# Uma Análise das Vulnerabilidades dos *Firewalls*

*Keesje Duarte Pouw*
E-mail: duarte@dcc.unicamp.br

*Paulo Lício de Geus*
E-mail: paulo@dcc.unicamp.br

Departamento de Ciência da Computação
Universidade Estadual de Campinas
Campinas, São Paulo

**Resumo**

Internet *Firewalls*[1] têm-se tornado um elemento indispensável para se proteger uma rede privada contra ataques externos. No entanto, muitas vezes não é claro o nível de protecão oferecido por tais dispositivos, bem como suas deficiências e circunstâncias nas quais estas estejam envolvidas. Tais questoes, agregadas a possíveis solucoes serão apresentadas e analisadas com o intuito de oferecer uma melhor visão das limitacoes e facilidades advindas do uso de *Firewalls* como elemento protetor de uma rede local.

**Abstract**

Internet Firewalls have became a must with respect to protect a private network from external threats. Nevertheless, it is not allways clear the level of protection offered for such devices, as well its vulnerabilities and the circumstances those are related. Such issues, with possible solutions will be presented and analised aiming (willing) to offer a better insight of the limitations and facilicilities that come along with the use of Firewalls.

## 1. Introducão

Para proteger uma rede privada contra as ameacas externas oriundas de uma conexão com a Internet, dois enfoques são possíveis: "reforcar" a seguranca dos sistemas de rede e servicos em geral; e isolar a rede interna, restringindo o acesso externo através de um *firewall.*

*Firewalls* são portanto um conjunto de sistemas (uma ou mais máquinas/roteadores) situado entre uma rede privada e a Internet, que têm como principal funcão interceptar

[1] Apesar do termo *Firewall* apresentar um conotacão genérica, o mesmo será usado neste artigo em referencia a Internet *Firewalls.*

todo tráfego entre ambos, e com base na política de segurança interna, permitir ou não a sua passagem [Ranum95].

Conforme o nível onde atuam, (modelo OSI) os *firewalls* são classificados em roteadores escrutinadores (*firewalls* a nível de rede) e *Application Gateways* (*firewalls* a nível de aplicação). Roteadores escrutinadores (*screening routers*) são roteadores com capacidade de efetuar filtragem de pacotes com base nas informações contidas até o nível de rede, tais como: endereço IP de origem e destino, protocolo de transporte (TCP/UDP), portas de origem e destino dos protocolos de transporte, etc. *Application Gateways* são programas específicos (*proxy applications*), implementados para cada aplicação que se queria permitir passagem através do *firewall*. Pelo fato de atuarem na camada de aplicação, os *application gateways* oferecem normalmente maior controle de acesso que os roteadores escrutinadores [Ranum95].

Na prática, uma série de combinações de configurações são possíveis na construção de um *firewall.* A*pplication level gateways* podem ser implementados em *dual-homed gateways* - máquinas com duas interfaces de rede e capacidade de rotear pacotes (*ip-forward*) desabilitada, isolando uma rede privada - ou com auxílio de roteadores escrutinadores (*screening routers*). Topologias que fazem uso apenas de *Firewalls* a nível de rede também são comuns. Vantagens e desvantagens do uso de cada um ou ambos dos modelos acima serão analisadas, sem no entanto ater-se a alguma topologia específica ou às várias combinações destas[Ranum92][Ranum93].

Como já fora indiretamente mencionado, o enfoque em um estudo sobre *Firewall* é a defesa de uma rede privada em relação ao mundo externo (Internet). *Firewalls* por si só oferecem pouca ou nenhuma proteção contra ataques oriundos de usuários da própria rede privada. Estes portanto, oferecem defesa ao perímetro de segurança mas não necessariamente incrementam o nível de proteção individual de cada máquina localmente. Preocupações com a proteção de cada máquina em si constituem um tópico de extrema importância, mas à parte do objetivo central deste artigo.

Infelizmente, devido a superficialidade com que o tema *firewall* foi introduzido será exigido do leitor alguma familiaridade com os conceitos e topologias básicas dos mesmos [Cheswick&Bellovin94][Chapman&Zwicky95], bem como com o protocolo TCP/IP [Comer95][Stevens94].

De uma maneira geral, as vulnerabilidades de um sistema operacional de rede[2] em si - descartando por exemplo falhas de configuração, instalação e decorrentes de ataques que exploram engenharia social, dentre outros - podem ser divididas basicamente em três áreas:

- Falhas do sistema operacional e *software* em geral;
- Vulnerabilidade dos protocolos TCP/IP;
- Deficiências de autenticação de usuários.

As vulnerabilidades dos *Firewalls* e sistemas de rede em geral serão vistas, dentro do possível, de maneira simples e generalizada. A idéia será definir classes de falhas e suas possíveis formas de prevenção.

---

[2] Em específico o sistema operacional UNIX e as redes TCP/IP.

Com intuito de oferecer uma visão global do artigo e de orientar o leitor através do texto, segue abaixo um esboço do mesmo:

- **Falhas de *software*:** i) Manter uma configuração "Minimal"; ii) Restringir ao Máximo os Privilégios dos Processos; iii) Manter *Software* Atualizado de correções e falhas;
- **Vulnerabilidades do Protocolo TCP/IP:** i) Criando Rotas para Ataque; ii) ICMP (Internet Control Message Protocol); iii) UDP (User Datagram Protocol); iv) Sequence Number Attacks (SNA); v) TCP Sequence Number Prediction Attack (SNPA); vi) DNS (Domain Name System);
- **Amenizando as Vulnerabilidades do Protocolo TCP/IP:** i) Isolar a Rede Privada; ii) Desativar Roteamento Dinâmico; iii) Evitar IP Spoofing Externo;
- **Vulnerabilidades do Meio Físico:** i) Grampeando a Rede; ii) Mapeamento entre Endereço IP e Endereço Ethernet (ARP);
- **Autenticação de Usuários:** i) Sequestro de Sessão (Session Hijacking Attack - SHA);
- **Negação de Serviço e "Ataques de Baixo Nível":** i) Negação de Serviço (Denial of Service Attacks); ii) "Ataques de Baixo Nível" e Stealth Scanning; iii) Packet State Filtering (PSF);
- **Adicionando criptografia aos Firewalls:** i) Criptografia de Firewall para Firewall; ii) Criptografia de Usuário para Firewall;
- **Conclusão.**

## 2. Falhas de *software*

Comportamento inesperado de programas, quer seja por falha de projeto e/ou implementação, tem invariavelmente permitido acesso não autorizado a sistemas que possuam tais falhas.

Quanto maior a complexidade e tamanho de um programa, mais difícil é "prever" seu comportamento e consequentemente garantir que este não apresente falhas que possam comprometer a segurança do sistema ao qual pertença. Existe ainda o complicador de que muitos destes programas são executados com excesso de privilégios, em geral como *root.*

Apesar de, no sistema operacional UNIX, vários *daemons* apresentarem tal problema, o caso mais notável da dificuldade de se desenvolver aplicações complexas de maneira segura é do programa *sendmail.* Suas falhas, conhecidas ou ainda por serem descobertas, foram e seguramente continuarão sendo exploradas por *hackers*[3].

Ao se instalar um *firewall,* os seguintes conceitos devem ser seguidos tendo em vista amenizar falhas de *software* na segurança dos mesmos:

---

[3] Mesmo sob risco de se criar controvérsia, o termo *hacker* será usado para designar pessoas que tenham por objetivo a invasão de sistemas alheios.

## 2.1 Manter uma configuracão "Minimal"

Partindo-se do pressuposto de que todos os programas possuem falhas, a maneira mais eficaz de se prevenir contra ataques que exploram este tipo de vulnerabilidade em um *firewall* é configurá-lo de forma "minimal". Deve-se executar o mínimo numero de processos necessários para garantir a funcionalidade desejada, e preferencialmente processos mais simples e pequenos, que possam assim ser testados durante o processo de implementacão.

*Smap* é um exemplo de um *wrapper client,* disponível no TIS *toolkit* - *FWTK* [Avolio&Ranum][Avolio&Ranum94][TIS93a][TIS93b], que implementa uma versão minimal do protocolo SMTP, aceitando mensagens via rede para posterior entrega pelo programa *sendmail*[TIS94a]. *Smap* foi projetado para executar suas funcões sem necessidade de privilégios extra (executar como *root)* e sua implementacão consumiu 700 linhas de codigo contra 20.000 de *sendmail.*

Apesar de algumas implementacões comerciais assim o fazerem, não nos parece adequada, por exemplo, a construcão de *Firewalls* com base em interfaces gráficas devido ao tamanho e complexidade do codigo envolvido. Isto sem mencionar o enorme numero de falhas de operacão e seguranca do pacote *X-Windows* e seus derivados até então já comprovadas.

Finalmente, um *firewall* construído sobre um sistema UNIX system V com configuracão mínima teria aproximadamente os seguintes processos em execucão:

```
# ps -ef
   UID   PID  PPID  C    STIME TTY      TIME COMD
   root     0     0 80 18:48:01 ?        0:02 sched
=> Scheduler, pai dos processos básicos init, pageout e fsflush.
   root     1     0 61 18:48:04 ?        0:01 /etc/init-
=> Define estado da máquina (single user, multi-user, etc.).
   root     2     0  0 18:48:04 ?        0:00 pageout
=> Controla funcoes de paginacão e swap.
   root     3     0 37 18:48:04 ?        0:00 fsflush
=> Efetua system call sync (sincronismo entre memoria e disco).
   root   106     1 80 18:48:48 console  0:02 -ksh
   root   140   106 22 18:52:28 console  0:00 ps -ef
   root    88     1 38 18:48:35 ?        0:00 /usr/sbin/syslogd
=> Sistema de "log" centralizado do UNIX.
   root    85     1 32 18:48:34 ?        0:00 /usr/sbin/inetd -s
=> Internet service daemon. Dispara servicos definidos no arquivo inetd.conf sob demanda.
   root    98     1 46 18:48:46 ?        0:00 /usr/sbin/cron
=> Sistema de automatizacão de tarefas do UNIX (crontab).
```

Os processos *sched*, *init*, *pageout* e *fsflush* são inerentes ao sitema operacional UNIX, estando presentes em qualquer configuracão deste sistema. Por questoes de desempenho, alguns servicos podem ser disponibilizados através de *daemons* específicos, tal como disponível na nova versão do FWTK (2.0 alpha).

## 2.2 Restringir ao Máximo os Privilégios dos Processo

É, sem dúvida, uma boa política configurar os processos com o mínimo de acesso e privilégios necessários no sistema, para que os mesmos possam desempenhar suas funcoes. Os danos que podem ser causados por uma falha de *software* em um

processo, que é executado como *root,* com certeza serão menores caso o processo seja executado como um usuário com menos privilégios.

Outro recurso disponível em sistemas UNIX é restringir o acesso de processos vulneráveis a uma determinada hierarquia do sistema de arquivos (*chroot system call)*, isolando-os dos arquivos de configuração e binários do sistema. Este tipo de precaução é normalmente empregado na instalação de serviços públicos na Internet: FTP anônimo, servidor HTTP, GOPHER, etc. No entanto, este mecanismo de proteção não é à prova de falhas, sendo possível a um *hacker* sofisticado burlar a limitação de hierarquia. A instalação de serviços públicos em geral deve preferencialmente ser efetuada fora do *firewall,* possivelmente em uma sub-rede isolada (conhecida por "zona desmilitarizada") e protegida por roteadores escrutinadores[Ranum92][Ranum93].

### 2.3 Manter *Software* Atualizado de correções e falhas

Finalmente, os programas em execução em um *firewall* devem corresponder à última versão no que se refere a correção de falhas de segurança. Listas de discussão na Internet[4] como *firewalls, bugtraq* e organizações como CERT (Computer Emergency Response Team) oferecem informações atualizadas sobre falhas de segurança e incidentes relacionados. Manter-se atualizado e bem informado é sem dúvida uma tarefa essencial a um *gatemaster*[5].

A título de ilustração, recentemente foi anunciado que o serviço *chargen* (*Character stream generator* - Porta 19 TCP/UDP), que até então era considerado inofensivo pela literatura, tem sido usado em ataques de negação de serviço.

## 3. Vulnerabilidades do Protocolo TCP/IP

A principal deficiência do protocolo TCP/IP é a incapacidade deste de autenticar uma máquina na rede. Em outras palavras, com base no endereço IP de origem de um pacote recebido, é impossível determinar com certeza a identidade da máquina que o tenha originado. Há também poucas garantias de que o conteúdo de um pacote recebido não tenha sido alterado, muito menos ainda que a privacidade dos dados nele contidos tenha sido preservada.

Os ataques que exploram tal falha têm como tática mais comum a personificação de uma máquina na rede. A finalidade é a de obter informações sigilosas como senhas, abusar da confiança que máquinas mantêm entre si, até ações mais sutis e sofisticadas como alterar o conteúdo dos dados que estejam de passagem para outros destinos.

O enfoque aqui abordado será analisar o impacto dessas vulnerabilidades na segurança de um *firewall,* bem como o papel destes na prevenção do problema. Para uma visão mais específica das deficiências e soluções adotadas para cada protocolo

---

[4] Referências [Cheswick&Bellovin94] e [Chapman95] apresentam em seus apêndices informações detalhadas de como acessar listas e organizações que tratam de falhas de *software* e incidentes relacionados a segurança em geral.

[5] Aquele, que por sugestão dos autores, deve ser chamado o responsável pela manutenção de um *firewall,* seguindo as linhas de *postmaster e webmaster.*

do pacote TCP/IP, o leitor interessado deve consultar o clássico artigo de Steven Bellovin sobre o assunto[Bellovin89]. Porém, antes de sugerir soluções, é necessário compreender um pouco mais a fundo os procedimentos que possibilitam personificar uma máquina na rede, e a partir daí, as formas de prevenção.

## 3.1 Criando Rotas para Ataque

A dificuldade em se personificar completamente uma máquina está em fazer com que pacotes enviados à máquina legítima cheguem ao falso destino. Para que isto seja possível, os roteadores entre o alvo de ataque e o *hacker* devem de alguma forma ser subvertidos. Isto pode ser efetuado de duas maneiras:

- Utilizando a opção *source routing* [Comer95] do protocolo IP.

Pacotes IP que contenham tal opção são por *default* roteados não com base nas tabelas de rota dos roteadores, mas conforme determinado a priori pelo originador do mesmo.

- Alterando as tabelas de rotas dos roteadores envolvidos.

Os protocolos de propagação de rotas (notavelmente RIP), tipicamente não têm como averiguar as informações recebidas. Isto torna então possível alterar todas as tabelas de rotas entre duas máquinas de maneira a personificar uma terceira.

Na seção seguinte será mostrado que ICMP *redirect messages*[Comer95] [Stevens94], podem ser exploradas com o mesmo intuito que os protocolos de propagação de rotas.

## 3.2 ICMP (Internet Control Message Protocol)

O protocolo ICMP pode ser usado para implementar ataques de negação de serviço através de mensagens *destination unreachable* ou por *address mask replies* falsos [Comer95][Stevens94]. Para amenizar o efeito deste tipo de ataque, mensagens ICMP deveriam sempre estar atreladas a uma conexão específica [Cheswick&Bellovin94]. Tendo em vista tal objetivo, os primeiros 8 bytes (no caso do Solaris, 64 bytes por *default,* definido pela variável: *icmp_return_data_bytes*) do cabeçalho da camada de transporte (UDP/TCP) são sempre incluídos nas mensagens ICMP. Contudo, implementações ICMP antigas não fazem uso desta informação e todas as conexões entre duas máquinas são então afetadas. Claramente, estas implementações são extremamente vulneráveis a ataques de negação de serviço que serão futuramente melhor abordados.

Mensagens geradas por roteadores escrutinadores, em consequência de pacotes enviados à portas bloqueadas, podem "derrubar" todas as conexões com uma determinada máquina. A geração de mensagens *destination unreachable* por parte de um *firewall* deve ser feita de maneira a não prejudicar conexões legítimas.

No entanto, o perigo maior reside na possibilidade de se injetar rotas falsas através de ICMP *redirect messages.* O que torna este tipo de ataque mais difícil de ser implementado é que este tipo de mensagem só pode ser gerada em função de conexões existentes, e deve ter sido originada de um *gateway* conectado na mesma sub-rede da máquina a ser atacada.

Em geral todas as implementacoes do protocolo ICMP realizam averiguacoes antes de alterar as tabelas de roteamento. Uma maquina 4.4BSD por exemplo, averigua as seguintes informacoes [Stevens94]:

- O novo roteador indicado pela mensagem deve estar em uma rede diretamente conectada;
- O *redirect* deve ter sido gerado pelo roteador atual do destino a ser alterado;
- A rota a ser modificada deve ser uma rota indireta;
- Redirect Messages so podem alterar rotas para maquinas e nao rotas para redes.

Considerando apenas o fator seguranca, roteadores e maquinas que implementam *firewalls*, nao deveriam atualizar suas tabelas de roteamento com base em *redirect messages*. Este tipo de mensagem, quando originada de redes externas, caracteriza problemas de configuracao ou tentativa de ataque. A melhor defesa neste caso e filtrar tais mensagens o quanto antes, de preferencia atraves de um roteador escrutinador.

## 3.3 UDP (User Datagram Protocol)

O protocolo UDP nao mantem nenhuma informacao (estado) a respeito das "conexoes" por ele estabelecidas. Em outras palavras, nao ha procedimento de estabelecimento de conexao, nem numeros sequenciais como no protocolo TCP. Isto torna UDP muito sucetivel a falsificacao de enderecos IP (IP *spoofing*).

Entretanto, por questoes de performance, o protocolo *Remote Procedure Call* da Sun (RPC)[RFC1057] foi implementado sobre UDP. Servicos de extrema importancia e sensibilidade do ponto de vista de seguranca, como NFS e NIS[Sun95], tiveram como base o RPC, e portanto padecem das mesmas limitacoes inerentes ao protocolo UDP[6]. Apesar de tambem apresentar problemas[LaMacchia&Odlyzko91], o protocolo *Secure RPC* e de longe mais confiavel que o RPC convencional e seu uso sempre que possivel deve ser encorajado.

## 3.4 *Sequence Number Attacks* (SNA)

Protocolos que mantem estado de suas conexoes (e.g. protocolo TCP) ou *queries* (e.g. DNS - Domain Name System e EGP - Exterior Gateway Protocol[Comer95]) por intermedio de numeros sequenciais sao, em geral, menos vulneraveis a IP *spoofing* que *stateless protocols*. Esta seguranca so existe no entanto, caso os numeros sequenciais usados nas transacoes nao possam ser previstos. E portanto de fundamental importancia que tais numeros sejam gerados da maneira mais aleatoria possivel.

Nao obstante, a maior parte dos geradores de numeros aleatorios utilizam realimentacao de suas saidas. Na verdade, os geradores de sequencias "aleatorias", que muitas vezes nem chegam a ser implementados, como no caso de muitos *resolvers* que sempre iniciam suas *queries* com um identificador igual a zero[Stevens94], sao facilmente previsiveis na maioria dos protocolos. Isto torna

---

[6] Afora as debilidades do protocolo UDP, os protocolos RPC, NFS e NIS apresentam vulnerabilidades proprias que fogem do escopo deste artigo.

possível a um *hacker*, como será descrito para o caso específico do protocolo TCP, prever a próxima sequência a ser utilizada por uma máquina e desta forma sobrepujar a "protecão" oferecida pelos números sequenciais.

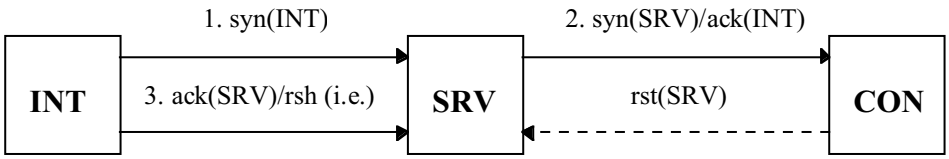
### 3.5 TCP *Sequence Number Prediction Attack* (SNPA)

TCP SNPA nada mais é que um caso específico de ataques SNA. No entanto, devido à importância do protocolo TCP e às peculiaridades envolvidas, este tipo de ataque será abordado em maiores detalhes.

Apesar de já ter sido previsto por Steve Bellovin[Bellovin89] em 1989, somente após o natal de 1994, quando Midnick usou desta técnica para penetrar nos computadores de Shimomura, que este ataque se tornou famoso, passando então, a não mais ser considerado uma proeza difícil de se implementar.

SNPA é baseado na capacidade de se prever o número sequencial inicial, a ser usado pela máquina alvo, no processo de estabelecimento de uma conexão TCP conhecido como "triplo aperto de mão"[Comer95] (*three way handshake*). Suponhamos então que um determinado intruso INT tenha previsto o próximo número sequencial inicial NSI a ser usado por um servidor SRV. O intruso (*hacker*) poderia então se passar por uma máquina confiável CON, emitindo comandos "R" de Berkeley, por exemplo, sem ter que se autenticar. O ataque se desenvolveria da seguinte forma (ver figura abaixo):

1. INT envia pacote: syn($NSI_{INT}$) para SRV, com endereco IP de origem de CON;
2. SRV responde com: syn($NSI_{SRV}$), ack($NSI_{INT}$) para CON;
3. Como INT "adivinhou" $NSI_{SRV}$, INT estabelece conexão enviando ack($NSI_{SRV}$) para SRV;
4. Dados maliciosos podem então ser enviados de INT para SRV. O interessante é que INT nem mesmo precisa receber os pacotes enviados por SRV.




NSI são passíveis de previsão, já que na maioria dos sistemas estes são derivados do último número sequencial usado. Então, a partir de uma conexão legítima é possível calcular o próximo número sequencial a ser usado por uma máquina, desde que novas conexões não se realizem. Não foi assim "por acaso" que o computador de Shimomura tenha sido invadido em uma noite de natal. Com acesso "exclusivo" a uma máquina, um *hacker* pode efetuar inúmeras conexões sucessivas até acertar um NSI definido por esta.

Apenas uma questão não foi considerada: ao receber a confirmacão da tentativa de estabelecimento de conexão vinda de SRV, que na verdade foi ilegalmente efetuada por INT, CON normalmente tentaria "derrubar" a conexão fraudulenta, enviando um pacote de *reset* (com a opcão RST habilitada) para SRV. Contudo, CON pode estar

inoperante, inclusive por obra de ataques de negacão de servico que serão tratados adiante.

### 3.6 DNS (Domain Name System)

Como DNS é um banco de dados distribuído, existe pouco controle sobre a veracidade das informacões divulgadas pelos inumeros domínios existentes na Internet. E possível, por exemplo, que um *hacker* que controle o mapeamento inverso de algum domínio, tenha algum endereco IP local (do *hacker*) sendo mapeado no nome de uma máquina confiável de um outro sistema qualquer (sistema alvo). Desta forma, este *hacker* pode se passar pela máquina confiável se a autenticacão no sistema alvo for baseada no nome de quem estabeleceu a conexão, como no caso dos comandos "R" de Berkeley [Bellovin95].

Tal tipo de ataque pode, e de fato é, evitado na maioria dos sistemas, comparando-se o mapeamento inverso com o mapeamento direto (*cross-check*) em sub-rotinas como *gethostbyname*. Mesmo assim, autenticacão baseada em nomes é ainda mais fraca que a baseada no enderecamento IP. Por este motivo sempre que possível arquivos de configuracão de sistema devem conter os enderecos IP das máquinas e não os seus nomes.

À parte o problema de autenticacão, DNS oferece em geral informacões preciosas (para um *hacker*) a respeito de uma rede privada. Uma boa política é a de não autorizar transferências entre servidores secundários (*zone transfer*)[Vixie95], mas isto pode não ser suficiente para um *hacker* determinado e paciente (eles geralmente o são). Este pode então "varrer" todo espaco de enderecamento via *queries* inversas.

Um enfoque mais "paranoico", proposto por Brent Chapman e descrito detalhadamente em [Chapman&Zwicky95], é ter um servidor de nome oferecendo o mínimo possível de informacão ao mundo externo e um outro servidor interno para uso doméstico. De forma bem resumida tem-se a seguinte situacão: as aplicacões no servidor externo seriam configuradas (através do arquivo /etc/resolv.conf) para enviar *queries* ao servidor interno. Este por sua vez, propagaria ao servidor externo (opcão *forward query*) questões relacionadas com os domínios externos. Desta maneira é possível isolar completamente as informacões de nomes internos e inclusive manter na rede privada, ainda que isto seja polêmico, enderecamento IP não oficial.

## 4. Amenizando as Vulnerabilidades do Protocolo TCP/IP

Mesmo com as falhas existentes, é possível em um ambiente local confiável proteger uma rede privada, de maneira a reduzir consideravelmente as chances de se sofrer uma invasão, em decorrência das vulnerabilidades do protocolo TCP/IP. Sem duvida, tal objetivo só é alcancado pela correta instalacão e configuracão de *Firewalls*. Vejamos a seguir algumas diretrizes básicas:

## 4.1 Isolar a Rede Privada

O objetivo principal de um *firewall,* é restringir o acesso direto de máquinas externas à rede privada. A idéia é reduzir os alvos potenciais de N para 1, no caso o *firewall.* Seguramente a disponibilização de acesso externo através de *application level gateways* é o que melhor oferece isolamento e controle de acesso específico para cada aplicação. A desvantagem é que cada serviço oferecido necessita de um *application gateway* próprio.

Ao se configurar um *dual-homed gateway,* grande cuidado deve ser tomado, no sentido de eliminar todas as formas possíveis de se rotear um pacote pelo *gateway.* No sistema operacional Solaris, por exemplo, isto envolve a desabilitação de pelo menos três parâmetros do *kernel: ip_forwarding, ip_src_routed, ip_forward_directed_broadcast.*

*Screening routers,* apesar de introduzirem flexibilidade na formação de topologias mais complexas podem, quando mal configurados, permitir a passagem de pacotes indesejados [Chapman92]. O importante é garantir através de testes exaustivos que nada além do programado possua trânsito entre a rede privada e a Internet. *Ip Filter*[Reed96] é um pacote de domínio público, para construção de um *screening router*[7] em um sistema UNIX (UNIX *box*), que apresenta ferramentas apropriadas para tais testes.

## 4.2 Desativar Roteamento Dinâmico

Neste sentido os cuidados a serem tomados são:

- Inibir roteamento baseado nas opções IP: *loose source e strict source routing* [Comer95];
- Rejeitar ICMP *redirect message* e mensagens ICMP em geral. Mais uma vez, em alguns sistemas operacionais, parâmetros do *kernel* podem ser definidos com tal intuito (*ip_ignore_redirect* no Solaris);
- Estabelecer roteamento estático. Na maioria dos casos é suficiente ao *firewall* conhecer apenas uma rota *default* para a Internet, que é geralmente definida em */etc/defaultrouter*. Caso necessário, rotas adicionais podem ser definidas manualmente pelo comando *route*.

## 4.3 Evitar *IP Spoofing* Externo

Em uma rede privada corretamente isolada, todos os pacotes entre esta e o mundo externo devem passar por um *firewall.* Deste modo é possível configurá-lo de maneira a recusar pacotes advindos da Internet que proclamem ter endereço IP de origem de alguma máquina da rede local.

Em *screening routers* é possível determinar por qual interface um pacote se originou. Assim, pacotes procedentes da interface conectada com a rede externa (provedor de serviço por exemplo), que apresentarem endereço IP de origem equivalente ao de

[7] I*p-filter é na realidade um State Packet Filter* que representa a última geração de *screening routers.*

alguma máquina interna, devem ser recusados e registrados como tentativa de *spoofing*. Mesmo em *dual-homed gateways* é possível evitar tal tipo de ataque, já que pacotes com origem e destino de máquinas da rede interna não deveriam passar pelo *firewall* (FWTK oferece tal protecão através da opcão *-dest* no arquivo de configuracão[TIS94b]).

É conveniente observar, que *Firewalls* implementados exclusivamente por *dual-homed gateways* não têm como proteger a si proprios, já que não possuem a capacidade de determinar por qual interface um pacote tenha sido originado. De qualquer forma, um *firewall* não deve estabelecer nenhuma relacão de confianca com qualquer outra máquina, tal como montar hierarquias via NFS ou consultar informacões pelo NIS.

# 5. Vulnerabilidades do Meio Físico

Até o presente momento nada foi discutido sobre as fragilidades das tecnologias de redes locais. Ataques que exploram estas vulnerabilidades são normalmente restritos às proprias redes locais, mas também podem ser usados, como descrito a seguir, para sobrepujar um *firewall*.

As deficiências da tecnologia *ethernet,* que constitui a maior parte das redes locais, expõem ainda mais as fragilidades da Internet. Os principais problemas estão relacionados com:

- Facilidade de se realizar grampo (*eavesdropping*);
- Falso mapeamento entre endereco de rede (IP) e endereco físico (ARP).

## 5.1 Grampeando a Rede

Sendo *ethernet* uma tecnologia de rede onde o meio físico é compartilhado, é possível configurar a interface de rede de uma máquina em modo "promíscuo", e assim receber todos os quadros transmitidos no meio. Em sistemas UNIX esta facilidade só esta disponível ao super usuário, mas em sistemas operacionais como DOS e MacOS não existe obviamente nenhuma restricão de acesso à interface.

Geralmente tem-se por objetivo obter informacões privilegiadas, como por exemplo senhas, mas também pode-se usar tal facilidade como passo na implementacão de ataques mais sofisticados, tal qual "sequestro de sessão", que será tratado adiante.

## 5.2 Falso Mapeamento entre Endereco IP e Endereco *Ethernet* (ARP)

O protocolo ARP (*Address Resolution Protocol*) mapeia um endereco IP em endereco *ethernet* enviando um *broadcast* com o endereco IP desejado. A máquina que tiver o endereco IP procurado, ou alguma outra agindo em nome daquela, responde com o par: endereco IP - endereco *ethernet*. Uma máquina mal intencionada pode então enviar respostas falsas, desviando todo o tráfego para si, tendo como objetivo personificar uma máquina, ou mais sutilmente, modificar os dados que estiverem sendo transmitidos entre duas outras máquinas. *Firewalls* e redes locais fisicamente inseguras devem mapear enderecos físicos de maneira estática. De qualquer forma

pouco pode ser feito em relacão a conexões externas que passem por redes *ethernet* inseguras.

# 6. Autenticacão de Usuários

Outra vulnerabilidade ainda não discutida é a autenticacão de usuários perante aplicacões e sistemas em geral. Senhas ainda continuam sendo usadas como principal forma de autenticacão na Internet. A facilidade de se forjar este tipo de sistema é notória e vem sendo explorada desde os tempos primordios da rede.

O princípio adotado por um conjunto de sistemas de autenticacão, conhecidos como *one time passwords*, é de se gerar "senhas" aleatorias que podem ser usadas apenas uma vez. Isto elimina o perigo de se ter a senha "adivinhada" ou descoberta de alguma outra forma (*eavesdropping* por exemplo).

Estes sistemas podem ser implementados de diversas maneiras. A forma original, desenvolvida pelas forcas armadas norte americanas, consiste de um dispositivo de bolso, podendo inclusive ser uma calculadora programável, que compartilha uma chave secreta com o sistema ao qual irá se identificar. Um desafio aleatorio é gerado após identificacão do usuário, este por sua vez é digitado no dispositivo que então computa a resposta (senha) a ser enviada. Há também implementacões que não requerem equipamento adicional. S/Key [Lamport81] é uma ferramenta (incluída no FWTK) que gera uma lista de senhas aleatorias impressas para cada usuário, a serem utilizadas quando o mesmo estiver fora da rede privada. Obviamente cada senha so pode ser utilizada uma unica vez.

## 6.1 Sequestro de Sessão (*Session Hijacking Attack* - SHA)

Mesmo que se tenha em mente *one time passwords,* não é possível evitar totalmente o acesso não autorizado à rede privada pela Internet. Para um *hacker,* que esteja estrategicamente posicionado entre um usuário legítimo fora da rede privada e o sistema alvo, existe a possibilidade de se poder tomar controle da sessão do usuário com o sistema, após este ter sido devidamente autenticado. Devido a restricão de "posicionamento" do *hacker,* este ataque também é classificado como uma espécie de *man_in_the_middle attack.*

Ao *hacker* é requerida a habilidade de acompanhar a sessão de autenticacão de um usuário legítimo, possivelmente grampeando uma rede *ethernet* que esteja entre ambos, e de interromper em tempo hábil a comunicacão entre o usuário e o sistema logo após este ter sido corretamente autenticado. Isto naturalmente pode ser feito, negando-se servico à máquina do usuário. Se for possível ao atacante, pelos métodos anteriormente descritos, alterar os dados enviados pelo usuário, não há sequer necessidade de se implementar negacão de servico.

Ainda que não sirva como consolo, é fundamental estabelecer que SHA é relativamente difícil de ser implementado devido às circunstâncias apresentadas anteriormente. Apesar de *one time passwords* estarem um patamar acima em termos de seguranca que senhas convencionais, o risco de se ter usuários acessando uma rede local via sistemas externos não deve ser desprezado no contexto atual da Internet.

# 7. Negacão de Servico e "Ataques de Baixo Nível"

## 7.1 Negacão de Servico (Denial of Service Attacks)

Talvez este seja um dos tipos de ataque mais difícil de se evitar por completo. É normalmente complicado até mesmo determinar com certeza que um ataque do gênero esteja ocorrendo ou tenha ocorrido. A geracão de "logs" e análise periódica dos dados armazenados, através de ferramentas como *swatch*[Hansen&Atkins93], é ainda a melhor forma de se lidar com este tipo de problema. Aliás, auditoria, juntamente com seguranca e autenticacão, é uma das funcões básicas que um *firewall* deve oferecer. A capacidade de agregar informacões a respeito do uso dos recursos do sistema (servicos em geral) por parte de cada usuário, é uma característica de grande valor que felizmente coincide com as necessidades de seguranca no controle de acesso externo. Infelizmente, ou felizmente dependendo do ponto de vista, caberia outro artigo para tratar o assunto.

Prevencões mais específicas normalmente referem-se à defesa da máquina em si (*defense in depth*)[Garfinkel&Spafford91] e ferem um pouco a linha geral deste artigo. Mesmo assim segue abaixo algumas consideracões para máquinas que venham a implementar um *firewall*:

- Restringir acesso de usuários comuns. De certa forma isto é uma condicão necessária (obviamente não suficiente) para se ter um *firewall*;
- Definir particões de disco especiais para diretórios ou arquivos que possam receber grande volume de dados como por exemplo arquivos de "log" e diretório PUB de FTP e mail;
- Mais uma vez procurar evitar instalar servicos públicos (FTP anônimo, servidor HTTP, etc.) em máquinas que facam parte do *firewall* em si;
- Limitar, quando disponível no sistema, número máximo de processos sendo executados e o acesso destes aos recursos do sistema, como percentagem de CPU e memoria;
- Procedimentos de *Tuning*, além de uma prática recomendada em geral, têm efeito significativo na disponibilidade do sistema, que na verdade é também um fator de medida de seguranca;
- Prevenir "ataques de baixo nível" (nível de rede) que serão tratados na secão seguinte.

## 7.2 "Ataques de Baixo Nível" e *Stealth Scanning*

Mesmo devidamente configurados inclusive a nível de *kernel* do sistema, os *Firewalls* estão sujeitos a ataques de baixo nível que exploram deficiências nas implementacões das camadas mais baixas do protocolo TCP/IP, e também a sondagens externas que procurem por tais vulnerabilidades.

*Internet Scanners,* como são chamadas as ferramentas de auditoria de rede, têm por objetivo procurar por servicos e falhas que possam comprometer um *firewall.* A

maior parte destas ferramentas baseia-se no estabelecimento de uma conexão TCP com todas as portas de uma máquina, de maneira a determinar se estas estão ativas ou não. Felizmente, TCP *wrapper*[Venema92] e outros programas afins, chamados *scanner detectors,* possuem a capacidade de contabilizar todas as conexões efetuadas, e a partir daí determinar a ação de tais *scanners.*

Por outro lado, não é necessário estabelecer uma conexão TCP para determinar se uma porta está ativa ou não. Enviando-se segmentos TCP em várias combinações de *flags-code bits* (SYN, ACK, FIN, RST, URG, PUSH), e sendo conhecida a resposta a ser devolvida por um determinado sistema, tem-se o mesmo efeito ou pior do que estabelecer uma conexão por completo, já que algumas combinações de *flags* não "permitidas", como por exemplo SYN/FIN, podem inclusive derrubar uma máquina (*denial of service attack*). Ao se estabelecer "conexões pela metade" enviando apenas um segmento SYN, o mesmo efeito pode ser alcançado estourando-se o *cache* de conexões semi-abertas do *kernel. Ipsend* é mais uma vez uma boa ferramenta para se procurar por tais vulnerabilidades.

De [Klaus95] foi extraída uma tabela com segmentos enviados e respostas recebidas[8] que exemplifica o que fora dito no parágrafo anterior:

| FLAG | RESP. PORTA ATIVA | RESP. PORTA INATIVA |
|---|---|---|
| SYN | SYN/ACK | RST ou nenhuma resposta |
| SYN/FIN | ACK ou SYN/ACK | RST |
| ACK | nenhuma resposta | RST |

*Firewalls* a nível de aplicação, ou melhor, *dual-homed gateways,* possuem a capacidade de melhor isolar uma rede privada e portanto evitar que as máquinas internas estejam sujeitas a este tipo de ataque. Em compensação estes não têm muitos recursos para protegerem a si próprios. Combinações de *application level gateways e packet state filters,* que serão tratados a seguir, constituem topologias mais seguras em geral, quando corretamente configuradas.

## 7.3 *Packet State Filtering* (PSF)

Os roteadores escrutinadores originalmente implementados tomavam suas decisões baseados unicamente nos pacotes recém-chegados. Nenhuma informação a respeito dos pacotes anteriormente tratados era levada em consideração no processo decisório vigente. Em outras palavras, nenhuma informação a respeito dos pacotes aceitos ou rejeitados era mantida.

Um dos primeiros problemas encontrados ocorre quando o tamanho de um pacote IP excede o limite estabelecido pelo maior quadro, que uma determinada rede local é capaz de transmitir (MTU - *Maximum Transfer Unit*)[Comer95]. Neste caso é necessário dividir o pacote IP em fragmentos menores que a MTU. Como o cabecalho

[8] As resposta variam de acordo com a implementação (sistema em questão). A Referência [Reed95] apresenta uma descrição completa dos vários pacotes possíveis e correspondentes respostas para vários tipos de sistemas.

dos protocolos de nível superior como TCP e UDP encontram-se no primeiro fragmento IP (*offset = 0*), alguma informação deve ser mantida caso se queira filtrar os demais fragmentos. Normalmente não é possível remontar o pacote IP original sem o primeiro fragmento, mas deixar os demais fragmentos passarem é um risco que deve ser evitado.

Com relação aos fragmentos IP, mais duas questões são interessantes de serem citadas. O primeiro fragmento (com *offset = 0*) pode chegar ao destino depois que algum outro já tenha chegado. Uma solução não ideal, porém segura, seria então descartar este fragmento [Reed95][Mogul89].

O outro problema refere-se a fragmentos que sejam menores que 68 octetos. Como o cabecalho IP pode ter até 60 octetos, 8 octetos de dados não são suficientes para caber todo o cabecalho do protocolo TCP (em 8 octetos é possível ter apenas os seguintes campos: porta de origem e destino e numero sequencial). Informações importantes como *flags* TCP (*code bits)* não têm assim como ser consideradas no processo de filtragem. Riscos também existem para fragmentos com *offset* = 1 (8 octetos), que podem desta forma sobrepor (alterar) informações do cabecalho, tais como os *flags* TCP, enviados em fragmento anterior, no processo de remontagem dos pacotes IP[Mogul91].

Os problemas de filtragem de fragmentos IP acima mencionados são mais claramente detalhados na RFC 1858[RFC1858] e não devem ser desconsiderados no processo de selecão de roteadores escrutinadores para construção de um *firewall.*

No protocolo TCP a decisão de se rejeitar ou aceitar uma conexão é efetuada sobre os pacotes que possuem o *bit* de sincronismo ligado. Os pacotes que não possuem tal *bit* ligado são incapazes de estabelecer conexão com qualquer serviço ativo, sendo neste sentido inofensivos. No entanto, estes pacotes podem ser usados, como visto anteriormente, na procura por serviços ativos - *stealth scanning.* Um *packet state filter* (PSF) que acompanhe os numeros sequenciais (*ack/sequence*) de uma conexão em andamento, pode evitar *stealth scanning* rejeitando os pacotes cujos numeros sequenciais não estejam dentro da janela correta.

Mesmo para protocolos *stateless* como UDP é possível, por intermédio de PSFs, adicionar alguma nocão de estado. A idéia é que, quando se envia um pacote UDP para fora, uma resposta é esperada dentro de um certo período de tempo no sentido oposto. PSFs podem com o princípio de *timeout* dar a idéia de conexão e direção no protocolo UDP, mesmo que ainda de forma um tanto rudimentar se comparado ao TCP. Mesmo assim como já fora mencionado, qualquer informação sobre o estado de uma "conexão" contribui para tornar esta transação mais segura na rede.

Por último, PSFs podem levantar listas do fluxo de pacotes gerados, e com isso oferecer uma análise de performance em baixo nível. Excessos de segmentos RST têm como ser detectados através dos log gerados e assim produzir pistas sobre o mau andamento das conexões e possivelmente tentativas de TCP SNPA ou *stealth scanning.*

# 8. Adicionando criptografia aos *Firewalls*

Solucões mais efetivas para os ataques que exploram as vulnerabilidades do protocolo TCP/IP certamente envolvem o uso de alguma técnica criptográfica. Infelizmente limitacões técnicas e comerciais (restricão de exportacão de *softwares* que usem criptografia) tendem a limitar o uso extensivo de criptografia.

Enquanto não se tem uma solucão de cunho genérico (se é que tal solucão realmente exista) talvez o melhor que se possa ter sejam "feudos seguros" defendidos por *Firewalls*. Novamente tem-se a questão de garantir acesso seguro a usuários que estejam fora dos limites de protecão, especialmente aqueles que estejam além dos limites físicos de uma rede privada e portanto sucetíveis a SHA (computacão nômade).

Uma vez que se faca uso de criptografia, tem-se por heranca suas fraquezas e ataques característicos. Uma análise destas vulnerabilidades, bem como um estudo sobre o assunto [Schneier96]em si, está fora dos limites propostos por este artigo. O objetivo principal não é tratar solucões específicas, mas apenas criar consciência para o problema em geral.

Há dois enfoques possíveis no uso de criptografia em *Firewalls*[Willoughby95]:

- Criptografia de *firewall* para *firewall,* e;
- Criptografia de usuário (externo) ao *firewall.*

## 8.1 Criptografia de *Firewall* para *Firewall*

Desta maneira é possível estabelecer comunicacão confiável entre dois *Firewalls* quaisquer na Internet. O que se tem na verdade é a definicão de uma rede privada virtual (*Virtual Network Perimeter*), ou seja a partir de duas redes locais separadas fisicamente, pode-se criar uma única rede virtual usando a estrutura de comunicacão existente na Internet.

A comunicacão entre *Firewalls* pode ser implementada em vários níveis do protocolo TCP/IP. Quando implementada a nível de aplicacão, apenas a parte de dados do protocolo TCP é cifrada. Pode ser interessante que o TCP *checksum* seja calculado a partir do texto original antes de ser cifrado. Assim pacotes adulterados podem ser detectados antes que cheguem à camada superior, onde seria necessário implementar alguma forma de retransmissão para tratar tal problema. Por outro lado, se totalmente implementado a nível de aplicacão, não há necessidade de se alterar o protocolo TCP original, o que pode tornar a implementacão mais fácil, especialmente quando não se tem acesso aos codigos do *kernel*.

A forma intermediária é ter todo o segmento TCP, incluindo o cabecalho, cifrado. Neste caso, uma camada responsável por cifrar e decifrar o conteudo dos dados do protocolo IP deve ser definida logo acima deste. Retransmissões e *checksums* são processados normalmente pelo protocolo TCP, possibilitando o uso de aplicacões existentes de modo transparente.

Contudo, o mais versátil e seguro é fazer tunelamento IP entre *Firewalls,* inclusive como forma de prevenir análise de tráfego entre os segmentos de rede. Um pacote IP

a ser enviado de uma rede local para outra, ao passar pelo primeiro *firewall* seria cifrado e encapsulado em outro pacote IP, que passaria então a ter como endereco de origem e destino os enderecos IP dos *Firewalls* envolvidos. No segundo *firewall* o pacote seria desencapsulado e posteriormente enviado ao destino final já na rede local remota. Mais uma vez é possível ter enderecamento IP não oficial, mas desta vez em todo o perímetro de uma rede virtual e não apenas em uma rede fisicamente isolada. Este esquema é normalmente implementado nos *Firewalls* comerciais que dispoem de recursos de criptografia.

## 8.2 Criptografia de Usuário para *Firewall*

Mesmo com o conceito de rede privada virtual, o problema de se ter um usuário fora do perímetro de protecão continua existindo. A melhor forma de garantir o acesso seguro de usuários externos, além do uso de forte autenticacão (*one time passwords* ou algum protocolo que faca uso de criptografia, por exemplo) é ter toda sessão, entre o usuário distante e o sistema, cifrada.

A idéia é evitar *spoofing* por autenticacão forte, e sequestro de sessão usando criptografia e possivelmente *session keys*[Schneier96]. Mesmo que um *hacker* tenha sucesso em um ataque do tipo SHA, os pacotes enviados por este seriam decifrados em lixo e a sessão seria então interrompida. Supoe-se naturalmente que o *hacker* não seja capaz de "quebrar" o sistema criptográfico usado, o que contudo é normalmente bem menos trivial que simplesmente explorar as falhas nativas do protocolo TCP/IP.

A forma de implementacão de criptografia tende a ser a nível de aplicacão para aplicação. Algum *application gateway* teria que ser responsável por tornar os servicos acessíveis aos usuários de maneira transparente pelo *firewall.*

A primeira dificuldade vislumbrada é com relacão à necessidade de se ter aplicacões cliente especialmente desenvolvidas para tal uso. É sem duvida um grande inconveniente não poder fazer uso das várias aplicacões clientes já existentes na Internet. A falta de padronizacão atual tende a levar a sistemas proprietários a serem totalmente incompatíveis entre si.

Sem querer aprofundar muito na questão, tais sistemas, assim como o Kerberos [Bellovin&Meritt90], não oferece protecão contra um atacante que compartilhe a mesma máquina do usuário remoto. O melhor seria usar sistemas operacionais monousuários, como DOS, Windows95, MacOs, etc. Apesar de computadores portáteis estarem se tornando cada fez mais acessíveis, isto não deixa de representar uma limitacão a mais.

Outras questões, como o problema de distribuicão de chaves, poderiam também ser abordadas, contudo, com o objetivo de se ser sucinto, e tendo em mente o objetivo principal outrora apresentado, será apenas acrescentado que mesmo em *Firewalls* comerciais tal enfoque é raramente explorado. Aparentemente apenas dois produtos comerciais[9] tratam do problema de garantir acesso seguro a transacões para usuários que estejam na Internet em geral.

---

[9] *SmarteWall* da V-ONE e *Eagle* da Raptor. Fonte:[Willoughby95].

## 9. Conclusão

*Firewalls,* quando usados para defender uma rede local que também apresente alto grau de proteção nas suas máquinas internas, isto é, tanto proteção em perímetro como em profundidade, reduzem drasticamente os riscos de se conectar à Internet.

Àqueles que desejam implementar seu próprio *firewall* existe relativamente bastante ferramentas de domínio publico para tal fim (FWTK, Ip Filter, S/Key, Socks, dentre outros[10]). A construção de um *firewall* a partir de tais instrumentos exige no entanto considerável esforço e conhecimento. FWTK, por exemplo, apresenta uma longa lista de "bugs" cujas "correções" se acham dispersas pelos arquivos da lista de discussão do produto (fwtk-users@tis.com). Uma nova versão com todos os erros corrigidos está sendo prometida para abril deste ano[11], mas mesmo assim, versões para diversas plataformas UNIX provavelmente terão ainda que ser desenvolvidas pelos próprios usuários, já que originalmente o pacote é oferecido apenas para SunOS.

Dos produtos básicos para construção de *Firewalls,* o que mais tem apresentado melhorias e aperfeiçoamentos é o Ip Filter. Para implementações a nível de rede (PSF), este pode inclusive ser comparado em termos de funcionalidade e sofisticação a produtos comerciais similares. Correções de falhas têm sido prontamente efetuadas e versões para diversas plataformas estão disponíveis (SunOS, Solaris e BSD). Não se deve esperar no entanto, de um produto gratuito, o mesmo suporte que se teria para um equivalente comercial. O suporte normalmente é oferecido, como nos demais pacotes, através de listas de discussão (ipfilter@coombs.anu.edu.au). E pressuposto todavia ao assinante, um mínimo de conhecimentos sobre segurança e do sistema operacional no qual se deseja construir o *firewall.* Recomenda-se ao leitor referir-se às literaturas apresentadas antes que literalmente se aventure por tais caminhos.

Grande espaço para pesquisa e desenvolvimento existe no sentido de agregar técnicas criptográficas e serviços mais sofisticados às ferramentas hoje disponíveis. Não é de conhecimento dos autores nenhuma ferramenta publica que possua as características apresentadas anteriormente com relação ao uso de criptografia nos *Firewalls.* De imediato uma forma de se ter tal recurso seria pela utilização de Secure Shell (SSH)[Koenig95], mas de maneira geral ainda há muito o que ser feito e pesquisado.

## 10. Agradecimento

Sinceros agradecimentos a Flávio Henrique de Sousa Lima pelo trabalho de revisão deste artigo, como por suas valorosas sugestões; e a José Roberto Menezes Monteiro pelo laborioso trabalho de acompanhar as listas de discussão sobre segurança (em especial *firewalls mailing list*), e selecionar dentre as mais de cem mensagens diárias aquelas de real valor.

---

[10] Referências [Cheswick&Bellovin94] e [Chapman95] apresentam relações completas das ferramentas disponíveis e lugares onde encontrá-las

[11] A versão 2.0 alpha já se encontra disponível em *ftp.tis.com.*

## 11. Referências

[Avolio&Ranum] Frederick M. Avolio and Marcus J. Ranum. A Toolkit and Methods for Internet Firewalls. Trusted Information Systems, Inc. (TIS).

[Avolio&Ranum94] Frederick M. Avolio and Marcus J. Ranum. A Network Perimeter with Secure External Access. In Proceedings of the Internet Society Symposium on Network and Distributed Systems Security, San Diego, California, February 1994.

[Bellovin&Meritt90] Steve M. Bellovin and M. Meritt. Limitations of the Kerberos Authentication systems. Computer Comunication Review. October, 1990.

[Bellovin89] Steven M. Bellovin. Security Problems in the TCP/IP Protocol Suite. Computer Communication Review, April 1989.

[Bellovin95] Steven M. Bellovin. Using the Domain Name System for System Break-ins. In Proceedings of the Fifth Usenix UNIX Security Symposium, Salt Lake City, UT. AT&T Bell Laboratories, 1995.

[Chapman&Zwicky95] D. Brent Chapman and Elizabeth D. Zwicky. Building Internet Firewalls. O' Reilly & Associates, Inc. 1995.

[Chapman92] D. Brent Chapman. Network (In)Security Through IP Packet Filtering. Proceedings of the Third Usenix UNIX Security Symposium. Baltimore, Maryland, September 1992.

[Cheswick&Bellovin94] William R. Cheswick and Steven M. Bellovin. Firewalls and Internet Security. AT&T Bell Laboratories. Addison-Wesley Publishing Company, Inc. Reading, Massachusetts, 1994.

[Comer95] Douglas E. Comer. Internetworking with TCP/IP: Principles, Protocols and Architecture, Volume I. Prentice-Hall, Englewood Cliffs, New Jersey, Third Edition, 1995.

[Garfinkel&Spafford91] Simsom Garfinkel and Gene Spafford. Practical Unix Security.
O' Reilly & Associates, Inc. 1991.

[Hansen&Atkins93] Stephen E. Hansen and Todd Atkins. Automated System Monitoring and Notification with Swatch. LISA, Monterey, CA, November 1993.

[Klaus95] Christopher William Klaus. Stealth Scanning - bypassing Firewalls and SATAN Detectors. Internet Security Systems, Inc, 1995.

[Koenig95] Thomas Koenig. Ssh (Secure Shell) FAQ - Frequently Asked Questions. December, 1995.

[LaMacchia&Odlyzko91] Brian A. LaMacchia and Andrew M. Odlyzko. Computation of Discrete Logarithms in Prime Fields. Designs, Codes, and Cryptography, 1991.

[Lamport81] Leslie Lamport. Password Authentication with Insecure Communications. Communication of the ACM, November 1981.

[Mogul89] Jeffrey C. Mogul. Simple and Flexible Datagram Access Control for UNIX based gateway. In Proceedings of Usenix UNIX Conference, Baltimore, Maryland, summer 1989.

[Mogul91] Jeffrey C. Mogul. Using *screend* to Implement IP/TCP Security Policies. Network Note NN-16, Digital Equipment Corporation. Network System Laboratory, July 1991.

[Ranum92] Marcus J. Ranum. A Network Firewall. In Proceedings of the World Conference on System Administration and Security, Washington, D.C., July1992.

[Ranum93] Marcus J. Ranum. Thinking About Firewalls. In Proceedings of the Second International Conference on Systems and Network Security and Management (SANS-III), April 1993.

[Ranum95] Marcus J. Ranum. *Firewalls* FAQ - Frequently Asked Questions. 1995.

[Reed95] Darren Reed. *Firewalls* mailing list: <199512041259.EAA21992@miles.greatcircle.com>, Dezembro 1995.

[Reed96] Darren Reed. Darren Reed home page.
ULR: http:\\coombs.anu.edu.au/~avalon/ip-filter.html. Version 3.0.2. February 1996.

[RFC1057] Sun Microsystems. RPC: Remote Procedure Call protocol especification: Version 2. RFC 1057, June 1988.

[RFC1858] Dareen Reed, Tom Fitzgerald and Paul Traina. RFC 1858: Security Considerations - IP Fragment Filtering. October, 1995.

[Schneier96] Bruce Schneier. Applied Criptography - Protocols, Algorithms and source code in C. John Wiley & Sons, Inc. 1996.

[Stevens94] TCP/IP Illustrated: the protocols, Volume I. Addison-Wesley Publishing Company, Inc. Reading, Massachusetts, 1994.

[Sun95] Sun Microsystems. Solaris 2.5 System Administrator Answerbook. Mountain View CA, 1995.

[TIS93a] Trusted Information Systems, Inc. (TIS). Secure External Access and Service: Providing Protection for the Network. September 1993.

[TIS93b] Trusted Information Systems, Inc. (TIS). TIS Firewall Toolkit: Building Blocks for Secure Internetworking and Connectivity. September1993.

[TIS94a] Trusted Information Systems, Inc. (TIS). TIS Firewall Toolkit: Configuration and Administration. February 1994.

[TIS94b] Trusted Information Systems, Inc. (TIS). FWTK man pages. September1994.

[Venema92] Wietse Venema. TCP WRAPPER: Network Monitoring, Access Control and Booby Traps. In Proceedings of the Third Usenix UNIX Security Symposium, Baltimore, Maryland, September 1992.

[Vixie95] Paul Vixie. DNS and BIND security issues. In Proceedings of the Fifth Usenix UNIX Security Symposium, Salt Lake City, UT, 1995.

[Willoughby95] Frank Willoughby. Internet Firewall Vulnerabilities. Fortified Networks Inc. November 1995.0