João Mário de Assis

Implementando VPN em Linux

Monografia de Pós-Graduação apresentada ao Departamento de Ciência da computação da Universidade Federal de Lavras como parte das exigências do Curso ARL- Administração em Redes Linux.

Orientador
Prof. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2003

João Mário de Assis

Implementando VPN em Linux

Monografia de Pós-Graduação apresentada ao Departamento de Ciência da computação da Universidade Federal de Lavras como parte das exigências do Curso ARL- Administração em Redes Linux.

Aprovada em *18 de Setembro de 2004*

Prof. Fernando Cortez Sica

Prof. Criatiano Leite de Castro

Prof. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil

Sumário

Lista de Figuras

Lista de Tabelas

Agradecimentos

A minha família, pelo apoio nos momentos decisivos.
Aos professores que contribuiram nesta árdua caminhada.
Ao Mestre Joaquim, pela orientação neste trabalho.
Aos colegas e amigos de turma.

Resumo

Este trabalho tem como objetivo, apresentar a implementação de VPN em Linux, usando o Red Hat Linux e FreeS/Wan.

Capítulo 1

Introdução

Inicialmente as redes locais de computadores, foram usadas para compartilhar alguns recursos como aplicações e impressora, sem a preocupação com o quesito segurança por parte dos administradores, uma vez que o número de usuários era pequeno.

Com o advento da internet comercial e com a necessidade de comunicação entre as redes locais afim de acessar sistemas remotos, surge a necessidade de uma política de segurança, que garanta esses sistemas funcionando, permitindo o acesso remoto de uma forma segura e confiável. Vários métodos foram desenvolvidos com este intuito, entre eles a VPN *(Virtual Private Network)* ou Rede Privada Virtual.

A VPN tem se tornado uma forma de diminuir os custos para interligar as redes das empresas, pois usa um meio público, normalmente a internet, para trafegar dados entre elas. A VPN cria "túneis virtuais"de comunicação entre essas redes, fazendo com que os dados trafeguem de forma criptografada pelos túneis, aumentando a segurança na comunicação.

Além disso, uma VPN possui a capacidade de expandir a quantidade de computadores que podem ter acesso a essa rede, sem investir em infra-estrutura extra. Ela permite suporte a usuários móveis, sem a utilização de bancos de *modem* ou servidores de acesso remoto, ajudando a aumentar a flexibilidade e diminuir os gastos com equipamentos extras.(SCRIMGER et al,2002).

Este trabalho tem como objetivo implementar uma VPN em Linux, utilizando o IPSec(*Internet Protocol Security*), visando analisar a funcionalidade e a segurança desta tecnologia na interligação de computadores usando um meio não seguro de transmissão.

Este trabalho está dividido em 8 capítulos, sendo que o capítulo 1 é uma introdução sobre o trabalho, o capítulo 2 descreve sobre redes de computadores, suas classes, tamanhos e topologia, o capítulo 3 descreve os protocolos de comunicação, o capítulo 4 sobre criptografia, o capítulo 5 descreve os protocolos e processos de tunelamento, o capítulo 6 fala sobre conceitos de VPN, o capítulo 7 descreve a implementação de uma VPN usando um *software* livre, no caso, o FreeS/Wan, o capítulo 8 é uma avaliação sobre o trabalho.

Capítulo 2

Redes de Computadores

2.1 Conceito

Uma rede de computadores é um conjunto de computadores autônomo interconectados capazes de compartilhar recursos e trocar informações. Uma rede de computadores , segundo [Tanembaum] é :

> Um conjunto de computadores autônomos interconectados. Os computadores são ditos autônomos, quando não existe uma relação mestre-escravo entre eles, se um computador puder iniciar, encerrar ou controlar outro computador não existirá autonomia.

As redes foram criadas e desenvolvidas com objetivo de compartilhar recursos aos usuários, como equipamentos, dados, aplicativos, independentemente da localização física tanto do usuário como do recurso. A Internet é um exemplo de uma grande rede difundida mundialmente, possui centenas de milhares de computadores interconectados trocando informações.

2.2 Classificação

As redes podem ser classificadas quanto à Aplicação, ao Tamanho a Topologia empregada, à Tecnologia de Transmissão, a Utilização e Emprego, Modo de Transmissão e Modo de Operação. Neste texto será destacado apenas alguns tópicos sobre Redes de Computadores, visto que o estudo das mesmas, não é o objetivo principal deste trabalho.

2.3 Quanto ao Tamanho

2.3.1 LAN

LAN (*Local Area Network*), são redes locais privadas que conectam computadores pessoais, estações de trabalho e outros dispositivos, em um ou mais edifícios, capazes de transmitir dados em alta velocidade mantendo uma baixa taxa de erro, possibilitando a troca de informações e recursos. Ela abrange uma distância relativamente pequena entre os edifícios separados, no máximo 2,5 km. Veja figura 2.1.

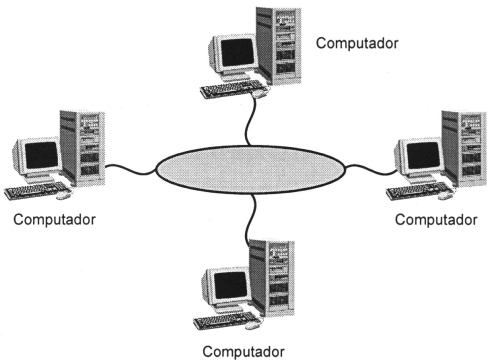

Figura 2.1: Rede Local

2.3.2 MAN

MAN (*Metropolitan Area Network*), é na verdade uma versão ampliada de uma LAN, pois utiliza a mesma tecnologia. Ela pode abranger um grupo de edifícios vizinhos ou uma cidade toda, podendo ser privada ou pública.

2.3.3 WAN

WAN (*Wide Area Network*), são redes geograficamente distribuidas, abrange uma grande área geográfica como um pais ou continente. Ela interligará as LANs e MANs. Veja figura 2.2.

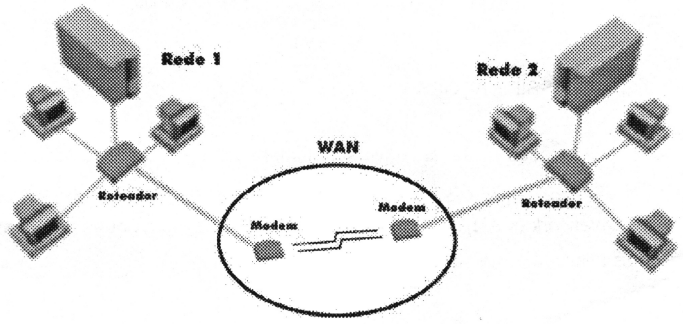

Figura 2.2: Rede Wan

2.4 Quanto à Topologia

2.4.1 Estrela

Esta topologia possui um nó central, que tem como função conectar os computadores e outros dispositivos como impressoras. Geralmente esse nó central pode ser um *hub* ou *switch*, que influenciam diretamente na performance da rede. Veja figura 2.3.

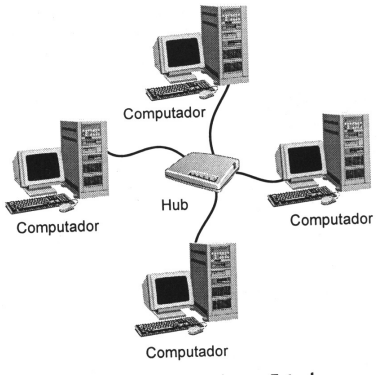

Figura 2.3: Rede Estrela

2.4.2 Anel

Redes com esta topologia, são caracterizadas pelo enlace formando um caminho fechado para conectar os computadores. Neste tipo de conexão, as estações

conectam-se ao anel através de um dispositivo físico que regenera o sinal que trafega na rede, o repetidor. Para melhorar a confiabilidade da rede, pode ser instalado um anel secundário, que será utilizado em caso de falha no anel primário. Veja figura 2.4.

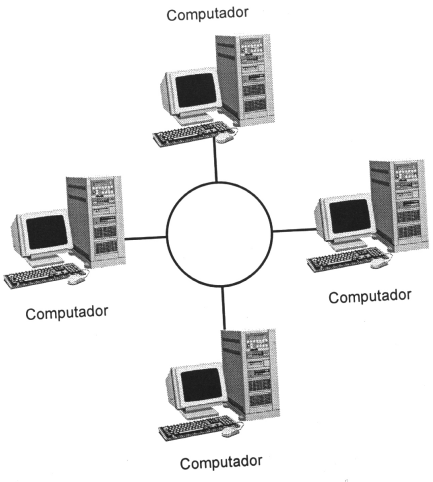

Figura 2.4: Rede Anel

2.4.3 Barra

É caracterizada pela conexão de todos os computadores ao meio físico, sem formar um caminho fechado. A rede funciona com a transmissão de mensagem por difusão, ou seja, as mensagens que trafegam na rede são ouvidas por todos os nós. A ligação ao meio é o ponto crítico, ela deve ser feita de forma a alterar o mínimo possível as caracteristicas elétricas do meio. O meio deve possuir em seus extremos uma carga igual à sua impedância caracteristica, a fim de evitar reflexões espúrias que interfiram no sinal transmitido. Veja figura 2.5.

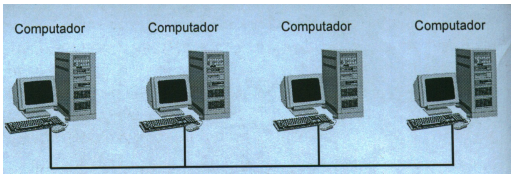

Figura 2.5: Rede Barra

2.5 Quanto à Aplicação

2.5.1 Ponto-a-Ponto

Também conhecidas como peer-to-peer ou P2P, são pequenas redes que não possuem uma centralização das informações, ou seja, ora um computador fornece recursos à outros computadores, ora usa os recursos dos demais, como arquivos ou periféricos. Esse tipo de rede não permite uma política de segurança eficiente, pode ocorrer que dois ou mais computadores possuam arquivos com o mesmo nome e não possuem um controle de acesso aos arquivos de uma forma segura.

2.5.2 Cliente-Servidor

Esse tipo de rede é conhecida pela forma centralizada com que armazena as informações, através de um computador dedicado a uma ou várias tarefas, chamado servidor. Tipos comuns de servidores são: impressão, arquivos, aplicações, correio eletrônico ou páginas *web*. Algumas vantagens da rede Cliente-Servidor podem ser destacadas, como a manutenção, feita de forma centralizada, reduzindo custo de administração e uma segurança mais eficaz ao acesso às informações.

Capítulo 3

Protocolos de Comunicação

Protocolo de comunicação é a forma com os que os dispositivos conectados em rede possam se comunicar, com isso vários protocolos foram criados, tais como: TCP/IP, NetBios, NetBEUI(Extended User Interface) e IPX/SPX(Interwork Packet Exchange/Sequenced Packet Exchange), UDP.

3.1 TCP/IP

Durante a década de 60, o departamento de defesa dos Estados Unidos da América, criou a *Advanced Research Project Agency* (ARPA), que desenvolveu uma rede, a ARPANET para interligar os vários centros de pesquisas: *University of California Los Angeles* (UCLA),*Stanford Research Institute* (SRI), *University of California Santa Barbara* (UCSB) e *Utah University*, para que em caso de uma gerra, se um dos centros fosse destruído, a rede continuaria funcionando. A ARPANET foi desenvolvida, mas apresentava inúmeros problemas, dentre eles, quedas constantes, isso possibilitou a pesquisa de um conjunto de protocolos mais estável e confiável.

Em meados da década de 1970 foi desenvolvido o TCP/IP, logo tornou-se popular por ser leve e por seu baixo custo de implantação em relação aos outros protocolos então disponíveis.

TCP/IP, anacrônico para o termo *Transmition Control Protocol/Internet Protocol*, é um conjunto de protocolos, onde dois dos mais importantes (IP e o TCP) deram seus nomes à arquitetura. Os protocolos TCP/IP podem ser utilizados sobre qualquer estrutura de rede, seja ela simples como uma ligação ponto-a-ponto ou

uma rede complexa.

O TCP/IP pode ser empregado em estruturas de rede como *Ethernet, Token-Ring, Frame-Relay, FDDI, PPP, ATM, X.25* e várias outras como meio de comunicação do protocolo TCP/IP. Assim como o modelo OSI (*Open System Interconnection*), a arquitetura TCP/IP realiza a divisão de funções do sistema de comunicação em estruturas de camadas denominadas, Rede, Inter-Rede, Transporte e Aplicação. Veja figura 3.1.

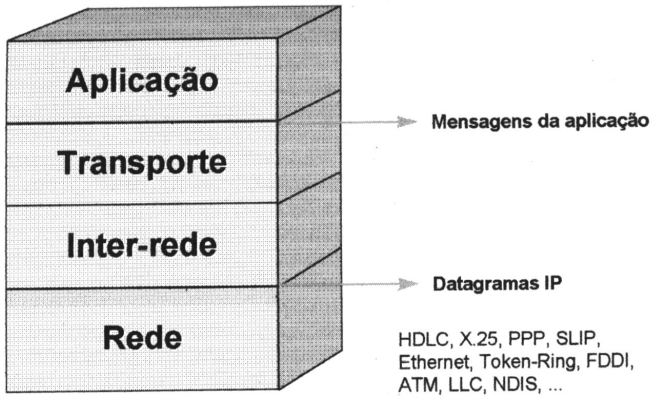

Figura 3.1: Camadas modelo TCP/IP

A arquitetura TCP/IP, baseia-se principalmente em um serviço de transporte orientado à conexão, fornecida pelo TCP e um serviço de rede não-orientado à conexão (datagrama não confiável), fornecido pelo IP. Os padrões da arquitetura TCP/IP, são coordenados por um corpo técnico da IAB (*Internet Activity Board*), formado por pesquisadores, tendo a maioria deles projetado e implementado os protocolos da arquitetura Internet.

3.1.1 Camada Host/Rede

A camada de Rede é responsável pelo envio dos datagramas construídos pela camada Inter-Rede, realiza também o mapeamento entre um endereço de identificação de nível Inter-Rede para um endereço físico ou lógico do nível de Rede. Protocolos que fazem parte desta camada, possuem a função de conectar o *host* à rede, para enviar pacotes IP. Dependendo do *host* e da rede o protocolo muda. Alguns protocolos da camada Host/Rede:

- Protocolos com estrutura de rede própria (*Frame-Relay, ATM, X.25*);
- Protocolos de Enlace OSI (PPP, *Ethernet, Token-Ring, FDDI, HDLC, SLIP*);
- Protocolos de Nível Físico (V.24, X.21);
- Protocolos de barramento de alta-velocidade (*SCSI, HIPPI,...*).

3.1.2 Camada Inter-Rede

Essa camada realiza a função de interligar os diferentes tipos de rede através do protocolo IP. Para identificar cada *host* e a própria rede, é definido um identificador chamado endereço IP.

Quando ocorre uma transmissão de pacotes entre redes, o protocolo IP é quem garante que eles sejam transmitidos, independente do destino. O protocolo IP pode entregar os pacotes fora da ordem original em que foram criados pelo emissor, devido a rotas diferentes para alcançar o destinatário, sendo assim, as camadas de níveis superiores do destinatário é quem coloca os pacotes recebidos na ordem original. Protocolos desta camada :

- Protocolo de transporte de dados: IP (*Internet Protocol*);
- Protocolo de controle e erro: ICMP (*Internet Control Message Protocol*);
- Protocolo de controle de grupo e endereços: IGMP (*Internet Group Management Protocol*);
- Protocolo de controle de informações de roteamento.

3.1.3 Camada de Transporte

A finalidade desta camada é permitir que pares de *hosts* mantenham uma comunicação fim a fim confiável. A função básica desta camada é receber dados da camada de Aplicação, se necessário, dividi-los em partes menores e repassá-los à camada Inter-Rede.

A camada de Transporte possui dois protocolos, UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*). Através do protocolo TCP realiza o controle de fluxo, para evitar que o emissor envie mais dados que o receptor

possa receber, controle de erro, multiplexação e sequenciamento ao acesso ao nível Inter-Rede. O protocolo UDP realiza apenas a multiplexação, para que várias aplicações possam acessar o sistema de comunicação de forma coerente.

3.1.4 Camada de Aplicação

Esta camada reune protocolos de alto nível que fornecem serviços de comunicação ao sistema ou ao usuário. Os protocolos de aplicação podem ser separados em protocolos de serviços básicos que atendem as necessidades do sistema de comunicação TCP/IP (DNS, BOOTP, DHCP) e protocolos de serviços para o usuário (FTP, HTTP, Telnet, SMTP, POP3, SNMP e outros).

3.2 IP

O protocolo IP é responsável pela comunicação entre *hosts* em uma estrutura de rede TCP/IP. O protocolo IP provê a capacidade de comunicação entre cada elemento da rede, para permitir o transporte de uma mensagem da origem até o destino. Ele atribui endereçamento independende do endereçamento e da topologia da rede utilizada e possui a capacidade de rotear mensagens entre os elementos que interligam a rede. A figura 3.2 apresenta um datagrama IP, versão quatro, cujos campos representam:

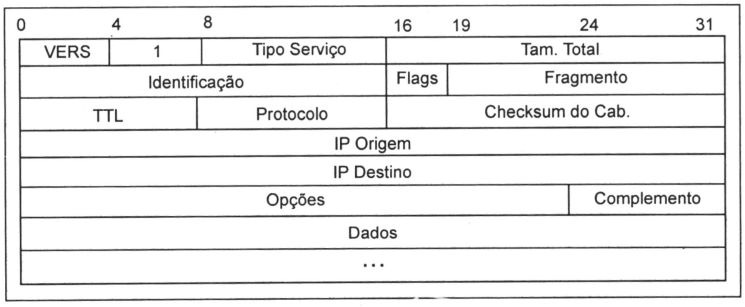

Figura 3.2: Datagrama IP

- O campo VERS, armazena o número da versão do datagrama. Importante nesse momento de mudança de versão de protocolo, para os equipamentos possam realizar o processamento adequado;

- O campo TAM. indica o tamanho do cabeçalho, visto que ele não é constante, o valor mínimo é 5 e o máximo é 15. O valor mínimo ocorre na ausência de opções e o máximo quando são utilizadas todas as opções disponíveis;

- O campo TIPO de SERVIÇO permite que os *hosts* informem essas opções, combinando opções de confiabilidade e velocidade;

- TAM. CAB., uma vez que o cabeçalho possui tamanho variável, este campo é usado para armazenar esta informação;

- O campo TAM. TOTAL informa o tamanho do datagrama, atualmente com valor máximo de 65.535 *bytes*;

- O campo IDENTIFICAÇÃO é usado para que os fragmentos de um datagrama gerados durante uma transmissão, que podem ser transferidos em ordens ou por caminhos diferentes, possam ser montados na ordem original pelo *host* destino;

- O Campo *FLAGS* possui três *bits*, um não utilizado o DF (*Dont fragment*) e MF (*More Fragmente*). O *bit* MF informa que o datagrama foi fragmentado, todos os fragmentos do datagrama utilizam esse *bit*. O controle de fragmentos é feito pelo campo FRAGMENTO, que indica a que ponto do datagrama o fragmento pertence;

- O campo TTL indica o tempo de vida em segundos que o datagrama possui, evita que um datagrama fique eternamente na rede. Na prática, o valor deste campo é decrementado toda vez que passa por um roteador até chegar ao valor zero e simplesmente ser descartado;

- O campo PROTOCOLO é usado para a identificação do protocolo utilizado pela camada de transporte. Essa informação é importante para o destinatário montar o datagrama;

- O campo *CHECKSUM* DO CAB é utilizado para verificação dos erros apenas do cabeçalho durante a transmissão dos datagramas;

- O campo IP ORIGEM, identifica o endereço do transmissor, enquanto que o IP DESTINO identifica o endereço do destinatário;
- O campo OPÇÕES refere-se aos endereços de rotas que não devem ser utilizados, endereços completos de rotas que o datagrama deve seguir e outras informações sobre roteadores. Esse campo possui tamanho variável, por isso é necessário o campo COMPLEMENTO para completar 32 *bits*;
- O campo DADOS é onde estão armazenados os dados transmitidos no datagrama.

Em redes basedas no protocolo TCP/IP, cada componente (computador, roteador, concentrador e outros) possuem uma interface de rede, que possui um número de identificação, o endereço IP. A versão 4 (Atual) do protocolo IP é formado por um número escrito em notação decimal de 32 *bits* (4*bytes*), separados por ponto. Desta forma o endereço IP 11000000 10100100 0011100 10100011 é representado por 192.160.28.63. Os endereços IP são divididos em classes, A, B, C, D e E, como mostra a figura 3.3.

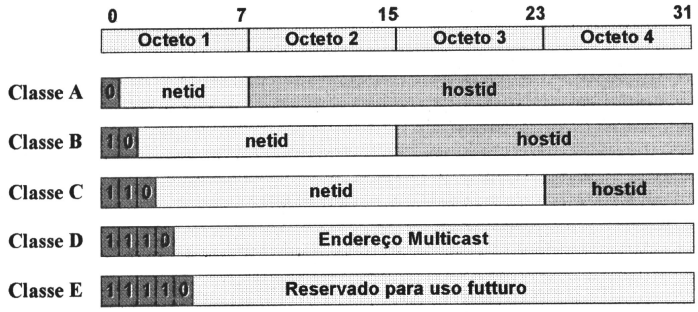

Figura 3.3: Classes de rede

Devido aos problemas existentes na versão 4 (IPv4), como números de endereços possíveis na Internet, foi criado a versão 6 (IPv6) do protocolo IP com seguintes objetivos:

- Aceitar bilhões de *hosts*;
- Reduzir o tamanho das tabelas de roteamento;
- Simplificar o processo de modo a permitir que os roteadores processem os pacotes com mais rapidez;

- Oferecer mais segurança (autenticação e privacidade) do que o IP atual;
- Dar mais importância ao tipo de serviço, especialmente para os dados em tempo real;
- Permitir *multcast*, possibilitando a especificação do escopo;
- Permitir que um *host* mude de lugar sem mudar de endereço ;
- Permitir que o protocolo evolua no futuro;
- Permitir a coexistência entre o novo e o antigo protocolo.

Os endereços do novo protocolo IP, IPv6, são maiores que o da versão atual, IPv4, possuem 16 *bytes*. O cabeçalho simplificado do protocolo, permite que os roteadores procecessem o cabeçalho mais rapidamente, melhorando a eficiência da transmissão. O datagrama IPv6, apresentado na figura 3.4, possui como campos:

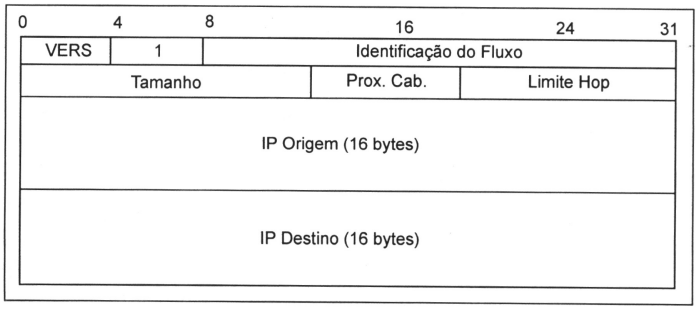

Figura 3.4: Datagrama IPv6

- O campo VERS informa a versão do protocolo, neste caso 6;
- O campo IDENTIFICAÇÃO do FLUXO, ainda em fase experimental, possibilita a criação de um circuito virtual entre o emissor e o receptor para a transmissão dos datagramas;
- O campo TAMANHO especifica o tamanho do datagrama. A diferença fundamental em relação à versão anterior é que nessa versão os 40 *bytes* não são contados nesse campo;

- O campo PROX. CAB. é utilizado quando algumas aplicações necessitam transmitir mais informações no datagrama;

- O campo LIMITE HOP que especifica o número limite de roteadores que um datagrama pode utilizar. Na verdade é o campo TTL da versão 4 que teve seu nome modificado;

- O campo IP ORIGEM especifica o endereço do emissor e o campo IP DESTINO especifica o endereço do receptor;

- O campo DADOS onde estão os dados transmitidos.

As redes podem ser divididas usando máscara de sub-rede para poder abrigar um número diferente de *hosts*, de acordo com a necessidade de cada empresa. A máscara define qual parte do endereço IP é relativa à rede e qual parte é relativa ao *host*. Os rotadores para obter o endereço de rede, efetuam uma operação lógica "*and*" entre a máscara de rede e o endereço IP.

Os pacotes IP não podem ser entregues diretamente aos equipamentos baseados em seu endereço IP, pois a camada de Rede só entende endereço dessa camada. Por exemplo, em redes como *Ethernet, Token-Ring*, os equipamentos possuem endereço próprio chamado endereço físico ou endereço MAC (*Medium Access Control*), de 6 bytes, atribuido pelo fabricante do equipamento.

Em uma rede *Ethernet*, os pacotes são entregues por difusão, utilizando os endereços Ethernet. Para que o pacote IP chegue ao destino é necessário que haja um mapeamento do endereço IP para o endereço *Ethernet*, isto é realizado pelo protocolo ARP (*Address Resolution Protocol*). Ele envia mensagens na rede, perguntando qual *host* é dono do endereço IP, o *host* dono responde, informando seu endereço *Ethernet*.

Uma situação inversa também pode ocorrer, obter um endereço IP a partir do endereço *Ethernet*, para isso é utilizado o protocolo RARP (*Reverse Address Resolution Protocol*). Um exemplo, é quando uma estação sem disco é inicializada, ela recebe uma imagem do sistema operacional através de um servidor, porém é preciso obter o endereço IP.

3.3 TCP

O protocolo TCP é um protocolo orientado à conexão, ou seja, permite a entrega dos pacotes entre o transmissor e o receptor sem erro, através de canal originado no transmissor para o receptor.

Para efetuar a transferência do fluxo de dados, tanto o transmissor como o receptor, criam pontos terminais chamados *socket*. Cada *socket* tem um número, que define o endereço IP do *host* mais um número de 16 *bits* local para este *host* chamado porta. Esta porta identifica o acesso a um serviço, que é utilizado quando é criado uma conexão explícita entre o emissor e o transmissor. O datagrama TCP, apresentado na figura 3.5 possui como campos:

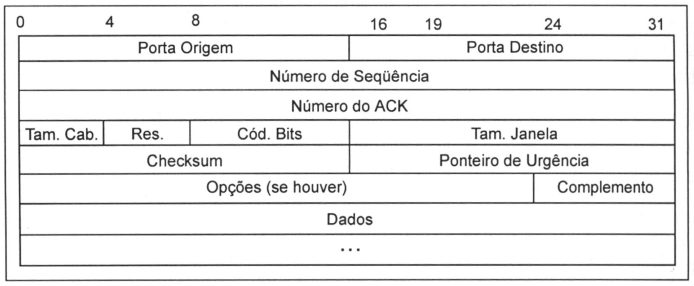

Figura 3.5: Datagrama TCP

- PORTA ORIGEM e PORTA DESTINO, identifica a porta origem e porta destino;
- NÚMERO de SEQUÊNCIA, utilizado para ordenar a montagem dos datagramas no receptor, uma vez que os datagramas podem ser transferidos fora da ordem original;
- NÚMERO de ACK, confirma a recepção do datagrama;
- TAM. CAB., uma vez que o cabeçalho possui tamanho variável, este campo é usado para armazenar esta informação;
- RES, este campo foi reservado para ser utilizado se ouvesse a necessidade de alguma correção. Ainda não precisou ser utilizado;

- COD. BITS, composto por 6 *bits* que podem indicar a urgência do datagrama, em conjunto com o ponteiro de urgência;
- PONTEIRO de URGÊNCIA, indica um deslocamento de *bit* no número de sequência;
- TAM. JANELA, campo utilizado no controle de fluxo, através do algorítmo de janela deslizante, informa quantos *bytes* podem ser enviados, a partir do *byte* informado;
- CHECKSUM, usado para verificar a confiabilidade dos dados tranmitidos no cabeçalho, na recepção é calculado o *checksum* e comparado com este campo;
- OPÇÕES, campo de tamanho variável usado para transmitir informações que não fazem parte do projeto original do protocolo;
- COMPLEMENTO, usado para completar 32 *bits*;
- DADOS, onde estão os dados transportados.

3.4 UDP

O UDP é um protocolo de transporte não orientado à conexão, ou seja, não cria um circuito virtual entre o emissor e o receptor. Por ser um protocolo não confiável, é utilizado em aplicações que não necessitam de controle de fluxo ou manutenção das mensagens enviadas, ou seja, aplicações que não necessitam de entrega precisa e sim de entrega imediata, como voz e imagem. O datagrama UDP, apresentado na figura 3.6, possui como campos:

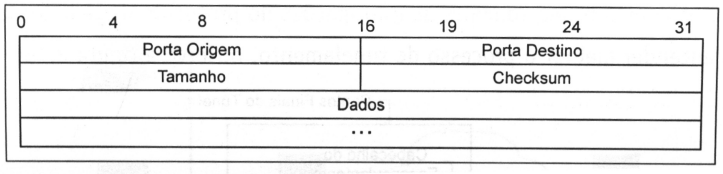

Figura 3.6: Datagrama UDP

- PORTA ORIGEM E PORTA DESTINO, identificam as portas utilizadas no transmissor e receptor, a mesma função que o TCP;
- TAMANHO, armazena o tamanho completo do datagrama, cabeçalho e dados;
- CHECKSUM, é calculado utilizando os campos de um pseudocabeçalho do datagrama que inclui também os endereços IP do transmissor e receptor;
- DADOS, possui a mesma função do campo em TCP.

Capítulo 4

Criptografia

4.1 Comentários Iniciais

A palavra criptografia é de origem grega, formada por duas palavras, *kryptós* que significa escondido e *graphos* que significa grafia, pode ser definida como a arte ou ciência que estuda as técnicas para garantir a segurança das mensagens, de forma que apenas pessoas autorizadas leiam. O objetivo da criptografia é prover uma comunicação segura, garantindo aos serviços confidencialidade, autenticidade, integridade e não repúdio. O processo de criptografia ilustrado na figura 4.1.

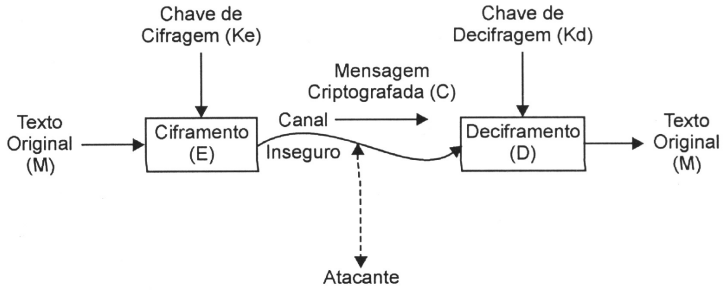

Figura 4.1: Processo Criptográfico

O processo de criptografia tem como finalidade através de uma mensagem gerada no emissor, criptografa-la através de uma chave e um algoritmo de cifragem, tendo como resultado uma mensagem cifrada, ou seja, compreensível apenas para quem tem autorização para le-la. No receptor o processo é inverso, chamado decifragem, obtendo a mensagem original.

A criptografia classifica-se em criptografia simétrica e criptografia assimétrica, dependendo do tipo de chave utilizada.

4.2 Criptografia Simétrica

Criptografia simétrica baseia-se na simetria das chaves do emissor e receptor, ou seja, a mesma chave usada para criptograr será usada para decriptografar a mensagem. Essa chave, denominada chave privada, é previamente trocada entre o emissor e o receptor, através de uma comunicação segura. Esse método apesar de sua simplicidade, possui alguns problemas, são eles:

- A mesma chave ser usada para cada par, emissor-receptor, se o número de pares for grande, será necessário um grande número de chaves, o que dificulta a administração das mesmas e compromete a segurança, visto que nem sempre é possível garantir que a chave será armazenada de forma segura;

- A criptografia simétrica não garante a identidade de quem enviou ou recebeu a mensagem.

Existem vários algorítimos simétricos que produzem chaves de tamanhos variados, quanto maior a chave, maior a segurança. A seguir alguns algoritmos:

- *Data Encryption Standard* (DES) - 56 bits;

- *Triple Data Encryption Standard* (3DES) - 112 bits;

- *Blowfish* - até 448 bits;

- *Twofish* - 128, 192 ou 256 bits;

- *Advanced Encryption Standard* (AES) - 128, 192 ou 256 bits.

Mais detalhes sobre esse algoritmos podem ser encontradas em:
`http://cacr.math.uwaterloo.ca/hac`
`http://bookpool.com/.x/es37oih3v1/ss/1?qs=applied+crypto\&Go.x=0\&Go=Go`

4.3 Criptografia Assimétrica

A criptografia assimétrica surgiu para contornar os problemas da criptografia si métrica, através de algorítmos que utilizam chave pública e privada. Pode-se utilizar qualquer das chaves para criptografar a mensagem, entretanto só a chave inversa pode ser usada para decriptografa-la.

Os algorítmos que implementam a chave pública e privada, exploram propriedades específicas dos números primos e a dificuldade de fatorá-los mesmo em equipamentos rápidos. O RSA (*Rivest Shamir Adleman*), composto por chaves de 512, 768, 1024 e 2048 *bits*. Esse algorítimo é a base da maioria das aplicações de criptografia assimétrica, mais detalhes em `http://www.impa.org.br/Publicacoes/Comput\protect\T1\textunderscoreMath/rsa/index.html`.

4.4 Função *Hashing*

A função *Hashing* tem como objetivo garantir a integridade da mensagem recebida e tornar a decifragem da mesma mais rápida, visto que, a criptografia assimétrica embora muito eficiente na cifragem da mensagem, é muito lenta na decifragem da mesma. Apesar disso, a função *hashing* não criptografa a mensagem. A função *Hashing* quando aplicada ao conteúdo de uma mensagem, gera um resumo chamado código *hash*.

Este código para ser eficiente deve ser único para cada mensagem com conteúdo diferente e deve ser tal, que recompor a mensagem a partir do código *hash* seja impossível, o que garante a integridade da mensagem. Os algorítmos que implementam a função *hash*, tem como objetivo fazer com que o resumo sofra uma grande modificação se algum caractere do conteúdo da mensagem for alterado.

Como exemplo de algoritmos *hash*, podem ser destacados o *Message Digest 5* (MD5), retorna um resumo de 128 *bits* o *Secure Hash Algorithm 1* (SHA-1) retorna um resumo de 160 *bits*, enquanto o *Secure Hash Algorithm 2* (SHA-2) retorna um resumo que pode ter 256, 384 e 512 *bits*.

A Função *Hashing* utilizada de forma isolada na transmissão de uma mensagem, pode não garantir a integridade, pois um intruso pode violar a mensagem,

calcular e substituir o código *Hash*. Este problema é resolvido unindo a criptografia assimétrica com os benefícios da função *Hashing* em um único processo chamado Assinatura Digital.

4.5 Assinatura Digital

O modelo computacional que permita assinar digitalmente um documento deve permitir que o receptor possa verificar a identidade do emissor, deve controlar a assinatura de forma que o emissor não possa repudiar o conteúdo e possa evitar fraudes no próprio sistema. A assinatura digital utiliza a criptografia assimétrica, pois utiliza um par de chaves, uma pública e outra privada, a chave privada é usada para assinar o documento, enquanto a pública verifica a assinatura. Em termos práticos, todos os algoritmos de criptografia assimérica podem ser utilizados para assinatura digital, porém o padrão adotado pelo mercado foi o RSA.

Para assinar um documento e mante-lo em segredo e evitar que outros usuários possam le-lo, é necessário não só assinatura digital, mas também a sua criptografia utilizando a chave pública do destinatário. Quando o documento é grande o processo de criptografia é demorado, isto aumenta o custo computacional. Para resolver este problema, utiliza-se funções *Hashing* que divide o documento em bloco fixo (128 a 512 *bits*) dependendo do algorítmo utilizado, gerando um *Hash*, criptografa-se o *Hash* gerado com uma chave privada, obtendo assim a assinatura digital.

A partir de um documento e sua assinatura digital, pode-se facilmente verificar sua autenticidade e integridade. Primeiro utiliza-se a mesma função *Hashing* aplicada ao documento na origem, obtendo assim o *Hash* do documento; Depois decifra-se a assinatura digital com a chave pública do remetente, que deve produzir o mesmo *Hash* gerado anteriormente. Com os valores *Hash* sendo iguais é determinado que o documento não foi modificado após a assinatura do mesmo, caso contrário, o documento ou a assinatura digital ou ambos foram modifica-dos.

4.6 Certificado Digital

Um certificado digital associa a identidade de uma pessoa ou processo, a um par de chaves criptográficas, uma pública e outra privada, que usadas em conjunto comprovam a identidade. O certificado digital é um arquivo assinado eletronicamente

por uma entidade certificadora chamada Autoridade Certificadora (AC).

O conteúdo e a autenticidade de um certificado emitido por uma autoridade certificadora, pode ser examinado por qualquer entidade que conheça a chave pública da AC. O certificado digital é protegido pela assinatura digital do emissor, Autoridade Certificadora. No certificado existem 6 campos obrigatórios, número serial, algorítmo de assinatura, o emissor, validade, chave pública, assunto e 4 campos opcionais, número da versão, dois identificadores e extensão.

A recomendação mais utilizada para emissão de certificados digitais é a X.509v3, descrita na RFC 2459(R. Housley; SPYRUS; W. Ford; Verisign; W. Polk; NIST; D. Solo; Citicorp; 1999) formulada pela *International telecommunication Union - Telecommunication Standardization Sector* (ITU-T), que introduziu as extensões (*flags*) que possibilita as autoridades certificadoras, utilizarem campos configuráveis.

Capítulo 5

Tunelamento

5.1 Processos de Tunelamento

O tunelamento é um método criado por protocolos utilizados em VPN, são responsáveis pela transmissão dos dados com segurança e confiabilidade, entre redes privadas através de redes públicas, geralmente a internet. Existem 3 processos que compõem o tunelamento :

- Encapsulamento - Os dados da rede local são encapsulados com os dados de destino;
- Transmissão- Os dados são enviados ao seu destino, por meio das informações do cabeçalho;
- Desencapsulamento - No destino, os dados são desencapsulados, deixando somente as informações do protocolo da rede local.

O tunelamento pode ser dividido em :

- Tunelamento Voluntário - Quando uma estação de trabalho ou um servidor utiliza um *software* para cliente de tunelamento para criar uma conexão até o servidor VPN. Esse método é utilizado por clientes *dial-up* que primeiro conectam-se à *internet*, para depois criar o túnel utilizando o *software*, com isso, o cliente passa ser o fim do túnel. Tunelamento voluntário ilustrado na figura 5.1;
- Tunelamento Compulsório - Quando existe um NAS (Servidor de Autenticação de Rede) na rede, ele é responsável pela configuração e autenticação.

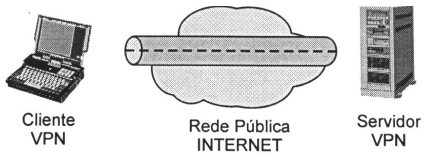

Figura 5.1: Tunelamento Voluntário

Assim não é necessário que os clientes da rede possuam *software* cliente para tunelamento. Desta forma, o final do túnel é no servidor NAS e não no cliente, que passa a acessar as informações da outra rede por meio do servidor de autenticação da rede. Tunelamento compulsório ilustrado na figura 5.2.

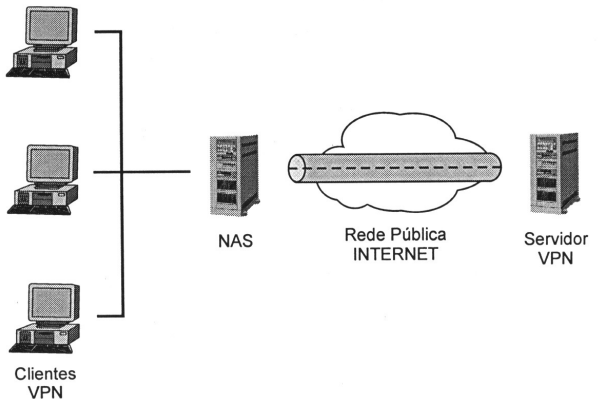

Figura 5.2: Tunelamento Compulsório

5.2 Protocolos de Tunelamento

Esta seção apresenta os principais protocolos de tunelamento. Esses protocolos estabelecem comunicações diretas entre duas redes separadas fisicamente de forma segura e confiável.

5.2.1 PPTP

Este protocolo, PPTP (*Point to Point Tunneling Protocol*), Protocolo de tunelamento Ponto a Ponto, desenvolvido por um consórcio que incluia, *3COM, Ascend Telematics, US Robotics e Microsoft*. É um protocolo de camada dois e utiliza os mesmos mecanismos de autenticação do PPP e é documentado pela RFC 2637 (K.Hamzeh; G. Pall; W. Verthein; J. Taarud; W. Little; G.Zorn, 1999). As conexões PPTP dispõem de autenticação de usuários e encriptação dos dado; para autenticar os dados pode-se utilizar os seguintes protocolos:

- MS-CHAP (*Microsoft Challenge Handshake Authentication Protocol*), RFC 2433 (G.Zorni; S.Cobb; Microsoft Corporation);
- EAP (*Extensible Authentication Protocol*), RFC 2284 (L.Blunk; J.Vollbreht; Merit Network inc; 1998);
- PAP (*Password Authentication Protocol*);
- CHAP (*Challenge Handshake Authenticatio Protocol*).

O protocolo PPTP utiliza o MPPE (*Microsoft Point to Point Encryption*), RFC 3078 (G.Pall; Microsoft Corporation; G.Zorni; Sisco System; 2001), que utiliza chaves de criptografia de 40, 56 e 128 *bits* para criptografar os dados. Por não oferecer serviço de criptografia, o PPTP pega os dados criptografados pelo MPPE e faz o encapsulamento utilizando uma versão do GRE (*Generic Routing Encapsulation*). Para manter o túnel, o PPTP utiliza o protocolo TCP.

O controle da conexão PPTP utiliza a porta reservada 1723 para estabelecer a conexão de controle, que fica entre o IP dinâmico do cliente da VPN e o endereço IP fixo do servidor. Após o estabelecimento da conexão, inicia a troca de mensagens entre o controle de conexão PPTP e o gerenciamento de mensagens PPTP. Os pacotes de controle de conexão PPTP, são compostos por um cabeçalho IP, um cabeçalho TCP e o controle de mensagens PPTP. Veja figura 5.3.

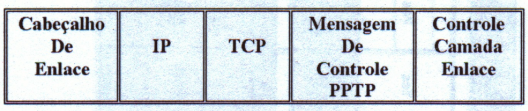

Figura 5.3: Quadro PPTP

O protocolo PPTP utiliza três níveis de encapsulamento para fazer o tunelamento dos dados, são eles :

- Encapsulamento do Frame PPP - O pacote PPP é encapsulado e criptografado com um cabeçalho PPP, originando um frame PPP. Esse frame recebe um cabeçalho GRE, que é utilizado para repassar informações de roteamento, para trafegar em redes IP;

- Encapsulamento do pacote GRE - Esse pacote é encapsulado por um cabeçalho IP, onde estão os endereços IP de origem e destino do pacote;

- Encapsulamento da camada Data-Link - Para que o pacote possa ir para uma LAN ou WAN, as informações desse cabeçalho são necessárias, pois são utilizadas para o envio desse pacote para a interface física.

O resultado deste encapsulamento é ilustrado pela figura 5.4.

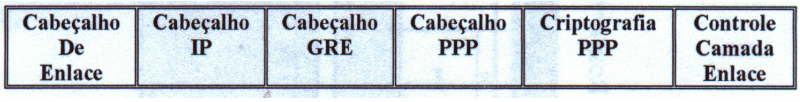

Figura 5.4: Quadro de encapsulamento

Uma vez entregue o pacote ao seu destinatário, é feita a remoção dos cabeçalhos Data-Link, IP, GRE e PPP, em seguida os dados são decriptografado, descomprimidos e podem ser entregues à rede local de destino.

5.2.2 L2TP

O L2TP (*Layer 2 Tunneling Protocol*), baseado no protocolo criado pela *Cisco System* L2F (*Layer 2 Forwarding*) e homologado pela IETF (*Internet Engineering Task Force*) como protoloco padrão. Ele utiliza o que há de melhor nos protocolos PPTP e L2F. Suporta protocolos como NetBEUI, IPX, ATM, Sonete *frame relay*, significando que é multiprotocolo. É descrito na RFC 2661 (W. Townsley; A. Valencia; A. Rubens; G. Pall; G. Zorn; B. Palter, 1999).

A Autenticação difere do PPTP pois é feita em duas etapas. Na primeira, antes do túnel ser instalado, o usuário é autenticado pelo provedor de acesso e na segunda, quando a conexão é estabelecida entre os *gateways*. No L2TP os dados

são criptografados antes da conexão ser estabelecida, para que isto aconteça, ele utiliza o DES (*Data Encryption Standard*).

Diferentemente do PPTP, o L2TP utiliza o protocolo UDP para manter o túnel, no Linux utiliza a porta 1701 para esta manutenção. Uma vez que o protocolo UDP não garante a entrega dos pacotes, o L2TP possui mensagens para certificar-se que os pacotes foram entregues, são elas: *Next Received Field* e *Next Send Field*. O formato do pacote L2TP é ilustrado pela figura 5.5.

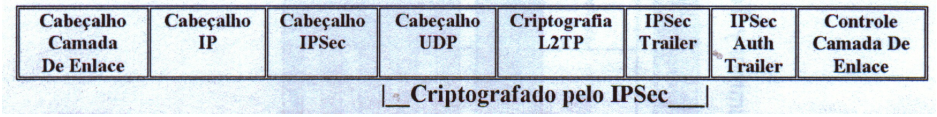

Figura 5.5: Formato do Pacote L2TP

Como no protocolo PPTP, o protocolo L2TP também possui alguns níveis de tunelamento, são eles:

- **Encapsulamento L2TP** - O pacote é encapsulado, recebendo um cabeçalho PPP, em seguida um cabeçalho L2TP;
- **Encapsulamento UDP** - O pacote L2TP é encapsulado pelo protocolo UDP, e enviado à porta 1701;
- **Encapsulamento IPSec** - Esta é a fase em que o pacote ganha segurança. Aqui os dados são criptografados e autenticados;
- **Encapsulamento IP** - Aqui o pacote ganha uma etiqueta com informações referentes ao destino e origem;
- **Encapsulamento de camada de transporte** - Por fim o pacote recebe um cabeçalho que representa os padrões utilizados pela rede na camada de transporte.

5.2.3 IPSEC

IPSec segundo Kolesnikov e Hatch (2002, p.133), "É um conjunto de protocolos desenvolvidos para proteger o tráfego dos pacotes IP". Sua principal caracterista é prover a privacidade, integridade e autenticidade dos dados na comunicação. Foi desenvolvido pela IETF com o intuito de ser o protocolo padrão de endereçamento

da próxima versão do IP (IPv6). Assim, a tecnologia IPSec é uma das opções para implementar VPNs e seus serviços podem ser implementados para quaisquer protocolos das camadas superiores como TCP, UDP, ICMP, etc.

Mais informações sobre o IPSec, pode ser encontrada nas RFC 2401 (S.Kent; R.Atkison, 1998) e RFC 2411 (R.Thayer; N. Doraswamy; R. Glenn, 1998), `http://www.ietf.org/rfc/rfc2401.txt` e `http://www.faqs.org/rfcs/rfc2411.html`.

O IPSec foi desenvolvido para prover serviços de segurança baseados em criptografia, tanto para o nível IP como para camadas superiores. Esses serviços são implementados com a utilização conjunta de protocolos de segurança de tráfego de dados, autenticação de cabeçalhos AH (*Authentication Header*) RFC 2402 (S.Kent; BBN Corp.; R.Atkinson; @Home Network; 1998), de encapsulamento seguro do conteúdo dos dados ESP (*Encapsulating Security Payload*), RFC 2406 (S.Kent; BBN Corp.; R.Atkinson; @Home Network, 1998), protocolos de gerência de chaves IKE (*Internet Key Exchange*), RFC 2409 (D.Harkins; D.Carrel; Cisco System, 1998) e ISAKMP (*Internet Security Association and Key Management Protocol*), RFC 2408 (D.Maughani; National Security Agency; M.Schertler; Securify inc; M.Schneider; J.Turner; RABA Tecn.; 1998), porém o IKE é o mais utilizado.

Um dos conceitos fundamentais do IPSec é o conceito de SA (*Security Association*). Uma SA é uma "conexão"que viabiliza o tráfego de serviços de forma segura entre computadores ou *gateways*, utilizando protocolos de segurança (AH, ESP ou ambos), quando é usado o AH e o ESP em conjunto, mais de uma SA deve ser definida.

Uma SA é definida por três parâmetros: o SPI (*Security Parameter Index*), o endereço IP de destino e o identificador do protocolo (AH ou ESP). Todos os dados das SAs são armazenados em um banco de dados chamado SAD (*Security Association Database*), tendo seu conteúdo restrito a pessoas autorizadas. O SPI é um número que identifica uma SA é definido durante a negociação que antecede o estabelecimento da mesma. Todos os membros de uma SA devem conhecer o SPI correspondente e usa-lo durante a comunicação.

O endereço IP de destino, pode ser *unicast, broadcast e multicast*, mas para a

definição dos mecanismos de gerenciamento de SA, o IPSec assume um endereço destino *unicast*, extendendo as definições para *broadcast* e *multicast*. O identificador de protocolo é um número, 51 para AH e 50 para ESP.

Outro ponto importante na arquitetura IPSec é o SPD (*Security Policy Database*), responsável por determinar qual ação a ser feita sobre o pacote ou seja, determinar se o mesmo seguirá em frente, seguirá após aplicar o protocolo IPSec, ou será descartado. Enquanto o SAD apenas referencia as políticas a serem usadas através de parâmetros, o SPD força política de segurança sobre os pacotes.

O tratamento de pacotes recebidos (tráfego que entra) é diferente do tratamento de pacotes enviados (tráfego que sai). No tráfego que entra, a identidade do emissor e a integridade do pacote serão verificados pelo SAD, para depois envia-los à regra SPD correspondente. Já no tráfego que sai, as regras são verificadas pelo SPD e determinada qual ação será aplicada ao pacote e se necessário, dependerá da políca de segurança, o IPSec consultará o SAD.

O IPSec possui dois modos de trabalho para envio de dados entre os pontos de comunicação, modo de Transporte e modo Túnel. No modo Transporte, a transmissão do pacote protegido pelo IPSec é feita de *host* para *host*, onde o *host* é o responsável pelo encapsulamento IPSec. Os pacotes criados neste modo, são adicionados cabeçalhos IPSec, (AH ou ESP), entre o cabeçalho IP original e os cabeçalhos dos protocolos de mais alto nível como, TCP ou UDP.

No modo Túnel, o *host* não possui suporte a IPSec, então o tráfego gerado é capturado por um *gateway*, que faz o encapsulamento IPSec e envia o pacote até outro *gateway*, que desencapsula e envia os dados ao *host*. Neste modo, é criado um cabeçalho IP externo que especifica o destino no contexto IPSec e o cabeçalho interno e os dados, serão encriptados pelo cabeçalho IPSec (AH ou ESP).Este modo é mais seguro que o modo transporte.

O protocolo AH, (*Authentication Header*), ver figura 5.6 é quem garante a autenticidade e integridade do pacote, ou seja, que este não foi alterado durante a transmissão.

A seguir a descrição dos campos que compõem o cabeçalho do protocolo AH:

- Próximo Cabeçalho - Contém o identificador do próximo cabeçalho;
- Comprimento do Conteúdo - Comprimento do *Payload*;

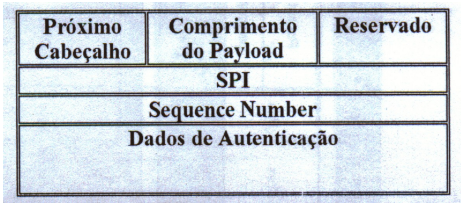

Figura 5.6: Cabeçalho do protocolo AH

- Reservado - 16 *bits* reservados para extensão do protocolo;
- SPI *Security Parameter Index* - Este índice em conjunto com o protocolo AH e o endereço fonte, identifica de forma única uma SA para um determinado pacote;
- *Sequence Number* - Contador que identifica os pacotes pertencente a uma determinada SA (usado como mecanismo *anti-replay*);
- Dados de Autenticação - Este campo tem comprimento variável e contém o ICV (*Integrity Check Value*) para este pacote, calculado seguindo o algorítmo de autenticação usado.

O protocolo AH adiciona autenticação, porém os dados trafegam na rede sem uma proteção e podem ser capturados. Este problema é resolvido com outro protocolo o ESP, que adiciona a confidencialidade. Alguns ataques podem ser evitados com a utilização do protocolo AH, são eles:

- *Replay*, quando o atacante intercepta um pacote válido e autenticado pertencente à uma conexão, replica-o e o reenvia. Este ataque é evitado através do campo *Sequence Number*, que enumera os pacotes que trafegam em uma determinada SA;
- *Spoofing*, quando um atacante assume o papel de uma máquina confiável para o destino, ganhando assim privilégios na comunicação. O uso de autenticação previne este tipo de ataque;
- *Connection hijacking*, ou "roubo de conexões", quando um atacante intercepta uma pacote no contexto de uma conexão e com isso passa a participar da comunicação. Mecanismos de autenticação previnem este tipo de ataque.

O protocolo ESP *Encapsulating Security Payload*, ver figura 5.7 é quem garante que somente os destinatários autorizados, terão acesso ao conteúdo do pacote, adicionando **autenticação e confidencialidade** ao mesmo.

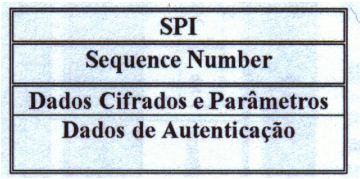

Figura 5.7: Cabeçalho do Protocolo ESP

A seguir os campos que compõem o cabeçalho ESP:

- SPI (*Security Parameter Index*) - Idem ao protocolo AH;
- *Sequence Number* - Idem ao protocolo AH;
- Dados Cifrados e Parâmetros - Contém os dados cifrados e os parâmetros utilizados pelo algorítimo de criptografia usado;
- Dados de Autenticação - Campo de comprimento variável que contém o ICV para este pacote, calculado a partir do algorítmo de autenticação usado.

O uso do ESP pode evitar os seguintes ataques:

- *Replay*, de maneira análoga ao AH, através do campo *Sequence Number*;
- Particionamento de pacotes cifrados, quando o atacante obtém parte de pacotes cifrados e consegue montar um pacote que pode ser aceito por um dos membros da conexão. A autenticação evita este tipo de ataque;
- *Sniffer*, quando o atacante obtém os pacotes que trafegam na rede, este tipo de ataque é evitado com a criptografia.

5.2.4 Comparacao entre Protocolos de Tunelamento

A tabela 5.1 mostra a potencialidade de cada protocolo de tunelamento.

Propriedades	Descricao	PPTP	L2TP	L2TP/IPSec	IPSec
Autenticação de Usuário	Consegue autenticar os usuários que queiram estabelecer uma conexão.	SIM	SIM	SIM	Implementação em andamento pelo grupo de trabalho IPSec da IETF[1]
Autenticação de Computadores	Autentica computadores envolvidos na conexão	SIM	SIM	SIM	SIM
Suporte a NAT	Passa por meio de um NAT para esconder os pontos finais da conexão	SIM	SIM	NÃO	NÃO
Suporte a Multiprotocolo	Define um método padrão para o tráfego IP e não IP	SIM	SIM	SIM	Implementação em andamento pelo grupo de trabalho IPSec da IETF
Atribuição Dinâmica de Endereço IP	Define uma negociação de endereçamento IP entre o servidor VPN e seus clientes. Isso elimina configurações manuais do protocolo IP.	SIM	SIM	SIM	Implementação em andamento pelo grupo de trabalho IPSec da IETF
Encriptação	Pode criptografar o tráfego corrente	SIM	SIM	SIM	SIM
Uso de PKI	Usa infra-estrutura de chave pública para implementar a criptografia e a autenticação	SIM	SIM	SIM	SIM
Autenticação de Pacotes	Provê um método de autenticação que garante que os pacotes não foram alterados durante a transmissão	NÃO	NÃO	SIM	SIM
Suporte a Multicast	Pode efetuar tráfego em multicast	SIM	SIM	SIM	NÃO

Tabela 5.1: Comparação entre protocolos

Capítulo 6

VPN - *Virtual Private Network*

6.1 Conceitos de VPN

Com o advento da internet, sua estrutura tem sido usada para interligar redes corporativas. O grande desafio está em garantir a segurança dos dados transmitidos, uma vez que a internet não prima pela segurança. Garantir principalmente que os dados não sejam modificados durante a transmissão, que as partes envolvidas na transmissão (origem e destino) sejam identificadas corretamente e manter o sigilo, isto é, não permitir que pessoas não autorizadas identifiquem o conteúdo da mensagem. A VPN surgiu com este propósito, garantir integridade, autenticidade, confidencialidade e controle de acesso, reduzindo o risco de ataques externo como, IP *Spoofing*, *man-in-the-middle*.

A VPN possue seus próprios protocolos de comunicação, dentre eles PPTP, L2TP e o IPSec, que atuam em conjunto com o TCP/IP, criando um túnel virtual onde os dados trafegam criptografados, garantindo a ilegibilidades dos mesmos à pessoas não autorizadas. Os protocolos de autenticação são usados, para garantir que as mensagens tenham vindo de usuários válidos e que se parte da mensagem for alterada, o pacote será descartado.

Algumas implementações de VPN utilizam *software* cliente nos computadores que se conectam à rede corporativa através da internet. Essas implementações são as chamadas VPDN(*Virtual Private Dial Network*), são ativadas pelo usuário após conexão com a internet através de *software* específico. Essa opção é muito usada por clientes móveis. Outras implementações utilizam criptografia entre *gateways*

VPN, não sendo necessária intervenção do usuário, muitas vezes ele nem sabe que esta usando VPN. Essa transparência permite que usuários, aplicações e computadores acessem recursos remotamente como se estivessem na rede local.

Uma das maiores vantagens em se implementar uma VPN é a redução de custo com o uso de serviços de rede oferecidos pelas operadoras de telecomunicações. Apesar disso, existem algumas desvantagens como consumir muito tempo para implementa-la, caso não haja um planejamento adequado, dificuldade na localização de defeitos, a relação de confiança entre as redes interconectadas e a disponibilidade da internet.

Pelo fato dos dados trafegarem criptografados em uma VPN, a localização de defeitos como, falhas de autenticação, a não sincronização das chaves, pacotes perdidos e sobrecarga do *gateway* VPN, pode ser um problema.

Outro ponto que deve ser bem planejado é a relação de confiança entre as redes, pois os recursos compartilhados de uma rede, ficarão acessíveis à outra. Isso pode ocasionar falhas de segurança na VPN, tornando-a vulnerável a ataques externo, caso uma das redes não possua uma segurança adequada.

As VPNs podem constituir uma alternativa segura na transmissão de dados através de redes públicas ou privadas, uma vez que oferecem recursos como autenticação e criptografia com níveis variados de segurança. Entretanto existem algumas aplicações em que o tempo de transmissão é crítico e que o uso de VPNs deve ser analisada com muito cuidado, pois podem ocorrer problemas com o desempenho ou atrasos na transmissão, comprometendo a qualidade do serviço oferecido.

6.2 FreeS/Wan

O projeto *FreeS/Wan* fundado pelo americano John Gilmore e desenvolvido por um grupo de canadenses, devido as leis americanas que proíbem a exportação de qualquer software criptográfico de computadores, sem a prévia autorização do governo americano, mesmo quando o software seja de domínio público. Segundo a documentação oficial, o *FreeS/Wan* é uma implemantação para Linux do protocolo IPSec, que provê criptografia e autenticação à serviços baseados no protocolo IP.

O IPSec difere de outros protocolos, protege todo o tráfego em uma rede baseada no protocolo IP. O IPSec pode ser implementado em *Gateway* em roteadores[1], *firewall* e computadores para usuários finais. Como exposto anteriormente, o IPSec utiliza os protocolos AH para prover serviços de autenticação baseados em níveis de pacotes, ESP para prover serviços de autenticação e criptografia e IKE que negocia parâmetros de conexão, incluindo trocas de chaves criptográficas.

O FreeS/Wan tem como padrão os algorítmos de chaves assimétricas RSA, simétrica 3DES, funções *hash* MD5 e SHA-1. Entretanto muitas funções extras são incorporadas ao programa original por programadores autônomos, com o intuito de aumentar a segurança ou mesmo permitir que o FreeS/Wan seja utilizados em vários ambientes. A criptografia de chave simétrica, AES, *Blowfish*, permitir a passagem pelo NAT *Network Address Translation* quando implementado, suporte a certificados digitáis X.509, são algumas destas funções. O Super FreeS/Wan como é conhecido, é um *mirror* do FreeS/Wan, onde podemos encontrar as principais funções exixtentes incorporada ao programa principal FreeS/Wan. Dois componentes são fundamentais para o FreeS/Wan:

- KLIPS (*Kernel IPSec Support*), responsável pela interação do IPSec com o kernel [2] do Linux, para o tratamento de pacotes IPSec, além da criação de cabeçalhos AH e ESP, tanto para pacotes que saem como para os que entram, encriptação e cálculos de autenticação de pacotes. Verifica também todos os pacotes que não são IPSec, para assegurar que eles não estão burlando as políticas de segurança;

- PLUTO *Daemon*[3]*IKE*, responsável por implementar o protocolo IKE, e pelo tratamento de associações seguras do ISAKMP. Executa as autenticações do *host*, cria as Associações Seguras do IPSec, repassa os dados requeridos ao KLIPS, adequa as configurações do *firewall* necessárias ao IPSec.

[1] Equipamento utilizado para interligar redes, locais e remotas

[2] Parte principal do Sistema Operacional, mais detalhes em http://www.kernel.org

[3] Um *Daemon* é um programa permanentemente ativo, que aguarda instruções externas para executar ações específicas

6.3 OpenVPN

Outra opção usada em construção de VPN é o *OpenVPN*. Ele é muito simples comparado ao FreeS/Wan. O *OpenVPN* simplesmente pega a informação a ser enviada, criptografa e envia pela *internet*, por um pacote UDP. A grande vantagem em relação ao FreeS/Wan é que ele não tem grandes problemas para passar por firewalls ou roteadores que fazem NAT e possui uma configuração mais simples. A seguir algumas características do *OpenVPN*:

- pode criar tunel para qualquer sub-rede IP ou adaptador *ethernet*, por um porte UDP ou TCP;
- Usa todas as caracteristicas de encriptação, autenticação e certificação da biblioteca *OpenSSL*, para proteger o tráfego privado pela *internet*;
- Qualquer cifragem, tamanho de chave, para datagrama de autenticação, suportado pela biblioteca *OpenSSL*;
- Pode-se escolher entre encriptação convencional, baseado em chave estática ou certificados baseados em encripatação de chave pública;
- Pode-se criar tuneis através de *firewalls*, sem ter que criar regras específicas.

Para maiores detalhes consulte `http://openvpn.sourceforge.net`.

6.4 OpensWan

O *OpensWan* é uma implementação do protocolo *IPSec* para o sistema operacional Linux, surgiu a partir do projeto *FreeS/Wan*. O *Openswan* suporta *Kernels* 2.0, 2.2, 2.4 e 2.6. Esse projeto está sendo apoiado pela Novell[4], que está expandindo seu comprometimento com os padrões *open-source*, maiores detalhes sobre o assunto acesse `http://novell.com/news/press/archive/06/pr04040.html`. A seguir algumas caracteristicas do *OpensWan*.

- Suporta *Linux Kernels* 2.4 via *KLIPS* e *Linux Kernels* 2.6 com *IPSec* nativo;
- A Versão 2.1, suporta certificados X509, NAT-T e XAUTH Cliente e Servidor.

Para maiores detalhes consulte `http://www.openswan.org`.

[4]Empresa fabricante de *software*

Capítulo 7

Implementando VPN em Linux

A implementação de uma VPN depende da escolha do sistema operacional que será usado nos *gateways* e a ferramenta usada para implementar um dos protocolos já citados anteriormente. Dependendo do sistema operacional utilizado, várias ferramentas podem ser encontradas, gratuitas ou pagas.

O sistema operacional escolhido neste trabalho foi o Linux, por ser um sistema bastante flexível e com caracteristicas como implementar regras de *firewall* a nível de *kernel*, possibilitando maior segurança da rede. Outro fator importante é que o Linux é gratuito, ou seja, não é necessário licença para utiliza-lo, podendo obte-lo através de *download* na internet ou em revistas de informática.

A distribuição escolhida foi *Red Hat* Linux, por sua reconhecida estabilidade, ser usada em servidores em todo mundo e servir de modelo para o desenvolvimento de outras distribuições. Para a configuração da VPN no sistema operacional Linux, foi escolhido a ferramenta FreeS/Wan(*Free Secure Wan*), que implementa o protocolo IPSec para proteger todo tráfego da rede baseado em IP. O *Red Hat* usado será a versão 9 e *kernel* 2.4.20-8.

A próxima etapa, será a implementação de uma VPN de forma experimental, mostrando o ambiente de testes, configurações e os resultados obtidos.

7.1 Ambiente de Teste

O ambiente de teste, simula duas redes distintas de uma mesma empresa, denominadas matriz e filial. Cada uma das redes possui um *gateway* VPN e um *host* (estação de trabalho). Para simular o meio de comunicação (*Internet*) entre as redes, será usado um computador que funcionará como roteador entre as mesmas.

Os computadores que serão usados como *gateway*, possuem duas placas de rede, uma voltada para rede interna, denominada Eth0 e outra voltada para o roteador (*Internet*) denominada Eth1. As estações de trabalho possuem apenas uma placa de rede em ambiente *Windows* [1].A implementação da VPN, foi baseada nas configurações de rede, como descrito abaixo :

Rede Matriz:

Gateway:
Hostname: GwMatriz *IP Gateway* : 200.140.0.2
Eth0 :
End. IP : 192.168.0.1 *Network* : 192.168.0.0
Netmask : 255.255.255.0 *Broadcast* : 192.168.0.255
Eth1 :
End. IP : 200.140.0.1 *Network* : 200.140.0.0
Netmask : 255.255.255.0 *Broadcast* : 200.140.0.255

Estação Matriz :

Hostname : Ematriz End. IP : 192.168.0.2
Netmask : 255.255.255.0 *Network* : 192.168.0.0
Gateway : 192.168.0.1 *Broadcast* : 192.168.0.255

ROTEADOR :

Hostname : Internet *Gateway* : 200.140.0.2
Eth0 :
End. IP : 200.140.0.2 *Network* : 200.140.0.0
Netmask : 255.255.255.0 *Broadcast* : 200.140.0.255

[1] Sistema Operacional da Microsoft

Eth1 :
End. IP : 200.140.1.2 *Network* : 200.140.1.0
Netmask : 255.255.255.0 *Broadcast* : 200.140.1.255

Rede Filial :

Gateway :
Hostname : GwFilial *Gateway* : 200.140.1.2
Eth0 :
End. IP : 192.168.1.1 *Network* : 192.168.1.0
Netmask : 255.255.255.0 *Broadcast* : 192.168.1.255
Eth1 :
End. IP : 200.140.1.1 *Network* : 200.140.1.0
Netmask : 255.255.255.0 *Broadcast* : 200.140.1.255

Estação Filial :

Hostname : EFilial End. IP : 192.168.1.2
Netmask : 255.255.255.0 *Network* : 192.168.1.0
Gateway : 192.168.1.1 *Broadcast* : 192.168.1.255

7.2 Instalando o FreeS/Wan

Existe duas formas de instalação do pacote FreeS/Wan: obter os fontes e fazer a compilação do *Kernel*, ou através de pacotes pré-compilados *Red Hat Packet Manager* RPM. Em ambos os casos é necessário o uso da conta *root*[2].

Os Arquivos fontes podem ser obtidos através do *site*:

* http://www.freeswan.org/downloads.html

Ou diretamente no Linux com o comando:

* ncftpget ftp://ftp.xs4all.nl/pub/crypto/freeswan/freeswan-/∗

[2]Conta de usuário que possui todos privilégios no Linux

Outro ponto importante é verificar a autenticidade do arquivo obtido, através dos arquivos de assinaturas, para isso, deve-se fazer o *download* do arquivo *freeswan-sigkey.asc* que é a chave pública do desenvolvedor e deve ser importada para o sistema do leitor pelo comando:

* pgp -ka freeswan-sigkey.asc

A autenticidade dos arquivos-fonte do FreeS/Wan, é verificado com o comando:

* pgp freeswan-2.06.tar.gz.sig freeswan-2.06.tar.gz

Após este comando, deve ser exibida uma mensagem semelhante a:

* Good signature from user linux FreeS/Wan Software Team (build@freeswan.org).
* Signature made 2002/03/29 20:50 GMT using 2047-bit key, key ID 46EAFCE1.

Uma vez verificado a autenticidade do arquivo, deve-se descompacta-lo usando o seguinte procedimento:

* mv freeswan-2.06.tar.gz /usr/src
* cd /usr/src
* tar -xzf freeswan-2.06.tar.gz.

Após a descompactação será criado o diretório /usr/src/freeswan-2.06. Caso seja necessário a inclusão de *patches*[3], como suporte a NAT, certificados X.509, entre outros, deve-se faze-lo agora. Caso haja instalação de *patches*, deve-se recompilar o *Kernel* e o FreeS/Wan.

Para compilar o FreeS/Wan como módulo use o seguinte procedimento:

* Mudar para o diretório /usr/src/freeswan-2.06
* Executar make menumod
* Executar make minstall

Para o *link* estático do FreeS/Wan no *Kernel* use o seguinte procedimento:

[3] Alteração de um programa, que acrescenta ou modifica uma pequena parte deste

* make menugo
* make minstall

O segundo método, adotado neste trabalho, é um pouco mais simples e não requer a compilação do *Kernel*, instalar o FreeS/Wan através de binários pré-compilados. Neste método são necessários dois arquivos que dependem da versão do *Kernel* e da distribuição utilizada. Neste caso a distribuição usada é a Red Hat 9.0 *Kernel* 2.4.20-8, os arquivos necessários são: freeswan-module-2.06_2.4.20-8.i386.rpm e freeswan-userland-2.06_2.4.20-8.i386.rpm.

Os arquivos RPM's podem ser obtidos através do endereço:

* ftp://ftp.xs4all.nl/pub/crypto/freeswan/binaries/RedHat-RPMs/2.4.20-8

Ou diretamente no Linux com o comando:

* ncftpget ftp:ftp.xs4all.nl/pub/crypto/freeswan/binaries/RedHat-RPMs /‘uname -r|tr -d ‘a-wy-z‘’/

Como dito anteriormente, é importante verificar a autenticidade do arquivo obtido, através dos arquivos de assinaturas. Para verificar a autenticidade do arquivo, deve-se fazer o *download* do arquivo *freeswan-rpmsign.asc* que é a chave pública do desenvolvedor e deve ser importada para o sistema do leitor pelo comando:

* rpm –import freeswan-rpmsign.asc (somente para RedHat 8.x ou superior)

Verifique a autenticidade dos arquivos RPMs, executando o comando:

* rpm –checking freeswan*.rpm

Após este comando, a mensagem exibida é semelhante a:

* freeswan-module-2.06_2.4.20-8.i386.rpm: pgp md5 OK
* freeswan-userland-2.06_2.4.20-8.i386.rpm: pgp md5 OK

Caso ocorra erro, deve-se fazer o *download* dos arquivos novamente.

Uma vez verificado a autenticidade e não ocorrer nenhum erro, os arquivos são instalados com o comando:

* rpm -ivh freeswan*.rpm

A instalação pode ser feita também através de ferramentas gráficas, disponíveis na maioria das distribuições Linux.

Para iniciar o FreeS/Wan, digite o comando:

* service ipsec start.

Para verificar se a instalação foi bem sucedida, digite o comando:

* ipsec verify

Com o comando digitado acima, deve-se ter a seguinte saída:

**Checking your system to see if IPsec got installed and started corretely*
**Version check and ipsec on-path* [OK]
**Checking for KLIPS support in kernel* [OK]
**Checking for RSA private key (/etc/ipsec.secrets)* [OK]
**Checking that pluto is running* [OK]

Se ocorrer algum erro, existe um guia disponível em:
**http://freeswan.org/freeswan_trees/freeswan-2.06/doc/trouble.html*

7.3 Configurando o FreeS/Wan

Antes de configurar o FreeS/Wan, é necessário alterar algumas configurações nos *Gateways* VPN. É necessário habilitar o repasse de pacotes IP (IP *Forwarding*) e desabilitar o filtro de caminho inverso (*Reverse Filter*). Este filtro é uma caracteristica de segurança do Linux que previne ataques DoS[4]. Ele permite que o *kernel* abandone os pacotes que entram na rede, cuja resposta não correspondam à mesma

[4]Visa interromper o serviço oferecido, enviando requisições em massa para um determinado computador, de modo que o mesmo não consiga responder a todas elas

interface daqueles que chegaram.

Para o FreeS/Wan isso é um problema, já que ele cria interfaces do tipo ipsecN, onde N corresponde a um número que começa em 0, fazendo com que o *Kernel* abandone os pacotes pertencentes à VPN. Para que isso não ocorra, é necessário editar os arquivos */etc/sysctl.conf e /etc/rc.d/rc.local* e alterar alguns parâmetros. Um exemplo desse arquivos é representado a seguir.

Parâmetros a serem alterados ao arquivo *Arquivo /etc/sysctl.conf*:

 # Controls IP packet forwarding
* *net.ipv4.ip_forward* = 0 (Alterar para 1)
 # Control source route verification
* *net.ipv4.conf.default.rp_filter* = 1 (Alterar para 0)

Linhas a serem adicionadas ao arquivo *Arquivo /etc/sysctl.conf*:

* *echo 0 > /proc/sys/net/ipv4/conf/ipsec0/rp_filter*
* *echo 0 > /proc/sys/net/ipv4/conf/eth1*[5]*/rp_filter*

Caso exista um *firewall* na rede, é preciso permitir todo tráfego pela porta UDP 500, utilizada pelo IKE para negociar as chaves e permitir os protocolos 50 (ESP) e o 51 (AH). O *script* abaixo, permite o acesso VPN através do *firewall*, baseado em *iptables*.

```
# Permite negociação pelo IKE
iptables -A INPUT -p udp –sport 500 –dport 500 -j ACCEPT
iptables -A OUTPUT -p udp –sport 500 –dport 500 -j ACCEPT

# Permite Protocolo ESP
iptables -A INPUT -p 50 -j ACCEPT
iptables -A OUTPUT -p 50 -j ACCEPT

# Permite Protocolo AH
iptables -A INPUT -p 51 -j ACCEPT
```

[5] interface que estiver sendo usada pelo ipsec

iptables -A OUTPUT -p 51 -j ACCEPT

Para a configuração do FreeS/Wan, dois arquivos são a base do funcionamento da VPN, o */etc/ipsec.conf* que armazena as configurações das conexões VPN (túneis), e o */etc/ipsec.secrets* que armazena as chaves privadas RSA. As chaves são geradas automaticamente durante a instalação, poupando tempo de administração. Nas versões anteriores a 1.98b do FreeS/Wan era necessário gerar as chaves RSA manualmente.

Outro ponto importante em relação à segurança, somente o usuário *root* deve ter acesso aos arquivos ipsec.conf e ipsec.secrets, os demais usuários não devem ter qualquer tipo de acesso. Para a, permissão digite o comando:

* *chown root.root /etc/ipsec.**
* *chmod 600 /etc/ipsec.**

O arquivo de configuração do FreeS/Wan */etc/ipsec.conf* utilizado nos *gateways*, divide-se em dois tipos de seções, *config e conn*. O primeiro tipo, a seção *config setup*, define as configurações básicas que serão utilizadas na inicialização do FreeS/Wan. O segundo tipo é formado por uma ou mais seções *conn*, que definem as conexões VPN que serão criadas pelo IPSec. A conexão *default* define os parâmetos que serão usados por todas as outras conexões.

O arquivo */etc/ipsec.secrets* por ser muito extenso, será apresentado apenas a parte que interessa, a *pubkey*, onde está presente a chave pública do *gateway*. Esta chave será usada posteriormente para a autenticação dos *gateways* VPN.

A seguir a *pubkey* do arquivo */etc/ipsec.secrets*.

pubkey=0sAQNpf20aHuT+Yh4O605x+ojSAFfghi/D2bDXQVQxaCDKpJSc
MlrlKYgKatZVXYlkS+iMwTsUkBHLQqAWAe7tIxdQL8wlmavqCMmDW
52OmujyojbXEo06kiSgWolL/0XLIKyGngSRmLSgcSjRoOTPvGXeCsEyP/
P+HwRwvIvcCMYHKKTL7NpA/SJMyVTuj4ZWamHg3ybIPK0hS77/uGb
B3FscpJSBX0YBHh9kv3BeYM4Mv8ZtG5lllC0ArmpO99BE3x2alQA/rjIo
Vr8DelJKJ3gcU7KNR0mqRc+g3e7CB/i2NHzcEWJyHic2Re21QLNlvY6v
qSUkpeoSDFHOlX6piPyE9375nOfFnMjHlBOHoGTQss5tYhVfutDoVXJe
5EC3FIEhcCM=

O arquivo */etc/ipsec.conf* do *gateway* VPN foi configurado da seguinte forma:

```
# basic configuration

config setup
Debug-logging controls: "none"for (almost) none, "all"for lots.
interfaces=%defaultroute
klipsdebug=none
plutodebug=none
uniqueids=yes
plutoload=%search
plutostart=%search

conn %default
keyingtries=0
disablearrivalcheck=no
authby=rsasig
leftrsasigkey=%dns
rightrsasigkey=%dns

# Conexão matriz-filial

conn matriz-filial
keylife=8h
left=200.140.0.2
leftsubnet=192.168.0.0/24
leftnexthop=200.140.0.1
leftrsasigkey=0sAQNpf20aHuT+..(Chave Pública gateway matriz)
leftid=@server.matriz.com
right=200.140.1.2
rightsubnet=192.168.1.0/24
rightnexthop=200.140.1.1
rightid=@server.filial.com
rightrsasigkey=0sAQOCmYYX9xmQ6No4..(Chave Pública gateway filial)
auto=add
```

A seguir um descritivo dos campo do arquivo */etc/ipsec.conf*.

- *Interfaces* - Define as interfaces físicas e virtuais que o IPSec utiliza. A opção %defaultroute, faz com que o programa procure a interface automaticamente ou *interfaces= "ipsec0=ethx"*.
- *Klipsdebug* - Determina quanto da saída de depuração do *KLIPS* deve ser armazenado. O valor *none* significa que nada será armazenado.
- *Plutodebug* - Determina quanto da sída de depuração será armazenado. O valor *none* significa que nada será armazenado.
- *Plutoload* - Identifica quais conexões são carregadas à base de dados do *Pluto* na inicialização do Freeswan. O parâmetro *search* define que as conexões que tenham *auto=route, auto=add* ou *auto=start* são carregadas.
- *Plutostart* - Identifica quais conexões tetam ser negociadas no inicio deo programa, onde o valor *%search* equivale a rotear todas as conexões que possuem as linhas *auto=route* ou *auto=start*, além de iniciar as que contenham *auto=start*.
- *Uniqueids* - Determina se o ID de um *gateway* deve ser único.
- *Keyingtries* - Determina o número de tentativas a serem feitas para negociar uma conexão. O valor 0 significa que o protocolo nunca desiste da negociação.
- *Authby* - Define como os *gateways* farão a autenticação entre eles. Os valores possíveis são *secret*,(utilizado para chaves secretas compartilhadas), *rsasig* (utilizado para assinaturas digitais RSA).
- *keylife* - Tempo em que a conexão estará ativa, após este tempo a conexão expirará. Este valor pode ser inteiro seguido da unidade de tempo (**s** para segundos, **m** para minutos, **h** para horas ou **d** para dias)
- *Left* - Endereço IP do *gateway* esquerdo. A ordem dos *gateways* (esquerda ou direita), não interfere no funcionamento da VPN.
- *Leftsubnet* - Endereço IP da subrede privada, atrás do *gateway* esquerdo
- *Leftnexthop* - Próximo endereço IP a ser alcançado pelo *gateway* esquerdo em direção à rede.

- *Leftid* - Define como o *gateway* esquerdo deve ser identificado para a autenticação.
- *Leftrsasigkey* - Contém a chave pública do *gateway* esquerdo.
- *Right* - Endereço IP do *gateway* direito.
- *Rightsubnet* - Endereço IP da subrede privada, atrás do *gateway* direito.
- *Rightnexthop* - Próximo endereço IP a ser alcançado pelo *gateway* direito em relação à rede pública.
- *Rightid* - Define como o *gateway* direito deve ser identificado para a autenticação.
- *Rightrsakey* - Contém a chave pública do *gateway* direito.
- *Auto* - Especifica que operação deve ser feita quando o *IPSec* for iniciado. A opção *start* indica que a conexao é iniciada quando o *IPSec* for iniciado. A opção *add* indica que a conexão é iniciada através do comando do *IPSec* " ipsec auto –up vpn ".

Após a configurar o arquivo */etc/ipsec.conf* em um dos *gateways*, deve-se copia-lo para o outro *gateway* e inicilizar o serviço com o comando:
* ipsec auto –up matriz-filial

Uma situação bastante comum é quando temos usuários remotos querendo conectar-se à rede corporativa sem um endereço IP fixo, os chamados *Road Warrior*. Neste caso a configuração dos arquivos */etc/ipsec.conf* nos *gateways* ficará da seguinte forma:

Arquivo do *gateway* matriz (IP fixo):

```
# Basic configuration

config setup
# Debug-logging controls: "none"for (almost) none, "all"for lots.
interfaces=%defaultroute
klipsdebug=none
plutodebug=none
```

```
uniqueids=yes

conn %default
keyingtries=0
disablearrivalcheck=no
authby=rsasig
# Conexão matriz-filial

conn matriz-filial
keylife=8h
left=200.140.0.2
leftsubnet=192.168.0.0/24
leftnexthop=200.140.0.1
leftrsasigkey=0sAQNpf20aHuT....
leftid=@server.matriz.com
right=%any
rightsubnet=192.168.1.0/24
rightnexthop=%defaultrout
rightid=@server.filial.com
rightrsasigkey=0sAQOCmYYX9xm....
auto=add
```

Arquivo do *gateway* cliente (IP dinâmico):

```
# Basic configuration

config setup
interfaces=%defaultroute
klipsdebug=none
plutodebug=none
uniqueids=yes

conn %default
keyingtries=0
disablearrivalcheck=no
authby=rsasig
```

```
# ConexÃo matriz-filial

conn matriz-filial
keylife=8h
left=%defaultroute
leftsubnet=192.168.1.0/24
leftnexthop=
leftrsasigkey=0sAQOCmYYX9xm.....
leftid=@server.filial.com
right=200.140.0.2
rightsubnet=192.168.1.0/24
rightnexthop=200.140.0.1
rightid=@server.matriz.com
rightrsasigkey=0sAQNpf20aHut.....
auto=add
```

Concluido o processo de configuração do FreeS/Wan, o próximo passo é inicializar as conexões e verificar se o túnel foi estabelecido. Executar nos *gateways* o comando:
* *ipsec auto –up matriz-filial*

Este comando provoca a seguinte saída nos *gateways* “matriz e filial” respectivamente:
* *gateway* matriz:

```
104 "matriz-filial"#1: STATE_MAIN_I1: initiate
106 "matriz-filial"#1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "matriz-filial"#1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "matriz-filial"#1: STATE_MAIN_I4: ISAKMP SA established
112 "matriz-filial"#2: STATE_QUICK_I1: initiate
004 "matriz-filial"#2: STATE_QUICK_I2: sent QI2,IPsec SA established
ESP=>0x2aee9326<0xa93e1378
```

* *gateway* filial:

```
112 "matriz-filial"#3: STATE_QUICK_I1:initiate
004 "matriz-filial"#3: STATE_QUICK_I2: sent QI2,IPsec SA established
ESP=>0xa93e1379<0x2aee9327
```

Em linha de comando, a opção *look* do comando ipsec, pode mostrar mais informações sobre conexões VPN. A seguir um exemplo do comando *look*

* *ipsec look*

GwMatriz Fri Sep 3 14:29:03 BRT 2004
0.0.0.0/0 –> 0.0.0.0/0 => %trap (0)
192.168.0.0/24->192.168.1.0/24=>tun0x1004@200.140.1.2 esp0x2aee9327@200.140.1.2(70) 200.140.0.2/32 -> 0.0.0.0/0 => %trap (70)
ipsec0->eth1 mtu=16260(1443)->1500
esp0x2aee9326@200.140.1.2 ESP_3DES_HMAC_MD5: dir=out src=200.140.0.2 iv_bits= 64bits iv=0xb3697ac60a6fc843 ooowin=64 alen=128 aklen=128 eklen=192 life(c,s,h)=addtime(5166,0,0) refcount=4 ref=14
esp0x2aee9327@200.140.1.2 ESP_3DES_HMAC_MD5: dir=out src=200.140.1.2 iv_bits=64bits iv=0x82b02573badc1bfa ooowin=64 seq=70 alen=128 aklen=128 eklen=192 life(c,s,h)=bytes(9520,0,0) addtime(5151,0,0) usetime(578,0,0)packets(70,0,0) idle=509 refcount=4 ref=24
esp0xa93e1378@200.140.0.2 ESP_3DES_HMAC_MD5: dir=in src= 200.140.1.2 iv_bits= 64bits iv=0x99ff980a05de7eef ooowin=64 alen=128 aklen=128 eklen=192 life(c,s,h)=addtime(5166,0,0) refcount=4 ref=9
esp0xa93e1379@200.140.0.2 ESP_3DES_HMAC_MD5: dir=in src= 200.140.1.2 iv_bits=64bits iv= 0xce3bcc70d96039c6 ooowin=64 seq=70 bit=0xffffffffffffffff alen=128 aklen=128 eklen=192 life(c,s,h)= bytes(7280,0,0) addtime(5151,0,0) usetime(578,0,0)packets(70,0,0) idle=509 refcount=74 ref=19
tun0x1001@200.140.0.2 IPIP: dir=in src= 200.140.1.2 policy=192.168.1.0/24->192.168.0.0/24 flags=0x8<>life(c,s,h)=addtime(5166,0,0) refcount=4 ref=8tun 0x1002@200.140.1.2 IPIP: dir=out src=200.140.0.2 life(c,s,h)=addtime(5166,0,0) refcount=4 ref=13
tun0x1003@200.140.0.2 IPIP: dir= in src = 200.140.1.2 policy=192.168.1.0/24->192.168.0.0/24 flags=0x8<>life(c,s,h)=bytes(7280,0,0)addtime(5151,0,0)usetime(578,0,0)packets(70,0,0)idle=509 refcount=4 ref=18 tun0x1004@200.140.1.2 IPIP: dir=out src=200.140.0.2 life(c,s,h)=bytes(7280,0,0)addtime(5151,0,0) usetime(578,0,0)packets(70,0,0) idle=509 refcount=4 ref=23

```
Destination–>Gateway—>Genmask——>Flags–>MSS->Window–>irtt->Iface
0.0.0.0——>200.140.0.1—->0.0.0.0———->UG—->0——>0———>0—–>eth1
0.0.0.0——>200.140.0.1—->128.0.0.0—–>UG—->0——>0———>0–>ipsec0
128.0.0.0—>200.140.0.1—->128.0.0.0—>UG—->0——>0———>0->ipsec0
169.254.0.0–>0.0.0.0———->255.255.0.0–>U——>0——>0———>0—->eth1
192.168.1.0->200.140.0.1–>255.255.255.0->UG–>0——>0———->0–>ipsec0
200.140.0.0->0.0.0.0———>255.255.255.0->U—->0——>0———>0—>eth1
200.140.0.0->0.0.0.0———>255.255.255.0->U—->0——>0———>0->ipsec0
```

Neste trabalho também foi realizado um teste de *ping*[6] entre os *hosts* das redes matriz e filial. Foi usado o *tcpdump* (*tcpdump -i eth1*) na placa de rede do *gateway* matriz que comunica com a internet, com intúito de monitorar o tráfego que passa pelo mesmo. O resultado pode ser visto a seguir.

```
22:48:53.512259 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x120)
22:48:53.514609 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x120)
22:48:54.518125 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x121)
22:48:54.519859 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x121)
22:48:55.517987 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x122)
22:48:55.519687 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x122)
22:48:56.517873 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x123)
22:48:56.519599 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x123)
22:48:57.517847 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x124)
22:48:57.519591 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x124)
22:48:58.505791 arp who-has 200.140.0.1 tell GwMatriz
22:48:58.506140 arp reply 200.140.0.1 is-at 0:c0:26:ee:5a:7f
22:48:58.517827 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x125)
22:48:58.519487 200.140.1.2 > GwMatriz: ESP(spi=0x291ea72d,seq=0x125)
22:48:59.517726 GwMatriz > 200.140.1.2: ESP(spi=0x5adca34a,seq=0x126)
```

7.4 Encriptação Oportunista

A encriptação oportunista não é um padrão do protocolo IPsec, mas está sendo proposta como uma extensão do mesmo. Sendo assim, ela só é possível quando

[6] Comando usado para determinar se um computador está ativo e em quanto tempo ele responde a uma solicitação.

existem pelo menos dois *gateways* com o FreeS/Wan instalados.

Este tipo de encriptação permite aos *gateways* criptografar todo tráfego existente entre eles, mesmo não havendo nenhuma relação entre os mesmos. Para que isso aconteça, eles precisam ter suas informações de autenticação configuradas no DNS e o FreeS/Wan esteja habilitado para encriptar oportunistamente.

A encriptação oportunista pode ser dividida em dois tipos, parcial e total. Na parcial apenas é possível iniciar conexões oportunista, enquanto que na total é possível tanto inicia-las quanto aceita-las.

Com a encriptação oportunista, a carga de administração sobre o IPSec reduz drasticamente, pois não são necessárias configurações específicas de túneis, apesar de existir a possibilidade de configura-los para casos específicos. Outra vantagem é que todo tráfego transmitido e recebido é encriptado, tornando o tráfego da internet mais seguro.

Capítulo 8

Conclusão

O ojetivo deste trabalho foi mostrar como implementar uma VPN em ambiente Linux usando o protocolo IPSec, através do FreeS/Wan, detalhando aspectos de configuração, segurança e funcionalidade da mesma.

Outros protocolos além do IPSec foram abordados neste trabalho, como o PPTP e L2TP. Apesar de mostrarem-se eficientes, pecam no que diz respeito à segurança. Já o IPSec mostrou-se mais robusto, apresentando características mais expressivas no que diz respeito a proteção do tráfego entre as redes interligadas.

O FreeS/Wan mostrou-se capaz de implementar os protocolos do IPSec e fornecer características de segurança, como encriptação oportunista. Contudo a configuração do FreeS/Wan é relativamente complicada e documentação oficial é um tanto confusa, fazendo com que busque-se ajuda em documentação de terceiros, nem sempre confiáveis.

Verificou-se também, que por utilizar recursos criptográficos, a comunicação entre as redes é mais lenta, pois o tamanho dos pacotes aumentam, porém o tráfego é seguro para transmissão de informações confidenciais. Este problema pode ser solucionado, usando outros algorítmos criptográficos, o AES, que segundo entidades como a NIST e a própria equipe do FreeS/Wan é mais rápido que o 3DES.

Cada vez mais as VPNs são incorporadas às organizações, sejam elas privadas ou governamentais. Os principais motivos estão relacionados à segurança das informações e o custo no tráfego das mesmas. Esta segurança deve-se principal-

mente aos algoritmos criptográficos e protocolos específicos, dificultando e muito, a ação de possíveis invasores. As VPNs garantem também integridade, confidencialidade das informações transmitidas, bem como a autenticação e controle de acesso aos *gateways*, permitindo uma confiança maior por parte de quem as usa.

Apesar de toda segurança oferecida pela VPN, sozinha ela não garante total segurança de uma rede. É importante que se faça um planejamento cuidadoso, envolvendo políticas rígidas de segurança, tanto para parte física quanto para parte lógica dos servidores.

Este trabalho mostrou também que o Linux é um sistema operacional muito eficaz, tornando-se uma solução atrativa para organizações que não dispõem de muitos recursos financeiros para gastar com soluções proprietárias. Para reforçar esta afirmação, neste trabalho foram usados equipamentos considerados obsoletos (*Pentium 166*) e os resultados foram satisfatórios.

Referências Bibliográficas

[1] Eduardo Bellincanta Ortiz e Ed'Wilson Tavares Ferreira. *VPN Vitual Private Network, Implementando Soluções em linux*. ÉRICA, 2003.

[2] Andrew S.Tanenbaum. *Redes de Computadores*. CAMPUS, 1997.

[3] Rob Scrimger. *TCP/IP, a Biblia*. CAMPUS, 2002.

[4] Marcelo L. Jucá *Implementação de Firewall em Linux*. BRASPORT, 2003.

[5] Joaquim Quinteiro Uchôa. *Segurança em Redes e Criptografia*. UFLA/FAEPE, 2002.

[6] Fernando Cortez Sica e Joaquim Quinteiro Uchôa. *Administração de Sistemas Linux*. UFLA/FAEPE, 2002.

[7] Dêner Lima Fernandes Martins. *Redes Privadas Virtuais Com IPSec*. http://www.cic.unb.br/docentes/pedro/trabs/vpn.pdf, acesso em Maio de 2004.

[8] Curso de Rede de Computadores. *Apostila de Internet e Arquitetura TCP/IP*. PUC RIO/CCE.

[9] Erick Dantas Rotole. *Arquitetura IP Security*. http://www.rnp.br/newsgen/9907/ipsec3.html, acesso em Maio de 2004.

[10] Marco A.G. Rossi e Oswaldo Franzin *Virtual Private Network*. http://www.gpr.com.br/cursos/vpn/vpn.html, acesso em Março de 2004.

[11] Eduardo Tude. *Tutorial VPN*. htpp://www.teleco.com.br/tutoriais/tutorialvpn/, acesso em Abril de 2004.

[12] Arpad Magosanyi. *The VPN HOWTO*. http://www.sunsite.unc.edu/mdw/HOWTO/mini/vpn.html, acesso em Abril de 2004.

[13] Openvpn. openvpn Project - https://openvpn.sourceforge.net, acesso em Maio de 2004.

[14] FreeS/Wan. FreeS/Wan Project - http://freeswan.org, acesso em Maio de 2004

[15] Alan T. Vasques e Rafael P. Schuler. *Vpn em Linux utilizando IPSec*. http://www.agatetepe.com.br/detalhes.php?q=808, acesso em Maio de 2004.

[16] Eduardo Assis e Antonio R.L. Gomes. *Manual de uma VPN utilizando o protocolo PPTP em Linux*. http://www.guiadoharware.net/artigos/277, acesso em Junho de 2004.

[17] Brian Hatch, James Lee e George Kurts. *Hackers Linux Expostos*. MAKRON BOOKS, 2002.

[18] Jose Adriano. *Criptografia*. FATEC Americana http://www.criptografia.rg3.net, acesso em Junho de 2004.

[19] Steve Burnett e Stephen Paine. *Criptografia e Segurança - O guia oficial RSA*. CAMPUS, 2002.

[20] UFRG. *Tutorial PGP*. http://penta2.ufrgs.br/ rbmaz/tutorialpgp/index.html, acesso em Junho de 2004.

[21] Unicamp. http://www.dca.fee.unicamp.br/pgp, acesso em Julho de 2004.

[22] RFC 2401 - Security Architecture for Internet Protocol. http://www.faqs.org/rfcs/rfc2401.html, acesso em Setembro de 2004.

[23] RFC 2411 - IP Security Document Roadmap. http://www.faqs.org/rfcs/rfc2411.html, acesso em Setembro de 2004.

[24] RFC 2661 - Layer Two Tunneling Protocol “L2TP”. http://www.faqs.org/rfcs/rfc2661.html, acesso em Setembro de 2004.

[25] RFC 2637 - Point-to-Point Tunneling Protocol. http://www.faqs.org/rfcs/rfc2637.html, acesso em Setembro de 2004.

[26] RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile. http://www.ietf.org/rfc/rfc2459.txt?number=2459, acesso em Setembro de 2004.

[27] RFC 2759 - Microsoft PPP CHAP Extention, Version 2. http://www.faqs.org/rfcs/rfc2759.html, acesso em Setembro de 2004.

[28] RFC 2433 - Microsoft PPP CHAP Extention http://www.ietf.org/rfc/rfc2433.txt?number=2733, acesso em Setembro de 2004.

[29] RFC 2284 - PPP Extensible Authentication Protocol (EAP). http://www.faqs.org/rfcs/rfc2284.html, acesso em Setembro de 2004.

[30] RFC 3078 - Microsoft Point-To-Point Encryption (MPPE) Protocol. http://www.faqs.org/rfcs/rfc3078.html, acesso em Setembro de 2004.

[31] RFC 2784 - Generic Routing Encapsulation (GRE). http://www.faqs.org/rfcs/rfc2784.html, acesso em Setembro de 2004.

[32] RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP). http://www.faqs.org/rfcs/rfc2408.html, acesso em Setembro de 2004.

[33] RFC 2406 - IP Encapsulating Security Payload (ESP). http://www.faqs.org/rfcs/rfc2406.html, acesso em Setembro de 2004.

[34] RFC 2409 - The Internet Key Exchange (IKE). http://www.faqs.org/rfcs/rfc2409.html, acesso em Setembro de 2004.

[35] RFC 2402 - IP Authentication Header (AH). http://www.faqs.org/rfcs/rfc2402.html, acesso em Setembro de 2004.

[36] RFC 1334 - PPP Authentication Protocol. http://www.faqs.org/rfcs/rfc1334.html, acesso em Setembro de 2004.