

# TUNELAMENTOS NÃO CONVENCIONAIS EM TCP/IP

Augusto Quadros Paes de Barros

Instituto de Pesquisas Tecnológicas do Estado de São Paulo

05508-901 São Paulo - SP

augusto@paesdebarros.com.br

## RESUMO

*Tunelamentos permitem a comunicação entre redes ou computadores através de um protocolo de comunicação encapsulado por outro protocolo de comunicação. Este artigo descreve algumas formas de tunelamento de tráfego TCP/IP em protocolos não projetados para tal, analisando os problemas existentes para fazê-lo e como os exemplos apresentados os resolvem. Os aspectos de segurança provenientes da utilização dessas formas de tunelamento, assim como técnicas de bloqueio e detecção, também são analisadas.*

## ABSTRACT

*Tunneling allows the communication between networks and computers using a communication protocol encapsulated in another communication protocol. This paper describes some methods of tunneling TCP/IP traffic in protocols that were not designed for that, analyzing the existing problems to perform it and how the presented samples solve them. The security aspects regarding the use of these tunneling methods, as well as detecting and blocking techniques, are also analyzed.*

## 1 INTRODUÇÃO

Uma das principais técnicas de segurança de redes é a aplicação do conceito de mínimo privilégio. Com isso, pontos de aplicação de políticas de segurança, como *firewalls*, trabalham com a filosofia de deixar passar apenas os pacotes necessários.

As políticas de segurança aplicadas evoluíram para minimizar o risco de acessos indevidos, bloqueando os endereços e os protocolos não utilizados pelas aplicações utilizadas pelos usuários da rede em suas atividades. Esta postura visa evitar não só acessos externos a uma rede protegida quanto acessos da rede protegida para a rede pública através de serviços não autorizados.

Em paralelo, diversas técnicas de tunelamento de tráfego foram criadas visando resolver problemas de incompatibilidade de protocolos, endereçamento, segurança ou roteamento. Os protocolos utilizados foram criados visando resolver esses problemas e tem o tunelamento como parte de suas atribuições. Alguns protocolos de tunelamento são apresentados na seção 2.

Tunelamentos, entretanto, não são utilizados apenas na solução dos problemas apresentados. As restrições aplicadas por políticas de segurança de redes são obstáculo para aqueles que tentam obter acesso não autorizado a essas redes ou aos seus computadores. Os usuários dessas redes que pretendem utilizar protocolos ou acessar endereços não autorizados também vêem as restrições como um obstáculo. Ambos podem valer-se de técnicas de tunelamento para burlar as restrições aplicadas à rede.

No entanto, por serem construídas baseadas no princípio do mínimo privilégio, as políticas de segurança de redes não costumam permitir o uso de protocolos de tunelamento, pois estes são

ferramentas para burlá-las. Com isso, os interessados em utilizar tunelamento são forçados a utilizar apenas os protocolos autorizados pela política, que não foram projetados para realizar tunelamento de tráfego. Assim, tunelamentos criados sobre tais protocolos são chamados de “não convencionais”. A criação de túneis sobre esses protocolos envolve uma série de desafios devido à natureza de uso diferente. A forma como tais desafios são resolvidos, assim como alguns exemplos de tunelamentos não convencionais são apresentados na seção 3.

A existência de tunelamentos não convencionais traz uma série de problemas para a aplicação da política de segurança de uma rede. Algumas formas de lidar com tais problemas, como a detecção e o bloqueio dos túneis, são apresentadas na seção 4.

## 2 PROTOCOLOS DE TUNELAMENTO

Protocolos de tunelamento podem ser implementados em diversas camadas. O protocolo mais utilizado para a construção de VPNs (Virtual Private Networks) é o IPSEC.

Originalmente desenvolvido para utilização com o IPv6, o IPSEC acabou sendo usado também sobre o IPv4. A funcionalidade de tunelamento do IPSEC é proporcionado pelo ESP, no chamado “modo túnel”. Neste modo, todo o pacote IP, quando recebido pelo gateway que faz o papel de “ponta” do túnel, é inserido dentro da área de dados, criptografado. Assim, todas as informações do pacote original são preservadas, inclusive os endereços de origem e destino. Com isso, pode-se conectar redes que utilizam endereçamento RFC1918 através da Internet. Por trabalhar tunelando pacotes IP, é possível transmitir todos os protocolos que trabalham sobre ele pelo túnel, como TCP, UDP e ICMP, por exemplo.

Também existem opções de tunelamento em camadas mais altas, como o SSH. Este protocolo trabalha sobre TCP e permite o tunelamento de outras aplicações que trabalham sobre TCP, pois o principal objetivo é a implementação de criptografia. O SSH trabalha através do tunelamento do fluxo de informação de e partir de conexões TCP estabelecidas para e a partir das pontas. Assim, o desenho de uma aplicação sendo tunelada sobre SSH segue o padrão da Figura 1.

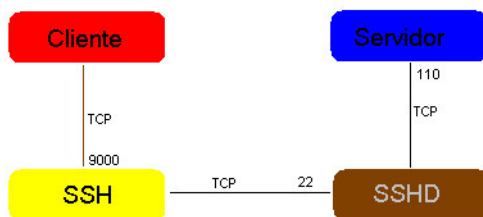


Fig.1 – Sockets TCP em um túnel SSH

O tunelamento da figura ilustra um cliente de e-mail acessando um servidor POP3, passando por um túnel SSH. O cliente conecta-se diretamente ao computador executando o cliente SSH, em uma porta arbitrariamente definida. A configuração do cliente SSH indica o endereço do servidor SSH (SSHD) e o endereço e porta destino da conexão. Assim, é estabelecida uma conexão SSH entre o cliente SSH e o servidor SSH e uma conexão TCP entre o servidor SSH e o servidor de e-mail. Essa abordagem de conexões TCP terminando e se iniciando nas pontas do túnel pode ser vista em diversas implementações de túneis não convencionais.

### 3 TUNELAMENTOS NÃO CONVENCIONAIS

Tunelamentos não convencionais são criados para burlar regras de segurança aplicadas à rede. Assim, um firewall que restringe os protocolos que saem da rede e direção à Internet pode ser driblado através de tunelamentos sobre os protocolos autorizados a trafegar.

Existem várias situações nas quais procura-se driblar restrições de um firewall entre elas:

- Invasor tentando aumentar seu nível de acesso a um computador invadido
- Usuário da rede tentando utilizar serviços não autorizados em outras redes
- *Spyware* tentando se comunicar com seu criador para enviar informações roubadas.

A criação de túneis sobre protocolos que não foram projetados para tal implica em lidar com alguns desafios, como:

- Sentido da conexão: Em uma comunicação comum, qualquer um dos lados pode enviar pacotes IP a qualquer momento. Alguns protocolos utilizados para tunelamento são baseados em na filosofia “requisição e resposta”, ou seja, um dos lados envia informações apenas após ter recebido algo do outro. A solução é a implementação de um algoritmo de *polling*, com requisições periódicas por parte do cliente, verificando se há informações do lado do servidor para serem enviadas.
- Tamanho dos dados: O protocolo ou os campos de mensagem que serão utilizados podem ter limites de tamanho das informações enviadas. A implementação do túnel deve ser capaz de fragmentar o tráfego de acordo com estes limites.
- Tipo de dados: Alguns protocolos têm restrições quanto aos dados que podem trafegar, como em casos onde são permitidos apenas caracteres ASCII aceitos para sistemas baseados na RFC822. Em situações como esta podem ser usados sistemas de codificação como Base32 ou Base64.

Resolvendo-se os pontos acima, praticamente qualquer protocolo que transmita dados a partir das duas pontas pode ser utilizado na criação de tunelamentos não convencionais, como poderá ser observado nos exemplos a seguir.

#### 3.1 Pingtunnel

As mensagens ICMP de tipo *Echo Request* e *Echo Reply* são utilizadas para diagnóstico de problemas de rede. Um host, ao receber uma mensagem *Echo Request*, deve devolver o conteúdo enviado como parte do pacote em uma mensagem de *Echo Reply*. Assim, o remetente poderá identificar se há conectividade com o host de destino e se há algum problema de integridade na transmissão. Este mecanismo de diagnóstico é implementado pela ferramenta Ping, presente em praticamente todas as plataformas que trabalham com TCP/IP.

A área de dados utilizada nessas mensagens pode carregar até 64kbytes. Para implementar um sistema de tunelamento, basta utilizar esta área para inserir os dados a serem transmitidos. Como mensagens *Echo Request* só podem ser enviadas após o recebimento de um *Echo Reply*, A ponta do túnel responsável pelo *Echo Request* deve enviá-lo mesmo quando não há dados a serem enviados.

A primeira ferramenta a implementar o conceito de tunelamento sobre ICMP foi batizada de *loki* (daemon9, 1996), que recebeu melhorias inclusive para envio de dados criptografados. Atualmente, o túnel sobre ICMP mais conhecido é o PingTunnel

(Stodle, 2004). A ferramenta é implementada como dois componentes, um cliente e um servidor, que fazem o papel das duas pontas do túnel. O servidor do PingTunnel recebe as configurações do túnel diretamente do cliente, também dentro da área de dados. Segue abaixo um exemplo de utilização:

Cliente:

```
./ptunnel -p proxy.pingtunnel.com  
-lp 8000 -da login.domain.com -dp  
22
```

Servidor:

```
./ptunnel
```

O exemplo cria um túnel ICMP entre o cliente e o host com nome “proxy.pingtunnel.com”. A conexão TCP que for estabelecida com a porta 8000 do cliente será redirecionada para a porta 22 do host “login.domain.com”. Com isso, uma estação poderá conectar-se ao serviço SSH desse host conectando-se à porta 8000 do cliente.

### 3.2 HTTP Tunnel

O Protocolo HTTP também pode ser utilizado para a criação de túneis. Pelo nível de complexidade, entretanto, há mais dificuldades na implementação. O maior problema é a característica *Request/Response* do protocolo. O HTTP 1.0 requer uma conexão TCP para cada par de requisição e resposta. Isso faz com que diversas conexões sejam necessárias para a manutenção do túnel, tornando o processo pouco performático. Com o HTTP 1.1, porém, foi introduzido o conceito de *Keep-Alive*, no qual diversas requisições e respostas poderiam ser trocadas em uma única conexão. Assim como no caso do ICMP, é necessário manter o lado cliente fazendo requisições mesmo sem dados para serem enviados, para permitir o envio por parte do servidor mesmo nessas situações.

Outro ponto de dificuldade de implementação de túnel sobre HTTP é a restrição de tipos de dados que podem ser enviados nessas conexões. Apenas caracteres ASCII 7-bit podem ser transmitidos, levando à necessidade de codificação de dados puramente binários para um fluxo de caracteres. A codificação BASE64 é utilizada nessas situações durante o uso comum do HTTP, e o mesmo é feito na implementação de túneis.

O uso do HTTP, entretanto, não traz apenas dificuldades para a criação de túneis. O HTTP prevê a existência de *Proxies*, o que pode facilitar o cumprimento de objetivos secundários como ofuscamento de origem ou inversão de sentido de conexões (conexão de fora para dentro de uma rede, através de exploração de *Proxies* mal configurados).

A implementação mais conhecida de túnel sobre HTTP é o HTTP Tunnel (Brinkhoff, 1999), disponível para diversas plataformas sob licença GNU. O HTTP Tunnel funciona de forma semelhante ao PingTunnel, com a criação de conexões TCP para o cliente e a partir do servidor para o servidor original, de acordo com o exemplo abaixo:

Cliente:

```
htc -F 9999 proxy.httptunnel.com:80
```

Servidor:

```
hts -F login.domain.com:22 80
```

O exemplo cria um túnel HTTP entre o cliente e o host com nome “proxy.httptunnel.com”. A conexão TCP que for estabelecida com a porta 9999 do cliente será redirecionada para a porta 22 do host “login.domain.com”. Com isso, uma estação poderá conectar-se ao serviço SSH desse host conectando-se à porta 9999 do cliente.

### 3.3 Ozzyman DNS

Um dos exemplos mais complexos de tunelamento não convencional e com implicações mais sérias de segurança é o uso de DNS. O protocolo DNS permite o uso de TCP e UDP, porém a maior parte das requisições de resolução de nomes acontece sobre UDP. O DNS está disponível para praticamente todas as estações em rede, até mesmo nas mais restritas. Uma diferença significativa no túnel sobre DNS é que as pontas do túnel não precisam ter conectividade direta, pois outros servidores podem fazer o transporte dos dados entre eles, através de requisições recursivas.

O DNS também requer a implementação de soluções que contornem a característica de *Request/Response*. Um problema adicional existente no DNS é o extensivo uso de *caching*, que faz com que não seja possível realizar diversas requisições pelo mesmo nome. O conteúdo dos registros também está restrito a um conjunto de caracteres menor do que aquele do HTTP, tornando até mesmo o uso do BASE64 impossível. A solução é o uso de codificações com um menor número de caracteres, como o BASE32, o que, entretanto, reduz a performance da solução.

A primeira demonstração pública da técnica é o conjunto de ferramentas Ozzyman (Kaminsky, 2004). Também implementada através de um componente cliente e outro servidor, a solução requer, para funcionar na Internet, que o servidor seja responsável por um subdomínio real. O cliente faz sucessivas requisições para registros TXT deste domínio, com o componente mais a esquerda do nome sendo um pedaço dos dados codificados com BASE32. A resposta é gerada dinamicamente pelo servidor e enviada como resposta às requisições. Em uma situação comum, o cliente fará as

requisições para um servidor DNS que permite requisições recursivas. Este mesmo servidor buscará a informação do servidor DNS que faz o papel de ponta do túnel e devolverá a resposta ao cliente. Uma configuração típica das ferramentas do conjunto Ozzyman pode ser vista no exemplo abaixo:

Cliente:

```
ssh -v -o ProxyCommand="/droute.pl  
sshdns.dnsmagic.zona.xyz"  
root@login.dominio.com
```

Servidor:

```
nomde.pl -i 127.0.0.1 dnsmagic.zona.xyz
```

O exemplo cria um túnel DNS entre o cliente e o servidor. O cliente SSH, ao invés de estabelecer uma conexão TCP, utiliza o utilitário “droute.pl” como entrada e saída de dados. Os dados são enviados pelo utilitário via requisições DNS para o servidor executando o “nomde.pl”, responsável pelo subdomínio dnsmagic.zona.xyz, que retira as informações das requisições DNS e entrega ao serviço SSHD local.

#### 4 TÉCNICAS DE BLOQUEIO E DETECÇÃO

O problema de segurança proveniente da possibilidade de criação de túneis sobre protocolos autorizados não tem solução trivial. Na maior parte dos casos não há diferenças entre pacotes comuns e pacotes com conteúdo tunelado, tornando complexa a decisão de bloqueio ou até mesmo a identificação de túneis. Algumas estratégias de detecção que podem trazer resultados são:

- Assinaturas de protocolos: Alguns protocolos trazem em seus processos de *handshaking* seqüências de bytes ou caracteres conhecidas. A identificação de tais seqüências em conexões de outros protocolos pode significar a presença de um túnel.
- Análise de conteúdo: Quantidade e tamanho anormais de requisições podem indicar a presença de túneis.

As estratégias acima podem ser implementadas por sistemas de detecção de intrusos (IDS), porém estão sujeitas a um alto nível de falsos positivos. Técnicas como criptografia e fragmentação de pacotes podem tornar ainda mais complexa a tarefa de detectar túneis.

O ponto de combate natural para túneis é o firewall, por ter a capacidade de bloquear pacotes que por ventura sejam identificados como túneis. Porém, a implementação de mecanismos de detecção nesse dispositivo sofre dos mesmos problemas que a implementação em IDS e tem o

inconveniente adicional da necessidade de alta performance, para evitar o descarte desnecessário de pacotes. Assim, a melhor ação a ser tomada na configuração do firewall para evitar túneis é limitar ao máximo os protocolos que serão permitidos, minimizando assim as opções de tunelamento disponíveis.

#### 5 CONCLUSÃO

A transmissão de dados na forma de tunelamento sobre protocolos que não foram projetados para isso é possível e a dificuldade de implementação varia de acordo com o protocolo utilizado. Alguns protocolos que trazem essa possibilidade, como o DNS, trazem sérias implicações de segurança, por terem uma grande liberdade de tráfego para e partir de redes protegidas por firewalls.

A proteção contra tunelamentos não convencionais ainda é limitada. A melhor maneira de evitá-los é adotar uma política de segurança de rede e controlar a execução de código nas estações, evitando assim que o software responsável pelo tunelamento seja executado.

#### REFERÊNCIAS BIBLIOGRÁFICAS

- COMER, Douglas E. *Internetworking with TCP/IP, Volume 1. 3<sup>rd</sup> edition*. New Jersey: Prentice-Hall, 1995.
- Daemon9. *Project Loki*. Phrack Magazine n.49, 1996.
- BRINKHOFF, Lars. GNU `httptunnel`. [<http://www.nocrew.org/software/httptunnel.html>]. 1999.
- KAMINSKY, Dan. *Black Ops of DNS*. In: Black Hat Briefings, Jul 2004, Las Vegas. [[http://www.doxpara.com/Black\\_Ops\\_DNS\\_BH.ppt](http://www.doxpara.com/Black_Ops_DNS_BH.ppt)]. 2004.