Revista de
Informática
Teórica e Aplicada

Segurança na Internet

Raul Fernando Weber
weber@inf.ufrgs.br*

Abstract

This tutorial presents the security aspects of computer networks in general and the Internet in particular. The basic communication mechanism is reviewed, and the principal Internet services (Telnet, FTP, E-mail, WWW and others) are discussed from their security point of view. The most disseminated form of attacks are briefly presented, and also the currently used countermeasures, as tcp-wrappers, firewalls, authentication mechanisms, intrusion detection, malicious fault detection and cryptographic methods.

Resumo

Este tutorial apresenta os conceitos básicos de segurança em redes de computadores, sendo especificamente orientado para os serviços mais comumente utilizados na Internet. Discute-se o mecanismo básico de comunicação via TCP/IP, os principais serviços (Telnet, FTP, correio eletrônico, WWW e outros), os principais tipos de ataques e quais as medidas utilizadas atualmente para aumentar o grau de segurança destes serviços. Entre estas medidas, destacam-se o uso de wrappers, firewalls, mecanismos de autenticação, mecanismos de detecção de falhas intencionais e métodos criptográficos.

Palavras-chave: Segurança, Internet, TCP/IP, Unix, Firewall, Autenticação, Criptografia

* Instituto de Informática-UFRGS, Caixa Postal 15064, 91501-970 Porto Alegre, RS

1. Introdução

Historicamente, a Internet nasceu em 1969, quando a DARPA (Defense Advanced Research Projects Agency) patrocinou um projeto de interconexão dos computadores das principais instituições de pesquisa, ensino e governamentais. Esta rede ficou conhecida como Arpanet, e fornecia os serviços básicos de correio eletrônico (*e-mail*) e transferência de arquivos. Durante a década de 70, a DARPA patrocinou pesquisas em protocolos de comunicação, o que resultou no TCP/IP, ou Transmission Control Protocol/Internet Protocol. Para difundir o uso destes protocolos, foi permitido que a Universidade da Califórnia incluísse o TCP/IP na sua implementação do Unix. Desde então, a Arpanet tem se expandido e passou a englobar diversas outras redes, recebendo, desde o fim da década de 80, a denominação de Internet.

Atualmente, assim como desde o seu início, a Internet é fortemente baseada em Unix e TCP/IP. O restante desta seção apresenta um breve resumo de como o endereçamento é realizado no TCP/IP, qual sua segurança e alguns dos principais arquivos de configuração do Unix. Para maiores detalhes sobre a estrutura da Internet, recomenda-se [18].

1.1 Estrutura da Internet

Os serviços da Internet estão baseados em uma arquitetura cliente/servidor, onde o programa cliente é responsável pela interação com o usuário, e o programa servidor é responsável pela execução das tarefas requisitadas pelo cliente. Normalmente, cliente e servidor executam em máquinas distintas, embora nada impeça que estejam em uma mesma máquina. Qualquer computador conectado à Internet pode atuar tanto como cliente como servidor, bastando para isto ter instalado os programas necessários, como por exemplo um *browser* de WWW, um leitor de *e-mail*, um *daemon* para servidor de FTP, etc.

A comunicação entre os computadores conectados à Internet é realizada através do protocolo IP (Internet Protocol), que é a base de 4 outros tipos de protocolos: TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ICMP (Internet Control Message Protocol) e IGMP (Internet Group Management Protocol).. O TCP é orientado a conexão e garante uma transferência confiável de dados, implementando mecanismos para recuperação de dados perdidos, recebidos fora de seqüência e/ou danificados. O UDP não é orientado a conexão, e não oferece garantias de entrega dos dados. A utilização de um ou outro protocolo depende das necessidades da aplicação. Os outros dois são utilizados para controle e operações básicas, e não serão mais discutidos neste artigo.

O protocolo IP está atualmente na sua quarta versão, o IPv4, em uso desde 1982. As crescentes necessidades de endereçamento e segurança impulsionaram o desenvolvimento da próxima versão, o IPv6 ou IPng (*next generation*). Observe-se que o IPv5 é um protocolo experimental e nunca foi utilizado na prática. No protocolo IP, os dados são

transmitidos em blocos denominados datagramas, ou, de forma mais coloquial, pacotes (*packets*). Como pode ser visto na figura 1, cada pacote contem um cabeçalho fixo (*header*) e um campo de dados (*contents*). O cabeçalho descreve, entre outras informações, quais os computadores de origem e destino, tipo de serviço, comprimento do pacote, etc. Depois que os pacotes alcançam seu destino, eles são remontados, para formar novamente um fluxo contínuo de dados; o processo de fragmentação e remontagem são transparentes ao usuário. Como comumente existem diversas rotas entre origem e destino, cada pacote pode percorrer uma rota diferente e levar um tempo distinto para alcançar seu destino. Note-se que não existe nenhum mecanismo implícito de segurança, e o conteúdo do pacote é visível em todos os computadores pelos quais o pacote circula.

Bits

0 – 4	4 – 8	8 – 12	12 – 16	16 – 20	20 – 24	24 – 28	28 – 31
Versão	IHL	Tipo de serviço	Comprimento total				
Identificação				Flags	Deslocamento		
Tempo de vida		Protocolo	Checksum do cabeçalho				
Endereço de origem							
Endereço de destino							
Opções							
Dados							

Figura 1 - Pacote e cabeçalho IP

Existem três maneiras distintas de dois computadores serem conectados utilizando-se IP:

- Os computadores estão conectados na mesma rede local. Duas redes comuns são Ethernet e Token Ring. Os pacotes IP são enviados na rede local, muitas vezes concorrentemente com outros pacotes, como Novell IPX ou Appletalk.
- Dois computadores podem ser interligados diretamente através de uma linha serial. Neste caso, os pacotes IP são enviados usando ou SLIP (Serial Line Internet Protocol), CSLIP (Compressed SLIP) ou PPP (Point-to-Point Protocol).
- Os pacotes IP podem ser encapsulados dentro de outros pacotes, como por exemplo ATM (Asynchronous Transfer Mode).

1.2 Endereços na Internet

Cada máquina conectada a Internet deve ter o seu próprio endereço, o número IP, constituído de 32 bits, agrupados em 4 octetos. Este endereços são descritos tipicamente como *a.b.c.d* onde *a, b, c* e *d* são números entre 0 e 255. Existem atualmente cinco tipos "clássicos" de endereçamento (classes A, B, C, E e E) e os endereços CIDR (Classless InterDomain Routing, que utilizam os k bits mais significativos para identificar a rede, e os

32-k bits restantes para identificar o computador). O método CDIR pode ser combinado com o esquema clássico, e foi definido como uma maneira de contornar a crescente falta de endereços IP.

Com o uso de classes, formam-se "buracos" de endereços, pois uma empresa raramente ocupa todos os endereços disponíveis. Para minimizar o efeito deste problema, a nova versão do IP reserva 128 bits para o endereço.

Com base nos endereços IP, os pacotes são roteados pela Internet, passando por diversas máquinas até atingir seu destino. Esta tarefa não é realizada de forma global, mas distribuída por diversas máquinas denominadas de *gateways* e roteadores. Cada uma analisa o endereço destino e, com base em tabelas, determina qual o próximo nó da rede que deve receber o pacote, levando-o assim cada na direção do computador destino.

Naturalmente, números são difíceis de serem lembrados ou manipulados pelos usuários, ainda mais se considerarmos o crescente números de pessoas leigas que se conectam à Internet. Nomes são mais fáceis de serem usados por pessoas, e assim no lugar de endereços IP costumam-se usar nomes para os computadores (*hostnames*). Cada *hostname* tem duas partes: o nome da máquina e o nome do domínio. Assim, por exemplo, o nome sirius.inf.ufrgs.br identifica a máquina sirius dentro do domínio do Instituto de Informática da UFRGS (que por sua vez pertence ao domínio da UFRGS, que está localizado no domínio do Brasil). A tradução entre nomes e números é realizada pelo DNS (Domain Name Service).

Utilizar nomes ao invés de endereços IP possibilita uma grande facilidade no uso dos serviços que o empregam. Ele elimina a necessidade de que usuários memorizem seqüências de números (endereços) em favor de nomes simbólicos, diminuindo erros e aumentado produtividade. Com o uso de serviços de nomes, o sistema fica mais fácil de ser administrado, pois possibilita a troca de máquinas e de suas configurações sem prejudicar o funcionamento normal da organização. Esta grande flexibilidade permite a troca de serviços de uma máquina para outra, a substituição de computadores por outros, o remanejo de serviços, etc, através de uma única alteração local, sem a necessidade de uma atualização global na Internet. Suponha-se, por exemplo, que uma empresa disponibilize para seus clientes acesso a uma determinada máquina chamada *base.empresa.com.br*. Por questões de performance, o administrador resolve trocar o serviço dos clientes para outro computador. Caso não fosse utilizado um serviço de nomes, todos os usuários deveriam saber desta alteração e conhecer o novo endereço IP para acesso. Entretanto, com serviços de nomes, esta alteração será transparente e tudo o que o administrador deve fazer é trocar o mapeamento do nome para o novo endereço IP.

Note-se ainda que um único computador pode ter mais de um endereço IP, e que um único endereço IP pode estar associado a mais de um *hostname*.

1.3 Protocolos TCP e UDP

O protocolo TCP fornece uma comunicação segura, ordena e confiável entre duas máquinas. Cada conexão TCP utiliza duas portas (*ports*), identificada por números de 16 bits. Assim, cada conexão é identificada por dois números de 32 bits (números IP de origem e destino) e dois de 16 bits (as portas na origem e no destino). As portas de 0 a 1023 são reservadas no UNIX, e um grande número de serviços utiliza números padrões de portas (veja-se seção 2).

O protocolo IP usa dois bits do cabeçalho, SYN e ACK, para marcar o início de novas conexões. Para iniciar uma conexão, o computador envia uma pacote com SYN ligado e ACK desligado. O computador destino responde ao pedido enviando um pacote com ambos bits ligados, e o computador que iniciou a conexão responde agora com ACK ligado e SYN desligado. Todos os demais pacotes possuem ACK ligado. Este processo permite distinguir entre os pacotes que iniciam uma conexão e os que pertencem a uma conexão já estabelecida. Esta propriedade é útil quando se deseja implementar *firewalls* baseadas em filtros de pacotes (ver seção 4), mas também permitem a um intruso detectar o início de uma sessão e eventualmente capturar senhas ou até mesmo toda a sessão.

Serviço	Protocolo	Porta
Systat	TCP	11
File Transfer (FTP)	TCP	20, 21
Telnet	TCP	23
Simple Mail Transfer (SMTP)	TCP	25
Access Control (TACAS)	UDP	49
Domain Name (DNS)	TCP, UDP	53
Trivial Transfer (TFTP)	UDP	69
Finger	TCP	79
HyperText Transfer (HTTP)	TCP	80
Post Office (POP)	TCP	109,110
Portmapper	TCP, UDP	111
Identification (auth)	TCP	113
Network News (NNTP)	TCP	119
Network Time	UDP	123
Simple Management (SNMP)	UDP	161,162
Remote Exec (rexec)	TCP	512
Remote login (rlogin)	TCP	513
Remote shell (rsh)	TCP	514
Routing (RIP)	UDP	520
X Window	TCP	6000-6063

Figura 2 - Serviços UNIX

Os serviços Internet que necessitam de um grande volume de transmissão de dados utilizam TCP. Assim, os serviços de transferência de arquivos, correio eletrônico, terminais remotos, etc utilizam TCP. Por outro lado, o protocolo UDP, apesar de não possuir a robustez do TCP, fornece uma taxa de transferência até 10 vezes maior, por apresentar menor *overhead*. O UDP é utilizado em serviços locais, ou serviços onde a perda de informação não é muito relevante (como *talk*, por exemplo). Assim como o TCP, o UDP também utiliza portas de 16 bits. A Figura 2 mostra alguns dos principais serviços do UNIX que utilizam TCP e UDP, assim como suas respectivas portas.

1.4 Segurança

A Internet foi projetada visando fornecer conectividade entre computadores para uma comunidade restrita de usuários que confiavam mutuamente entre si. Ela não foi projetada para um ambiente comercial, para tráfego de informações valiosas ou sensíveis, ou para resistir a ataques mal-intencionados. Durante a década de 80, antes da popularização da Internet, os computadores foram alvos de ataques individuais e isolados. A solução adotada foi relativamente simples: incentivar os usuários a escolherem boas senhas, prevenir o compartilhamento indiscriminado de contas e arquivos e eliminar os *bugs* de segurança de programas como *sendmail, finger* e *login* à medida que eles iam sendo descobertos.

Na década de 90, entretanto, os ataques estão se tornando mais sofisticados e organizados:

- Senhas e outras informações importantes são capturadas por *network sniffers*.
- Computadores são invadidos ou paralisados por ataques do tipo *IP spoofing*.
- Sessões são desviadas através de *connection hijacking*.
- Dados são comprometidos pela inserção de informação espúria via *data spoofing*

Estes ataques são diretamente relacionados ao protocolo IP que não foi projetado para o ambiente atual da Internet:

- Embora projetado para ser tolerante à falhas de hardware, o IP não possui muita resistência contra ataques intencionais.
- O IP não foi projetado para fornecer segurança; assumiu-se que esta tarefa seria realizada por protocolos de maior nível de abstração.
- IP está em permanente evolução; futuras versões provavelmente fornecerão a segurança e a confiabilidade requeridas. Esta caraterística, entretanto, também tem suas desvantagens, uma vez que o IP está sendo usado em ambientes para os quais não foi originalmente projetado.

Atualmente, a única maneira segura de proteger os pacotes contra espionagem é utilizando criptografia. Existem vários métodos:

- Cifragem a nível de *link*: os pacotes são automaticamente cifrados quando enviados por um canal inseguro. Esta prática é comum para enlaces de rádio, mas quase não são utilizadas em Ethernet ou FDDI.
- Cifragem entre origem e destino: os pacotes são cifrados pela máquina origem e decifrados somente na máquina destino.
- Cifragem a nível de aplicativo: a cifragem é realizada pelo próprio programa aplicativo

Infelizmente, a cifragem dos dados não resolve todos os problemas. Ainda existem muitos problemas causados pela falta de um mecanismo de autenticação robusto. Atualmente utiliza-se o DNS para verificar a validade de nomes e números IP. Muitos programas realizam uma operação reversa de DNS, para autenticar se um determinado número IP corresponde a um domínio registrado. Programas mais sofisticados realizam uma dupla verificação, traduzindo o nome obtido novamente para um número IP e comparando-o a seguir com o número original.. Esta autenticação não é confiável, pois pode ser iludida por vários ataques, como por exemplo:

- *Host flooding*: um computador pode ser paralisado através de um grande número de requisições (principalmente se utiliza UDP). Assim, um atacante pode paralisar um DNS e fornecer uma resposta falsa no lugar.
- Informação falsa: muitos DNS armazenam em uma cache qualquer resposta que recebam, quer seja requisitada ou não. Um atacante pode assim carregar uma série de endereços incorretos para serem utilizados em futuras requisições.
- Servidores falsos: o simples fato de uma máquina executar DNS não significa que pode-se confiar nos resultados. Hoje em dia conectar uma máquina a Internet é uma tarefa simples e que pode ser executada até por um PC comum.

A falta de um mecanismo efetivo de autenticação permite que atacantes possam alterar ou falsificar a origem de diversas conexões. Isto é particularmente efetivo para correio eletrônico, serviços de News e WWW.

2. Serviços da Internet e suas vulnerabilidades

A maioria dos serviços do UNIX são fornecidos por programas denominados *servers* ou *daemons*, que são associados a uma porta e a um protocolo. Muitos destes servidores estão sempre ativos, e normalmente estão associados a serviços que manipulam uma grande quantidade de informação ou necessitam fornecer uma resposta rápida. Exemplos destes casos são o *sendmail*, responsável pelo correio eletrônico, e o *nfsd* (network file system daemon).

Entretanto, a medida que o número de serviços da rede crescia, tornou-se impraticável manter um programa executando para cada serviço. Isto levou ao desenvolvimento de um único programa, p *inetd* (Internet daemon), que examina o tráfego

em várias portas e ativa o programa necessário quando uma conexão é solicitada. Um trecho típico do arquivo /etc/inetd.conf é visto na figura 3.

O primeiro campo especifica o nome do serviço, que é mapeado para portas através do arquivo /etc/services. Os dois campos seguintes indicam se a conexão é por TCP ou UDP. O quarto campo especifica se um novo servidor deve ser iniciado a cada requisição (nowait) ou se o servidor é único e o cliente deve esperar até ele estar livre (wait). O quinto campo indica sob qual usuário o serviço é executado. Note-se que um grande número de serviços executa sob privilégios de root, o que pode causar problemas se o serviço está mal configurado ou se um atacante consegue subverter o serviço. O último campo indica a localização do executável a ser ativado e quais parâmetros devem ser passados ao serviço.

```
telnet      stream    tcp    nowait    root      /etc/telnetd telnetd
ftp         stream    tcp    nowait    root      /usr/etc/in.tcpd ftpd
name        dgram     udp    wait      root      /usr/etc/in/tnamed in.tnamed
finger      stream    tcp    nowait    nobody    /usr/etc/in.tcpd in.fingerd
daytime     stream    tcp    nowait    root      internal
pop         stream    tcp    nowait    root      /usr/local/etc/popper popper
nntp        stream    tcp    nowait    root      /etc/nntpd nntpd
```

Figura 3 - Exemplo (parcial) do arquivo /etc/inetd.conf

O *inetd* não possui nenhuma caraterística de segurança, pois ele simplesmente identifica o serviço que está sendo requisitado e inicia o servidor correspondente. Qualquer requisito de segurança deve ser implementado no servidor propriamente dito. Um novo *daemon* está em desenvolvimento, o *xinetd*, que permite, entre outras caraterísticas:

- bloquear conexões provenientes de um número ou de uma faixa de números IP.
- permitir acesso somente a usuários autorizados
- rejeitar conexões fora de horários especificados
- registrar dados sobre conexões realizadas e conexões rejeitadas (como por exemplo números IP, horários de início e fim, nome do usuário, etc).

Outras técnicas de controlar o acesso aos serviços incluem o uso de *tcpwrappers* e *firewalls* (veja-se seção 4).

Para uma correta configuração dos serviços é necessário entender seu funcionamento e estar ciente de seus pontos críticos. Entre todos os serviços suportados pelos protocolos TCP/IP, existem aqueles que devem merecer atenção especial por parte dos administradores, visto que são mais visíveis por usuários externos ao domínio e, portanto, mais expostos a ataques. Existem outros que são empregados somente no interior do domínio, sendo o seu impacto na segurança global muito pequeno. A seguir são examinados alguns dos serviços padrões de um sistema UNIX, com uma análise do seu impacto na segurança do sistema como um todo. Note-se que cada serviço tem suas

características próprias no tocante a segurança, e que sempre existe o risco de que alguma característica ainda não explorada ou conhecida possa vir a se tornar um risco sério ao sistema. A lista a seguir não pretende ser completa ou exaustiva, ela visa somente ilustrar os serviços mais comuns. Para uma análise mais completa recomenda-se consultar bibliografia específica [3, 4, 9, 10, 14].

2.1 Telnet

O serviço de telnet permite que usuários acessem sistemas remotamente, como se estivessem utilizando um terminal diretamente conectado a ele. O telnet somente emula um terminal e não uma estação gráfica e assim só permite acesso a aplicações baseadas em texto. Atualmente o telnet não é considerado um serviço seguro, pois todos os dados enviados e recebidos durante um sessão não são cifrados, sendo vulneráveis à qualquer tipo de escuta nos meios de comunicação ou em máquinas não confiáveis. Particularmente vulneráveis à esta escuta são os dados de identificação e autenticação do usuário, como nome e senha (*password*). Deste modo a utilização do telnet sem qualquer mecanismo extra de segurança não é aconselhável, pois o fluxo de dados pode passar por inúmeras redes intermediárias, nas quais não se pode confiar.

Uma forma de se evitar este tipo de problema é alterar o mecanismo de autenticação e utilizar um método de *one-time password*, no qual para cada conexão uma nova senha será requisitada. Com isto, senhas capturadas por atacantes não poderão ser reutilizadas. Note-se que com este mecanismo as senhas estarão seguras, mas o resto da sessão ainda estará desprotegida e dados importantes que circulam pela conexão poderão ser capturados. Para proteger integralmente um sessão telnet, todos os dados devem ser cifrados.

Outro perigo referente ao uso do telnet é a técnica chamada *session hijacking* utilizada por atacantes para apoderar-se de sessões que estão ativas. Intrusos esperam até que o usuário legítimo acesse o sistema, e então dominam o controle da sessão, comprometendo assim a máquina remota. Existem algumas técnicas para realizar o "roubo" de uma sessão Telnet, entre elas a utilização de uma ferramenta desenvolvida especialmente para este fim. Para maiores detalhes, pode-se consultar os CERT Advisories em http://www.cert.org.

2.2 FTP

O File Transfer Protocol (FTP) é utilizado para transferência de arquivos entre duas máquinas. Pode ser usado para qualquer tipo de arquivos, incluindo executáveis, gráficos, texto, sons, vídeos e animações. Ele pode ser utilizado para duas razões: permitir que os seus usuários internos acessem arquivos remotos ou permitir que usuários remotos acessem dados internos.

Do ponto de vista de segurança, utilizar FTP para acessar arquivos remotos não representa um grande risco, exceto pelo fato de poderem ser buscados arquivos com vírus até mesmo programas piratas, consumindo assim recursos de conexão e tempo de processamento. Por outro lado, permitir que pessoas externas acessem informações internas pode ser mais perigoso. Existem dois tipos de acesso FTP para esta finalidade: *user ftp* e *anonymous ftp* (FTP anônimo). O *user ftp* é usado por usuários já cadastrados no sistema, devendo estes fornecerem sua identificação e senha para efetuar uma conexão. Já o FTP anônimo permite que qualquer pessoa consiga acessar dados em uma área restrita do sistema. Esta área geralmente contém artigos técnicos, informações sobre produtos, softwares de domínio público, e tudo aquilo que se deseja compartilhar com o mundo.

O *user ftp* pode ser considerado um método seguro para transferência de arquivos. Entre as suas fraquezas está a transmissão de senhas sem criptografia, assim como no telnet Durante a sua configuração é importante que se retire permissões de acesso para contas não relacionadas com pessoas (*root, uucp, news, lib*) pois estas dispõem de privilégios especiais e não devem ser usadas diretamente.

Para que uma pessoa tenha acesso a um serviço FTP anônimo, basta que se identifique como o usuário "*anonymous*" e que forneça seu endereço completo como senha. A principal preocupação é permitir que apenas arquivos que se deseje compartilhar sejam compartilhados. Quando configurando o servidor de FTP, deve-se limitar o acesso somente para a área pública, não permitindo que o visitante consiga visualizar qualquer outra informação. Caso o usuário consiga sair da área pública e acessar arquivos de configuração do sistema, tal como o arquivo de senhas, ele disporá de informações preciosas para futuramente lançar um ataque.

Entretanto, uma das maneiras mais usuais de usuários anônimos terem acesso a arquivos confidenciais é obtê-los diretamente da área pública. Estes arquivos são depositados incorretamente nesta região por usuários internos, que desejam compartilhá-los com apenas um pequeno grupo, acreditando estarem executando uma operação segura. Para que se possa contornar esta situação é recomendável que somente uma pessoa autorizada tenha direitos para gravar arquivos na área pública do sistema. Esta pessoa, na maioria das vezes o administrador, recebe pedidos dos usuários, e, depois de uma análise, pode aceitá-los ou negá-los.

Uma das mais importantes regras para se configurar o serviço FTP anônimo é garantir que nenhum arquivo ou diretório na área pública seja possuído pela conta *ftp*, pois o FTP anônimo executa sob este "usuário". Se qualquer um destes diretórios for possuído pelo *ftp* e não for protegido contra gravação, usuários anônimos poderão, sem nenhum esforço, alterar o ambiente do FTP, removendo, substituindo ou criando novos arquivos. Com estes poderes, um visitante poderia escrever no diretório do *ftp* um arquivo *.rhosts*, e após executar um *shell* remoto com o usuário *ftp*, ou substituir executáveis por cavalos-de-tróia.

Diversos domínios possuem diretórios com permissão de escrita na área de FTP anônimo, lugar onde um visitante pode escrever ou ler arquivos, conforme a sua necessidade. Entretanto estes diretórios devem ser utilizados com parcimônia, pois é grande a probabilidade que sejam utilizados pelo *underground* para repositório e distribuição de softwares piratas ou material pornográfico. Segundo [3] existem várias maneiras de se contornar este problema:

- verificar se realmente é necessária a área para escrita na região do FTP anônimo. Outros mecanismos para transferência de arquivos podem ser utilizados em substituição ao FTP, entre eles o correio eletrônico.
- remover todos os arquivos da área para escrita periodicamente, impossibilitando o atacante de utilizar o computador como repositório. Uma alternativa a este mecanismo é mover todos os arquivos que foram gravados na área de escrita para um diretório não público onde poderão ser analisados. Deste modo o administrador do sistema poderá saber se o seu site7 está ou não sendo utilizado para fins não legítimos.
- criar diretórios de escrita escondidos. Neste caso, usuários legítimos gravam seus dados em diretórios escondidos, cujos nomes foram previamente acertados entre usuário e administrador. Assim, estes diretórios não poderão ser abusados por atacantes, pois eles desconhecem sua existência.
- permitir que arquivos gravados na área de escrita possam ser lidos apenas pelo administrador. Com isto, a máquina não poderá ser utilizada como distribuidor de material dos atacantes, pois uma vez gravado os arquivos, não poderão ser lidos.

FTP anônimo pode ser um serviço valioso se bem configurado e administrado. Assim, é importante que, durante sua configuração, exista um guia no qual o administrador possa se basear. Os tópicos seguintes são recomendações do CERT descritos em http://www.cert.org:

- usar a versão mais recente do *daemon* FTP.
- o diretório raiz do FTP anônimo (~/ftp) e seus subdiretórios não devem ser possuídos pela conta ftp. Este é um problema de configuração muito comum.
- utilizar versões modificadas dos arquivos *passwd, group* na área do FTP anônimo.
- utilizar algum mecanismo para tratar com subdiretórios liberados para escrita.

2.3 Correio eletrônico

O correio eletrônico é, sem dúvida, um dos mais importantes serviços utilizados na Internet, mas é também um dos mais vulneráveis. Ele pode ser alvo de uma grande variedade de ataques.

Com os mecanismos de correio eletrônico disponíveis atualmente, forjar uma mensagem pode ser considerada uma tarefa trivial. Atacantes podem abrir uma conexão

TCP em uma máquina alvo e manualmente digitar protocolos e enganar servidores. Deste modo, não é possível confiar-se na veracidade do remetente ou do conteúdo da mensagem. É preciso ter mecanismos em um nível superior para obter-se privacidade e autenticação do remetente. Um exemplo deste mecanismo é o PGP, implementado por Philip Zimmerman [8].

Mensagens falsas abrem caminho para dois tipos de ataques: contra a reputação e de engenharia social. Ataques contra a reputação são geralmente lançados por empregados demitidos ou descontentes que desejam se vingar de sua companhia. A imagem da empresa pode, então, ser denegrida através de mensagens forjadas contendo opiniões que não condizem com a realidade. Através de engenharia social, mensagens fraudulentas podem ser utilizadas para enganar usuários, que, como resposta, podem executar alguma ação ou fornecer alguma informação para o atacante.

Existe, hoje em dia, uma grande preocupação com a forma em que o correio eletrônico é utilizado. Houve, nos últimos anos, um acréscimo considerável no número de mensagens trocadas entre pessoas. Entretanto, elas ainda desconhecem as vulnerabilidades e os perigos do serviço e o utilizam inadvertidamente. Mensagens de correio eletrônico, como qualquer outro dado que percorre a Internet, são sujeitas a interceptação. Devido à natureza do serviço, mensagens são armazenadas em computadores intermediários, e não só roteadas através destes computadores Assim, é necessário que usuários empreguem o serviço com muitas ressalvas, não devendo utilizá-lo para transmissão de informações confidenciais. Educação e conscientização são as soluções para estes problemas.

O sistema de correio eletrônico se compõe de três módulos, cada um deles com suas próprias vulnerabilidades.

O servidor (*server*) é encarregado de se comunicar com outros servidores e comandar o recebimento e a entrega das mensagens. Para realizar seu trabalhado é necessário que aceite comandos provenientes de computadores externos, sendo, assim, passível de ataques onde o atacante envia comandos ao servidor, alterando o seu funcionamento normal, e utilizando falhas de implementação conhecidas ou de comandos especiais não documentados que poderão dar-lhe algum tipo de acesso.

O agente de entrega (*delivery agent*) coloca as mensagens recebidas pelo servidor nas caixas de correio correspondentes. Para isto, necessita de permissões especiais já que precisa gravar nos diretórios dos usuários e, por isso, é constantemente alvo de atacantes. Como o agente de entrega não recebe comandos, não pode atacado como o servidor. Entretanto o conteúdo das mensagens pode trazer perigos, através de ataques *data-driven*. Este ataque permite, por exemplo, que um intruso execute comandos no sistema alvo apenas enviando mensagens de correio eletrônico especialmente formatadas. O agente de entrega, se mal configurado, interpreta o corpo da mensagem e a entende como uma solicitação de execução de comando vinda do usuário.

O agente do usuário (*user agent*) permite que o usuário leia suas mensagens e que componha outras que serão postadas. O agente do usuário executa como um usuário comum, ou seja, com poder e acesso limitados. Entretanto, ele pode executar outros programas em resposta a dados recebidos e deste modo ser alvo de ataques. Leitores de correio eletrônico (*mail readers*) que suportam extensões multimídia são mais sujeitos a este tipo de ataque, pois chamam outros programas para tratar tipos de dados mais complexos, como imagens, sons e vídeos.

Uma situação apresentada em [4] aponta problemas com *readers* que implementam extensões MINE (*Multipurpose Internet Mail Extensions*): se um *reader* MINE recebesse uma mensagem com a informação estruturada abaixo mostrada, a busca do arquivo rfc1470.txt em ds.internic.net/rfc através do serviço FTP anônimo seria realiza automaticamente.

```
Content-Type: Message/External-body
    name="rfc1480.txt";
    site="ds.internic.net";
    access-type="anon-ftp";
    directory="rfc"
```

Suponha-se agora que um atacante enviasse uma mensagem com o seguinte código para um usuário alvo:

```
Content-Type: Message/External-body
    name=".rhost";
    site="ftp.visigoth.org";
    access-type="anon-ftp";
    directory="."
```

Deste modo, o *reader* MINE dispararia automaticamente a busca do arquivo .rhost na máquina do atacante (ftp.visigoth.org), o que sobrescreveria o arquivo .rhost legítimo do usuário, abrindo brechas para futuros ataques com os serviços de login remoto e execução remota.

Sistemas podem ser protegidos contra ataques *command channel* através de *firewalls*. Com eles, pode-se reduzir o número de máquinas com as quais um atacante pode abrir um canal de comando e prover, nestes hosts, servidores seguros. Por outro lado, ataques *data-driven* são difíceis de serem evitados pois é impossível diferenciar uma mensagem legítima de uma que esconda algum perigo. Em alguns casos é possível filtrar mensagens que contenham caracteres estranhos vindos no cabeçalho, que podem indicar a presença de um ataque. Entretanto, utilizar versões recentes de servidores e agentes e educar os usuários irá evitar qualquer tipo de ataque *data-driven*.

O protocolo utilizado para correio eletrônico é o SMTP (*Simple Mail Transfer Protocol*). O SMTP não representa nenhum problema de segurança, mas suas implementações sim. O servidor SMTP mais utilizado em sistemas UNIX é o *sendmail* que,

historicamente, foi alvo de inúmeros ataques, incluindo o verme da Internet [5]. Seu principal problema é que ele foi projetado para ser adaptável a quase qualquer necessidade ou solicitação, e isto resultou em um processo extremamente complexo, executado como superusuário, que aceita conexões de qualquer máquina na Internet e possui uma poderosa e rica linguagem de comandos. Estas características fazem com que seja alvo de inúmeras tentativas de ataques, tendo, ao longo do tempo, sido acumulada uma grande lista de vulnerabilidades. Versões antigas do *sendmail*, por exemplo, permitem que uma mensagem de correio eletrônico possa ser redirecionada para qualquer arquivo no sistema; entre eles arquivos críticos tais como *passwd*, *.rhost* e *hosts.equiv*. Este arquivos podem assim ser alterados por um atacante através do envio de uma simples mensagem com o objetivo de "preparar o terreno" para uma futura investida.

O *sendmail* suporta uma *wizard password*, especificada no arquivo de configuração. Ela habilita o usuário a executar funções que não são solicitadas normalmente ao programa *sendmail*, sendo usada por programadores e depuradores de implementações do protocolo. Atacantes também fazem uso desta característica e, uma vez dispondo desta senha podem iniciar um *shell* sem realizar um login. Esta opção é ativada através do comando WIZ que deve estar sempre desabilitado, exceto em situações muito especiais.

O *sendmail* pode ser compilado em modo de depuração e ter deste modo o comando DEBUG ativo. A utilização do DEBUG pode fornecer a um atacante poder irrestrito no sistema alvo. É necessário, então, que os administradores do sistema se certifiquem que não esteja habilitado.

2.4 Finger

O serviço finger é utilizado para obter dados sobre pessoas. Quando utilizado sem argumentos mostra informações sobre todos os usuários atualmente utilizando um determinado sistema. Quando acompanhado de um nome, mostra informações detalhadas sobre este usuário. Finger é uma maneira fácil de divulgar informações para outras pessoas. Entre elas estão nome completo, *username*, *e-mail*, telefone, data e local do último *login*.

Além de fornecer informações a usuários com boas intenções, o finger possibilita a atacantes uma maneira de conseguir uma lista detalhada sobre usuários de um sistema. Sob este ponto-de-vista, o finger é um dos serviços mais perigosos, pois ele é muito útil para investigar um alvo em potencial. Informações pessoais são usadas por intrusos para adivinhar senhas (alguns usuários escolhem suas senhas baseadas nestas informações para facilitar sua memorização, tornando-as fáceis de serem descobertas). Informações sobre último *login* ajudam atacantes a selecionar contas inativas ou que raramente são utilizadas. Estas são alvos perfeitos pois dificilmente qualquer atividade hostil será descobertas. Informações disponibilizadas por finger podem ainda servir de base para ataques de engenharia social. Deste modo deve existir um grande cuidado sobre quais dados

disponibilizar através do serviço. Educação dos usuários e supervisão e intervenção do administrador reduzem as chances de se ceder informações indevidas através do finger.

Em virtude dos seus perigos, muitos administradores optam por desabilitar ou substituir o serviço finger:

- desabilitar o servidor finger através da remoção da linha correspondente em */etc/inetd.conf*. Esta pode não ser uma boa maneira de se contornar o problema, pois o principal uso do finger também será bloqueado.
- substituir o servidor finger por um *script* que envie uma mensagem padrão instruindo as pessoas como contatar seu domínio, para então obter a informação desejada.
- utilizar um servidor *phone book* no lugar do servidor finger. Estes servidores permitem que informações sejam colocadas em bases de dados e que se especifique quais delas devem ser retornadas para requisições provenientes de dentro e de fora do domínio.

Existem alguns clientes finger que não verificam o tamanho dos dados recebidos, esperando que este controle seja realizado pelo servidor. Assim, eles são passíveis de ataques do tipo *denial-of-service*, caso sejam enviados dados que ultrapassem os limites de *buffers* internos. Outros clientes são suscetíveis a ataques *data-driven*. Mensagens customizadas pelos usuários (*.plan)* podem conter controles de terminal e deste modo alterar o funcionamento normal de terminais ou de emuladores.

2.5 TFTP

O TFTP (Trivial File Transfer Protocol) é uma versão simplificada do FTP, planejada principalmente para transferir arquivos de configuração durante a inicialização do sistema. Por causa disto, não utiliza identificação ou autenticação do solicitante do serviço. Sua implementação é tão simples que cabe em uma pequena memória ROM, o que o torna ideal para realizar o *boot* remoto de terminais e estações sem disco. Não possuiu absolutamente nenhuma segurança. Suas primeiras versões permitiam o acesso a qualquer arquivo do sistema; as versões atuais limitam as transferências de e para diretórios específicos. Por tudo isto, deve ser evitado disponibilizar o serviço TFTP para fora da rede interna.

2.6 DNS

O DNS (Domain Name Service) é o serviço padrão para acessar e armazenar informações sobre máquinas na Internet. Ele é uma base de dados distribuída cujo principal objetivo é mapear nomes de máquinas em endereços IP. Entretanto ele pode também realizar o caminho inverso, ou seja, mapear endereços IP em nomes de máquinas. Outras atribuições do DNS são informar versão de sistema operacional e tipo de CPU dos

computadores, informar a máquina encarregada de fornecer serviço de nome para o domínio e a encarregada de receber correio eletrônico. O DNS permite assim cada domínio manter informações sobre seus computadores e possibilita que informações referentes a máquinas externas sejam encontradas.

Para resolver um *hostname*, ou seja, tentar encontrar o endereço IP através do nome fornecido, tem-se os seguintes passos básicos:

1. serviço cliente faz uma requisição para o servidor DNS local.
2. o servidor local tenta resolver o *hostname* com base nas informações que têm.
3. caso consiga, (o servidor local conseguirá resolver o *hostname* se este estiver no domínio de sua autoridade ou estiver em sua cache) o servidor local responde ao cliente.
4. caso não consiga, o servidor local repassa a requisição para servidores externos de nível mais alto na hierarquia DNS. Estes, por sua vez, tentam solucionar o *hostname*.
5. Após receber resposta dos servidores externos, o servidor local coloca-a em sua cache e responde ao cliente.

Para o cliente, todo o mecanismo de busca em servidores externos fica transparente. É importante notar que um servidor DNS pode agir, dependendo da ocasião, como servidor ou cliente. Um ou mais registros são retornados em resposta a uma requisição. Se um cliente pedir um nome que já esteve no cache mas ficou defasado, o servidor voltará ao servidor de nome com o domínio da autoridade e obterá a vinculação novamente.

Além da resolução de nomes, servidores DNS provem mecanismos para transferência de toda sua base de dados. Este processo é chamado *zone transfer*, e é realizado entre servidores que disponibilizam a mesma informação. Deste modo, seu principal uso é por servidores DNS secundários que querem ter uma cópia de todo o conhecimento do servidor DNS primário. Diversos domínios utilizam servidores secundários como um mecanismo de tolerância à falhas viso que realizam a mesma tarefa do primário. Estas transferências, entretanto, podem ser um risco de segurança, pois podem fornecer a atacantes uma completa lista de todos os computadores de uma organização conectados em uma rede interna. Eles a utilizam para investigar sobre futuros alvos.

O principal risco quando provendo serviços DNS é fornecer mais informação do que o necessário. Quando um cliente pesquisa sobre um determinado *hostname* pode ter como resposta, além do endereço IP, um outro tipo de registro (HINFO) que fornece informações sobre o hardware e software presentes nesta máquina. Estas informações, muito úteis para o administrador do sistema, podem ser de grande valia para atacantes. Elas são utilizadas para traçar um perfil completo da máquina alvo. Sabendo a versão do sistema operacional (e os seus *bugs* de segurança) o intruso tem maior chance de conseguir acesso. Portanto, o

encarregado da segurança deve cuidadosamente configurar o DNS para não prover este tipo de informação para requisições vindas de fora.

Os atacantes podem obter informações de um servidor DNS contatando-o e, fingindo ser um servidor secundário, requisitar uma *zone transfer*. Soluções são:

- configurar filtros de pacote para não permitir requisição de conexão TCP na porta em que o servidor DNS está escutando. Transmissões *zone transfer* são realizadas através do TCP, enquanto *lookups* são realizadas através do UDP.
- limitar o número de servidores secundários aptos a realizarem *zone transfer*. A diretiva *xfenets* informa quais servidores estão autorizados a requisitar uma *zone transfer*. Infelizmente, autenticação baseada em endereço fonte é o mecanismo empregado para realizar a verificação dos servidores, o qual pode ser facilmente enganado.

Administradores devem se preocupar com informações falsas repassadas por servidores DNS comprometidos por atacantes. Muitas implementações de clientes e servidores não verificam se a informação que chega é resposta de uma requisição realizada ou se provem de um servidor confiável. Simplesmente aceitam-na, colocando-a no cache. Consultas posteriores serão respondidas erroneamente com estes dados falsos.

2.7 Rlogin, rexec, rsh

Os serviços *rlogin, rexec* e *rsh*, inicialmente projetados para comunicação entre Berkeley Unix, são utilizados para acesso remoto a sistemas e execução remota de programas. A diferença básica entre estes serviços e os demais, é o sistema de autenticação utilizado. Para iniciar um sessão *rlogin, rexec* ou *rsh* em uma máquina remota, não é necessário fornecer um *username*, pois ele é automaticamente transmitido no início da conexão. Este *username* será o do usuário no sistema local que está executando o processo cliente destes serviços. Caso a conexão se origine em uma máquina confiável, não é necessário fornecer nenhuma espécie de senha.

Estes serviços utilizam o conceito de máquinas e usuários confiáveis para realizar autenticação, o qual permite que sessões entre duas máquinas sejam iniciadas sem a necessidade de senhas. Para implementar este conceito os sistemas Unix se utilizam de dois arquivos, */etc/hosts.equiv* e *.rhosts*. As entradas no arquivo */etc/hosts.equiv* correspondem a nomes de máquinas que são confiáveis pelo sistema. Se o nome da máquina cliente de uma sessão *rlogin* ou *rsh* estiver presente no arquivo */etc/hosts.equiv* da máquina servidora, então ela é considerada confiável e conexões podem ser iniciadas. Entretanto, para que um usuário consiga abrir uma sessão, é necessário que esteja cadastrado nos dois sistemas, possuindo em ambos a mesma identificação (*username*).

Embora *rlogin, rexec* e *rsh* funcionem muito bem do ponto de vista do usuário, proporcionando uma grande economia de tempo, apresentam alguns sérios riscos de

segurança, tornando-os perigosos, principalmente quando utilizados em ambientes não confiáveis. Como este serviços utilizam autenticação baseada em endereços, eles são suscetíveis a ataques do tipo *IP spoofing* e *DNS spoofing*. Estes ataques fornecem maneiras de convencer o servidor que conexões provenientes de máquinas não confiáveis estão, na realidade, se originando a partir de um computador confiável.

Atacantes podem usar serviços de transferência de arquivos mal configurados, tais como FTP, uucp, TFTP, para depositar determinada entrada em */etc/hosts.equiv* ou em algum *.rhosts* e conseguir acesso em uma máquina alvo. Outro problema de serviço mal configurado envolve NFS. Se o diretório raiz de um usuário é exportado com direitos de leitura/escrita, de modo que o atacante consiga montá-lo, *.rhosts* pode ser alterado a seu favor.

Um atacante que já comprometeu o sistema, pode utilizar a confiança entre máquinas e modificar */etc/hosts.equiv* ou *.rhosts* para configurar uma rota alternativa de entrada nesta máquina para ser utilizada se sua rota principal for complicada ou se ela for descoberta e fechada. Os atacantes dão principal atenção aos arquivos *.rhosts* de contas que são pouco utilizadas devido a menor probabilidade de serem descobertos.

Com a utilização dos arquivos *.rhosts* uma parcela da segurança do sistema fica nas mãos dos usuários, que podem enumerar livremente hosts em quem confiam. O problema é que muitas vezes usuários permitem acesso para computadores externos ao domínio, burlando a política de segurança da corporação. Deste modo, alguns administradores suspendem o uso do *.rhosts*, deixando a configuração dos serviços integralmente nas suas mãos através da modificação do arquivo */etc/hosts.equiv*.

A utilização dos serviços *rlogin, rexec* e *rsh* pode ser considerada segura somente quando empregados em ambientes nos quais todas as máquinas são confiáveis, sendo usados principalmente em redes locais. Assim, a sua utilização através da Internet, que pode ser considerado um ambiente não confiável, é inapropriada e deve ser proibida. Outros serviços podem ser usados como métodos alternativos a *rlogin, rexec* e *rsh.* Uma opção é o uso do Telnet, que mesmo dispondo de algumas vulnerabilidades, pode ser considerado mais seguro do que os serviços remotos.

2.8 WWW

O serviço WWW (World Wide Web) foi inicialmente desenvolvido nos laboratórios do CERN, em Genebra, Suíça. O trabalho foi liderado por Tim Berners-Lee e culminou na definição do protocolo HTTP (HyperText Transfer Protocol). Em 1992 o CERN colocou seu trabalho em domínio público, o que estimulou dezenas de organizações a criar seus próprios produtos.

Como toda a estrutura na Internet, no WWW tem-se também o programa cliente (*browser*), que faz as requisições dos documentos e os interpreta, e o programa servidor

(*Web server*). Diferente dos demais programas clientes, o *browser* WWW é extremamente simples de operar, e substitui com vantagens os serviços de FTP e Gopher. As capacidades e potencialidades do WWW foram ainda mais expandidas permitindo-se que os servidores executem programas por trás de páginas Web. Estes programas são criados com o protocolo CGI (Common Gateway Interface), e podem executar tarefas tão simples como atualizar um contador de acessos ou tão complexos como processar pedidos comerciais ou realizar consultas a banco de dados. Capacidade semelhante foi dada ao cliente através da execução de *applets* Java recebidos do servidor. Aumentando ainda mais a complexidade dos *browsers*, suas últimas versões processam ainda correio eletrônico, *news* e vários outrso serviços.

Se o serviço WWW é um dos mais úteis e difundidos na Internet, ele também representa um sério desafio à segurança:

- um atacante pode explorar *bugs* conhecidos ou erros de configuração de um servidor WWW ou *scripts* CGI para obter acesso não autorizado a outros arquivos do sistema, ou até mesmo obter controle do computador.
- informação confidencial pode ser inadvertidamente disponibilizada pelo servidor WWW.
- informação confidencial em trânsito entre cliente e servidor pode ser interceptada.
- um servidor WWW mal intencionado pode utilizar *bugs* conhecidos do *browser* ou *applets* Java para controlar o computador cliente.
- o processamento automático de diversos tipos de arquivos pelo *browser* ou por *plug-ins* nele instalados pode alterar o funcionamento do computador cliente ou colocar nele cavalos-de-tróia.

Devido a estes problemas, recomenda-se que o servidor WWW seja executado em uma máquina própria, onde não seja executado nenhum outro serviço. A utilização de uma máquina exclusiva para WWW dificulta a um atacante invadir o sistema, e reduz os danos causados por uma eventual invasão. Além disto, diversos outros pontos podem ser observados para minimizar os riscos:

- não permitir aos usuários a execução de programas ou *scripts* no servidor.
- os *scripts* CGI devem ser cuidadosamente projetados para somente executar as funções desejadas e testar exaustivamente entradas errôneas ou fora das especificações, possuindo um bom mecanismo de retorno de mensagens de erro.
- criar um usuário, *www*, que seja o administrador do servidor, e dono de todos os arquivos e diretórios. Não executar o servidor sob a conta *root*.
- eliminar do servidor todas as contas desnecessárias.
- eliminar do servidor todos os compiladores e demais programas que não sejam necessários.

- instalar o mínimo possível de serviços de rede; não montar diretórios de outras máquinas.
- utilizar HTPPS (Secure HTTP) quando informação sensível deve transitar pela rede.
- quando for necessário autenticar usuários, usar técnicas de criptografia. Não utilizar esquemas simples de nome e senha, que podem ser capturados.

O cliente WWW também apresenta problemas de segurança. A maioria deles diz respeito à manipulação automática de documentos:

- não permitir a execução automática de programas.
- não permitir o salvamento automático de arquivos executáveis; eles podem conter vírus ou cavalos-de-tróia.
- não processar automaticamente planilhas eletrônicas ou documentos de processadores de texto, pois estes podem conter macros maliciosas (como os vírus de macro do Word).
- desabilitar o processamento de *applets* Java quando acessar domínios não confiáveis ou que sejam capazes de conter código malicioso. Em casos extremos, somente habilitar Java quando acessar domínios de completa confiança.

Maiores informações sobre segurança em WWW podem ser obtidas nos seguintes endereços:

- http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html
- http://www.primus.com/staff/paulp/cgi-security
- http://hoohoo.ncsa.uiuc.edi.cgi/security.html
- http://www.cs.purdue.edu/coast/hotlist.html#securi01
- http://hopf.math.nwu.edu/docs/manual.html (sobre o WN server, um servidor seguro)

2.9 Principais vulnerabilidades

Entre as vulnerabilidades mais comuns encontradas nos serviços baseados em TCP/IP que são disponibilizados na Internet estão:

- má configuração dos serviços, o que permite a pessoas sem autorização ter acesso a dados confidenciais. Podem também causar a queda de todo o sistema caso recebam mais dados do que consigam tratar.
- serviços não utilizam nenhuma forma de criptografia nos dados a serem transmitidos pela Internet. Dados confidenciais que trafegam pela Internet podem ser interceptados, incluindo senhas de usuários.

- serviços utilizam mecanismos de autenticação fáceis de serem enganados. Enganando o sistema de autenticação, pessoas podem se passar por usuários legítimos ou máquinas confiáveis.
- implementação de clientes e servidores dos serviços apresentam *bugs*. Explorando estes *bugs*, pessoas podem executar ações às quais não têm permissão.
- serviços são baseados em TCP/IP o qual têm seus próprios problemas de segurança. Permite que uma conexão já estabelecida e corretamente autenticada pelo legítimo usuário seja roubada e utilizada por pessoas não autorizadas.
- alguns serviços são difíceis de serem usados com filtros de pacotes (veja-se seção 4). Dependendo das características do serviço, as regras para sua filtragem podem se difíceis de serem formuladas. Esta dificuldade pode acarretar em brechas nos sistemas de defesa.

É dever de todo administrador identificar os principais problemas de segurança no seu domínio e empregar mecanismos para solucioná-los. Para reduzir a probabilidade de erros e reduzir o número de brechas no sistema, é importante que se siga um padrão bem definido durante a configuração dos serviços. É importante, também, considerar alternativas mais robustas, como uso de criptografia, quando tarefas críticas estiverem envolvidas.

De todas as vulnerabilidades apresentadas acima, talvez as que mais sejam exploradas por pessoas mal intencionadas são aquelas relacionadas com erros de implementação. Ao longo dos anos, diversos *bugs* foram encontrados nos mais distintos produtos de diferentes fornecedores, sendo utilizados por atacantes para conseguir acesso. É função dos fornecedores liberar versões que os corrijam, e é função dos administradores de sistemas implantá-las. Caso esta implantação demore, atacantes poderão se valer de *bugs* já divulgados e amplamente discutidos.

3. Principais ataques

Um ataque pode ser dividido em algumas fases distintas, cada uma delas possuindo suas próprias características [14]:

1. selecionar um alvo.
2. coletar informações sobre alvo. Existem diversos métodos para conseguir informações sobre um alvo. Um ponto de partida é a utilização de serviços como finger, showmount e rpcinfo. Todas as ferramentas empregadas por administradores para gerenciamento são também de grande valor para o atacante. DNS, whois, sendmail, ftp e uucp são outros serviços que fornecem ao atacante valiosas informações durante a fase de obtenção de dados.
3. lançar um ataque sobre o alvo. Para tentar entrar em um sistema, o atacante utiliza vários métodos que exploram vulnerabilidades conhecidas de serviços, servidores

e sistemas operacionais. Alternativamente, o atacante pode simplesmente tentar adivinhar senhas de usuários na máquina alvo.

4. destruir evidências da invasão. Geralmente, as ações realizadas pelo atacante para conseguir entrar na máquina ficam registradas em arquivos de log. É importante que eles sejam alterados de modo a encobrir estes passos e evitar sua análise pelo administrador do sistema.
5. obter senhas de outras contas. Quanto mais contas forem comprometidas pelo atacante melhor. Na maioria das vezes, um simples ataque do dicionário no arquivo */etc/passwd* é necessário para obtenção de um considerável número de senhas.
6. obter acesso *root* na máquina invadida. Tendo acesso a uma conta de usuário, o atacante terá direitos limitados. Assim, é importante conseguir acesso *root*, pois deste modo disporá de todo o sistema, empregando-o para realizar os passos seguintes e lançar futuros ataques. Para conseguir direitos *root*, atacantes podem empregar diversos métodos, explorando geralmente *bugs* existentes em programas.
7. instalar ferramentas para captura de senhas. Para isto, atacantes podem instalar monitores de rede ou cavalos-de-tróia. Entre os programas candidatos para serem substituídos por cavalos-de-tróia estão *login, telnet e ftp*. Caso estes estejam protegidos e sua alteração seja impossível, atacantes podem, ainda, explorar erros tipográficos dos usuários. Para isto, basta colocar programas chamado *telent* (ou *logn, fpt,* etc) em diretórios utilizados pelas vítimas.
8. configurar caminhos secundários de entrada na máquina invadida; para o caso da rota principal ser descoberta e fechada. Dispondo de direitos *root*, fica extremamente fácil para um atacante construir rotas alternativas para entrar no sistema. Entre elas estão a inserção de um novo usuário ou a reconfiguração de alguns serviços.
9. encontrar outras máquinas que confiam na máquina invadida. Para isto, basta analisar arquivos como *.rhosts* e *hosts.equiv*.
10. utilizar a máquina invadida como base para lançar outros ataques.

3.1 Engenharia Social

A engenharia social é um método de ataque que consiste em enganar as vítimas através do emprego de informações adquiridas por pesquisa. O atacante se vale destas informações para induzir pessoas de boa fé a executam ações indevidas e perigosas. O intruso pode utilizar-se de vários meios para lançar um ataque de engenharia social, sendo telefonemas e mensagens de correio eletrônico falsificadas os mais comuns.

3.2 Denial-of-Service

Um ataque *denial-of-service* tem como objetivo impedir o uso legítimo do sistema. Para isto, um intruso inunda o sistema ou a rede com mensagens, processos ou requisições, de modo que nenhum trabalho possa ser realizado. Isto pode ser obtido de várias maneiras distintas:

- Ping of Death - enviando-se um pacote maior que o limite de 64 Kbytes.
- Teardrop - enviando-se pacotes com dados fragmentados que são impossíveis de serem reunidos.
- Syn Flood - saturando-se um computador com pedidos de conexão (SYN) que nunca são efetivados.
- Smurf Attack - falsificando-se um pedido de ping em broadcast; a resposta dos demais computadores satura a máquina alvo com múltiplas respostas.
- UDP Flood - criando-se um fluxo de dados inúteis entre duas máquinas (como chargen e echo).

Na maioria das vezes, impedir ataques *denial-of-service* é impossível. Se uma máquina aceita conexões do mundo externo (correio eletrônico, FTP anônimo, etc) ela é passível de ser inundada. Assim, é importante que se configure os serviços de modo que se um deles for inundado, o resto do site permaneça operacional enquanto as pessoas encarregadas tentam encontrar e solucionar o problema. Uma alternativa para minimizar as chances de ataques *denial-of-service* é fornecer recursos suficientes (poder de processamento, espaço em disco) para que, mesmo em circunstâncias de extrema carga, o sistema continue funcionando.

3.3 Exploração de Bugs

Uma das maneiras mais comuns de se conseguir acesso em um sistema é explorar furos de implementação presentes em programas e sistemas operacionais. Escrever software correto ainda é uma meta a ser atingida pela grande maioria dos desenvolvedores.

Como quaisquer outros, programas de comunicação que implementam segurança possuem *bugs*. Por constatação, pode-se verificar ainda que quanto maior o seu tamanho, maior o número de bugs. Assim, máquinas expostas a ataques devem executar o mínimo necessário de programas, e estes, preferencialmente, devem ser bem escritos e exaustivamente testados.

Durante o desenvolvimento de um software deve-se levar em consideração técnicas que objetivam a eliminação de erros. Quando escrevendo programas que envolvem alguma espécie de segurança, estas devem ser reforçadas para garantir que o produto final seja seguro e estável. A falha de apenas um componente é o suficiente para que o sistema inteiro seja comprometido por um atacante. Entre estas técnicas estão:

- desenvolver programas utilizando boas práticas de engenharia de software.
- verificar alguns erros comuns que podem causar grandes danos. Não permitir estouros de *buffers* ou acessos fora dos limites. Tratar falta de memória e falta de espaço em disco.
- verificar as entradas. É recomendável que se utilize ferramentas, tais como *lex* e *yacc*, para definir e implementar todas as possíveis entradas. Muitos programas apresentam comportamento espúrio quando tratando entradas não previstas pelo desenvolvedor.
- implementar a lei do menor privilégio. Não fornecer a processos mais poder do que necessitam.

3.4 Exploração de protocolos

O protocolo TCP/IP representa um risco de segurança simplesmente porque ele permite que usuários remotos acessem arquivos e dados de outros equipamentos. Quando foi projetado, não foi levado em consideração o aspecto segurança, pois naquela época as pessoas envolvidas não tinham idéia de quão utilizado ele seria. Assim, atacantes exploram algumas de suas características para conseguir acesso não-autorizado em sistemas de computação. Muitos desses são derivados de falhas no mecanismo de autenticação.

3.4.1 IP Spoofing

No *IP spoofing,* o atacante utiliza o endereço IP de uma máquina confiável juntamente com algum protocolo que faz autenticação baseada em endereços. Deste modo, este ataque permite que pessoas utilizando qualquer máquina se passem por usuários legítimos de uma máquina que é confiável em determinada rede.

A idéia básica deste ataque é estabelecer uma conexão com o alvo, enganando o *handshake* inicial do protocolo TCP. O principal ponto do *sequence number attack* é "adivinhar" o ISN (Initial Sequence Number) gerado pelo host alvo. Para isto,é utilizada uma característica da implementação do 4.2BSD Unix, juntamente com os protocolos TCP/IP. No Unix de Berkeley, existe uma variável global para o número inicial de seqüência, que é incrementada por 128 a cada segundo e por 64 após o início de cada conexão. Deste modo, se o atacante criar uma conexão real com o alvo (possivelmente com um serviço que está disponível para ele), descobrirá o estado atual da variável. De posse desta informação, soma-se 64 a ela, sendo o resultado, com alto grau de confiabilidade, o ISN da próxima conexão.

Mas existe ainda um problema para o atacante; o pacote de sincronização enviado pelo alvo para a suposta origem da conexão. Este host verá este pacote como parte de uma conexão não existente, tentando, então, terminá-la através do envio de pacotes com flag RST (*reset*) setado. Isto irá causar o encerramento da conexão fraudulenta. Neste caso, uma

opção é retirar de operação a máquina, inundando a porta com requisições de conexão. Isto causará *overflow* de filas e causará a perda da mensagem do alvo para a máquina confiável. É importante observar que existem inúmeras outras formas de se tirar uma máquina de operação, sendo possível ainda, lançar o ataque quando a máquina confiável estiver desligada.

Sejam três hosts, *X*, *S*, *C*. O atacante está lançando o ataque a partir de *X* em *S*, personificando *C*, em quem *S* confia.

1. X retira de operação C, enchendo a porta 21 (canal de controle do FTP) com requisições de conexão.
2. *X* estabelece uma conexão real com *S* e grava o número inicial de seqüência retornado por *S*.
3. *X* envia um pacote SYN tendo como origem a porta 21 e destino porta 514 (servidor de execução remota). O endereço fonte contido no cabeçalho deste pacote é alterado para conter o endereço IP de *C*.
4. *S* faz a autenticação da requisição de conexão baseada em endereço. *S* envia um pacote de resposta para *C*, que é ignorado pois *C* está fora de operação.
5. *X* envia um pacote ACK para *S*, com o número de seqüência "adivinhado", ou seja, o número previamente adquirido somado com 64.
6. *X* tem uma conexão legítima com *S* na porta 514.
7. *X* envia dados contendo um comando a ser executado em *S*.
8. *S* "pensa" que recebeu uma comando *rsh* (*remote shell*) de *C*, uma máquina confiável, executando-o.

O ataque descrito acima se utiliza de uma característica de implementação, tendo êxito somente em máquinas que estejam utilizando o 4.2BSD Unix como sistema operacional. Porém, existem generalizações que permitem ao atacante, com um pouco mais de trabalho, forjar conexões em qualquer outro host, independente do seu sistema operacional. Para isto, o atacante deve fazer algumas conexões reais de teste, para tentar descobrir a lei de geração dos números de seqüência e conseguir predize-lo na hora do ataque.

Existe um grande número de possíveis defesas:

- a geração do número inicial de seqüência pode ser randômica. Quanto mais randômico for, mais difícil será para o atacante "adivinhá-lo". (A especificação do TCP diz que a variável deve ser incrementada 250.000 vezes por segundo)
- colocar barreiras (Firewall) que impeçam a entrada de pacotes cujo endereço IP origem seja de máquinas internas ao site.
- colocar barreiras (Firewall) que impeçam conexões externas em serviços que realizam autenticação baseada em endereços.

- implantar serviços com outros mecanismos de autenticação (por criptografia, por exemplo) caso eles sejam indispensáveis.

3.4.2 DNS Spoofing

Em ataques ao DNS, a idéia básica do *DNS Spoofing* é subverter o servidor de nomes, e com isto permitir que máquinas não confiáveis (as do atacante) se passem por máquinas confiáveis. Para lançar este ataque, o intruso deve ter inicialmente controle sobre o *host* servidor de DNS e saber o nome de uma máquina em quem o alvo confia. Com isto altera-se o registro do DNS que mapeia o endereço IP da máquina confiável para o seu nome, modificando-o para que contenha o endereço da máquina atacante. A partir deste momento o intruso terá livre acesso em serviços que realizam autenticação baseada em nomes.

A maioria dos novos sistemas possuem métodos contra este tipo de ataque, utilizando para isto uma técnica conhecida com *cross-chek*. Nela, o nome retornado pela consulta é submetido novamente ao serviço de nomes. Se o endereço utilizado para a conexão é diferente do retornado pelo *cross-chek*, a conexão é abortada e uma violação de segurança é apontada. O *cross-chek* pode ser implementado no servidor de nomes ou nos servidores dos serviços com autenticação baseada em nomes. Entretanto, existem variantes do *DNS Spoofing attack*. Nelas, o intruso contamina o cache das respostas do DNS na máquina alvo ou inunda o servidor de DNS com pedidos, com o objetivo de enganar o *cross-check*.

3.4.3 Source Routing Attack

Este ataque se utiliza dos mecanismos de roteamento disponíveis e da opção *loose source route* do protocolo IP para induzir a máquina alvo a acreditar que o ataque é, na realidade, uma operação legítima proveniente de uma outra máquina confiável. A opção *loose source route* disponibiliza um mecanismo para que a origem de um datagrama possa fornecer informações de roteamento usadas pelos *gateways* para direcioná-lo ao destino. Deste modo um processo pode iniciar uma conexão TCP fornecendo um caminho explícito para o destino, sobrescrevendo o processo usual de roteamento.

Sejam três máquinas, *X*, *A* e *B*. Seja a situação em que o intruso em *X* lança um *Source Routing attack* em *A* que confia em *B*.

1. O intruso, com algum tipo de ataque, desestabiliza *B*, ou espera até que ela esteja desligada.
2. O intruso configura *X* para que contenha o endereço IP de *B*. Neste momento o atacante está personificando a máquina confiável.

3. O intruso utiliza *rlogin* ou *rsh* para acessar *A*. Para isto deve utilizar pacotes IP com opção *loose source route* ativada e corretamente configurada, contendo um caminho válido de *X* para *A*.
4. *A* aceita as requisições de *X*, pensando que elas são de *B*. A partir deste momento *X* detêm uma conexão legítima com *A*.
5. As respostas de *A* são enviadas para *X* através do caminho inverso descrito na opção *loose source route* dos datagramas recebidos por *A*.

A opção *loose source route* pode ser empregada, também, para alterar a rota dos pacotes e, deste modo, desviar dos sistemas de segurança de um domínio. Isto pode ser impedido configurando-se um *firewall* como sendo o único ponto de comunicação com o exterior. Duas defesas são:

- utilizar versões dos servidores *rlogin* e *rsh* que não aceitam conexões com a opção *loose source route* ativada.
- utilizar filtros de pacote que não permitam que pacotes com a opção ativada entrem no site. O uso deste enfoque é aconselhado porque o domínio pode disponibilizar outros serviços com autenticação baseada em endereços que não dispõem do mecanismo citado acima.

3.4.4 RIP

Para que hosts em diferentes redes consigam se comunicar, é necessária a presença de um roteador (ou de um conjunto deles) cuja função é identificar o melhor caminho que um pacote deve seguir a fim de encontrar seu destino. Para isto, deve ser configurado de modo que contenha as informações necessárias em suas tabelas internas. Inicialmente um roteador só conhece as redes a que está diretamente conectado, sendo as demais conhecidas através de rotas estáticas ou de protocolos de roteamento. Rotas estáticas são caminhos nas tabelas de roteamento inseridas manualmente, sendo mudada somente através de nova intervenção humana. Quando se utiliza protocolos de roteamento, informações são trocadas automaticamente entre roteadores e quaisquer mudanças de caminho são atualizadas dinamicamente.

Entre os protocolos de roteamento mais utilizados está o RIP (*Routing Informatin Protocol*) cujas atualizações são realizadas em intervalos de tempo fixos e pré-definidos e que geralmente não verifica a veracidade dos dados recebidos. Esta fragilidade do RIP pode ser utilizada por atacantes para enganar o sistema de roteamento.

O ataque mais provável que se utiliza do RIP faz com que o intruso consiga personificar uma máquina específica, através do envio de informações de roteamento falsas para *gateways* que envolvam caminhos entre o alvo e o atacante. Deste modo pacotes destinados ao computador confiável são direcionados para a máquina do atacante. Em um primeiro momento, o atacante personifica uma máquina em quem o alvo confia. Para isto

basta alterar o seu endereço IP. Após, informações de roteamento falsas são enviadas a todos os roteadores entre o alvo e o intruso. Estas informações alteram o estado das rotas que indicam os caminhos que pacotes a partir do alvo devem seguir para chegar à máquina confiável. O novo estado indica que esta rota é entre o alvo e o atacante.

É importante observar que após a alteração das rotas a máquina que foi personificada pelo intruso não conseguirá fazer qualquer comunicação com o alvo, pois as respostas serão redirecionadas para o atacante. Com isto, o ataque poderá ser descoberto. Deste modo, o intruso deverá personificar uma máquina que, de preferência, esteja desligada, esteja à algum tempo em *idle* ou esteja inoperante, em virtude de algum ataque do tipo *denial-of-service*.

Existe uma variante mais sutil e perigosa deste ataque. Nela, o atacante personifica uma máquina que está ativa. Deste modo, os pacotes do alvo para ela serão enviados para o atacante. Neste momento eles poderão ser visualizados ou alterados pelo intruso. Estes pacotes poderão conter dados importantes, tais como informações confidencias a respeito de uma empresa. Após são enviados para o destino correto através de *IP source address routing*. Uma característica interessante deste ataque é que conexões requisitadas pelo alvo também passarão pelo atacante, sendo possível ao intruso analisar os pacotes que, ocasionalmente, poderão conter senhas de usuários ou outras informações relevantes.

Já existem hoje em dia roteadores que aceitam o protocolo RIP versão 2, que especifica um melhoramento em diversos aspectos em relação ao RIP, incluindo autenticação nas mensagens de atualização. Entretanto ele não torna os sistemas de roteamento completamente seguros; apenas torna o trabalho do atacante um pouco mais difícil.

3.4.5 ICMP

O ICMP (*Internet Control Message Protocol*) é um protocolo utilizado para fazer a manutenção de conexões TCP/IP. Ele também é usado para gerar uma grande quantidade de ataques. Entres as mensagens ICMP mais exploradas por atacantes estão a *redirect* e *destination unreachable*.

A *redirect* é utilizada por gateways para avisar máquinas sobre melhores rotas. Os ataques produzidos com esta mensagem subvertem o sistemas de roteamento, se assemelhando aos ataques no RIP. Um ataque viável utilizando a *redirect* é configurar uma falsa rota para um host confiável *A*, através de um *gateway* secundário, já de posse do atacante. Para isto sejam os seguintes passos: o atacante envia para o alvo um pacote de requisição de conexão falsificando o endereço de *A*. O alvo, dando seqüência ao *handshake*, envia seu próprio pacote para *A*, através do roteador primário. Enquanto o pacote estiver em trânsito, o atacante falsifica uma mensagem *redirect* afirmando ser do gateway primário e referindo-se à falsa conexão. Esta mensagem é enviada para o alvo, que a aceita como sendo um controle legítimo. Caso o alvo faça as mudanças ditadas pela

redirect em suas tabelas globais de roteamento, existirá então um caminho entre o alvo e o intruso, que poderá ser utilizado para conexões legítimas. Na realidade, não é necessário que o atacante disponha de um *gateway* secundário. É suficiente que ele detenha a posse de qualquer máquina na rede local do alvo, utilizando-a como sendo o *gateway*.

A mensagem *destination unreacheable* é utilizada para informar a um host que o destino da sua conexão não pode ser encontrado. Esta mensagem pode ser enviada por um roteador, caso nenhuma rota para o destino esteja disponível, ou se a máquina destino for inexistente ou se está inativa. A *destination unreacheable* pode, também, ser enviada pelo próprio host destino. Neste caso, o host está afirmando que a porta a que a conexão se refere não pode ser encontrada. A *destination unreacheable* pode ser utilizada para terminar conexões específicas existentes e, deste modo, promover ataques do tipo *denial-of-service*. Para isto, basta que o intruso conheça os números da porta local e remota de uma conexão TCP e falsifique um pacote ICMP. Estas informações são, as vezes, disponíveis através do serviço *netstat*.

3.5 Sniffers

Na grande maioria das redes, os pacotes são transmitidos para todos os computadores conectados ao mesmo meio físico, e cada máquina é programada para somente tratar os pacotes destinados para ela. Entretanto, é possível reprogramar o interface de rede de uma máquina para que ele capture todos os pacotes que circulam pelo meio, não importando o destino. O interface de rede que opera deste modo é dito em modo promíscuo, e esta técnica é denominada de *sniffing*.

Como em um ambiente de rede normal trafegam pela rede muita informação sensível, como por exemplo nomes de usuários e senhas (*username* e *passwords*), fica fácil para um programa *sniffer* obter estas informações. Naturalmente, o volume de dados a tratar é enorme, mas o trabalho pode ser facilitado por regras simples de filtragem e pelo fato de ser extremamente fácil detectar o início de uma conexão TCP. Um *sniffer* bem conhecido é o programa esniff.c, desenvolvido para operar no sistema SunOS, e que captura os primeiros 300 caracteres de todas as conexões telnet, ftp e rlogin.

A utilização de *sniffers* é difícil de ser detectada. Se ele estiver somente coletando dados e não respondendo a nenhuma informação, a única maneira é percorrer fisicamente todas as conexões da rede. Uma maneira de detectar-se um *sniffer* em operação é através dos imensos arquivos gerados por ele. Note-se ainda que normalmente um *sniffer* requer recursos somente disponíveis as superusuário, o que impede o seu uso pelos usuários normais de um sistema. Entretanto, com a popularização de sistemas UNIX para IBM-PCs, é relativamente simples a um atacante conectar um PC à rede e a partir desta máquina, sem o conhecimento do administrador da rede, executar diversos *sniffers* (e outros ataques).

Diversas medidas podem ser adotadas contra *sniffers*, mas a maioria delas requer o uso de hardware específico, como *hubs* ativos, que enviam a uma rede ou sub-rede somente

os pacotes destinados à máquinas localizadas nestas rede, ou então interfaces de rede que não possuam modo promíscuo. No lado dos aplicativos, podem ser utilizados programas que utilizem criptografia ou então realizam autenticação com métodos criptográficos, como as *one time passwords*.

3.6 Ataque do dicionário

Um dos arquivos mais cobiçados por atacantes é o /etc/passwd, devido ao fato dele conter as senhas de todos os usuários de uma máquina ou rede. Naturalmente, as senhas não são armazenadas na forma de um texto normal, mas sim são cifradas através de um algoritmo unidirecional, que não permite a reversão do texto cifrado novamente para o texto normal.

Este algoritmo, o *crypt(3)*, é uma variante do DES (Data Encryption Standard), que usa a senha (de 8 caracteres) como chave de cifragem para criptografar um *string* de 64 zeros binários. Para adicionar uma maior complexidade a este processo, ele é repetido 25 vezes. A cada vez o texto resultante é reutilizado como texto de entrada, e a cada vez a senha do usuário é usada como chave. O resultado final, de 8 bytes, é transformado em 11 caracteres visíveis (cujo bit mais significativo é zero, e de onde se eliminam caracteres de controle).

Para impedir que usuários com a mesma senha tenham o mesmo texto cifrado, o UNIX utiliza ainda uma pitada de sal (*a grain of salt*). A cada usuário é atribuído um número randômico de 12 bits, o *salt number*, cuja única finalidade é produzir saídas distintas para usuário distintos. Assim, se dois usuários usarem por acaso a mesma senha, este fato fica camuflado no arquivo /etc/passwd. Para que o processo de verificação possa reproduzir a cifragem, o *salt* é armazenado como os dois primeiros caracteres do texto cifrado. Quando um usuário efetua o *login*, o número *salt* é recuperado e utilizado juntamente com a *password* fornecida pelo usuário para gerar um texto cifrado. Se o texto resultante é igual ao texto armazenado, o usuário é autenticado como legítimo.

Entretanto, a eficácia de todos estes cuidados fica comprometida pelo hábito das pessoas de utilizar senhas facilmente memorizáveis, como nomes próprios ou palavras de uso corriqueiro. Justamente neste fato se baseia o ataque do dicionário. O atacante compõe um dicionário e, dispondo de suficiente poder de processamento, experimenta todas as palavras deste dicionário contra o texto cifrado armazenado no arquivo /etc/passwd. Se um usuário com número *salt s* possui como texto cifrado o *string x*, então no momento em que a palavra *w* do dicionário, em conjunção com o número *s* produzir como resultado *x*, então o atacante sabe que a senha correspondente é *w*.

Para impedir este ataque, os usuários devem ser conscientizados para não usarem palavras "fáceis", ou seja, que tenham grande probabilidade de constar em um dicionário. Boas sugestões são combinações de letras e números, uso de palavras com grafia errada, ou letras iniciais das palavras de uma frase.

4. Mecanismos de proteção

A definição de uma política de segurança é o primeiro passo para que se possa escolher e implementar quais os mecanismos de proteção serão utilizados. É necessário que as seguintes questões sejam profundamente consideradas:

- o que se está querendo proteger?
- o que é preciso para proteger?
- qual a probabilidade de um ataque?
- qual o prejuízo se o ataque for bem sucedido?
- implementar procedimentos de segurança irá ser vantajoso no ponto de vista custo-benefício?

Cada uma destas questões devem ser muito bem discutidas. Qualquer medida de segurança que for implantada deve levar em consideração o usuário, pois é ele quem utiliza o sistema no dia-a-dia. Do ponto de vista de segurança, quatro posturas podem ser encontradas, que definem os 4 P's da segurança [2, 15]:

- Paranóico: Tudo é proibido, mesmo aquilo que deveria ser permitido. Como regra, a conexão à Internet nunca deveria ter sido estabelecida.
- Prudente: Tudo que não é explicitamente permitido é proibido. É a melhor postura atual, apesar de requerer uma boa administração.
- Permissivo: Tudo que não é explicitamente proibido é permitido. Esta era a postura até o início da década de 90.
- Promíscuo: Tudo é permitido, inclusive o que deveria ser proibido.

A política de segurança deve estar sempre sendo revisada, pois ao longo do tempo as necessidades se alteram. Note-se que a segurança através do desconhecimento (*security through obscurity*), um enfoque muito adotado atualmente, é uma postura muito perigosa. Adotando este enfoque, muitos domínios acreditam estar seguros pelo fato de imaginarem que ninguém sabe a seu respeito. Assim, realizam pouco, ou nenhum trabalho voltado à segurança da organização. Uma vez conhecidos, serão alvos fáceis.

A segurança pode ser implementada a nível de *hosts* ou a nível de rede. Com segurança de *hosts*, cada máquina é protegida isoladamente. Este enfoque funciona bem quando implantado em domínios pequenos, mas normalmente é um processo complexo, demorado e caro tornar cada máquina segura. Esta tarefa se torna ainda mais complexa se uma grande variedade de fornecedores de máquinas, sistemas operacionais e programas está envolvida.

Na segurança a nível de rede, todas as máquinas de um domínio são protegidas por apenas um mecanismo que está presente entre os canais de comunicação que conectam máquinas internas com máquinas externas. Em outras palavras, este mecanismo representa

uma barreira entre todas as conversações entre a rede interna e a rede externa. Com a adoção deste enfoque, o domínio estará protegido independentemente da configuração e das vulnerabilidades das máquinas internas. Teoricamente, uma barreira pode proteger uma grande quantidade de computadores contra ataques. Entretanto, como nenhuma segurança é absoluta, é importante que as máquinas internas apresentem mecanismos de segurança a nível de *host*. Atualmente a segurança de rede é implementada através de *firewalls*.

Para implementar a política de segurança em um domínio, pode ser necessário a aplicação de alguma estratégia de segurança. As mais comuns são [3]:

- *least privilege*: dar a usuários, administradores e programas somente os privilégios que são necessários para que suas tarefas sejam realizadas. Nunca se deve dar mais poder do que o necessário.
- *defense-in-depth*: nunca confiar em somente um mecanismo para realizar a segurança. Utilizar dois ou mais mecanismos é uma boa alternativa pois implementa redundância. Caso um componente falhe, o sistema ainda estará protegido pela presença dos demais. Note-se que não é aconselhável que se implemente *defense-in-depth* utilizando dois componentes de um mesmo fabricante. Isto facilita a entrada de atacantes devido a erros de configuração ou *bugs* no software.
- *choke point*: uma estratégia *choke point* força que toda a comunicação entre a rede interna e a Internet passe por apenas um canal. Neste canal devem estar presentes componentes de segurança e monitoramento a fim de torná-lo seguro (como qualquer outra comunicação, tentativas de ataque também passarão por ele).
- *weakest link*: deve-se eliminar todos os pontos fracos do sistema. Pontos fracos são os alvos preferidos dos atacantes, devido às suas vulnerabilidades. Se não for possível eliminá-los, deverão ser muito bem monitorados.
- *fail-safe*: caso um componente falhe, ele deve parar de funcionar de modo a não permitir o acesso do atacante. Até que seja consertado ele negará também acesso de pessoas autorizadas.

Muitos mecanismos podem utilizados para implementar segurança. Neste artigo são apresentados somente os mais empregados atualmente: *wrappers*, *firewalls*, mecanismos de autenticação, ferramentas de detecção de falhas e métodos criptográficos.

4.1 Wrappers

Wrappers (ou TCP Wrappers) [9, 6, 17] são na realidade um conjunto de programas que "encapsulam" os *daemons* dos serviços de rede visando aumentar sua segurança. Eles funcionam como um filtro e estendem o serviço original, dando por exemplo uma maior capacidade de *log*, verificação de falsificação de números IP ou nomes de *hosts*, aumento nas restrições dos serviços, etc.

O uso de *wrappers* é bem simples, bastando alterar a configuração do serviço que se deseja proteger no arquivo inet.conf. Quando da requisição de um serviço, o *inetd* irá então ativar o *wrapper*, que após a verificação da requisição ativará o serviço original. O controle de acesso do *wrapper* pode ser feito por serviço ou por host, e é definido através de um sistema de regras. Isto permite funções interessantes, como por exemplo executar um finger em cada usuário que tentar utilizar o *finger daemon*, ativar um servidor com *daemon* em português caso o domínio de origem termine em ".br", cancelamento automático de qualquer conexão que utilize *source routing*, criação de *banners* ou mensagens personalizadas para diversos serviços e muitos outras aplicações.

O *wrapper* não pode ser considerado uma ferramenta para segurança total. Seu uso principal visa suprir deficiências dos servidores atuais e aumentar o controle sobre sua utilização. Além disto, ele é uma perfeita ferramenta para *log*.

A medida que os *daemons* atuais vão sendo substituídos por versões mais robustas, com maiores capacidades ou maior segurança, os *wrappers* irão ter sua importância diminuída.

4.2 Firewalls

Um *firewall* [3, 4, 9] é um conjunto de componentes colocados entre duas redes e que coletivamente implementam uma barreira de segurança. Seu nome é emprestado das portas corta-fogo que são utilizadas para diminuir a velocidade de propagação de um incêndio, e sua finalidade é exatamente a mesma: retardar os efeitos de um ataque até que medidas administrativas contrárias sejam executadas. Um *firewall* pode ser considerado um *choke point* pois todo o tráfego entre as redes interna e externa (Internet) deve passar por ele. Assim, é um ótimo local onde aplicar a política de segurança. Por exemplo, se a política de segurança impede o uso de FTP para o exterior, todos os pedidos de conexão com servidores FTP externos serão filtrados pelo *firewall* e não serão passados para à Internet.

O objetivo básico da implementação de um *firewall* é defender a organização de ataques externos. Como efeito secundário, ele também pode ser utilizado para regular o uso de recursos externos pelos usuários internos. Entretanto um *firewall* não pode proteger contra:

- usuários internos. Qualquer forma de ataque proveniente da própria rede interna não pode ser detectada pelo *firewall*. A razão é óbvia: o fluxo de dados de um ponto da rede interna para outro ponto da rede interna não passa pelo *firewall*.
- vírus. Os componentes de um *firewall* analisam cabeçalhos dos pacotes que chegam, e não os seus dados.

Normalmente um *firewall* pode ser implementado utilizando dois mecanismos: filtragem de pacotes e servidores *proxy*. Servidores *proxy* são programas que "conversam"

com servidores externos em nome dos clientes internos. Clientes *proxy* conectam-se em servidores *proxy* que conectam-se nos servidores reais. O servidor *proxy* recebe resposta do servidor real e a redireciona para o cliente *proxy*. Servidores *proxy* geralmente executam em *bastion hosts*, ou seja, computadores que devem ser altamente seguros porque são expostos à Internet, sendo o ponto de comunicação dos usuários com o mundo externo. Por estarem expostos, são alvos de grande número de ataques.

Filtros de pacote realizam um roteamento seletivo de pacotes. Como roteadores normais, eles redirecionam o pacote analisando o endereço destino. Porém, com base em outros dados (endereço destino, endereço fonte, porta origem, porta destino) eles podem aceitar ou negar um pacote. Caso o pacote seja aceito, o procedimento é o normal. Caso ela seja negado o filtro de pacote simplesmente o descarta. Filtros de pacote são uma maneira barata para se implementar a política de segurança. Geralmente, características que permitem filtragem acompanham os roteadores. Assim, a decisão de se implantar um filtro de pacotes não irá ter custo adicional.

Filtros de pacote são flexíveis, dando ao administrador a chance de realizar filtragens baseada em serviços ou endereços de máquinas. Realizando filtragem baseada em serviços ele pode, por exemplo:

- permitir que usuários internos utilizem telnet para acessar *hosts* externos e impedir o inverso.
- permitir que máquinas externas acessem o servidor de *e-mail* da organização e impedir acesso externo em serviços perigosos, como TFTP, NFS, rlogin, etc
- bloquear qualquer tráfego do interior para o exterior.

Com filtragem baseada em endereços, o administrador pode:

- permitir qualquer tráfego proveniente da rede *x.y.z.w* que é considerada confiável pela política de segurança.
- impedir a entrada de qualquer pacote cujo endereço representa uma máquina interna (esta regra impede a maioria dos ataques *IP spoofing*).
- permitir que a máquina externa *r.s.t.u.v* se conecte no servidor de *e-mail* da organização. Esta regra é muito comum, e permite que um servidor de mail confiável faça a transferência das mensagens.
- principal problema quando utilizando filtros de pacote é configurá-lo da maneira correta para que reflita a política de segurança da organização, por exemplo, uma universidade (figura 4). Muitas vezes, isto não é uma tarefa trivial. Cada fornecedor disponibiliza sua própria linguagem para formulação das regras, as quais, muitas vezes, são confusas. Se regras erradas forem inseridas, elas podem

abrir um "buraco" no filtro de pacote, por onde o atacante pode entrar ou podem negar serviços liberados para usuários internos.

ação	fonte	porta	destino	porta	flags	comentários
allow	secondary_dns	*	primary_dns	53	tcp	permite acesso pelo DNS secundário
block	*	*	*	53	tcp	mas bloqueia qualquer outra zone transfer
allow	*	*	*	53	udp	permite perguntas ao DNS (via udp)
allow	ntp_outside	123	ntp.inside	123	udp	permite acesso para o network time
block	*	*	*	69	udp	bloqueia acesso ao servidor TFTP
block	*	*	*	87	tcp	sem serviço externo de link
block	*	*	*	111	tcp	sem chamadas externas ao rpc (via tcp)
block	*	*	*	111	udp	sem chamadas externas ao rpc (via udp)
block	*	*	*	2049	udp	sem uso externo do NFS (via udp)
block	*	*	*	2049	tcp	sem uso externo do NFS (via tcp)
block	*	*	*	512	tcp	sem uso externo do rexec
block	*	*	*	513	tcp	sem uso externo do rlogin
block	*	*	*	514	tcp	sem uso externo do rsh
block	*	*	*	515	tcp	sem uso externo do lpr
block	*	*	*	540	tcp	sem uso do uucpd
block	*	*	*	6000-6100	tcp	sem uso externo dos serviços X window
allow	*	*	admin_net	444	tcp	permite acesso cifrado à administração
block	*	*	admin_net	*	tcp	mas bloqueia qualquer outro tráfego
block	pclab_net	*	*	*	tcp	estudantes no laboratório não tem acesso externo
block	pclab_net	*	*	*	udp	nem via serviços UDP
allow	*	*	*	*	tcp	todo outro tráfego TCP é permitido
block	*	*	*	*	udp	mas o tráfego UDP é bloqueado

Figura 4 - Regras de filtragem típicas

Note-se que as regras são aplicadas de cima para baixo. Se uma regra bloqueia o pacote, nenhuma outra regra é executada. Os nomes usados para fonte e destino são fictícios e tem somente propósitos ilustrativos no exemplo.

Quando um *firewall* for construído, várias arquiteturas diferentes podem ser montadas. As mais comuns são [4], [9]:

- **Screening router**: utiliza somente um filtro de pacotes separando a rede interna da externa. Não é efetivo a nível de aplicativos, pois não examina os conteúdos dos pacotes.
- **Dual homed host**: utiliza somente um *bastion host* separando a rede interna da externa. O roteamento direto (*IP forwarding*) é desabilitado e todo roteamento é realizado a nível dos aplicativos. Todos os serviços são realizados por servidores *proxy*.
- **Screened subnet**: utiliza dois filtros de pacotes e uma sub-rede para os *bastion hosts*, como mostrado na figura 5. É a arquitetura mais cara mas também a mais recomendada, pois cada componente individual (filtros e *hosts*) tem funções simples e portanto são fáceis de configurar.
- **Screened host**: utiliza um filtro de pacotes entre as redes externa e interna. Este filtro direciona todo o tráfego externo para um ou mais *bastion hosts*, que por sua vez realizam o roteamento a nível dos aplicativos, atuando como servidores *proxy*. É basicamente uma arquitetura screened subnet sem o roteador (filtro) interno.

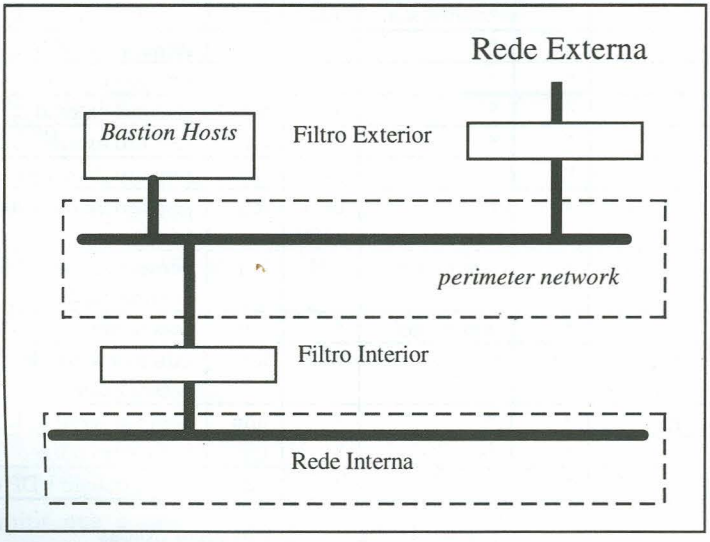

Figura 5 - Screened Subnet

A arquitetura mais segura é a *screened subnet,* que emprega um *bastion host* (ou mais) e dois filtros de pacote. Ela adiciona uma camada extra de segurança, através de uma *perimeter network* que constitui uma subrede entre a rede interna e a Internet. A principal vantagem deste esquema é garantir segurança mesmo no caso dos *bastion hosts* serem comprometidos. Os *bastion hosts* são as máquinas mais expostas e por isso as mais sujeitas a ataques. Mesmo sendo utilizados métodos robustos de segurança de *host*, existe alguma

chance delas serem violadas. Porém, estando o atacante de posse de um *bastion host*, o seu acesso será restrito a *perimeter network*.

4.3 Mecanismos de autenticação

A grande maioria dos serviços da Internet que necessitam identificar um usuário utilizam o mecanismo "alguma coisa que você sabe", ou seja, senhas. Apesar de ser um método fácil, rápido e barato, a utilização de senhas é um grande problema de segurança, pois elas podem ser capturadas por *sniffers* ou descobertas por um ataque do dicionário. Uma solução possível é utilizar senhas que não são reusáveis, como as *one time passwords* [10]. A senha ainda pode ser capturada, mas ela não terá nenhuma utilidade para o atacante.

A utilização destas senhas de uso único requer naturalmente o uso de programas clientes e servidores que as suportem, ou então o emprego adicional de um servidor de autenticação ativado antes do serviço real por um *wrapper* adequado. Tipicamente um sistema deste tipo apresenta ao usuário um desafio (*challenge*). O usuário realiza cálculos sobre este desafio, incluindo uma *pass phrase* secreta, e devolve o resultado ao servidor, que verifica a exatidão do resultado. Como a *pass phrase* não circula pela rede, não há o perigo dela ser capturada. Naturalmente, os cálculos necessários estão fora da capacidade do usuário comum, e por isso eles são realizados pelo próprio programa cliente ou através de um aplicativo específico (uma calculadora especial).

Três sistemas estão disponível pela própria rede:

- S/Key, disponível em ftp.bellcore.com
- OPIE, disponível em ftp.nrl.navy.mil
- LogDaemon, disponível em ftp.win.tue.nl, one também podem ser encontrados outros utilitários como TCP Wrapper, Portmap, Crack, COPS, Tiger e SATAN.

Além da autenticação de usuários, outro problema é a autenticação de *hosts*. As técnicas atuais, de autenticação via DNS reverso, são insuficientes e podem ser enganadas por ataques ao nível do protocolo IP. Uma forma de solucionar o problema é utilizando criptografia. Com isto tem-se privacidade, pois os dados circulam cifrados, e ao mesmo tempo autenticidade, pois somente as máquinas que possuem as chaves corretas conseguirão decifrar corretamente os dados.

4.4 Ferramentas de detecção de falhas

Diversos programas foram desenvolvidos ao longo da história da Internet para verificarem aspectos relacionados com segurança e analisarem todas as máquinas de um domínio à procura de falhas de configuração ou utilização. A lista s seguir ilustra algumas destas ferramentas [6]:

- SATAN (Security Analysis Tool for Auditing Network): é o sistema de auditoria mais completo disponível em domínio público. Ele pode ser utilizado em três modos (leve, normal e pesado), dependendo da profundidade da análise que se deseja. SATAN procura em todos os serviços oferecidos por erros de configuração e *bugs* conhecidos.
- COPS (Computer Oracle and Password Program): outro sistema de auditoria, que inclui verificação do uso de *passwords* "fáceis".
- TIGER: é um conjunto de *scripts* e programas que também visam a auditoria de um sistema, mas sua ênfase maior está em erros de permissões de arquivos.
- Tripwire: é um analisador de integridade de arquivos e diretórios. Tem como objetivo detectar alterações indevidas, como as que ocorrem tipicamente em uma intrusão do sistema. Utiliza funções de hash unidirecionais, como MD5, SHA e CRC de 32 bits.
- Crack: visa testar as senhas de um sistema Unix, realizando sobre elas o ataque do dicionário.
- Npasswd e Passwd+: programas que realizam uma série de testes (programáveis) sobre as senhas no momento que os usuários as registram.
- Swatch (Simple Watcher): monitora os arquivos de *log* e permite ao administrador realizar ações específicas em resposta à ocorrência de diversos eventos.
- Trimlog: gerencia arquivos de *log*.

4.5 Métodos criptográficos

O fato da informação trafegar pela rede em forma clara permite que um intruso qualquer capture pacotes e leia esta informação, como uma mensagem de *e-mail*, um formulário WWW, os comandos executados via telnet, etc. Uma maneira de impedir este tipo de ataque é utilizar métodos criptográficos, com criptografia de chave única (ou simétrica), criptografia de chave pública, assinatura digital e funções de hash unidirecionais [12]. Alguns sistemas que utilizam estes métodos são listados a seguir:

- PGP (Pretty Good Privacy): desenvolvido por Philip Zimmerman em 1991, destina-se a comunicação segura via correio eletrônico. Utiliza o algoritmo de chave única IDEA, a função de hash MD5 para integridade e o algoritmo de chave pública RSA para gerenciamento de chaves e assinatura digital. As chaves podem variar entre 512 e 1024 bits. Não utiliza nenhuma autoridade de certificação de chaves, mas é o próprio usuário que distribui suas chaves públicas [8].
- RIPEM (Riordan's Internet PEM): uma implementação das normas PEM (Privacy Enhanced Mail). Utiliza MD2 ou MD5 para verificação e integridade, DES para cifragem com chave única e RSA para gerência de chaves. Sua especificação pode ser encontrada nos RFC 1421 a 1424 [12].

- SHTTP (Secure HTTP): um conjunto de protocolos adicionados ao HTTP, que suportam vários métodos para estabelecimento de chaves e múltiplos algoritmos de criptografia.
- SSL (Secure Sockets Layer): desenvolvido pela Netscape Communications, tem como objetivo gerar segurança e privacidade entre duas aplicações, como HTTP, telnet ou FTP. No início da comunicação são trocadas entre as aplicações quais a versão do protocolo e os algoritmos a serem utilizados, é realizada a autenticação mútua e finalmente a negociação das chaves de criptografia.
- PCT (Private Communication Technology): desenvolvido pela Microsoft Corporation, é semelhante ao SSL.
- Kerberos: um protocolo de autenticação, desenvolvido originalmente no MIT. Um servidor Kerberos atua como um árbitro confiável, mantendo uma base de dados de clientes e suas chaves secretas [12]. Assim, duas pessoas (ou computadores) podem se autenticar mutuamente através do Kerberos. Seu código fonte é disponível no MIT.
- SET (Secure Electronic Transaction): desenvolvido pela Visa e Mastercard para implementar segurança em transações comerciais. Sua referência pode ser encontrada em http://www.mastercard.com/set/set.html

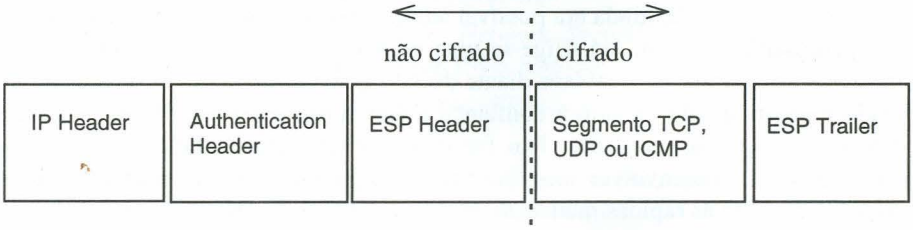

Figura 6 - pacote *secure IP*

- IPv6: a nova versão do Internet Protocol. Possui suporte a autenticação e privacidade, implementadas através de extensões do cabeçalho, seguindo o cabeçalho principal [16]. O campo destinado a autenticação é o *authentication header* (AH) e o para privacidade é o *encapsulating security payload* (ESP), conforme ilustrado na figura 6. O campo AH provê suporte para autenticação dos pacotes. O seu principal campo é o *authentication data* que contém dados dependentes do algoritmo de autenticação empregado. Este campo é calculado sobre todo o pacote IP, excluindo campos que são alterados quando em trânsito. O ESP provê suporte para privacidade. Este cabeçalho começa com o campo SPI (*Security Parameters Index*); o restante, se existir, contém parâmetros que são dependentes do algoritmo de criptografia utilizado. Dependendo das necessidades

do usuário, o mecanismo pode cifrar o segmento de transporte (TCP, UPD, ICMP) ou todo o pacote IP. No primeiro caso, será criado um *transport-mode ESP*, e no segundo um *tunnel-mode ESP*. Maiores detalhes podem ser encontrados nos RFCs 1825 a 1829.

5. Conclusões

Com o contínuo crescimento da Internet, com a interligação das redes internas a ela, e com a existência de um espectro de aplicações muito maior que os seus projetistas jamais sonharam. Infelizmente, com o crescimento da Internet cresceram também, e em uma escala muito maior, os seus problemas de segurança. As palavras de Tsutomu Shimomura, que sofreu ataques de *hackers* e realizou uma caçada pela rede que culminou na prisão de Kevin Mitnick, ilustram bem este problema: "A rede de computadores conhecida como Internet começou como um experimento único de construir uma comunidade de pessoas que compartilhavam uma escala de valores sobre a tecnologia e o papel que computadores podem desempenhar para mudar o mundo. Essa comunidade era baseada, em grande parte, em um senso de confiança mútua. Hoje, as muralhas eletrônicas que se erguem por toda a parte na rede constituem a prova mais clara de que esta confiança e esta comunidade se perderam. É uma perda para todos nós." [13]

Se na década de 80 ainda era possível administrar um domínio da Internet com uma postura permissiva, hoje em dia exige-se uma grande dose de prudência. Os aspectos de segurança se tornam mais complexos diante do fato que a maioria dos serviços da rede foi projetado e desenvolvido quando a confiança mútua ainda existia na rede, e os casos de vandalismo eram raros e isolados. Isto faz com que um administrador de uma rede ou domínio tenha que desenvolver uma política de segurança séria e estar continuamente atento para adaptá-la às rápidas mudanças que ocorrem na Internet.

Este artigo somente abordou superficialmente os principais tópicos relacionados com segurança. Para uma detalhamento mais profundo, recomenda-se a leitura da bibliografia indicada ao final do artigo. E, se é da Internet que provêm as maiores ameaças, também é nela que as ferramentas necessárias para aumentar a segurança podem ser encontradas. Recomenda-se os grupos de *news* comp.security.*, comp.virus, sci.crypt, alt.security.*, e alt.hacker assim como as páginas na WWW do CIAC (Computer Incident Advisory Capability - http://ciac.llnl.gov), COAST (Computer Operations, Audit and Security Technology - http://www.cs.purdue.edu/coast/coast.html), FIRST (Forum os Incident Response and Security Teams - http://www.first.org/first) e CERT (Computer Emergency Response Team - http://www.cert.org).

Informações sobre o *toolkit* para *firewalls* da TIS (Trusted Information Systems) pode ser encontrada em http://www.tis.com; outras informações sobre *firewalls* podem ser encontradas em ftp://greatcircle.com/pub. De interesse é também a pesquisa da AT&T, disponível em http://www.research.att.com. RFCs podem ser obtidos por ftp em

server.ds.internic.net, sendo de interesse para segurança, entre outros os de números 1535 a 1537 (sobre DNS), 1507 a 1510 (autenticação), 1421 a 1424 (PEM), 1319 a 1321 (message digest), 1244 e 1281 (segurança), 1825 a 1829 (IP versão 6).

Referências

[1] Bruno, Lee. Internet Security: how much is enough? Data Communications, april 1996.

[2] Bryan, J. *Build a Firewall.* Byte International Edition, vol.20, no.4, april 1995, p.91-103.

[3] Chapman, D. B. e Zwicky, E.D. *Building Internet Firewalls.* O'Reilly & Associates, setembro 1995, 517p.

[4] Cheswick, William e Bellovin, Steven M. *Firewalls and Internet Security: repelling the Wily Hacker.* Addison Wesley, abril 1995, 306p.

[5] Communications of ACM. *Special Issue on the Internet Worm.* Vol.32, no.6, june 1989.

[6] Dandrea, Marcelo Mercio. *Ferramentas para Segurança na Internet.* Trabalho Individual no.609, Pós-Graduação em Ciência da Computação, Instituto de Informática, UFRGS, jdezembro 1996, 42p.

[7] Gallen, Bob e Sutterfield, Lee. *Network Security Points of Failure.* Unix Review, vol.14, no.12, november 1996, p.47-53.

[8] Garfinkel, Simson. *PGP: Pretty Good Privacy.* O'Reilly and Associates, 1995, 393p.

[9] Garfinkel, Simson e Spafford, Gene. P*ractical Unix & Internet Security, 2nd edition.* O'Reilly & Associates, abril 1996, 971p.

[10] Hare, Chris e Siyan, Karanjit. *Internet Firewalls and Network Security, 2nd edition.* New Riders Publishing, 1996, 631p.

[11] Kaufman, Charlie, Perlman, Radia e Speciner, Mike. *Network Security: Private Communication in a Public World.* Prentice Hall, 1995, 504p.

[12] Schneier, Bruce. *Applied Cryptography, 2nd edition.* John Wiley & Sons, 1996, 758p.

[13] Shimomura, Tsutomu e Markoff, John. *Contra-ataque: a história da captura do pirata cibernético mais procurado dos Estados Unidos.* Companhia das Letras, 1996, 343p.

[14] Sparta de Souza, Rafael Mattos. U*m estudo de vulnerabilidade de serviços da Internet.* Trabalho de Diplomação, Bacharelado em Ciência da Computação, Instituto de Informática, UFRGS, janeiro 1997, 62p.

[15] Spohn, Marco Aurélio. *Internet Firewalls*. Trabalho Individual no.554, Pós-Graduação em Ciência da Computação, Instituto de Informática, UFRGS, junho 1996, 69p.

[16] Stallings, William. *Interner Armour*. Byte International Edition, vol.21, no.12, december 1996, p.127-134.

[17] Strauch, Suzana B. M. *Análise de Segurança na Internet*. Trabalho Individual no.632, Pós-Graduação em Ciência da Computação, Instituto de Informática, UFRGS, janeiro 1997, 44p.

[18] Tanenbaum, Andrew. Computer Networks, 3rd edition. Prentice Hall, 1996, 814p.

Agradecimentos

Por todo o auxílio que têm prestado na minha pesquisa sobre segurança, e pela grande ajuda na confecção deste artigo, desejo aqui agradecer ao formando do Curso de Ciência da Computação da UFRGS Rafael Mattos Sparta de Souza e aos alunos do curso de mestrado do Pós-Graduação em Ciência da Computação da UFRGS Marco Aurélio Spohn, Suzana Beatriz de Miranda Strauch e Marcelo Mercio Dandrea.