição do pacote e informações retiradas do pacote capturado.

Por padrão, o tcpdump tenta resolver o nome por DNS reverso. Então, é mais eficiente usar o parâmetro -n, pedindo que mostre no modo numérico. Por coincidência, o parâmetro é o mesmo do traceroute.

Exemplo: em uma linha de terminal, digite os comandos ping 127.0.0.1. Depois abra outro terminal e digite o comando:

```
tcpdump -ni lo
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
15:23:13.703348 IP 127.0.0.1 > 127.0.0.1: ICMP echo request, id 33044,
seq 1, length 64
15:23:13.703379 IP 127.0.0.1 > 127.0.0.1: ICMP echo reply, id 33044, seq
1, length 64
15:23:14.714358 IP 127.0.0.1 > 127.0.0.1: ICMP echo request, id 33044,
seq 2, length 64
15:23:14.714384 IP 127.0.0.1 > 127.0.0.1: ICMP echo reply, id 33044, seq
2, length 64
15:23:15.713876 IP 127.0.0.1 > 127.0.0.1: ICMP echo request, id 33044,
seq 3, length 64
15:23:15.713903 IP 127.0.0.1 > 127.0.0.1: ICMP echo reply, id 33044, seq
3, length 64
```

Depois de transcorrido algum tempo, pressione Control C. Isto irá parar o tcpdump. Faça o mesmo no console com o ping.

Vejamos um exemplo com um “dump” de um traceroute.

Em um console deve ser disparado um tcpdump e em outro, o traceroute.

O comando traceroute foi:

```
traceroute -n 200.239.245.55
traceroute to 200.239.245.55 (200.239.245.55), 30 hops max, 38 byte
packets
 1  10.10.0.1  1.264 ms  0.788 ms  0.769 ms
 2  10.102.0.1  88.278 ms  125.116 ms  115.582 ms
 3  10.100.0.2  50.844 ms  84.851 ms  55.236 ms
 4  200.239.245.118  117.250 ms  123.197 ms  61.403 ms
 5  200.239.245.55  103.008 ms  99.445 ms  37.088 ms
```

E o respectivo dump foi:

```
tcpdump -np
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
19:15:16.433166 IP 10.10.0.30.36815 > 200.239.245.55.33435: UDP, length
10
19:15:16.433349 IP 10.10.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:16.434166 IP 10.10.0.30.36815 > 200.239.245.55.33436: UDP, length
10
19:15:16.434611 IP 10.10.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:16.435105 IP 10.10.0.30.36815 > 200.239.245.55.33437: UDP, length
10
19:15:16.435540 IP 10.10.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:16.436225 IP 10.10.0.30.36815 > 200.239.245.55.33438: UDP, length
10
19:15:16.524417 IP 10.102.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.524552 IP 10.10.0.30.36815 > 200.239.245.55.33439: UDP, length
10
19:15:16.649532 IP 10.102.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.649700 IP 10.10.0.30.36815 > 200.239.245.55.33440: UDP, length
10
19:15:16.765151 IP 10.102.0.1 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.765318 IP 10.10.0.30.36815 > 200.239.245.55.33441: UDP, length
10
```

```
19:15:16.815993 IP 10.100.0.2 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.816199 IP 10.10.0.30.36815 > 200.239.245.55.33442: UDP, length
10
19:15:16.900919 IP 10.100.0.2 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.901076 IP 10.10.0.30.36815 > 200.239.245.55.33443: UDP, length
10
19:15:16.956183 IP 10.100.0.2 > 10.10.0.30: ICMP time exceeded in-
transit, length 36
19:15:16.956347 IP 10.10.0.30.36815 > 200.239.245.55.33444: UDP, length
10
19:15:17.073452 IP 200.239.245.118 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:17.073638 IP 10.10.0.30.36815 > 200.239.245.55.33445: UDP, length
10
19:15:17.196698 IP 200.239.245.118 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:17.196868 IP 10.10.0.30.36815 > 200.239.245.55.33446: UDP, length
10
19:15:17.258138 IP 200.239.245.118 > 10.10.0.30: ICMP time exceeded in-
transit, length 46
19:15:17.258305 IP 10.10.0.30.36815 > 200.239.245.55.33447: UDP, length
10
19:15:17.361174 IP 200.239.245.55 > 10.10.0.30: ICMP 200.239.245.55 udp
port 33447 unreachable, length 46
19:15:17.361350 IP 10.10.0.30.36815 > 200.239.245.55.33448: UDP, length
10
19:15:17.460656 IP 200.239.245.55 > 10.10.0.30: ICMP 200.239.245.55 udp
port 33448 unreachable, length 46
19:15:17.460826 IP 10.10.0.30.36815 > 200.239.245.55.33449: UDP, length
10
19:15:17.497781 IP 200.239.245.55 > 10.10.0.30: ICMP 200.239.245.55 udp
port 33449 unreachable, length 46
```

Observações interessantes:

- a porta destino não é fixa, mas a porta origem é;
- a resposta ICMP para o último pacote que chega ao destino informa que a porta destino não está alcançável, ou seja, não existe nenhum serviço atendendo nesta porta.

Texto 4 - NMAP

Esta ferramenta procura por serviços disponíveis nas máquinas contra as quais é disparado. Em versões mais recentes, ganhou a funcionalidade de descobrir a versão do software que presta alguns serviços e o tipo e versão do sistema operacional que roda na máquina alvo.

Por padrão, o nmap procura apenas pelas portas mais conhecidas, de 1 a 1024 e as relacionadas no arquivo /usr/share/nmap/nmap-services.

A quantidade de parâmetros existente é considerável e são apresentados no manual do nmap. No entanto, para uma utilização simples e rápida, pode-se usar o exemplo:

```
nmap -A localhost

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-06-02 16:19
GMT
Interesting ports on localhost (127.0.0.1):
(The 1661 ports scanned but not shown below are in state: closed)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.1 (protocol 1.99)
53/tcp open  domain
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.5.25 - 2.6.3 or Gentoo 1.2 Linux 2.4.19 rc1-rc7),
Linux 2.6.3 - 2.6.8
Uptime 0.003 days (since Fri Jun  2 16:14:37 2006)

Nmap finished: 1 IP address (1 host up) scanned in 12.435 seconds
```

No caso, estavam ligados o seviço de nomes de domínios (domain - DNS) e o servidor de SSH. A opção -A liga várias opções que permitem ao nmap descobrir a versão do sistema operacional e a versão dos serviços.

Faça um tcpdump de uma sessão de nmap, de forma similar ao feito com o ping e o traceroute. Comente no fórum.

Texto 5 - Ethereal

O ethereal possui a mesma finalidade do tcpdump: capturar e analisar pacotes. Mas é uma ferramenta muito mais poderosa, possuindo interface gráfica e permitindo uma real análise de protocolos, inclusive de aplicação. Ela pode ser obtida no site http://www.ethereal.com/.

Por possuir versão para o Windows, constitui uma ferramenta bastante popular entre técnicos de operação de rede e de desenvolvimento de protocolos e serviços/servidores. Possui recursos de análise e de estatística poderosos, gráficos, inclusive para os mais recentes protocolos, a exemplo dos utilizados em Voz sobre IP (VoIP).

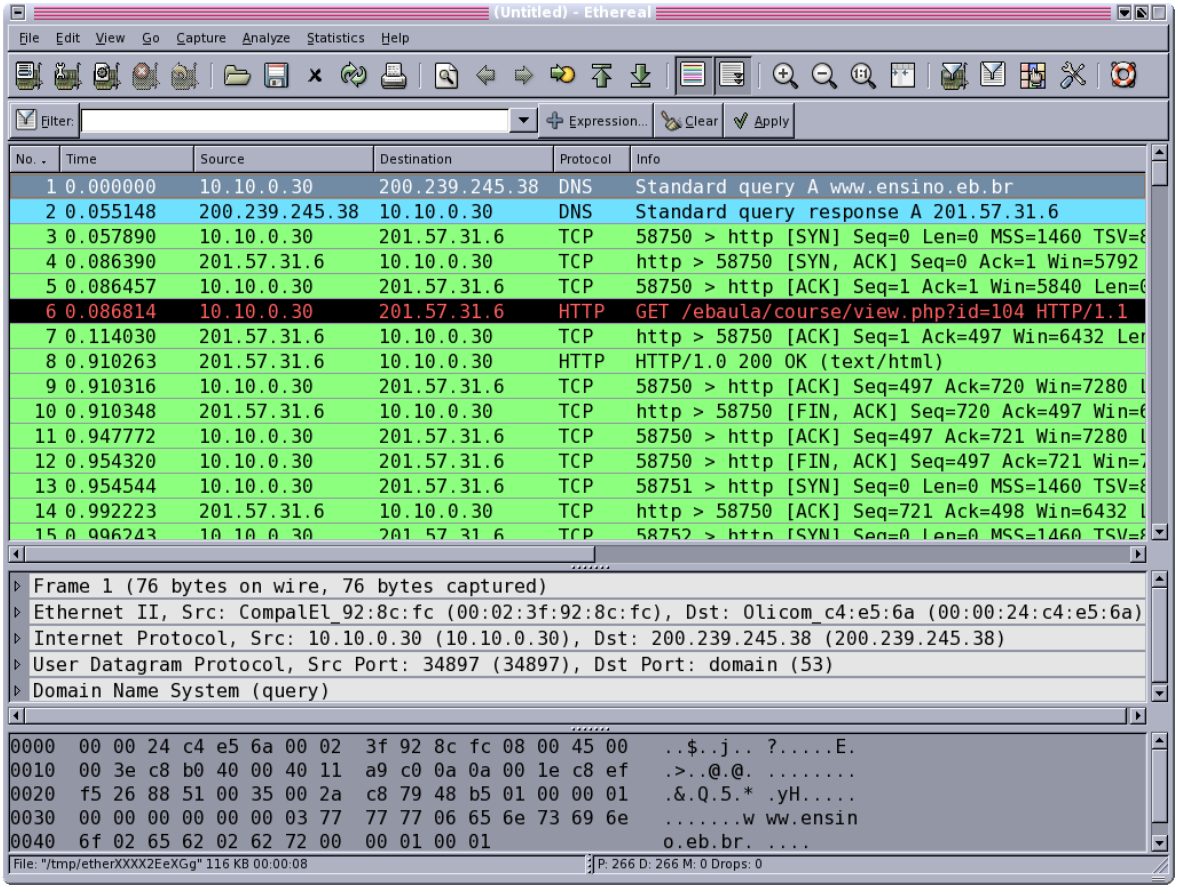

Figura 15 - Ethereal: um poderoso analisador de protocolos.

A Figura 15 mostra uma tela do Ethereal. O aplicativo possui uma área superior com menu, barra de botões e barra de manipulação de filtro de pacotes a serem mostrados.

As janelas de análise são três:

- a primeira, a maior na figura, mostra o cabeçalho e uma sucinta descrição de cada pacote;
- a segunda, quando clicamos com o mouse na janela anterior (no caso, o pacote número 1), mostra a dissecação do pacote que está em destaque;
- a janela inferior, mostra o pacote aberto em hexadecimal na janela da esquerda e sua possível tradução para caracteres ASCII, à direita.

A Figura 15 apresenta o início de uma sessão de HTTP. Os primeiros pacotes são a requisição DNS para a resolução do nome do servidor e, a partir do terceiro pacote (número 3 na primeira coluna a esquerda), é iniciada a conexão TCP na porta http (80), seguido de um pacote com [SYN, ACK] e outro com [ACK]. Estabelecida a conexão TCP, começam a trafegar os pacotes HTTP, com o GET.

Como o usuário não estava logado no EBaula e tentou acessar diretamente a página do curso, o servidor informou isto na resposta ao GET, no pacote de número 8 (não aparece nesta figura) e, em seguida, fechou a conexão no pacote 10, sendo finalizada pelo cliente no pacote 12.

Outra conexão começa a ser perdida no pacote de número 13. Não vamos ver a continuação.

Na janela de dissecação, pode-se observar cada um dos cabeçalhos existentes no pacote. Neste exemplo:

- Ethernet II: o nível de enlace;
- Internet Protocol: o IP, com os endereços origem e destino;
- UDP, com as portas origem e destino;
- a requisição (*query*) do DNS.

Ao lado esquerdo, existe uma seta triangular em cada linha. Se clicada, irá abrir a camada a que se refere, com todos os detalhes do cabeçalho, e informação sobre o que significa cada byte ou grupo de bytes.

Conhecer bem os protocolos é fundamental para o bom uso da ferramenta. No entanto, ela também pode ser usada para o aprendizado de protocolos.

Texto 6 - IDS - *Intrusion Detector System*

Um IDS (*Intrusion Detector System*) de rede tem como principal função analisar pacotes que já passaram pelo Firewall e não foram filtrados, ou seja, são pacotes permitidos pelo Firewall. A análise verifica se os pacotes que estão passando possuem dados que indicam a tentativa de explorar alguma vulnerabilidade conhecida.

Podemos definir IDS como uma ferramenta inteligente capaz de detectar tentativas de invasão em tempo real.

Tais sistemas atuam de várias formas:

- somente alertam as tentativas de invasão;
- além de alertar, agem de modo reativo, aplicando ações necessárias contra o ataque.

  Uma forma simples de coibir um ataque em andamento é enviar um pacote com um TCP Reset para o cliente e para o servidor de uma conexão em que se detectou um ataque. Outra forma de coibir um ataque é alterar regras do Firewall, bloqueando os pacotes atacantes.

Em resumo, o IDS constitui um sistema de configurações e regras que tem como objetivo gerar alertas ao detectar pacotes que possam fazer parte de um possível ataque.

Há diversos tipos de ferramentas de IDS para diferentes plataformas. Porém, elas trabalham basicamente de modo parecido, ou seja, analisando os pacotes que trafegam na rede e comparando-os com assinaturas já prontas de ataques, identificando de forma fácil e precisa qualquer tipo de anomalia ou ataque que possa vir a ocorrer na rede ou computador.

Uma ferramenta IDS serve para nos trazer informações sobre a rede, tais como:

- quantas tentativas de ataques sofremos por dia;
- que tipo de ataque foi usado;
- qual a origem dos ataques.

Cuidado com o uso ineficiente de um IDS pois, diretamente ligado na Internet, sem a filtragem de um Firewall, a quantidade de alertas será astronômica. Por exemplo, em um teste realizado em 1998 com um IDS na entrada da rede Internet de uma grande Universidade, o número de alertas por dia passou de onze mil. Hoje, certamente passaria de uma centena de milhar. O próprio nome já diz: detector de intrusos, ou seja, detector de algo que passou pelas proteções da entrada (Firewall) e já está dentro da rede protegida: um intruso. Colocar um IDS do lado de fora, recebendo diretamente os ataques da Internet, é tão eficiente e inteligente quanto colocar um alarme detector de vazamento de água dentro da caixa d'água.

> Nunca se usa uma ferramenta de IDS no ambiente externo, antes da filtragem do Firewall.

Embora possa parecer absurdo, uma vez entendido o real uso de um IDS, diversos vendedores ou “consultores” de segurança de rede defendem o uso incorreto, porque mostra que o equipamento funciona, ao gerar uma quantidade inútil de alarmes, a maioria em portas que serão barradas pelo Firewall e que, por isso, não apresentam nenhum perigo. Enquanto a maioria das empresas provedoras de acesso Internet não vigiarem também a saída, impedindo que usuários façam ataques, a Internet será um mar de tentativas, desde a procura por portas abertas (scan) até tentativas mais focadas.

Um ótimo exemplo de software livre que faz este trabalho é o Snort, desenvolvido por Martin Roesch. É bastante popular por sua flexibilidade nas configurações de regras e constante atualização frente às novas ferramentas de invasão. Outro ponto forte do Snort é o fato de ter um grande cadastro de assinaturas – considerado um dos maiores – leve e pequeno.

O Snort pode gravar os alertas em banco de dados SQL, o que facilita a vida de um administrador, principalmente para gerar relatórios. Há softwares que permitem automatizar a geração de relatórios e se comportam como um *front-end* para o Snort. Você encontra referências, no site do Snort.

Texto 7 - Detector de vulnerabilidades

Outra tarefa de um administrador de rede, preocupado com a segurança, é determinar se seus sistemas, tanto internos quanto externos, estão com alguma vulnerabilidade conhecida, principalmente de rede, ou seja, rodando um software obsoleto que presta um serviço para a rede interna ou externa. Vimos, na disciplina *Segurança em Redes 1*, que este é um dos grandes responsáveis pelo sucesso de um ataque.

Para os softwares conhecidos e documentados, uma solução é estar cadastrado em listas de divulgação (email) dos fabricantes dos softwares usados na empresa. Assim, a cada nova versão publicada, o administrador será avisado e poderá analisar os bugs concertados e as novas funções, a compatibilidade com a atual versão e fazer o update, se necessário.

Em servidores e máquinas de uso crítico, nunca se usam os chamados “updates automáticos”, devido às falhas ainda existentes nesses procedimentos, aliado à possível necessidade de reboot e outros fatores que podem comprometer a disponibilidade.

Mas, podemos ter mais alguns problemas: e se um servidor, ou um software, foi instalado na rede, em qualquer máquina, sem o conhecimento do

administrador? E tal software apresentar uma vulnerabilidade explorável pela rede? Ou então, se, por uma falha na documentação da empresa, ou por culpa do próprio administrador, um determinado serviço não estiver documentado e todos se “esquecerem” que ele existe? Quando isso ocorre, uma vulnerabilidade só será detectada por acaso, ou quando for explorada, trazendo, neste último caso, inevitáveis prejuízos.

Uma resposta rápida seria usar o nmap para procurar serviços funcionando em todas as máquinas da rede sob teste. Contudo, o nmap só indica que existe um serviço respondendo e, até, pode informar a versão do software. Ótimo para uma primeira análise e para fazer uma documentação do que está rodando na rede. Porém, será que a versão informada pelo nmap é a última disponível para aquele servidor? E se não for a última versão disponível, será que está vulnerável a algum ataque?

Reunir um banco de dados de informações com todas as vulnerabilidades conhecidas de praticamente todos os principais sistemas operacionais e servidores/serviços de rede, aliando este banco a um sistema de procura por portas abertas, é o trabalho desenvolvido pelos fabricantes de detectores de vulnerabilidades. Um detector deste tipo pode procurar pelas portas abertas e gerar um relatório completo sobre as vulnerabilidades encontradas e seu nível de severidade, bem como disponibilizar as ações que o administrador deve tomar para a segurança, imediatas e de médio prazo.

Também é possível fazer testes mais agressivos. Neste tipo de teste, um servidor vulnerável pode mesmo ter um serviço derrubado ou travar completamente (este último, muito comum em algumas vulnerabilidades do Windows). Por isso, testes completos só devem ser realizados em horários previamente agendados, fora do horário de uso do sistema.

Em se tratando de um sistema dito 24 x 7, que não pára nunca, abre-se uma janela de manutenção, avisando com antecedência pertinente aos usuários, que poderá haver uma indisponibilidade. Em sistemas mais críticos, que trabalham com sistemas backup flutuantes, o teste pode ser realizado no sistema backup. Mesmo assim, a janela de manutenção deve ser aberta com aviso aos usuários pois, se houver uma falha real no sistema principal exatamente no momento do teste do backup, o sistema todo deixará de

responder. E nunca diga ou pense: mas isso nunca acontece!

No mundo do software livre, um grande impulso neste sentido foi dado por Renaud Deraison, que criou e mantém um software chamado de Nessus (http://www.nessus.org). Esse software teve as idéias herdadas do “Satan” (*Security Administrator Tool for Analyzing Network*), que inspirou os criadores do “Saint” (*Security Administrator's Integrated Network Tool*) e, finalmente, do “Nessus”.

Há várias notícias na Internet sobre ele:

http://software.newsforge.com/software/05/02/02/1412203.shtml

Veja uma pequena tradução:

> O Nessus oferece a funcionalidade necessária para detectar as falhas difíceis de encontrar, específicas de aplicativos e sistemas operacionais. Quando combinado com outras ferramentas de código aberto como o Snort para detecção de intrusão e o NMap para inspeção de portas, o Nessus pode ajudar a tornar sua infra-estrutura de TI “à prova de balas” em relação aos ataques a vulnerabilidades conhecidas.

A seguir, mais um link, para ler um pequeno manual sobre o Nessus, embora em português de Portugal – o que pode gerar algumas “graças”. O artigo possui, no fim, uma denúncia sobre o uso incorreto do software livre em questão. Divirtam-se com o “Português”[6].

http://www.tldp.org/linuxfocus/Portugues/November2001/article217.shtml

Este software ficou tão popular e é tão utilizado para prestar serviços por consultores de segurança, que seu autor resolveu ganhar dinheiro. Atualmente, o software possui uma versão paga, em que as assinaturas de vulnerabilidades são liberadas imediatamente, com garantia de prazo máximo de vinte e quatro horas entre uma vulnerabilidade ser publicada na Internet e a assinatura estar disponível. Para os utilizadores da versão totalmente *free*, sem custos, as assinaturas somente são liberadas com uma semana de atraso, em relação ao sistema pago. Assim, os prestadores de serviços de detecção de vulnerabilidade que têm em seus contratos a obrigação de estarem

6 Tal comentário refere-se às diferenças entre o português falado no Brasil e o de Portugal.

totalmente atualizados, devem pagar para obterem rapidamente as assinaturas das vulnerabilidades descobertas “nas últimas 24 horas”. Veja em:

http://www.tenablesecurity.com/products/direct.shtml

Outros softwares

Para pesquisar na rede, citamos alguns outros softwares existentes.

Notem que todos podem ser utilizados para monitoração e prevenção ou para ataque. O conhecimento é distribuído, exatamente para que a maioria determinada a fazer o que é correto, o faça antes daqueles que estão sempre buscando novas armas de ataque na rede.

Ettercap

Trata-se de um conjunto de ferramentas para “ataques no meio do caminho”, a ser usado em redes locais. Possui decodificação de diversos protocolos do nível de aplicação, permitindo inclusive ao atacante inserir dados em uma conexão em andamento. Ele engana switches que não estejam com MAC amarrado por porta, de modo que pode-se capturar pacotes de qualquer conexão. Sua funcionalidade benéfica é a utilização em auditoria ou vigilância de suspeitos comprovados. Também possibilita verificar se existe algum outro Ettercap ligado à rede. Seu uso pode infringir várias leis, quando feito de forma arbitrária.

Etherape

É um visualizador gráfico de conexões e banda consumida por conexão, que funciona em tempo real. Sua tela causa impacto aos visitantes. Pode ser obtido em http://etherape.sourceforge.net/

LOGCHECK

Permite automatizar a verificação e classificação dos logs de um sistema Unix, enviando email em cadência predeterminada com:

- eventos não usuais (O que é isto?);

- violações de segurança em potencial (Atenção, verifique!);
- ataques ao sistema (Provável ataque em andamento!!!).

Pertence à família do portsentry, um detector de vulnerabilidades para redes (como o Nessus) e o hostsentry, um detector de vulnerabilidades para hosts. A Psionic, empresa responsável pelos softwares citados, foi comprada pela Cisco. Os sistemas ainda continuaram em desenvolvimento livremente, por algum tempo. Disponível em http://sourceforge.net/projects/sentrytools/

Unidade 4 - Protocolos de criptografia & PKI

Agora, caminharemos para a utilização prática da criptografia vista nas disciplinas anteriores, sendo que nos concentraremos nos problemas relacionados ao gerenciamento das chaves criptográficas e às formas encontradas para automatizar e melhorar este gerenciamento.

Citaremos e descreveremos sumariamente diversos protocolos. Cabe a você procurar mais informações nas fontes de consultas.

Texto 1 - Gerência de chaves

Os problemas que devem ser solucionados pelos especialistas preocupados com gerência de chaves começam na geração das chaves, passando pela guarda e entrega (transporte) da chave até os usuários finais.

Geração das chaves

A melhor chave é totalmente aleatória, mas dificilmente conseguimos aleatoriedade. Em sistemas computacionais, há duas formas tradicionais de obtê-la:

- entrada de usuário, por exemplo, através de movimentos do mouse e/ou medição do tempo obtido na digitação entre uma tecla e outra, de uma seqüência de teclas;
- eventos pseudo-randômicos: combinação de diversos fatores, como interrupções ao processador, hora do sistema, PID.

As senhas geradas por pessoas também são usadas como chaves, mas costumam ser fracas. Para fortalecer uma senha criada por um ser humano, pode-se passá-la por um HASH. Veja um exemplo. Usando Linux, digite:

```
md5sum
minha senha
^D  (Control D)
b8a5c586c970614fbf7cd9b71db354f3  -
```

A seqüência de caracteres obtida é uma chave bem melhor do que “minha senha”.

Armazenagem da chave

Além da geração, um problema a ser vencido é a armazenagem da chave. Podemos avaliar as seguintes opções de armazenamento:

- em memória – a segurança da máquina e do sistema operacional é importante;
- em disco – armazenadas em arquivo cifrado;
- em hardware especializado (smart cards);
- em hardware dedicado: HSM - *Hardware Security Module*;
- offline.

Outra forma de aumentar a segurança da chave está em utilizar segredo compartilhado: a senha/chave é quebrada em n partes, sendo que m (m<n) são suficientes para se gerar uma nova senha. Assim, se apenas um número menor que m de partes for comprometido, a chave, como um todo, não estará comprometida.

Distribuição das chaves

Uma vez criadas as chaves, muitas vezes será necessário entregá-la aos usuários finais, seja pessoa ou máquina/software. Nesse transporte, o sigilo e a inviolabilidade da chave precisam ser observados. Algumas possibilidades são:

- entrega em mãos – pode ser o mais seguro, mas, na maioria das vezes, é impraticável;

- enviada por correio – qual a confiabilidade deste processo? E o tempo que levará?

  utilizar o conceito de chave pública – porém, como garantir a autenticidade desta chave pública? Esse método pode ser oneroso devido à necessidade de alto poder computacional, para grande utilização;

- sistema misto – considerado o melhor método. Criam-se senhas/chaves de sessão de processamento leve e a entrega utiliza um sistema de chave pública. De tempos em tempos, a chave de sessão é trocada.

Texto 2 - Força de um sistema de criptografia

Em um bom sistema de criptografia, a segurança depende apenas do sigilo das chaves. Isso porque pressupõe que o algoritmo é forte o suficiente para resistir à criptoanálise, ou seja, uma mensagem não pode ser descriptografada sem o conhecimento da chave.

Para medir a força de um sistema, suponha que o algoritmo é bem conhecido. Logo, atacantes poderão encriptar textos e obter o resultado, mas serão incapazes de reverter o processo, de achar um algoritmo relativamente rápido que permita decifrar os textos, sem possuir a chave.

Assim, um sistema forte de criptografia depende apenas da chave, e a segurança, de seu sigilo.

Ataque de força bruta

Em um ataque de força bruta, o agressor tenta todas as chaves de criptografia possíveis. Este método torna-se impraticável se a chave de criptografia for grande.

Por exemplo, utilizando DES (*Data Encrypted Systems*) com chave de 56 bits,

teremos 72.100.000.000.000.000 de possibilidades. Precisamos tentar cada uma das combinações, para achar uma que faça abrir a mensagem. Como as chaves normalmente são baseadas em números aleatórios, primos e escolhidos “com cuidado”, provavelmente a chave não será um número fácil de achar.

Contudo, sabendo-se que um grupo da DEC criou um chip capaz de realizar 16.000.000 de operações DES por segundo, uma máquina com 1.000 desses chips seria capaz de quebrar uma mensagem criptografada com DES de 56 bits em menos de oito semanas.

Importância do tamanho da chave

A quebra de algoritmos é assunto regular da *RSA DATA SECURITY CONFERENCE*, realizada anualmente. O tempo para quebra por força bruta dos algoritmos mais utilizados depende sempre do poder computacional disponível para tal experimento. A seguir, mostramos alguns dados retirados dessas conferências, ao longo dos anos, e o tempo necessário (calculado) para quebrar os algoritmos:

- Em Janeiro de 1997
  - RC5 (40 bits) - 3 1/2 horas
  - RC5 (48 bits) - 313 horas
  - DES (56 bits) - 140 dias
- Em Julho de 1998
  - DES (56 bits) – 35 minutos (máquina de US$ 1.000.000,00)
  - Com duas chaves DES, com uma encriptação em cima da outra,com a mesma máquina: estimado em 4 bilhões de anos.
- Em Janeiro de 1999
  - DES (56 bits) - 22 horas e 15 minutos, com equipamentos de “mercado”

Tamanho mínimo de chaves

Como a força de um bom algoritmo está diretamente relacionada com a chave e o poder computacional disponível, a validade do que está escrito a seguir vai depender de quão rápido os sistemas computacionais se tornarão mais poderosos, após esta publicação.

Um fator que também deve ser levado em consideração, além do poder computacional disponível, é o quão perecível é a própria informação. Explicando: se o resultado de uma "loteria" fosse calculado no dia anterior a sua publicação, este dado seria considerado sigiloso somente durante vinte e quatro horas. Depois da publicação, nenhuma importância existe em quebrar a mensagem criptográfica que guardava o resultado.

Já os dados de projeto de uma usina nuclear ou de um estabelecimento militar precisa ser preservado por anos.

Com isso em mente, em 2004, recomendava-se:

- criptografia simétrica:
  - para dados perecíveis em um dia: 75 bits;
  - para dados perecíveis em 20 anos: 90 bits;
- para sistemas assimétricos:
  - 768 a 1024 bits.

Em setembro de 2005 (DC), o cert.br publicou em sua cartilha de segurança, o seguinte (disponível em: http://cartilha.cert.br/conceitos/sec8.html#subsec8.5):

> Atualmente, para se obter um bom nível de segurança na utilização do método de criptografia de chave única, é aconselhável utilizar chaves de no mínimo 128 bits. E para o método de criptografia de chaves pública e privada é aconselhável utilizar chaves de 2048 bits, sendo o mínimo aceitável de 1024 bits. Dependendo dos fins para os quais os métodos criptográficos serão utilizados, deve-se considerar a utilização de chaves maiores: 256 ou 512 bits para chave única e 4096 ou 8192 bits para chaves pública e privada.

Neste texto, chave única refere-se a chave simétrica.

Quanto ao uso de chaves maiores, devemos lembrar da limitação imposta pelos softwares de mercado, e seu suporte ao tamanho das chaves. Assim, para algumas aplicações mais críticas, preciamos pensar em adequar um software de criptografia, se não houver disponibilidade de configuração nos sistemas de mercado. Felizmente, a implementação dos algoritmos de criptografia caminha para uma parametrização confortável, em que o tamanho da chave pode ser escolhido.

A pergunta que fica é: até quando estes tamanhos de chaves utilizados e comercializados serão suficientes?

Texto 3 - Introdução a PKI

Como vimos, o principal problema na utilização prática de criptografia reside basicamente, na gerência das chaves, seu sigilo, guarda e entrega. Para o emprego dos sistemas de criptografia no mundo dos negócios, foi necessário criar uma série de procedimentos que atacasse e resolvesse de forma precisa, pública e segura todos os problemas relacionados com as chaves. Com esta finalidade, foi criada a Infra-estrutura de Chave Pública (ICP) – *Public Key Infrastructure* (PKI).

Os sistemas PKI provêem os níveis de segurança necessários à concretização destes negócios, garantido a confidencialidade, autenticidade, integridade e não- repudiação para transações comerciais e financeiras na grande rede. As facilidades e benefícios dos sistemas PKI não estão restritos somente à Internet, podendo ser empregados nos processos de comunicação segura entre funcionários de uma empresa, fornecendo os meios legais para garantir a validade de documentos armazenados em formato digital, implementando métodos de autenticação “forte” e logon único.

Uma Infra-estrutura de Chaves Públicas é um conjunto de regimes normativos, procedimentos, padrões de formatos, algoritmos e protocolos e, finalmente,

implementação de serviços que disponibilizam e/ou viabilizam o uso interoperável e escalável da criptografia assimétrica para redes abertas, compatíveis com tais padrões.

São funções de um PKI:

- estabelecer políticas de segurança que definam as regras nas quais os sistemas criptográficos devem operar;
- especificar produtos para gerar, armazenar e administrar chaves;
- normatizar os processos que especificam como as chaves e os certificados devem ser gerados, distribuídos e utilizados.

Como características do PKI, citamos:

- seu padrão de segurança na Internet tem aceitação global;
- possui funcionalidade para assinaturas digitais;
- é uma plataforma comum de segurança para redes corporativas e públicas;
- facilita a transição para as tecnologias em processo de desenvolvimento.

Então, um PKI é uma combinação de produtos de hardware e software, procedimentos e políticas de segurança. O PKI fornece a segurança necessária para comércio eletrônico: parceiros que não se conhecem podem trocar informações de modo seguro por meio de cadeias de credibilidade.

O PKI utiliza certificados de chave pública ou certificados digitais que vinculam os usuários a uma chave pública.

Atualmente, a utilização de Infra-estrutura de Chaves Públicas associada a *smart cards* é a forma mais segura de conduzir negócios pela Internet.

O Brasil está caminhando no sentido de ter seu próprio controle de chaves, sobretudo para utilização nas transações governamentais (mas não só), através do ICP-Brasil.

O que é o ICP-Brasil

“É um conjunto de técnicas, práticas e procedimentos, a ser implementado

pelas organizações governamentais e privadas brasileiras, com o objetivo de estabelecer os fundamentos técnicos e metodológicos de um sistema de certificação digital baseado em chave pública." (Disponível em: http://www.icpbrasil.gov.br/).

O ICP Brasil foi criado pelo **Instituto Nacional de Tecnologia da Informação**. Para conhecê-lo, acesse http://www.iti.gov.br/

Texto 4 - Padrões para criptografia de chave pública

Esse título é mais conhecido como PKCS: *Public Key Cryptography Standards.*

Os primeiros documentos foram divulgados em 1991, como parte dos acordos que estabeleceram os padrões de Criptografia por Chave Pública.

A padronização resultou do trabalho conjunto de empresas e universidades, tais como: Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell, Sun.

Revisados inicialmente em 1993, vários documentos sofreram posteriores correções e alguns novos foram publicados depois. Desde 1996, há grupos de trabalho atualizando as especificações.

A seguir, resumimos alguns documentos e o foco de seu conteúdo. Cabe a você, interessado no assunto, buscar detalhes.

Em uma Unidade adiante, esses protocolos serão mencionados sem maiores explicações, assim como acontece em editais, licitações e consultas públicas. Veja exemplo em:

http://www.iti.br/twiki/bin/view/Main/ConsultaPublicaPadroesSoftware.

PKCS #1: Criptografia RSA

Recomendações para implementação de um sistema de criptografia de chave pública baseado no algoritmo RSA.

- V1.5 (1993): descreve os procedimentos básicos do SSL, S/MIME e PKIX;
- V2.0 (1998): acrescenta a criptografia OAEP (*Optimal Asymmetric Encryption Padding*) de Bellare-Rogaway;
- V2.1: inclui a criptografia B-R PSS (*Probabilistic Signature Scheme*).

PKCS #7: Sintaxe de mensagens criptografadas

Desenvolvida para e-mail seguro (*Privacy-Enhanced Mail*).

- V1.5 (1993): orientada para administração de chaves RSA, utilizada em S/MIME e protocolos PKIX;
- RFC 2630 do IETF: inclui o gerenciamento de chaves Diffie Hellman;
- V1.6bis: aceita protocolo SET.

PKCS #10: Requisição de certificados

Especifica o padrão da sintaxe para a solicitação de certificados. A certificação inclui Nome único (DN), chave pública e conjunto opcional de atributos, assinados digitalmente pela entidade solicitante.

Este padrão explicita que a CA transforma a solicitação em um certificado X.509 ou em um certificado PKCS#6.

No entanto, não define a forma pela qual a CA envia o certificado para o solicitante.

PKCS #11: (Criptoki) Cryptographic Token Interface

Divulgado em 1995, este padrão especifica uma interface de programação (API), chamada Cryptoki, para SmartCards e outros dispositivos que irão

armazenar informações criptográficas e realizar operações de criptografia. Cryptoki – pronunciado como *cripto-key*, nome curto para *cryptographic token interface* – segue uma simples aproximação orientada a objetos, com o objetivo de ser independente de tecnologia (para qualquer tipo de dispositivo) e com recursos compartilhados (múltiplas aplicações acessando múltiplos dispositivos), apresentando para as aplicações uma visão lógica comum para qualquer dispositivo chamado de *token* de criptografia (*cryptographic token*).

Várias versões já foram publicadas. Exemplos:

- V1.0 (1995): descreve os métodos básicos;
- V2.01 (1997): acrescenta mecanismos de criptografia e formas de administração;
- V2.10: detalha melhor a interface e acrescenta novos mecanismos criptográficos;
- V2.20 CorreçãoAmendment 1: PKCS #11 mechanisms for One-Time Password Tokens.

PKCS #15: Formato de arquivos para dados criptografados

Especifica o formato de arquivos para dados criptografados (*Cryptographic Token Information Format*) em SmartCards e outros dispositivos.

- V1.0 (1998): adotada pelo WAP Forum e para cartões EDI, foi desenvolvida em conjunto com o SEIS - *Secured Electronic Information in Society;*
- V1.1: inclui formatos para compatibilização com aplicações.

PKCS: Outros Documentos

Relacionamos a seguir outros documentos e os títulos. Detalhes podem ser obtidos em http://www.rsasecurity.com/rsalabs/node.asp?id=2124

- PKCS #3: chaves Diffie-Hellman;
- PKCS #5: criptografia baseada em passwords;
- PKCS #8: sintaxe da informação da chave privada;

- PKCS #9: tipos de atributos selecionados;
- PKCS #10: sintaxe de solicitações de certificados;
- PKCS #12: sintaxe da troca de informações pessoais;
- PKCS#2 e PKCS#4 estão obsoletos e foram incorporados ao PKCS#1;
- PKCS#13: criptografia com curvas elípticas;
- PKCS#14: em desenvolvimento.

Texto 5 - Certificados Digitais

Certificados Digitais são os equivalentes eletrônicos às provas físicas de identificação, como passaporte[7] e cédulas de identidade. Auxiliam a autenticação de usuários em redes de comunicações e são elementos essenciais em um PKI, podendo estar armazenados em uma estação, um disquete ou em um dispositivo de segurança como o smart-card.

A Figura 16 ilustra o conteúdo de um certificado digital.

7 Passaporte: documento emitido pelo governo que tem credibilidade em outros países.

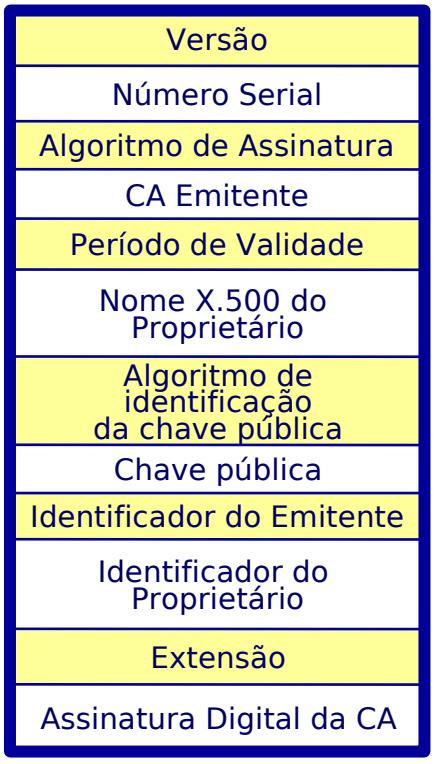

Figura 16 - Conteúdo de um certificado digital.

Credibilidade do certificado digital

O certificado digital é um tipo de passaporte que identifica e autentica o proprietário. Assim, um certificado digital emitido por uma Autoridade Certificadora confiável (com fé pública, a exemplo dos cartórios tradicionais) também tem credibilidade.

Validação de um certificado

A validação de um certificado segue os passos:

1. o receptor obtém o certificado da origem da comunicação e o software executa as seguintes funções:
2. obtém o certificado da CA que assina o certificado recebido;
3. decripta a amostra do certificado usando a chave pública da CA;
4. calcula a amostra do certificado;

5. compara as amostras;

6. verifica a data de expiração do certificado;

7. verifica se o certificado foi revogado.

Veremos adiante que os certificados das CAs mundialmente reconhecidas, mencionados no passo 2, normalmente já estão cadastrados nos softwares que irão trabalhar com certificação digital, de modo que obter este certificado representa tão somente acessar um arquivo em disco local.

Certificados – revogação

Pode ser necessário revogar um certificado, antes que termine seu prazo de validade, fora do qual, o certificado é automaticamente considerado revogado. As possíveis razões para se fazer uma revogação explícita incluem, entre outras:

- comprometimento da CA;
- comprometimento da Chave Privada;
- mudança de Status;
- suspensão.

A lista de certificados revogados é chamada de CRL (*Certificate Revocation List*). Obrigatoriamente, as CRLs são emitidas e assinadas pela CA. A CRL deve ser verificada sempre que um certificado é recebido.

Implementação de infra-estruturas de chaves públicas

Uma infra-estrutura de chave pública simples, ilustrada na Figura 17, consiste de:

- Autoridade Certificadora;
- Autoridade de Registro;
- Diretório;
- Aplicações PKI;
- Políticas & procedimentos.

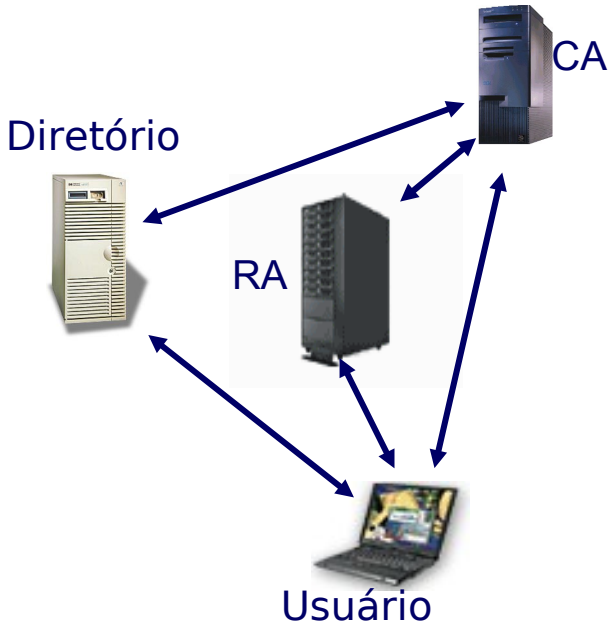

Figura 17 - Relações entre as entidades de um PKI.

Uma Autoridade Certificadora (CA) é a base de credibilidade do PKI e administra todo o ciclo de vida dos certificados de chave pública.

Uma Autoridades de Registro (RA) faz a interface entre a CA e o usuário final. Ela identifica o usuário para a CA e obtém a garantia, a autenticação de que o usuário é quem ele diz ser, normalmente através de documentos tradicionais, tais como: cédulas de identidade, contrato social (empresas), CGC/CNPJ etc.

A qualidade do processo de autenticação do usuário determina o nível de credibilidade que pode ser atribuído a um certificado.

Já a Autoridade de Registro é responsável por manter os registros, por meio de cópias dos documentos apresentados pelos usuários.

Os Diretórios funcionam como ponto de distribuição para certificados e para as relações de certificados revogados. Podem estar distribuídos em redes. Toda informação armazenada segue o padrão X.500. Se necessário, armazenam também outras informações.

As Aplicações PKI são representadas pelos softwares que provêem a forma de utilizar, na prática, o PKI. Dentre elas, citamos as funções:

- a comunicação entre servidores web e browsers;
- E-mail criptografado e/ou assinado digitalmente;
- *Electronic Data Interchange* (EDI);
- transações de cartões de crédito na Internet;
- Redes Virtuais Privadas (VPN) seguras.

Procedimentos e políticas

Quando se trata de credibilidade, um documento registrado em papel pode representar valores de vários milhões dependendo de:

- quem emitiu;
- quem assinou;
- facilidade de falsificar;
- embasamento legal do documento.

A credibilidade dos certificados digitais depende de vários fatores e coloca uma chave em diferentes níveis de credibilidade. A seguir, relacionamos alguns desses fatores:

- política de segurança (resistência da chave, armazenamento de chaves);
- critérios de identificação do solicitante de um certificado;
- obrigações contratuais entre as partes;
- legislação sobre certificados e assinaturas digitais;
- tamanho e método de geração (aleatoriedade) das chaves;
- uso das chaves;
- método para transporte;
- método de Importação da chave no usuário final;

- armazenamento e recuperação;
- troca das informações;
- possibilidade de comprometimento da chave;
- procedimentos administrativos;
- responsabilidades e compensações (contratuais);
- controle de acessos;
- cargos e responsabilidades;
- hardware;
- segurança física;
- níveis de serviços;
- planos de contingência;
- critério de emissão manual/automático;
- critério de identificação do requisitante da chave;
- tipos de certificados;
- tempo de vida;
- presteza e confiabilidade dos serviços de revogação e CRLs;
- qualidade dos serviços de diretórios.

Custódia ou Armazenamento de Chaves

A custódia (key escrow) considera as necessidades de governos e empresas de:

- controlar o uso da criptografia – em muitos países a criptografia é considerada como uma arma bélica;
- possibilitar, sob garantias, decriptar um texto cifrado.

Da mesma forma, o armazenamento de chaves (*key archive*) constitui uma necessidade comercial para permitir que as empresas se recuperem de:

- chaves perdidas;
- passwords esquecidas;
- perda de funcionários.

As chaves públicas são sempre utilizadas em pares: a chave pública e a chave privada. Podem-se criar dois pares de chaves com as seguintes funcionalidades:

- um par para criptografia;
- um par para não repúdio (assinatura).

Quando se deseja armazenar chaves privadas, temos o problema da perda da credibilidade (repúdio), pois alguém pode afirmar que o sistema de armazenamento teve falha de segurança e a chave vazou. A solução para este impasse é deixar a chave privada de criptografia ser armazenada ou custodiada, porém a chave privada de assinatura deve ser de controle exclusivo do usuário.

Unidade 5 - Usando a criptografia

Nesta unidade veremos exemplos de sistemas que utilizam criptografia, empregados em sistemas comerciais, tendo por base software livre, o que permitirá a você criar seu próprio certificado digital, ou gerar uma requisição para ser assinada por uma CA com fé pública ou, ainda, criar seu próprio PKI.

Texto 1 - PGP

O PGP (*Pretty Good Privacy*) foi criado para encriptação de mensagens de texto em emails. Utiliza a criptografia baseada em chave pública e chave privada. Assim, permite assinatura da mensagem, que pode ser conferida com a chave pública. Como possui suporte disponível em diversos programas de email, seu uso pode facilmente se tornar comum.

O PGP é um utilitário prático de criptografia e assinatura (certificado) digital, baseado nos mais modernos, seguros e eficientes sistemas de criptografia. Originalmente criado e desenvolvido por Philip R. Zimmermann (http://www.philzimmermann.com/), em 1991, só foi possível difundir mundialmente esta tecnologia, driblando as rigorosas restrições de exportação de tecnologia de segurança dos EUA. Para isso, todo o código do PGP foi impresso em um livro, para que o software pudesse ser digitalizado (OCR) e reconstruído independentemente fora dos Estados Unidos, livre das restrições.

Em função disso, o Governo dos EUA agiu legalmente contra Phil Zimmermann. Somente em janeiro de 1996, a Suprema Corte dos EUA abandonou o caso. Em março daquele ano, foi formada a PGP Inc., cuidando das versões comerciais do PGP, tendo Phil Zimmermann como CEO. Em dezembro de 1997, a Network Associates Inc. (NAI) adquiriu a propriedade da empresa PGP Inc. e Phil permaneceu por um bom tempo na NAI como Membro Senior, provendo orientação técnica no desenvolvimento contínuo do PGP e

garantindo a idoneidade das versões lá desenvolvidas (até a 7.0.3), livre de *back doors* e com divulgação pública de todo o código fonte.

Em fevereiro de 2001, Phil deixou a NAI para dar prosseguimento a seus projetos de proteção de privacidade pessoal. Lançou o *OpenPGP Consortium* (http://www.openpgp.org/), que visa facilitar a interoperabilidade de implementações do padrão *OpenPGP*. Em junho de 2002, os produtos e propriedade intelectual do PGP foram adquiridos da Network Associates por uma nova empresa, a PGP Corporation, onde Zimmermann agora atua como orientador especial e consultor. E o próprio Phil Zimmermann agora também é revendedor do PGP.

O PGP possui um equivalente totalmente *free*. O GnuPG (http://www.gnupg.org/). Há *frontends* gráficos e *plugins* disponíveis para programas de email. Assim , o GnuPG pode ser usado com facilidade.

O GPA, por exemplo, constitui uma dessas ferramentas disponíveis no site do GnuPG. Possui um gerenciador de chaves e um manipulador de arquivos criptografados. A Figura 18 ilustra o gerenciador de chaves.

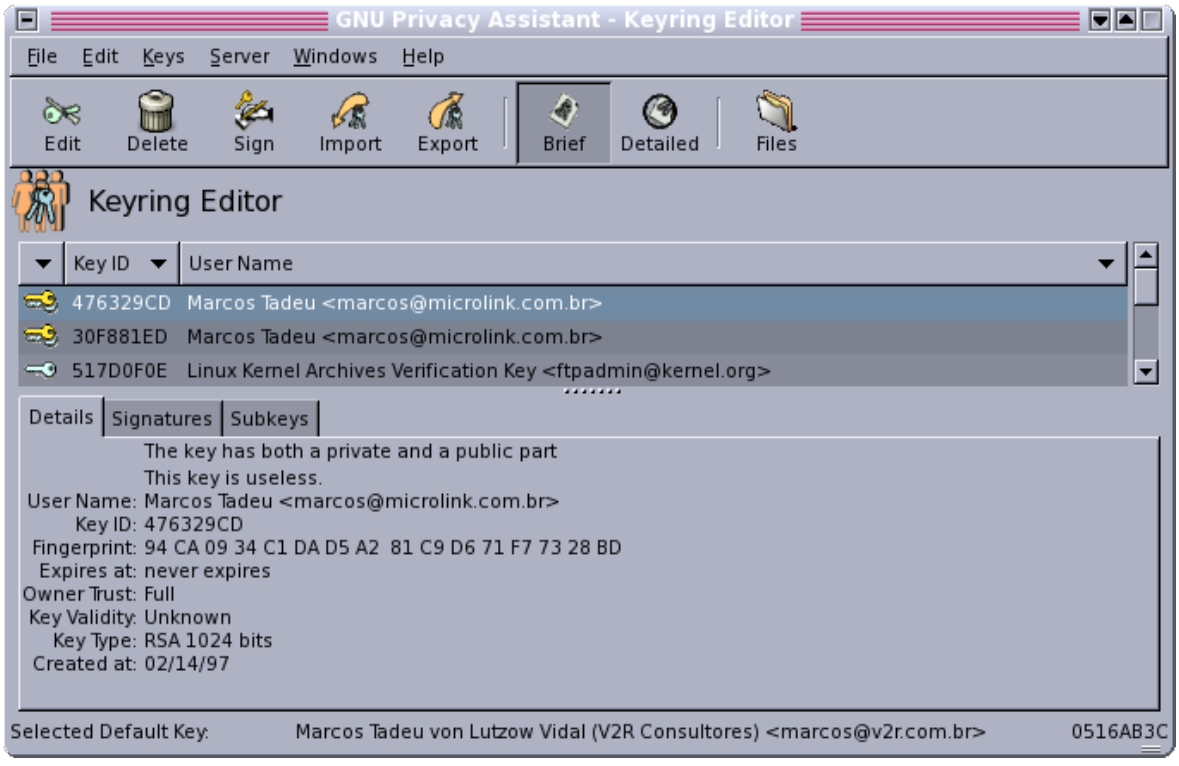

Figura 18 - GPA - o gerenciador de chaves.

Texto 2 - SSL

O SSL (*Secure Socket Layer*), ou Camada de Soquetes Segura, foi desenvolvido pela Netscape em 1994 com o objetivo de permitir aos clientes (normalmente os navegadores Web) e servidores HTTP comunicarem-se sobre uma conexão segura. O SSL oferece encriptação, autenticação e integridade dos dados no intuito de proteger a informação trocada em redes públicas inseguras.

Há várias versões do SSL:

- SSL versão 2.0 possui algumas fraquezas de segurança, não sendo muito utilizado atualmente;
- SSL 3.0 é universalmente suportado;
- finalmente, o *Transport Layer Security* (TLS), um melhoramento do SSL 3.0, tem sido adotado como padrão na Internet e é amplamente suportado por quase todos os softwares recentes.

Figura 19 - A subcamada SSL na pilha TCP/IP.

Conforme ilustrado na Figura 19, sua utilização é realizada como uma subcamada da camada Aplicação da arquitetura TCP/IP, ou seja, o SSL recebe os dados da Aplicação, trata-os (criptografa) e envia o resultado à camada TCP, para ser transportado. Do outro lado, no destino, a camada TCP entrega os pacotes com dados criptografados à camada SSL. Esta, por sua vez, desfaz a criptografia e entrega os dados “originais à Aplicação”. Então, o SSL pode

ser usado seletivamente e criptografa, apenas, os dados que esta aplicação desejar proteger.

Negociação da encriptação

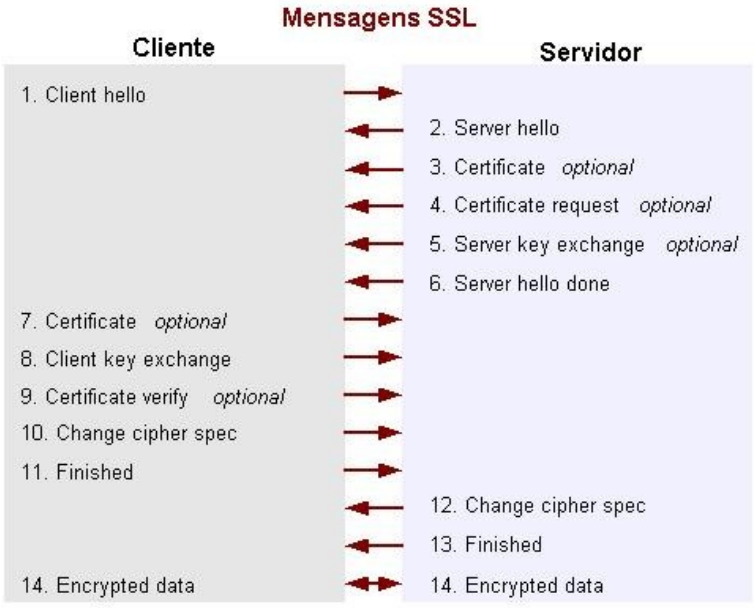

Figura 20 - Estabelecimento de uma sessão SSL.

Após estabelecida a conexão TCP, a camada usuária desta, que agora é o SSL, passa a trocar informações com a camada SSL da outra ponta, com o objetivo de estabelecer os parâmetros de criptografia que serão utilizados na fase de transferência de dados. A autenticidade do servidor e/ou do cliente também pode ser analisada antes de ocorrer qualquer transferência de dados.

A Figura 20 ilustra este *handshake*, que passa pelas seguintes fases:

- cliente e servidor trocam mensagens de saudação (*Hello*) que contém dados aleatórios;
- servidor envia ao cliente sua chave pública, junto a um certificado assinado; cliente poderá verificar a autenticidade deste certificado;
- cliente gera dados aleatórios para usar como um "segredo compartilhado",

enviando, em seguida, estes dados ao servidor, criptografados com a chave pública recebida dele, em uma mensagem denominada *Client Key Exchange*;

- cliente e servidor utilizam uma combinação dos dados aleatórios trocados e dos dados secretos gerados pelo cliente para gerar as chaves criptográficas que serão utilizadas;
- cliente envia uma mensagem, denominada *Change Cypher Spec,* para determinar o serviço/algoritmo de criptografia a ser utilizado, e uma mensagem de fim da etapa de estabelecimento de conexão, *Finished;*
- o servidor responde com o envio de *Change Cypher Spec* e mensagem *Finished*.

A partir daí, as camadas de Aplicação do cliente e do servidor podem começar a enviar dados de um lado para o outro.

Texto 3 - SSH

Substitui o telnet e o ftp por similares seguros, onde os dados trafegam criptografados e permite a criação de túneis seguros (criptografados).

O SSH tornou-se comercial, mas há uma vertente *free* do projeto em http://www.openssh.org. No site também encontraremos clientes para Windows (putty e winscp).

Um uso importante do telnet é na configuração remota de equipamento, sobretudo de roteadores. Vale ressaltar que os novos roteadores já possuem servidor SSH incorporado. Assim, o perigo existente com o uso do telnet (que passa tudo em texto plano) pode ser eliminado com o uso do SSH. Abandone o telnet!

O SSH possibilita a criação de túneis onde os dados passam criptografados.

Em uma aplicação que irá usar um túnel, a única modificação está na configuração do endereço IP do servidor, que deve ser trocado pelo IP da máquina onde roda o cliente SSH. Normalmente, tal máquina é a mesma onde está o cliente e usa-se o endereço localhost. Um exemplo de utilização túnel será apresentado com o SSL, mais adiante. O método de utilização a ser apresentado vale também para o SSH.

Rodando o SLAX CD, digite:

```
/etc/rc.d/rc.sshd
```

O SSH irá gerar as chaves de criptografia e colocar informações na tela. Isto só acontece na primeira vez em que o sshd (*daemon ssh*) roda, por não encontrar as chaves. No CD, isso acontece a cada boot, pois as chaves não são gravadas.

Recomendamos estudar o SSH e suas funcionalidades, usando os comandos:

```
man ssh
man sshd
```

Veja exemplos de utilização e documentação em:

- http://www.fisica.ufc.br/rede/acesso/
- http://www.fisica.ufc.br/rede/virtual/
- http://www.crc.dcc.ufmg.br/tutoriais/tunelssh_mysql/

Texto 4 - OpenSSL

O projeto OpenSSL disponibiliza um *toolkit* em código livre que implementa o protocolo SSL e vários algoritmos e primitivas criptográficas de uso comum, incluindo algoritmos de troca de chaves, funções de hash, algoritmos simétricos e assimétricos. O toolkit possui duas bibliotecas e um conjunto de programas (executáveis) que implementam as rotinas por elas disponibilizadas. Os mecanismos do SSL estão implementadas na libssl e os outros algoritmos, na libcrypto.handshake

Veja em http://www.openssl.org/ a definição do projeto, assim como alguma documentação.

O projeto OpenSSL foi tão bem aceito que grande parte dos servidores HTTP que rodam apache – o servidor opensource utilizado em mais de 60% dos servidores do mundo[8] – e que possuem a parte de servidor seguro (HTTPS), o fazem com as bibliotecas do OpenSSL.

Um outro projeto de grande importância na utilização da criptografia disponibilizada pelo OpenSSL é o projeto mod_ssl. Criado em 1998 por Ralf S. Engelschall, foi originalmente derivado do software desenvolvido por Ben Laurie para ser usado no Apache-SSL, um outro projeto de servidor seguro HTTP. O pacote mod_ssl é licenciado no estilo da licença BSD, o que quer dizer, basicamente, que pode ser usado para qualquer propósito, seja comercial ou não.

O site do mod_ssl (http://www.modssl.org/) possui extensa documentação. Podemos dizer que a documentação disponibilizada reúne praticamente todo o conhecimento utilizado, na prática, sobre criptografia para servidores HTTP. Seu estudo detalhado resume bem o assunto do curso de criptografia.

O comando openssl

Mesmo sem acesso à Internet, mas tendo acesso ao CD do SLAX, é possível estudar bastante. Digite no terminal:

```
openssl
OpenSSL>
```

O comando chamado de openssl é o principal programa executável do projeto OpenSSL, citado anteriormente. Note que mudou o *prompt* para

OpenSSL>

Isto significa que o comando openssl possui uma interface interativa, além de poder ser usado com parâmetros em linha de comando, o que é mais comum.

Digite “h” e tecle “Enter” – na verdade, poderíamos digitar qualquer coisa que não fosse um dos comandos do openssl. Será listado uma série de comandos que o openssl aceita.

8 Disponível em: http://news.netcraft.com/archives/web_server_survey.html.

Exercício: verifique, nessa listagem, se reconhece siglas que tenham sido estudadas em disciplinas anteriores. Relacione, pelo menos, cinco, descrevendo seu significado. Publique sua resposta no Fórum aberto no EBaula, especificamente para este exercício.

Após realizada a pesquisa, verifique no manual do openssl o significado de algumas siglas, principalmente as que ainda não foram estudadas, por se tratar de comandos próprios do openssl, e não de protocolos padronizados de criptografia. Que manual?

```
man openssl
```

Texto 5 - Gerando certificados

Vamos gerar alguns certificados. Ao fim, teremos um certificado auto-assinado que será usado como CA, um certificado assinado pela CA e uma requisição que poderia ser enviada para ser assinada por uma Autoridade Certificadora comercial e vir a se transformar em um certificado assinado com reconhecimento e validade públicos.

Todo certificado raiz de Autoridade Certificadora tem que ser auto assinado, pois sendo a autoridade máxima, não existe nenhum nível acima para a certificar ou garantir. Por isso, diz-se que uma CA tem "fé-pública" reconhecida por si só (*la garantía soy yo!)*. Que fique claro que nossa CA não terá validade pública.

Mas, é comum uma empresa criar seu próprio sistema de PKI interno, o que é feito gerando o próprio CA e importando este CA nos sistemas internos que usam criptografia, ou seja, nos clientes e servidores. Assim, todos os certificados dos servidores que forem criados poderão ser assinados pelo CA que, dentro da empresa, tem validade devido a total confiabilidade. Com a CA importada nos clientes, será fácil validar os demais certificados e, assim, ter

uma rede interna de confiabilidade.

Os certificados que serão gerados em nossos exemplos poderiam, por exemplo, ser usados no servidor web do EBaula, para que todas as transferências de dados, incluindo login e senha, passassem pela Internet criptografadas. O exemplo que executaremos é um dos caminhos para criar certificados para servidores seguros.

Como primeiro passo, vamos gerar um par de chaves RSA, para uma hipotética autoridade certificadora "EB". O arquivo de destino contém a chave privada e as informações necessárias para a geração da chave pública correspondente. Para iniciarmos já com disciplina, vamos eleger o diretório /etc/ssl como base de nossos experimentos com certificados. Na verdade, este diretório existe exatamente para isto, já com alguns arquivos de configuração do openssl, que alteraremos mais tarde.

Chave RSA de 2048 bits

Vamos começar gerando uma chave privada RSA de 2048 bits. Ela será criptografada com 3DES, usando como chave da criptografia 3DES uma senha *(pass phrase*) que será pedida e que você deverá digitar. A chave gerada será gravada no arquivo de nome colocado após o parâmetro “-out”, no caso, “chave_privada_ca.pem”. Execute os dois comandos a seguir (mostramos, também, o resultado visto na tela):

```
cd /etc/ssl
openssl genrsa -des3 -out chave_privada_ca.pem 2048
Generating RSA private key, 2048 bit long modulus
..............................+++
.........................................+++
e is 65537 (0x10001)
Enter pass phrase for chave_privada_ca.pem:
Verifying - Enter pass phrase for chave_privada_ca.pem:
```

Criptografar a chave com 3DES é importante para garantir a inviolabilidade da chave e impedir que ela seja usada por qualquer um. Com isso, sempre que o openssl for usar a chave, a senha será pedida. Sendo uma chave que será usada como chave privada de uma CA, a senha escolhida é essencial.

Convenhamos que, no nosso caso, é apenas um exemplo/teste. Assim, para facilitar, colocamos a senha como 1111 – o mínimo para a senha são quatro caracteres e pode ter até 8191 caracteres!

Veja como ficou o arquivo com a chave, com o comando "cat". Note as informações sobre o algoritmo. Com certeza, a chave gerada em seu sistema será totalmente diferente deste exemplo e, por isto, retiramos o miolo da chave na impressão a seguir.

```
cat chave_privada_ca.pem
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,40FF783805D3B417

mxY8oqImrBHSMPjt+JHah512nVn1Zcp29lbVbt42lAKrgFGPIxG+RF1mjnQncYE/
.......... miolo retirado ......
bWMbG+SI0ZQpC4IZmyCmHky8DKbGiPo8WKG+i8C6GHFO4ISyEGnt1zT/5a6FEwKD
-----END RSA PRIVATE KEY-----
```

A seqüência de caracteres é a própria chave RSA de 2048 bits, criptografada com 3DES e gravada em sua representação em ASCII. Este formato de gravação de chaves e certificados em arquivos é chamado PEM (daí a extensão do nome do arquivo que usamos).

Para ver detalhes da chave, digite o comando a seguir. Ele indica ser uma operação com o algoritmo RSA e usa como entrada (nome após o "-in") o arquivo onde está a chave. O parâmetro "-text" serve para indicar que os detalhes da chave devem ser listados em modo texto. Os números aparecem em notação hexadecimal, onde os bytes são separados por dois pontos (as linhas intermediárias com os números foram subtraídas, em prol da economia de papel, mesmo porque sua chave é diferente e o importante são os campos que ela contém):

```
openssl rsa -in chave_privada_ca.pem -text
Enter pass phrase for chave_privada_ca.pem:
Private-Key: (2048 bit)
modulus:
    00:e3:f0:ba:b5:05:ca:ec:61:df:f1:b7:fb:1a:7a:
    ..............
    5e:9f
```

```
publicExponent: 65537 (0x10001)
privateExponent:
    00:cc:3b:05:0e:ed:01:c4:8a:6d:7c:c6:bb:d6:1e:
    ..............
    ec:01
prime1:
    00:f9:63:30:d5:51:e7:3c:89:50:c1:d8:5b:e6:46:
    ..............
    bf:2d:5b:9e:41:b1:91:24:0f
prime2:
    00:e9:fb:f5:58:57:85:f8:22:02:4d:d8:81:57:37:
    ..............
    03:ba:e8:a9:c4:35:5a:4c:71
exponent1:
    01:8c:7b:26:27:ff:0e:a2:2f:ad:34:81:c5:99:54:
    ..............
    7d:b4:9e:f3:fb:7d:52:43
exponent2:
    63:84:78:6b:64:7e:64:75:02:89:dd:85:d5:74:9e:
    ..............
    fa:61:2e:a3:77:15:ba:e1
coefficient:
    00:90:11:64:3c:2c:4a:2c:2e:50:e9:cd:e8:ec:1f:
    ..............
    11:b9:7d:0f:f4:b9:a3:b1:fa
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA4/C6tQXK7GHf8bf7GnrgeLdnZ/4EIOc34gh8xW6VBg88B63J
..............
XwCGuWnJFn46NxI47HKNLQ/C5ge/bhy/ne1nOpIzfhG5fQ/0uaOx+g==
-----END RSA PRIVATE KEY-----
```

Agora é hora de recordar, usando o módulo de Criptografia Geral.

Exercícios:

1. Identifique, pelo nome, todos os campos mostrados nesta chave, com o processo teórico de geração estudado. Discuta no Fórum.
2. Gere chaves de diferentes tamanhos (512, 1024, 2048, 4096 bits) e verifique as mudanças que ocorrem no tamanho do conteúdo da chave, listando-as com o comando recém-aprendido. Perceba também a diferença

no tempo gasto para a geração das chaves de diferentes tamanhos.

3. Gere várias chaves de mesmo tamanho, por exemplo de 2048 bits, sem senha (sem criptografar com 3DES), gravando no arquivo /dev/null. Qual o comando para isto? Explique o motivo de haver diferenças no tempo de geração, se as chaves são do mesmo tamanho. O que são os pontinhos (.) e os sinais de mais (+)?

Certificado auto-assinado

A partir da chave RSA, vamos gerar um certificado para nossa CA "EB". Após o comando, será necessário fornecer a senha da chave RSA e, depois, responder a uma série de perguntas. Nelas, um valor padrão é oferecido, aparecendo entre colchetes "[]". Onde os colchetes estiverem vazios, nenhum valor padrão foi oferecido e o campo ficará igualmente vazio, simplesmente pressionando-se *Enter*. Os valores padronizados ficam no arquivo de configuração do openssl (/etc/ssl/openssl.cnf) e podem ser alterados editando-se o arquivo. Como todo arquivo de configuração no UNIX, é um arquivo texto passível de alterações com qualquer editor de texto.

```
openssl req -new -x509 -key chave_privada_ca.pem -out certificado_ca.pem
-days 365
Enter pass phrase for chave_privada_ca.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Rio de Janeiro
Locality Name (eg, city) []:Niterói
Organization Name (eg, company) [Internet Widgits Pty Ltd]:EB
Organizational Unit Name (eg, section) []:EAD
Common Name (eg, YOUR name) []:CA EB
Email Address []:
```

Vamos ver como ficou o CA, usando o subcomando x509 do openssl. Observe o que é possível fazer com o x509, usando:

```
man x509
```

Vejamos o certificado, com a linha de comando a seguir:

```
openssl x509 -in certificado_ca.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            8d:96:24:3b:e8:c3:dc:36
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=BR, ST=Rio de Janeiro, L=Niter\xF3i, O=EB, OU=EAD, CN=CA EB
        Validity
            Not Before: Jun 10 11:30:24 2006 GMT
            Not After : Jun 10 11:30:24 2007 GMT
        Subject: C=BR, ST=Rio de Janeiro, L=Niter\xF3i, O=EB, OU=EAD, CN=CA EB
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:e2:7e:88:3e:28:e9:cb:d1:e6:ea:12:b9:a9:ce:
                    ............
                    3a:6d:bc:bb:20:f9:d3:d6:1b:14:ff:6b:9e:05:6d:
                    e5:0f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                F5:8C:CD:05:61:61:38:04:D2:71:D8:5F:90:08:88:E1:EF:D0:E0:9A
            X509v3 Authority Key Identifier:
                keyid:F5:8C:CD:05:61:61:38:04:D2:71:D8:5F:90:08:88:E1:EF:D0:E0:9
A

                DirName:/C=BR/ST=Rio de Janeiro/L=Niter\xF3i/O=EB/OU=EAD/CN=CA
EB

                serial:8D:96:24:3B:E8:C3:DC:36

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        b8:17:05:64:cf:c0:25:f4:63:bf:62:ba:64:5e:7c:6a:08:80:
        ............
        8f:ab:bb:6a
-----BEGIN CERTIFICATE-----
MIIEEjCCAvqgAwIBAgIJAI2WJDvow9w2MA0GCSqGSIb3DQEBBAUAMGMxCzAJBgNV
```

```
.....................
SFrGU3XNCp2N2XX8CuvDRh6FZGLCN/8S4eHL2QT80TmklY+ru2o=
-----END CERTIFICATE-----
```

A primeira lição a ser aprendida com o exemplo é que não deve haver acentos nas palavras usadas nos campos de certificados (Note como ficou a palavra Niterói, ao visualizarmos o certificado da CA).

Exercícios:

1. Usando como referência o manual (“man openssl” e “man x509”), identifique cada parâmetro dos dois últimos comandos openssl utilizados.
2. Identifique e comente o significado dos campos listados da CA.

Chave do servidor

Agora vamos gerar uma chave (par público/privado) RSA para ser usada como certificado para o servidor.

```
openssl genrsa -out chave_privada_sv.pem 2048
Generating RSA private key, 2048 bit long modulus
.............................................+++
.............................................................+++
e is 65537 (0x10001)
```

Criptografar ou não a chave

Note que foi retirada a criptografia 3DES (-des no comando), ou seja, esta chave não está protegida por senha e pode ser usada por qualquer um que tenha acesso a ela.

Também, a não proteção por senha possibilita que o futuro certificado assinado seja roubado para ser usado em um servidor falso. Se ocorrer o roubo do certificado, e o fato for descoberto, impedimos sua validação pela CA, incluindo-o em uma lista de certificados revogados.

Entretanto, a chave protegida por senha tem uma desvantagem prática: toda vez que o servidor, ou o processo que presta o serviço e utiliza a chave, for iniciado, alguém terá que digitar a senha do certificado. Assim, é comum não

usar essa segurança e garantir que a segurança por acesso físico e por permissões de acesso ao arquivo da chave seja bem cuidada.

Então, para o arquivo com a chave, coloque permissão de leitura somente para o dono.

Requisição de certificado

Vamos gerar uma requisição de certificado, usando a chave gerada no item anterior. Essa requisição é para ser assinada pela autoridade certificadora EB. Os dados pedidos agora serão usados no certificado do servidor. Cada servidor deve ter seu próprio certificado. O nome colocado no campo “Common Name (eg, YOUR name)” deve ser o nome da máquina ou o nome do serviço. Em um servidor HTTP, o nome pelo qual é chamado (URL), por exemplo: www.ensino.eb.br.

```
openssl req -new -key chave_privada_sv.pem -out pedido_sv.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [BR]:
State or Province Name (full name) [Rio de Janeiro]:
Locality Name (eg, city) [Niteroi]:
Organization Name (eg, company) [EB]:
Organizational Unit Name (eg, section) []:EAD
Common Name (eg, YOUR name) []:www.ensino.eb.br
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Verifique como ficou a requisição:

```
openssl req -in requisicao_sv.pem -text
```

Assinando a requisição com o certificado da CA

Agora vamos usar a autoridade certificadora “EB” para assinar a chave. Em um PKI real, é um momento importante.

Antes, porém, precisaremos criar alguns arquivos, pois o openssl já possui internamente controle para operação de um mini PKI. Esse controle inclui o número serial de certificados que a CA já assinou e um índice com um resumo dos certificados assinados. Tais arquivos são utilizados com um mini-banco de dados para controlar listas de revogação. Procure no arquivo openssl.cnf a sessão [CA_default].

Verifique a linha “dir = ./demoCA”. Ela indica que este é o diretório de trabalho, que deve existir com os dois arquivos citados. Vamos criar o diretório e os arquivos e, no arquivo serial, colocar “01”, já que é nosso primeiro certificado.

```
mkdir demoCA
touch demoCA/index.txt
cat > demoCA/serial
01
Crtl D   #digite Control D
```

Após isso, nossa assinatura já pode ser realizada (atenção: o comando a seguir ocupa as duas primeiras linhas, mas deve ser todo digitado e, somente ao fim, pressionar enter):

```
openssl ca -outdir . -keyfile chave_privada_ca.pem -cert
certificado_ca.pem -out certificado_sv.pem -in requisicao_sv.pem

Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for chave_privada_ca.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 257 (0x101)
        Validity
            Not Before: Jun 10 15:12:28 2006 GMT
```

```
            Not After : Jun 10 15:12:28 2007 GMT
        Subject:
            countryName               = BR
            stateOrProvinceName       = Rio de Janeiro
            organizationName          = EB
            organizationalUnitName    = EAD
            commonName                = www.ensino.eb.br
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                3F:8D:0D:CB:0C:9C:23:63:60:7B:E6:D4:51:87:AD:B1:55:AD:2E:67
            X509v3 Authority Key Identifier:
                keyid:F5:8C:CD:05:61:61:38:04:D2:71:D8:5F:90:08:88:E1:EF:D0:E0:9
A
                DirName:/C=BR/ST=Rio de Janeiro/L=Niter\xF3i/O=EB/OU=EAD/CN=CA
EB
                serial:8D:96:24:3B:E8:C3:DC:36

Certificate is to be certified until Jun 10 15:12:28 2007 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

A senha pedida é a senha da chave do CA (para a qual sugerimos 1111). Esqueceu a senha do CA? Se sim, terá que criar outro, retornando à página 102. Usar a seta para cima facilita, pois permite ver e reutilizar comandos anteriores.

Veja como ficou o certificado com o comando:

```
openssl x509 -in certificado_sv.pem -text
```

O arquivo “requisicao_sv.pem” já é uma requisição de certificado que pode ser enviada para uma autoridade certificadora reconhecida mundialmente, para assinar a requisição, de devolvendo-a como um certificado para ser usado no servidor seguro. A diferença entre usar um certificado assinado pelo nosso CA “de fundo de quintal” ou por uma autoridade comercial reside no conseqüente reconhecimento público, ou não, do certificado.

Por exemplo, se um certificado assinado por uma CA interna for usado em um servidor HTTP público, os aplicativos clientes (*browsers*) irão apresentar a janela que indica que o certificado não pode ser validado e pedir ao usuário permissão para continuar a usar a criptografia com este certificado não válido. A Figura 21 mostra um exemplo deste aviso.

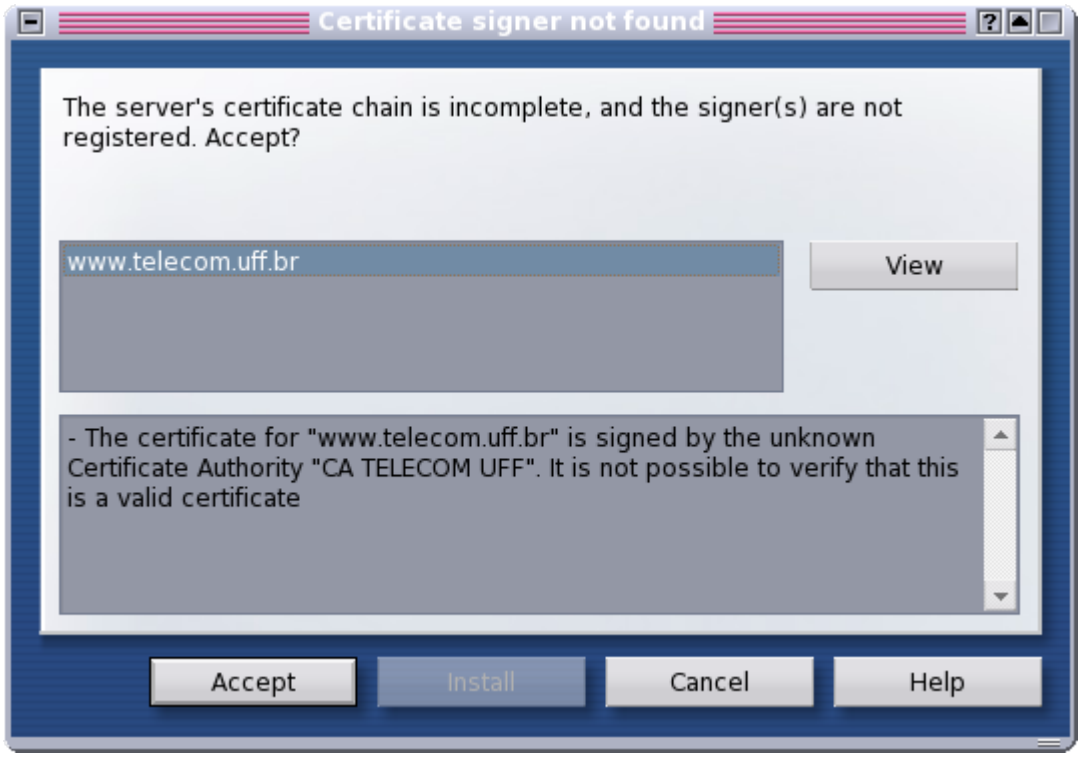

Figura 21 - Certificado assinado por uma CA não reconhecida.

Clicando no botão “View”, será possível verificar os detalhes do certificado.

Reveja a etapa de validação de um certificado, no texto sobre SSL. No uso mais freqüente do SSL, isto é, em servidores HTTPS, a verificação é feita pelo navegador, que possui em arquivo as chaves públicas dos CAs reconhecidos mundialmente, permitindo, assim, a verificação da assinatura de um certificado.

É possível importar a chave pública de um certificado, pois ela é enviada durante a etapa de estabelecimento da sessão SSL. Ao fazer a importação, o usuário estará, com esse ato, dizendo ao navegador que este CA é reconhecido por ele (usuário) e que os certificados assinados por tal CA devem ser considerados válidos. A Figura 22 mostra o mesmo certificado anterior,

após a chave pública da CA ter sido importada no navegador. A janela é conseguida clicando no cadeado fechado que fica na parte inferior direita do navegador (no caso, o Netscape 7.2).

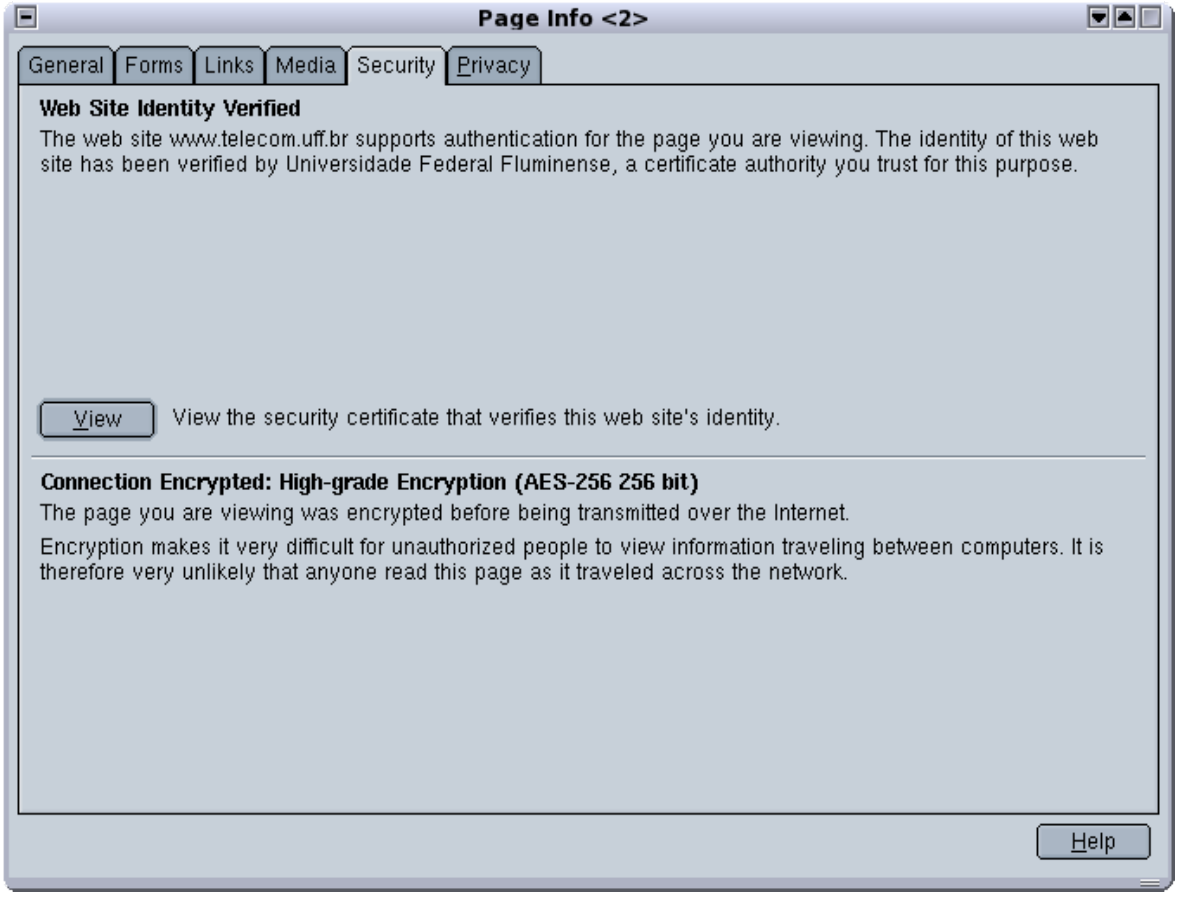

Figura 22 - Certificado assinado por uma CA reconhecida.

Normalmente é possível verificar detalhes do certificado e da CA que assinou o certificado. No exemplo, ao clicar o botão "*View*" (Figura 22), outra janela irá mostrar os detalhes, conforme a Figura 23. Mais detalhes podem ser vistos na aba "*Details*". Contêm alguns dos campos que foram vistos ao examinar o certificado com o comando openssl.

Scripts prontos

Sendo os comandos openssl um tanto complicados para os iniciantes, principalmente para aqueles que não têm o estudo sobre criptografia que

temos, ou não encontram exemplos prontos como os que vimos, fica difícil criar sozinho as linhas de comandos aqui utilizadas.

Figura 23- Detalhes do certificado e do CA.

Então, para popularizar o uso da criptografia com openssl, foram elaborados scripts que facilitam a criação de uma mini PKI, com sua CA e demais certificados. Estes scripts estão no diretório /etc/ssl/misc. Os comandos lá existentes devem ser chamados estando no diretório /etc/ssl (já estamos nele, certo?). Vamos, então, começar tudo de novo, apagando o que foi criado até agora. Você pode executar o comando a seguir ou, estando no CD Live do SLAX, dar um boot, já que nada é gravado em disco.

```
rm -fr *.pem demoCA
```

Agora execute:

```
./misc/CA.pl -h
usage: CA -newcert|-newreq|-newreq-nodes|-newca|-sign|-verify
```

Os parâmetros possíveis, que estão separados pelo *pipe* (barra vertical), são razoavelmente auto-explicativos, não? A seqüência dos próximos três comandos cria uma nova árvore para o PKI e um certificado auto-assinado

para a CA; gera uma requisição de certificado e o assina com a CA.

```
./misc/CA.pl -newca
./misc/CA.pl -newreq-nodes
./misc/CA.pl -sign
```

Por que não foi mostrado antes? Porque não é toda distribuição Linux que vem com esses scripts. Conhecer o comando openssl e saber utilizá-lo é importante.

Os arquivos do certificado estão no diretório corrente, com os nomes “newcert.pem” e “newreq.pem”. O certificado da CA está no diretório demoCA (o diretório foi automaticamente criado), com o nome “cacert.pem” e sua chave, está no diretório demoCA/private com o nome “cakey.pem”.

Exercícios:

1. Verifique e comente as permissões dos arquivos criados.
2. Verifique a requisição, o certificado, a chave da CA e o certificado da CA, com o comando openssl.
3. Crie um certificado assinado pela CA para o servidor mail.empresa.com.br.

Unidade 6 - VPN – Rede Virtual Privada

O assunto tem relevância para o curso, quando se alia VPN à criptografia. Por essa razão, nesta unidade abordaremos protocolos e aplicações práticas, fazendo uma rápida retrospectiva dos protocolos, inclusive alguns antigos, mais utilizados.

VPN – A idéia básica

Facilitar a interligação de sistemas remotos, sem que estes tenham conhecimento da infra-estrutura de comunicação (arquitetura da rede). A segurança pode ser uma premissa, mas nem todos os métodos de VPN possuem suporte à criptografia.

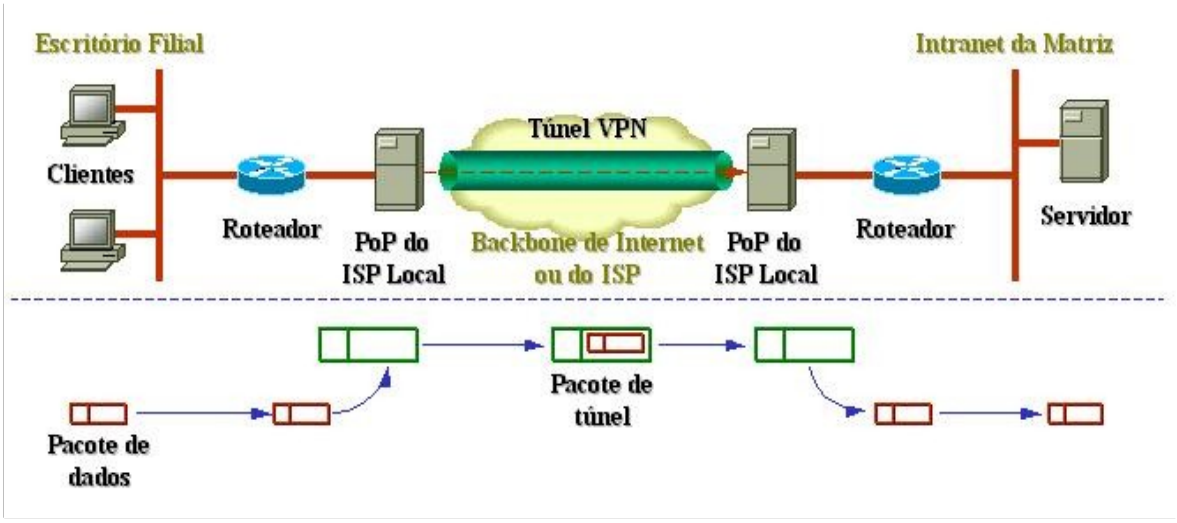

Figura 24 - VPN: a idéia básica.

A Figura 24 mostra essa idéia básica. O “tunelamento” independe do uso ou não de criptografia.

No entanto, o uso de criptografia aumenta a segurança na transmissão de dados e o uso de certificação aumenta a segurança na abertura de sessão, identificando e validando as pontas.

Texto 1 - Protocolos antigos

Alguns protocolos apresentados, embora antigos, são usados até hoje como suporte ou transporte para outros mais atuais.

GRE (Generic Routing Protocol)

É um protocolo muito utilizado como suporte para transporte de outros pacotes.

Os túneis GRE são geralmente configurados entre roteadores fonte e roteadores destino (pacotes ponto-a-ponto).

Os pacotes designados para serem enviados através do túnel, já encapsulados com um cabeçalho de um protocolo – como o IP, por exemplo – são encapsulados por um novo cabeçalho (cabeçalho GRE + IP). Isto forma o túnel. O endereço de destino no novo cabeçalho é IP do final do túnel.

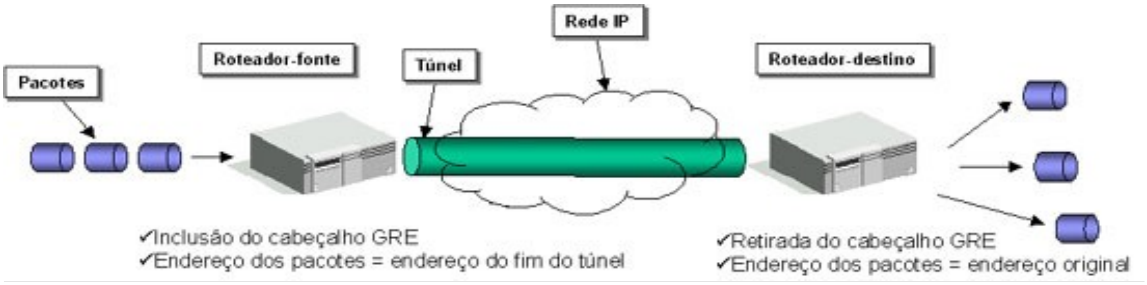

Figura 25 - Túnel GRE.

Ao chegar a este fim, os pacotes são desencapsulados (retira-se o cabeçalho GRE) e continuam seu caminho para o destino determinado pelo cabeçalho original. A Figura 25 ilustra um túnel GRE.

SOCKS

É um protocolo desenvolvido para proxy, que pode ser encarado como VPN não segura. Possui autenticação para o acesso.

Constitui um protocolo que opera em nível mais alto do que os demais, que tipicamente operam no nível dois ou três. Funcionando em nível mais alto,

permite facilmente aos administradores limitar o tráfego na VPN a certas aplicações. No entanto, na época da sua criação, isto foi considerado uma falha de “marketing” pela dificuldade de uso, pois é necessário compilar as funcionalidades do SOCKS nos sistemas (kernel do sistema operacional) e/ou aplicações. Seu uso é bastante comum em sistemas UNIX/Linux, e existem clientes para diversos sistemas, incluindo Windows:

http://archive.socks.permeo.com/cgi-bin/download.pl

O Protocolo SOCKSv5 é um padrão público (RFC-1928) para realizar proxy de rede no nível de transporte. Possui possibilidade de autenticação mesmo quando se utiliza Firewall com NAT (*Authenticated Firewall Traversal* – AFT) e foi criado para melhorar o SOCKSv4.

Como características relevantes, citamos:

- autenticação forte;
- negociação do método de autenticação;
- proxy de resolução de endereços;
- proxy para sessões baseadas em UDP.
- padrões relacionados aos métodos de autenticação:
- usuário/senha para SOCKSv5 (rfc1929);
- GSS-API (*Generic Security Service Application Programming Interface*) para SOCKSv5 (rfc1961).

L2F-Layer-2 Forwarding

Trata-se de um dos primeiros protocolos desenvolvidos especificamente para criar VPNs. Assim como o PPtP (veremos adiante), o L2F foi projetado como um protocolo de tunelamento entre usuários remotos e corporações. Uma grande diferença entre os dois é o fato do PPtP não depender de IP e, por isso, ser capaz de trabalhar diretamente com outros meios físicos/enlace, tais como FRAME RELAY ou ATM.

Utiliza conexões PPP para a autenticação de usuários remotos, mas também

inclui suporte para TACACS+ (lê-se tacacs plus) e RADIUS para autenticação desde o início da conexão (métodos utilizados em sistemas de autenticação com base de dados única, a exemplo do SSO).

Na verdade, a autenticação é feita em dois níveis: primeiro, quando a conexão é solicitada pelo usuário ao provedor de acesso; depois, quando o túnel se forma, o gateway da corporação também irá requerer autenticação.

Os túneis podem suportar mais de uma conexão, o que não é possível no protocolo PPtP.

O L2F também permite tratar outros pacotes diferentes de IP, como o IPX e o NetBEUI por ser um protocolo baseado na camada 2 do modelo OSI.

Sua grande utilização é feita em conexão via linha telefônica discada.

Na etapa de conexão ocorre a negociação do protocolo (Figura 26). Quando um usuário deseja se conectar ao *gateway* da intranet corporativa:

- primeiro estabelece uma conexão PPP com o NAS (Servidor de Acesso a Rede) do ISP (Provedor de Acesso a Internet);
- a partir daí, o NAS estabelece um túnel L2F com o *gateway*;
- finalmente, o *gateway* autentica o nome de usuário e senha do cliente, e estabelece a conexão PPP com o cliente.

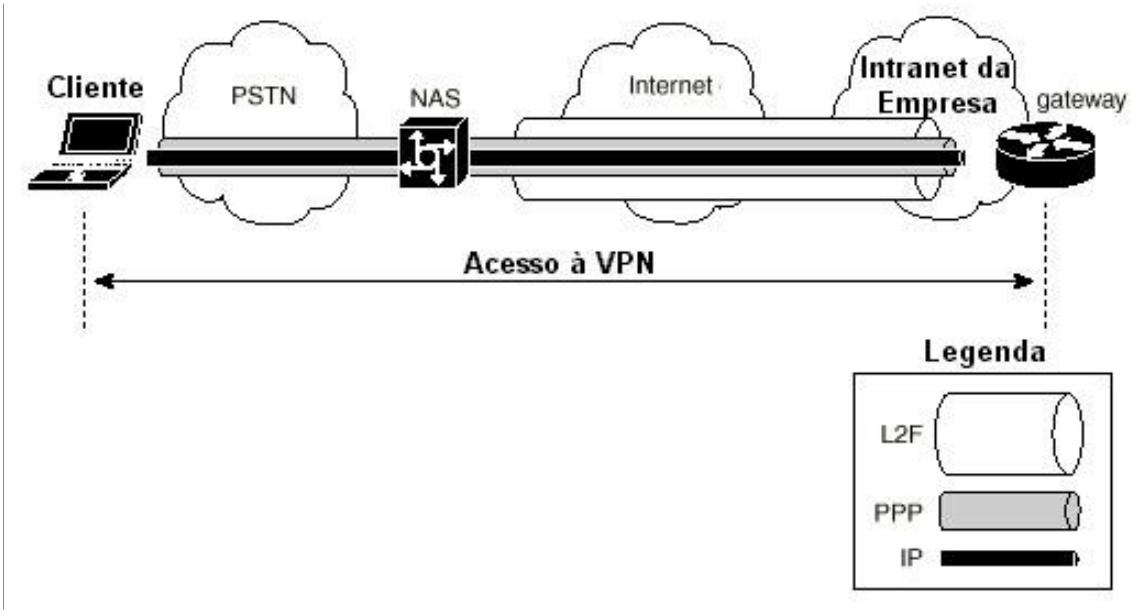

Figura 26 - Túnel L2F.

O NAS do ISP local e o *gateway* da Intranet estabelecem um túnel L2F que o NAS utiliza para encaminhar os pacotes PPP até o *gateway*.

A VPN de acesso se estende desde o cliente até o *gateway*. No entanto, a parte que trafega dentro da PSTN (Rede Pública Telefônica Comutada) não é criptografada.

Vejamos como acontece a autenticação. No início de uma sessão VPN – L2F, o cliente, o NAS e o *gateway* da *intranet* usam um sistema triplo de autenticação via CHAP (*Challenge Handshake Authentication Protocol*), da seguinte forma:

- inicialmente, o NAS contesta o cliente e o cliente responde;
- em seguida, o NAS encaminha esta informação de CHAP para o *gateway*, que verifica a resposta do cliente e devolve uma terceira mensagem de CHAP (sucesso ou fracasso na autorização) para o cliente;
- se tudo correu bem, a conexão e a VPN estão estabelecidas. Caso contrário, a conexão é desfeita e a ligação telefônica é derrubada.

PPtP

O Protocolo de Túnel Ponto-a-Ponto (*Point-to-Point Tunneling Protocol*) foi criado para criar-se uma VPN sobre o protocolo ponto-a-ponto PPP.

O projeto original possui abertura que possibilita o uso de qualquer protocolo para autenticação e criptografia. Parte do projeto foi implementada pela Microsoft no NT e ganhou boa parcela do mercado.

O funcionamento básico é o seguinte: o PPtP encapsula os pacotes da rede virtual dentro de um pacote PPP, que, por sua vez, são encapsulados em pacotes GRE.

Existe um canal de controle em que é iniciada uma conexão pelo cliente, no servidor na porta TCP/1723. Normalmente, essa conexão passa a ser usada como canal bidirecional (requisições e respostas nos dois sentidos).

O PPtP implementado pela Microsoft (MS-PPtP) sofreu alterações (minimizações e simplificações) se comparado ao projeto original e possui as características:

- autenticação – três opções:
  - senha em texto plano;
  - senha modificada - HASH;
  - convite/resposta: A Microsoft usa uma implementação proprietária do CHAP (*Challenge Handshake Authentication Protoco*l) – MS-CHAP;
  - Microsoft Authentication.

O protocolo MPPE (*Microsoft Point-to-Point Encryption*) pressupõe a existência de uma chave secreta compartilhada (40 ou 128 bits) e usa RC4 para criptografar os dados. É negociado por uma opção do Protocolo de Controle de Compressão (CPP) do PPP.

No MS-PPtP, o MPPE só pode ser usado se a autenticação escolhida for a CHAP e o cliente estiver rodando NT ou superior. Alguns clientes Win-95 não conseguem estabelecer sessões criptografadas, se não tiverem instalado o suporte para windows NT hash.

É importante observar que vários pacotes do protocolo PPP, tais como LCP, PAP, CBCP, CHAP e IPCP, não são criptografados pelo MPPE.

Agora, veja como são geradas chaves compartilhadas. A geração da chave compartilhada de 40 bits no PPtP – MPPE é realizada da seguinte maneira: o sistema gera uma chave determinística a partir do *Lan Manager hash* da senha do usuário, usando SHA. Depois, coloca os 24 bits mais significativos em 0xD1269E.

Agora, veja como é gerada a chave compartilhada de 128 bits no PPtP – MPPE: o sistema concatena o NT hash da senha do usuário com um número aleatório gerado durante a negociação do protocolo MS-CHAP. A seguir, o número é enviado para o cliente e ambos geram uma chave determinística de 128 bits usando SHA.

Há uma implementação free do PPtP, compatível com autenticação e criptografia da Microsoft (http://www.poptop.org/). Suas características principais são:

- MSCHAPv2;

- MPPE 40 bit;
- MPPE 128 bit;
- criptografia RC4;
- suporte para múltiplas conexões de clientes;
- integração com ambiente de rede Microsoft (LDAP, SAMBA) com o uso do plugin RADIUS;
- funciona com clientes PPtP Windows 95/98/Me/NT/2000/XP;
- funciona com clientes PPtP Linux.

MS-PPtP – Análise

O NT usa duas funções hash para proteger as senhas:

- *Lan Manager hash*, que foi desenvolvida para o OS2 pela IBM e integrada ao windows for workgroups;
- Windows NT hash, desenvolvida para o NT e com base no MD4 de uma via. É usado em vários protocolos de autenticação no NT (além do PPtP).

O funcionamento do *Lan Manager Hash* segue os passos:

1. Converte a senha do usuário para 14 dígitos, isto é, completa com branco se ela tiver menos de 14 caracteres;
2. Converte todos os caracteres alfabéticos para maiúsculos;
3. Divide a seqüência de 14 caracteres em duas palavras de 7 dígitos (um byte por dígito);
4. Usando cada string de 7 bytes como uma chave DES de 56 bits, criptografa uma constante string com cada chave, obtendo duas strings criptografadas de 8 bytes;
5. Concatena as duas strings obtidas no passo anterior em uma de 16 bytes.

O ataque por dicionário é trivial com captura dos pacotes que transportam a senha digitada, pois o processo é determinístico. As facilidades para isso incluem:

- senhas fáceis (digitadas por seres humanos - usuários);
- conversão para maiúsculo: diminui número de possibilidades;
- não existe semente aleatória (*salt*);
- duas pessoas com a mesma senha produzem a mesma senha hash;
- as duas strings de 7 bytes são computadas separadamente, o que reduz o ataque por força bruta a 7 caracteres, pois a maioria das senhas digitadas possui menos de 8 caracteres. Quando maiores que 7, dificilmente chegam a 14 e a segunda string é facilmente quebrada.

Assim, é fácil precomputar dicionários muito grandes com hash e, após a captura no estabelecimento da conexão, buscar rapidamente o hash correspondente à senha. A velocidade para quebrar uma senha depende unicamente do I/O de disco!

O funcionamento do Windows NT hash tem dois passos:

1. converte de até 14 caracteres para unicode. O NT hash suporta até 128 caracteres, mas o *Lan Manager* limita a senha digitada em 14 caracteres;
2. é feito um hash da senha com MD4, obtendo um hash de 16 bytes.

Embora melhorado, ainda não há *salt* e o ataque a senhas com hash precomputado continua sendo possível.

O ataque ainda é facilitado pois, devido a um erro de projeto e implementação (com a finalidade de manter a compatibilidade com o sistema do LAN Manager hash), os dois hash da senha são enviados. Então, basta descobrir a senha pelo sistema mais simples e, uma vez descobertos quais são os caracteres, refinar o ataque buscando por combinações de maiúsculos e minúsculos.

Já se for utilizado o MS-CHAP, que utiliza oito bytes aleatórios para criptografar a senha, fica impossibilitado o ataque por dicionário reverso. No entanto, o cliente sempre envia o *Lan Manager hash*: trivial!

No MS-CHAP, a senha do cliente é dividida em três campos criptografados separadamente. Isto possibilita um ataque ao próprio MS-CHAP.

Pode-se dizer que o MS-PPtP tem implementação frágil, com um protocolo

falho. Devido a isto, a Microsoft publicou em boletim de segurança que deve-se desligar *Lan Manager* (hash) em todas as máquinas.

O MS-CHAPv2 foi criado para sanar as falhas da versão anterior (v1), porém a segurança continua fortemente apoiada na senha digitada pelo usuário. O sistema ficou bem mais complicado, com funções de hash sendo usadas recursivamente, mas um ataque por dicionário ainda é possível e relativamente fácil.

Por isso, nos casos em que o nível de segurança dos dados é relevante, recomenda-se: o MS-PPtP não deve ser usado!

L2TP-Layer 2 Tunneling Protocol

Foi criado pela IETF (*Internet Engeneering Task Force*, organização que desenvolve os padrões da Internet), com o objetivo de resolver as falhas do PPTP e do L2F. Utiliza os mesmos conceitos do L2F, tendo sido desenvolvido para transportar pacotes por diferentes meios: X.25; frame-relay; ATM e outros protocolos de nível de rede: IPX; NetBEUI; L2TP.

O L2TP cria um túnel “compulsório”. Este túnel é criado pelo provedor de acesso, não permitindo ao usuário qualquer participação.

Analise o procedimento básico na Figura 27.

O usuário disca para o provedor de acesso. De acordo com o perfil configurado para o usuário (SSO) e, ainda, em caso de autenticação positiva, um túnel L2TP é estabelecido dinamicamente para um ponto predeterminado, onde a conexão PPP tem seu ponto final (*end point*).

O L2TP é usado pelas operadoras de telefonia para prestar um serviço de acesso remoto, via linha discada, para empresas, sendo normalmente cobrado.

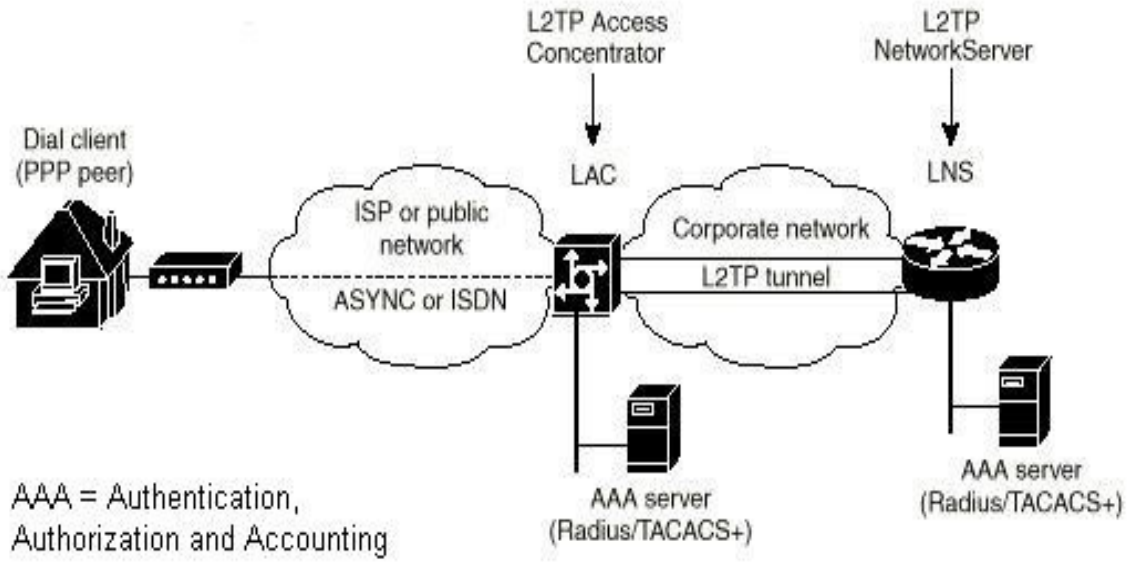

Figura 27 - Túnel L2TP.

O L2TP utiliza autenticação baseada no PPP, devido a este ser o protocolo de enlace usado em conexões discadas. Assim, o L2TP inclui mecanismos de autenticação dentro do PPP: os protocolos PAP e CHAP.

Outros sistemas de autenticação também podem ser usados, como o RADIUS e o TACACS. Porém, o L2TP não inclui processos para gerenciamento de chaves criptográficas exigidas para a criptografia em suas especificações de protocolo. Como conseqüência, a chave simétrica usada deve ser compartilhada “manualmente”, normalmente por uma ligação telefônica entre o administrador do roteador da empresa e o técnico da operadora.

Texto 2 - Túnel SSL

O SSL, aliado aos certificados que acabamos de estudar, permite criar canais seguros de comunicação para quaisquer serviços que utilizem TCP como protocolo de transporte, mesmo que o serviço não possua qualquer suporte a SSL ou criptografia. Isso é possível com a utilização de um software chamado

STunnel (http://www.stunnel.org). Tal software viabiliza a criação de um túnel seguro, criptografado, com chaves assinadas (certificados) que podem, opcionalmente, serem validadas, tanto no servidor quanto nos clientes.

Usando NAT/PAT, pode-se redirecionar uma porta para o STunnel, transmitir criptografado, descriptografar na outra ponta e redirecionar para uma porta ou aplicativo servidor, mesmo que este seja acionado pelo inetd (o STunnel é capaz de chamar um aplicativo como se fosse o inetd).

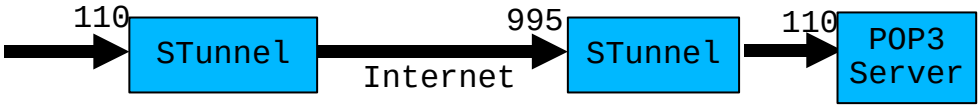

Figura 28 - Criando túneis seguros.

A Figura 28 mostra um exemplo para o serviço POP3 que utiliza, por padrão, a porta tcp/110. Os dois programas STunnel podem estar rodando em qualquer máquina:

- de um lado, perto do cliente (que não aparece, mas estaria à esquerda);
- do outro lado, perto do servidor POP3.

Contudo, será obtida maior segurança se o STunnel à esquerda estiver rodando na mesma máquina do programa cliente POP3, e o STunnel à direita estiver rodando na máquina do servidor POP3. Assim, nenhum dado passará pela rede sem estar criptografado.

Com tal finalidade, o cliente de email é configurado para conectar no servidor localhost (127.0.0.1). Neste endereço, encontrará o STunnel que roda na máquina do cliente. O STunnel cliente está configurado para conectar na porta 995 do servidor, que possui um STunnel rodando ouvindo esta porta. Este STunnel desfaz a criptografia, conectando na porta 110 do servidor. Nem o cliente de email, nem o servidor possuem suporte para criptografia, mas os dados na rede estarão seguros.

Cabe lembrar que o software possui versões para Linux e para Windows.

Um alerta para quem ainda usa sistemas obsoletos: devido a um bug, o NT 4.0 costuma travar a pilha TCP/IP e sair de rede, ao se utilizar o STunnel.

Desafio: baixar, instalar, gerar certificados e usar o STunnel.

Texto 3 - IPSec

Pode-se dizer, resumidamente, que o IPSec (*IP Security Protocol*) é um *framework* (um conjunto de diversas ferramentas, compondo um sistema) de padrões abertos, que visa a assegurar uma comunicação segura em redes IP. Baseado em padrões desenvolvidos pela IETF, o IPSec busca garantir confidencialidade, integridade e autenticidade nas comunicações de dados em uma rede IP pública.

Os padrões do IPSec foram desenvolvidos originalmente para o Ipv6. Mas devido à demora na adoção mundial do IPv6, as idéias de segurança do IPv6 foram exportadas, gerando o que hoje é conhecido como IPsec.

São utilizados os cabeçalhos de extensão específicos do protocolo Ipv6:

- cabeçalho de autenticação (AH - *Authentication Header*);
- cabeçalho de encapsulamento de dados de segurança (ESP - *Encapsulations Security Payload Header*).

Além dessas duas estruturas, o IPSEc utiliza o conceito de associação de segurança (SPI), que permite a comunicação entre duas ou mais entidades comunicantes e descreve todos os mecanismos de segurança a serem utilizados – por exemplo: qual algoritmo e modo de autenticação aplicar no cabeçalho de autenticação, chaves usadas nos algoritmos de autenticação e cifragem, tempo de vida da chave, tempo de vida da associação de segurança, nível de sensibilidade dos dados protegidos, entre outros.

Assim, quando uma entidade deseja estabelecer uma associação de segurança, utiliza um SPI e o endereço de destino da entidade na qual se quer fazer a comunicação segura, e envia essas informações à entidade com que deseja estabelecer o canal seguro. Dessa maneira, para cada sessão de comunicação autenticada serão necessários dois SPIs, ou seja, um para cada sentido, devido ao fato de que a associação de segurança é unidirecional.

Os algoritmos de cifra ou autenticação utilizados não possuem apenas uma estrutura específica. Essa flexibilidade permite que sejam empregadas sempre às normas mais recentes disponíveis, incrementando ainda mais a segurança.

Por padrão, os algoritmos utilizados são HMAC-MDR e HMAC-SHA-1, para autenticação, e DESC-CBC, para a cifra usada no cabeçalho.

É importante destacar que os cabeçalhos usados são de extensão e serão adicionados a um cabeçalho IP. Dessa forma, os encaminhadores (roteadores) do pacote poderão interpretá-los como se fossem parte integrante dos dados, permitindo a utilização de equipamentos que conhecem IP e desconhecem IPsec .

IPSec: Cabeçalho de Autenticação (AH)

Tem por funcionalidade validar a identidade de entidades comunicantes, identificando o emissor e destino corretos para certificar se o emissor é realmente quem diz ser. Como este cabeçalho é apenas adicionado ao datagrama IP, por si só não oferece segurança contra ataques de análise de tráfego ou confidencialidade. Para tratar esse problema, é feita uma ação junto com o cabeçalho de encapsulamento.

IPSec: Cabeçalho de Encapsulamento de Dados de Segurança (ESP)

Tem por finalidade fornecer integridade e confidencialidade aos datagramas IP através da cifra dos dados contidos no datagrama. Existem dois modos de utilização do ESP, ilustrados na Figura 29 e citados a seguir.

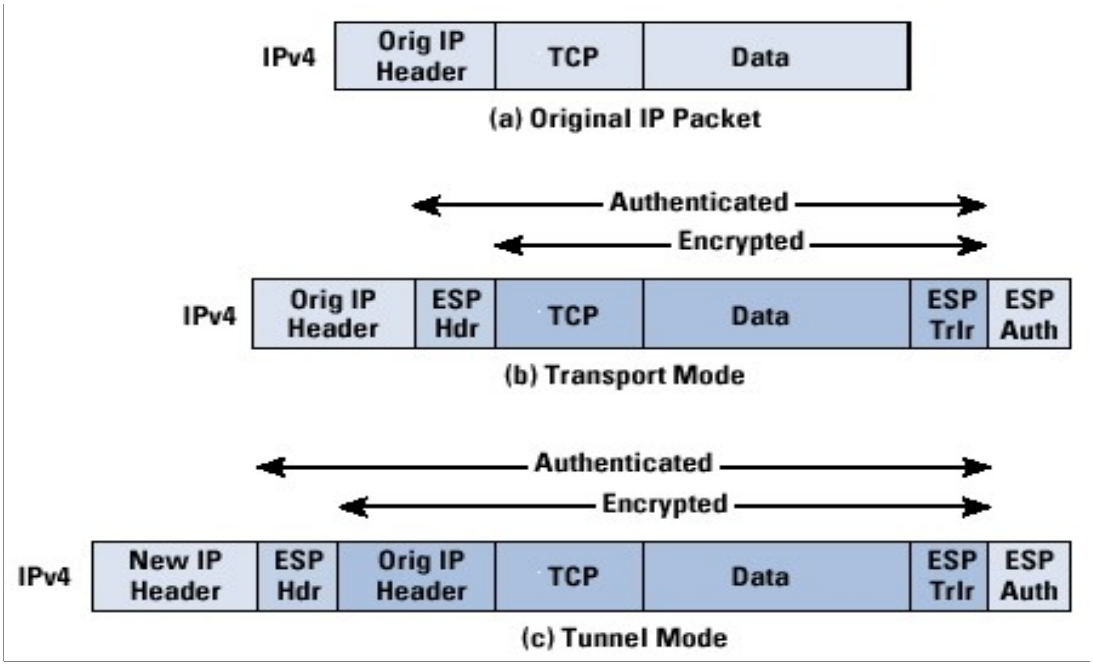

Figura 29 - IPSec - Os novos cabeçalhos.

- Modo de transporte

  Proporciona proteção principalmente aos protocolos da camada superior. Na maioria das vezes, é utilizado em comunicações entre cliente e servidor. A informação do protocolo da camada de transporte é cifrada, adicionando-lhe um novo cabeçalho IP não cifrado. Por isso, se torna vantajoso em redes pequenas, onde os servidores e os nodos existentes implementam IPSec.

- Modo de túnel

  Proporciona proteção ao pacote IP. Para isso, depois da adição dos campos de ESP ao pacote IP, todo o pacote é tratado como o módulo de dados de um novo pacote IP. Deste modo, pode ser usado para enviar dados cifrados através de um túnel, o que permite enviar dados independentemente da infra-estrutura utilizada. Um exemplo de utilização seria o envio de pacotes IP através de canais virtuais criados numa rede IP pública, como Internet, dando, assim, segurança a um grupo de nodos que não implementam IPSec.

Além disso, existem mecanismos de gestão de chaves, responsabilizando-se pela criação, eliminação e alteração das chaves para autenticação e validação de informações. Embora o IPSec original não integre um mecanismo de gestão

de chaves, a IETF definiu como norma de gestão o protocolo híbrido ISAKMP/Oakley, que também é denominado de IKE (*Internet Key Exchange*).

O IKE utiliza a porta UDP/500 para interagir com os demais mecanismos de segurança IPSec, através de associações de segurança para diversos protocolos e outras associações de segurança, possibilitando transparência na associação de diferentes mecanismos de segurança – sem envolver as entidades participantes na comunicação.

O IPSec está amplamente disponível em diversos sistemas operacionais e em sistemas comerciais fechados. No software livre, um bom representante é o OpenSwan (http://www.openswan.org/). O kernel versão 2.6 do Linux já possui suporte para o IPSec, o que facilita seu emprego com o OpenSwan. Para o kernel versão 2.4, o OpenSwan acrescenta os módulos de kernel necessários.

O estudo detalhado do IPSec, através da documentação disponível no site do openswan, fecha com chave de ouro o assunto Criptografia e Segurança de Redes.

Desafios finais:

1. Recomendamos a você instalar o Linux, Slackware, por exemplo, em uma máquina dedicada. Instale, nele, o OpenSwan.
2. Crie um túnel IPSec entre esta máquina e uma outra, que esteja rodando Windows 2000 ou XP, pois ambos possuem suporte a IPSec incorporado. Procure na Internet a documentação necessária.
3. Configurar, gere chaves e certificados (CA e do servidor) e coloque em operação um servidor web seguro (https).

Bons estudos! Bom trabalho!

Referências

CHAPMAN, D. Brent; ZWICKY, Elizabeth D. *Building internet firewalls*. Disponível em: <http://www.unix.org.ua/orelly/networking/firewall/>. Acesso em: 15 jun.-2006.

Índice Alfabético

A
Anti-spam ... 17
Application Gateway ... 14
C
Cadeia de regras ... 40
Certificate Revocation List ... 89
Chain ... 40
Computador de uso genérico ... 17
CRL ... 89
D
Destination NAT ... 56
DMZ ... 9, 19
DNAT ... 56
F
Firewall
  Alta Disponibilidade ... 9
  Packet Filters ... 10
  Stateful Inspection ... 12
FTP
  Modo Passivo ... 11
I
ICP ... 82
ICP-Brasil ... 83
Infra-estrutura de Chave Pública ... 82
INPUT ... 40
INPUT, FORWARD e OUTPUT ... 40
Intranet ... 54
Iptables ... 39
K
Kernel 2.4 ... 17
L
Linux ... 17
N
NAT ... 54
Netfilter ... 39
P
Ping ... 59
PKCS ... 84
PKI ... 82
Postrouting ... 56
Prerouting ... 56
Public Key Cryptography Standards ... 84
Public Key Infrastructure ... 82
R
RFC-1918 ... 52p.
RFC-3330 ... 52
Riscos ... 9
Rota ... 22

S.....
SNAT..... 56
Source NAT..... 56
T.....
Traceroute..... 61
TTL..... 61
V.....
Vulnerabilidade..... 18
Z.....
Zona desmilitarizada,..... 18
Zona neutra..... 18
Á.....
Área desmilitarizada..... 9

Marcos Tadeu von Lützow Vidal

Engenheiro elétrico com ênfase em Telecomunicações, pela Universidade Federal Fluminense – UFF, em 1988.

Mestre em Engenharia Elétrica/Redes de Computadores, pela Universidade Federal do Rio de Janeiro – UFRJ, em 1994.

Professor do Departamento de Engenharia de Telecomunicações da UFF, desde 1988.

Presidente do Grupo de Segurança da Informação, do Núcleo de Tecnologia da Informação e Comunicação – NTI, da UFF.

Consultor de empresas na área de redes e segurança da informação.