Um modelo para proteção do tráfego de serviços baseado em níveis de segurança

Jansen Carlo Sena

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Um modelo para proteção do tráfego de serviços baseado em níveis de segurança

Jansen Carlo Sena

Maio de 2002

Banca Examinadora:

- Prof. Dr. Paulo Lício de Geus
  Instituto de Computação, UNICAMP (Orientador)
- Prof. Dr. Raul Fernando Weber
  Instituto de Informática, UFRGS
- Prof. Dr. Ricardo Dahab
  Instituto de Computação, UNICAMP
- Prof. Dr. Edmundo Madeira (Suplente)
  Instituto de Computação, UNICAMP

Um modelo para proteção do tráfego de serviços baseado em níveis de segurança

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Jansen Carlo Sena e aprovada pela Banca Examinadora.

Campinas, 16 de Maio de 2002.

Prof. Dr. Paulo Lício de Geus
Instituto de Computação, UNICAMP
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Para minha esposa, meus pais e meu irmão.

Tudo o que um homem pode imaginar outros homens poderão realizar.

Júlio Verne

É difícil dizer o que é impossível, pois a fantasia de ontem é a esperança de hoje e a realidade de amanhã.

Robert H. Goddard

Agradecimentos

A Deus, pela presença constante e por me fazer acreditar nesta conquista.

À minha esposa, Amanda, pela coragem de ter enfrentado comigo este desafio e, acima de tudo, pelo carinho, dedicação, companheirismo e amor.

À minha mãe, Áurea, amiga de sempre, pelas palavras de incentivo e por todo amor e carinho dedicados durante toda minha vida.

Ao meu pai, Odenildo, além de grande amigo e companheiro, meu ídolo maior na vida, pelo exemplo de caráter e honestidade e pelo apoio incondicional.

Ao meu irmão, Jean, mais que amigo, irmão fiel e companheiro, pela força e incentivo.

Ao meu orientador, Paulo Lício de Geus, pela oportunidade, aprendizagem e atenção.

À minha prima, Neylla, pela amizade, e a minha priminha, Rachel, que na inocência da infância, constantemente enche nossos corações da mais pura alegria.

Aos meus tios Antônio e Vera e a meus irmãos de coração Fernanda, Janaína, Jardel e Marcelo pelo apoio e carinho acolhedor de sempre e por fazerem-me sentir perto de casa.

Aos meus queridos amigos Helena e Silvano, pelas bagunças desde infância, pela força e apoio, pela amizade fraterna e verdadeira.

Aos meus tios Dores e Fernandes, pelo carinho que sempre dispensaram a mim.

A Amílcar e Ivanei, pela valiosa amizade conquistada em Campinas e pelos jogos do Corinthians às quarta-feiras.

Aos amigos do LAS, Alessandro, Cleymone, Diego, Edmar, Fabrício, Flávio, João e Marcelo, pela companhia diária, pelas discussões enriquecedoras, pela convivência nestes dois anos de trabalho e pela amizade que estabelecemos.

Aos professores do Instituto de Computação, em especial a Ricardo Dahab e Edmundo Madeira pelas dúvidas esclarecidas e pelo sugestões enriquecedoras.

A todos os colegas e funcionários do Instituto de Computação.

Ao CNPq pelo apoio financeiro.

Resumo

O IPSec foi especificado com a finalidade de prover, através de algoritmos criptográficos, os serviços de autenticação, integridade e confidencialidade ao conteúdo de pacotes IP. O nível de granularidade e os parâmetros de segurança específicos a serem utilizados em cada tipo de tráfego devem ser definidos pelo administrador. Contudo, a miscelânea de opções pode fazer com que a proteção aplicada não contemple os requisitos necessários e o uso deste protocolo fique restrito a ambientes pré-definidos como as redes privadas virtuais. Uma tentativa de driblar tal dificuldade é utilizar políticas de segurança genéricas onde um mesmo conjunto de parâmetros é aplicado a todo o tipo de tráfego. Tal solução não leva em consideração os diferentes graus de proteção entre as diversas aplicações de uma rede de computadores. Por outro lado, a definição e a manutenção de uma política de segurança para o IPSec baseada em cada serviço representa uma tarefa demasiadamente complexa para o administrador de sistemas.

Com base neste cenário, o presente trabalho apresenta o SLM (*Security Level Model*), um modelo que visa racionalizar o uso do IPSec através de níveis de segurança que encapsulam parâmetros com graus de proteção semelhantes e de descrições da política IPSec em uma linguagem de alto-nível, centralizadas em um servidor, permitindo ao administrador de sistemas abster-se dos detalhes de configuração do IPSec e, conseqüentemente, viabilizando seu uso como tecnologia para proteção adequada do tráfego dos serviços utilizados em sua rede.

Abstract

IPSec has been specified to provide, through the use of cryptographic algorithms, authentication, integrity and confidentiality services for the contents of IP packets. The granularity level and specific security parameters to be used for each kind of traffic must be defined by the administrator. However, the miscellany of options may result in the protection applied not addressing the necessary requirements and in the use of this protocol being restricted to predefined environments, such as virtual private networks. An attempt to circumvent this obstacle is to use a generic security policy where a single parameter set is applied to every kind of traffic. Such solution does not take into account the different levels of protection required by the diverse applications on a computer network. On the other hand, the definition and maintenance of an IPSec security policy based on each service poses a rather complex task for the administrator.

In the light of the above, this work presents the SLM (Security Level Model), a model that aims at rationalizing IPSec's use through security levels that encapsulate sets of parameters with similar protection abilities and through high-level descriptions centralized on a server. These allow the system administrator to not have to deal with details of IPSec configuration and, consequently, making its use viable as an adequate protection for the traffic of services being used on the network.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A enorme popularização da Internet nas últimas décadas trouxe consigo a possibilidade de explorar vulnerabilidades contidas em suas diversas estruturas básicas de funcionamento, dentre as quais o protocolo IP. Responsável pela entrega de pacotes da camada de rede, esse protocolo mostrou-se, com o passar dos anos, extremamente frágil diante de ataques não detectados na época do seu desenvolvimento.

A falsificação de endereços e dados, a ausência de serviços capazes de garantir o sigilo das informações e a possibilidade do reenvio de pacotes capturados são alguns dos problemas que afetam as aplicações que fazem uso do IP. Com base neste cenário, o IPSec foi especificado para incorporar, dentre outros serviços, autenticação, integridade e confidencialidade às informações contidas em pacotes IP. Contudo, a complexidade de configuração, decorrente da miscelânea de parâmetros necessários para a utilização do IPSec como tecnologia de proteção do tráfego de serviços em uma rede de computadores, tem limitado este protocolo a uma ferramenta para construção de ambientes cujos extremos da comunicação, na maioria dos casos, são *gateways* de redes virtuais privadas. Além disso, da mesma maneira que outros protocolos para comunicação segura, o IPSec provê seus serviços através de vários algoritmos criptográficos que nem sempre possuem graus de proteção semelhantes, permitindo que configurações menos criteriosas façam com que um mesmo tipo de transmissão seja protegido ora por algoritmos demasiadamente seguros (e possivelmente caros computacionalmente), ora por algoritmos suscetíveis a ataques de força-bruta, por exemplo.

De maneira geral, utilizar o IPSec para proteger adequadamente o tráfego dos serviços presentes em uma rede de computadores requer a definição de uma política de segurança que esteja de acordo com os requisitos de proteção de cada aplicação. Por outro lado, a criação e a manutenção dessas políticas podem se tornar uma tarefa extremamente problemática, custosa e sujeita a erros. Diante disso, neste trabalho é proposto o SLM (*Security Level Model*), cujo objetivo é facilitar o processo de construção e manutenção de

políticas de segurança para o IPSec, tornando-o viável para proteger o tráfego de serviços contra ataques comuns sob o protocolo IP.

1.1 Organização do trabalho

No Capítulo 2 são apresentadas outras tecnologias, alternativas ao IPSec, para a criação de canais seguros para a transmissão de informações e uma comparação entre as diversas soluções, no intuito de justificar a escolha do IPSec como base para o desenvolvimento deste trabalho. A seguir, no Capítulo 3, são descritas as características básicas do IPv6, versão do protocolo IP utilizada em conjunto com o IPSec para a implementação do modelo a ser proposto.

Dada a estreita relação do IPSec e deste trabalho com os sistemas criptográficos de chave secreta, no Capítulo 5 são apresentados seus aspectos fundamentais e alguns dos seus representantes utilizados, divididos entre algoritmos de confidencialidade e algoritmos de autenticação e integridade.

Com base nos conceitos explicitados nos capítulos anteriores, no Capítulo 6, o SLM, modelo desenvolvido como resultado deste projeto, é detalhado elucidando-se seus objetivos, estruturas, componentes e mecanismos de funcionamento. Em seguida, no Capítulo 7, são feitas considerações que dizem respeito a características e decisões de projeto referentes à implementação do SLM. No Capítulo 8, são feitas considerações finais sobre o trabalho desenvolvido e possíveis extensões identificadas para o SLM.

No Apêndice A estão listadas as estruturas e configurações do LDAP, protocolo utilizado para armazenar as bases de dados do SLM, e no Apêndice B estão contidos os detalhes relativos ao SLC, um dos componentes do SLM, incluindo manual de utilização, resultados produzidos e código-fonte.

Capítulo 2

Protocolos para o estabelecimento de canais seguros

Os esforços dispendidos no desenvolvimento de tecnologias para adicionar segurança à comunicação em redes de computadores não foram sincronizados muito menos centralizados em uma única organização. Além do IPSec, a carência de mecanismos capazes de proteger o tráfego de aplicações culminou na especificação de muitos padrões, públicos e privados, que visam impedir o sucesso de ataques desenvolvidos ao longo dos anos.

Algumas soluções são versões com características de segurança de padrões existentes, como é o caso do HTTPS. Outras, são espeficicações completas de protocolos a serem utilizados em conjunto com aqueles desprovidos de serviços de segurança, como o SSL, TLS e o próprio IPSec. Além disso, em determinadas situações, é possível combinar soluções no intuito de obter o conjunto de serviços desejados. De maneira geral, cada um dos protocolos de segurança foi desenvolvido objetivando a proteção de aspectos específicos da comunicação em rede, e a escolha da solução adequada para cada cenário constitui-se uma decisão fundamental para o ambiente e/ou aplicação que se deseja proteger.

Para justificar a escolha do IPSec para o desenvolvimento deste trabalho, em detrimento de outras soluções possíveis, neste capítulo são apresentadas as características básicas e uma breve análise de segurança de alguns protocolos que, como o IPSec, são capazes de estabelecer canais seguros. Em seguida, são feitas comparações acerca das vantagens e desvantagens de cada solução em relação ao IPSec.

2.1 SSL e TLS

Desenvolvido em 1993 pela *Netscape Communications Corporation*, o SSL (*Secure Socket Layer*) [20] tem como principal objetivo adicionar segurança às mensagens de protocolos de transporte orientados à conexão, como o TCP. Entre os serviços oferecidos estão: au-

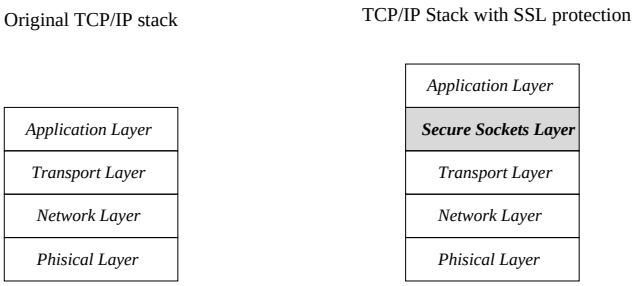


Figura 2.1: Posicionamento da camada de proteção provida pelo SSL na pilha TCP/IP.

tenticação do servidor e/ou do cliente, integridade, confidencialidade e proteção contra o reenvio de mensagens. Atualmente, o SSL é considerado um padrão *de facto* para o desenvolvimento de aplicações seguras na Internet, como as soluções de comércio eletrônico, por exemplo [75]. Além disso, diversos aplicativos anteriores a este protocolo o têm incorporado como tecnologia para prover segurança à comunicação.

A partir da arquitetura TCP/IP [69, 68], o SSL integra-se como uma camada adicional localizada entre as camadas de transporte e aplicação, conforme mostrado na Figura 2.1. Sua estrutura é baseada em duas sub-camadas: a de registro e a do protocolo de *handshake*. A primeira é responsável pelas operações criptográficas de cifragem e decifragem, cálculo das MACs e verificação de integridade e autenticidade das mensagens. Os dados gerados por uma aplicação protegida pelo SSL são primeiramente repassados à camada de registro, para que sejam adicionados os serviços de segurança, e, em seguida, são encaminhados às camadas inferiores. Após a transmissão dos dados e o seu processamento pelas camadas de rede e transporte, a camada de registro valida o conteúdo e, finalmente, repassa a mensagem para a aplicação. Caso a checagem de integridade falhe, a conexão é encerrada.

O protocolo de *handshake*, por outro lado, deve ser utilizado antes do estabelecimento da comunicação segura de dados. Sua função principal é sincronizar e negociar um conjunto de parâmetros necessários à proteção da comunicação entre duas entidades, entre eles: os algoritmos e as chaves criptográficas, a versão do SSL a ser utilizada, entre outros. Além disso, é responsabilidade deste protocolo autenticar o cliente e o servidor. A autenticação de ambos os extremos é opcional. Desta forma, é possível que um cliente mantenha-se anônimo durante uma conexão. Em contrapartida, quando pelo menos um dos dois *hosts* permanece anônimo, o processo de negociação de parâmetros torna-se vulnerável a ataques de *man-in-the-middle* [56, 43].

As chaves criptográficas e os números de seqüência utilizados no serviço de proteção contra o reenvio de mensagens são criados separadamente para cada sentido da comunicação. Os últimos são gerados novamente a cada troca de chaves, impedindo sua pre-

visão a partir da obtenção do valor utilizado por uma conexão anterior.

Da mesma maneira que o IPSec, o SSL não é restrito a um conjunto fixo de algoritmos criptográficos. Porém, não é possível valer-se de todas as combinações possíveis. A definição do SSL disponibiliza grupos de algoritmos que podem ser utilizados em conjunto, denominados *ciphersuites*. As *ciphersuites* suportadas podem ser encontradas em [20].

A possibilidade de utilizar grupos distintos de algoritmos pode permitir que aplicações selecionem o grupo que mais se adequa aos seus requisitos de segurança, de acordo com a relação "proteção x desempenho". Além disso, aplicações que utilizam grupos que contêm algoritmos com falhas de segurança recém-detectadas podem, rapidamente, passar a utilizar outro grupo menos vulnerável.

2.1.1 Análise de segurança do SSL 2.0

Primeira versão amplamente utilizada do SSL, o SSL 2.0 após diversas análises de segurança, passou a ser considerado uma especificação insegura do protocolo. Vulnerabilidades detectadas em diversas estruturas e procedimentos tornaram-no inadequado para a proteção do tráfego de aplicações.

Durante o processo de negociação realizado pelo protocolo de *handshake*, o SSL 2.0, permite, por exemplo, que um atacante posicionado entre duas entidades que estão estabelecendo parâmetros de proteção altere os grupos de algoritmos propostos. Desta forma, é possível fazer com que seja escolhido o grupo que contém os algoritmos menos seguros, fragilizando a proteção da comunicação entre as duas entidades. Este procedimento é conhecido como *cipher suite rollback attack* [71].

Outra grave vulnerabilidade do SSL 2.0 apresenta-se quando este protocolo é utilizado nos modos de exportação contendo somente os algoritmos permitidos pelo governo norte-americano. Nestas circunstâncias, as chaves dos algoritmos de autenticação são reduzidas a 40 *bits*. Chaves maiores poderiam ser utilizadas sem qualquer risco de infração às leis norte-americanas. Em outras palavras, o próprio protocolo reduz, sem qualquer necessidade, a segurança das informações sob sua proteção.

Fragilidades relativas aos procedimentos de utilização dos algoritmos criptográficos também foram detectadas no SSL 2.0. Por exemplo, o campo que armazena os *bytes* inseridos para o alinhamento das MACs geradas não é autenticado. Conseqüentemente, um atacante pode, após capturar a mensagem, alterar o valor deste campo no intuito de fazer com que o destino desconsidere alguns *bytes* válidos do fim da mensagem.

De maneira geral, sob o aspecto da segurança de sistemas, o uso do SSL 2.0 não é recomendado para a proteção dos dados de aplicações [75].

2.1.2 Análise de segurança do SSL 3.0

Os diversos problemas de segurança detectados no SSL 2.0 fizeram com que a *Netscape* especificasse uma nova versão para o protocolo SSL, incluindo correções das fragilidades encontradas na versão anterior bem como novas características de segurança, resultando no SSL 3.0 [20].

Em relação ao *ciphersuite rollback attack*, o SSL 3.0, por exemplo, autentica todas as mensagens geradas pelo protocolo de *handshake*, evitando que um atacante possa forjar informações no intuito de fazer com que a proteção seja aplicada com base num grupo de algoritmos vulneráveis a ataques.

Quando utilizado no modo de exportação, as chaves dos algoritmos de autenticação utilizados para a criação dos MACs possuem pelo menos 128 *bits* ao invés dos 40 *bits* utilizados pelo SSL 2.0. Chaves simétricas iguais ou maiores de 128 *bits* são consideradas seguras contra ataques de força-bruta desempenhados pela maioria das organizações [75, 15].

A abrangência dos campos protegidos pelo serviço de autenticação também foi revista no SSL 3.0, incluindo, por exemplo, o campo que contém o número de *bytes* de *padding* inseridos para o correto funcionamento dos algoritmos que geram os MACs. Como conseqüência, esta versão do SSL não permite que um atacante altere o número de *bytes* validados pelo destino, ao contrário do que pode ocorrer no SSL 2.0.

Apesar de resolver graves problemas de segurança do SSL 2.0, análises de segurança têm mostrado que o SSL 3.0 também possui vulnerabilidades que podem ser exploradas para prejudicar os seus serviços [71]. Dentre as fragilidades detectadas, quatro merecem maior destaque:

- Análise de tráfego do tamanho das requisições GET: o comprimento do texto cifrado contendo o GET de uma conexão de um *Web browser* protegida pelo SSL 3.0, aliado a observações do tráfego que podem detectar o servidor Web acessado e o tamanho dos dados retornados como resposta a solitação de GET, pode ser informação suficiente para um atacante descobrir a página Web acessada, possivelmente contendo informações sigilosas. Para isto, é suficiente inundar o servidor Web com requisições do mesmo tamanho daquela feita pelo *Web browser* e verificar as respostas que contêm tamanhos correspondentes à obtida por ele. Este ataque é possível pelo fato do tamanho da mensagem cifrada corresponder ao tamanho da mensagem original. A inserção de um número aletório de *bytes* para evitar a correspondência do comprimento entre as duas versões de uma mensagem é provida pelo SSL 3.0 somente para os algoritmos de cifragem em cadeia;

- *Change cipher spec-dropping*: realizada a troca inicial de informações do protocolo de *handshake*, onde todos os dados são enviados sem qualquer proteção crip-

tográfica, cada extremo da comunicação deve enviar uma mensagem denominada *change cipher spec* para confirmar os parâmetros a serem utilizados na proteção da comunicação. Caso os grupos de algoritmos propostos provenham somente autenticação, um atacante posicionado entre os dois extremos pode capturar e deletar as mensagens de *change cipher spec*, fazendo com que a comunicação prossiga sem qualquer proteção de autenticação e integridade das informações transmitidas;

- Falsificação do campo *KeyExchangeAlgorithm*: um servidor deve sinalizar ao cliente o algoritmo a ser utilizado para a troca de chaves bem como os parâmetros públicos específicos do algoritmo escolhido. Apesar do campo que inclui os parâmetros ter sua integridade garantida, o campo *KeyExhangeAlgorithm* que identifica o algoritmo não é protegido. Sendo assim, é possível que um atacante faça com que um cliente processe parâmetros advindos do servidor para o uso com o algoritmo Diffie-Hellman como sendo parâmetros do RSA. Este ataque pode ser realizado através da captura da mensagem *server key exchange*, que contém o campo *KeyExchangeAlgorithm*. Após alterar o valor deste campo, trocando a identificação do algoritmo, basta reenviar a mensagem para o cliente. Como conseqüência do processamento incorreto efetuado pelo cliente, é possível que o atacante obtenha acesso ao *pre_master_secret*, valor utilizado para a geração das chaves criptográficas a serem utilizadas durante a comunicação segura, habilitando-o a decifrar e falsificar qualquer informação entre o cliente e o servidor. Apesar de algumas implementações mais robustas não serem vulneráveis a este ataque, o fato da especificação do SSL 3.0 não previnir nada neste sentido, é possível que uma parcela considerável das implementações seja suscetível a esta vulnerabilidade;

- *Version rollback attack*: por questões de compatibilidade, servidores SSL 3.0 possuem suporte ao SSL 2.0. A mensagem *hello* enviada por um cliente a um servidor indica a versão do SSL suportada e é utilizada pelo servidor para discernir entre qual versão do protocolo utilizar em uma dada conexão. Porém, dado que o campo de controle de versão não é protegido pelo serviço de integridade, um atacante pode, ao capturar a mensagem *hello* de um cliente que está tentando negociar uma conexão com o SSL 3.0, alterar a versão no intuito de provocar que a conexão seja estabelecida com o SSL 2.0, expondo a comunicação a todas as vulnerabilidades já conhecidas desta versão do SSL.

Apesar do resultado de análises de segurança do SSL 3.0 evidenciar algumas vulnerabilidades neste protocolo, ele é considerado, atualmente, uma alternativa viável para o estabelecimento de conexões seguras entre aplicações voltadas para a Internet [71, 75].

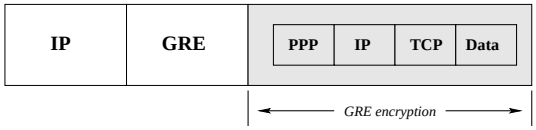


Figura 2.2: Encapsulamento de um pacote IP feito pelo PPTP.

2.1.3 *Transport Layer Security*

A grande popularização do SSL fez com que o IETF desenvolvesse, mediante pequenas modificações sobre o SSL 3.0, um protocolo para a proteção de conexões entre aplicações na Internet denominado TLS (*Transport Layer Security*) [12]. Além de representar uma alternativa para a comunicação segura, outra razão que impulsionou o projeto do TLS foi a possibilidade de estabelecer um padrão não-proprietário, diferentemente do SSL, que é patenteado pela *Netscape* [8].

As principais diferenças entre o SSL 3.0 e o TLS são de caráter técnico, envolvendo, por exemplo, diferenças nas funções de geração de números pseudo-aleatórios nos mecanismos de composição dos MACs e no processo de inserção de *bytes* de *padding*. Apesar das poucas diferenças entre os dois protocolos, o TLS 1.0 não é capaz de interagir diretamente com o SSL 3.0. Por outro lado, a especificação do TLS 1.0 descreve um mecanismo pelo qual é possível que uma entidade com TLS 1.0 possa estabelecer uma comunicação segura com outra que utiliza o SSL 3.0.

2.2 PPTP

O PPTP (*Point-to-Point Tunneling Protocol*) [24], cuja principal finalidade é prover um mecanismo para o tunelamento seguro de tráfego IP, foi desenvolvido com base nos protocolos PPP (*Point-to-Point Protocol*) [64] e GRE (*Generic Routing Encapsulation*) [16, 25, 26].

Sempre utilizado sobre o IP, o PPTP exige que haja conectividade IP entre os *hosts* que possuem algum tráfego protegido por seus serviços. Antes do envio de um pacote IP, o PPTP encapsula este em um pacote PPP que, por sua vez, é cifrado e encapsulado em um pacote GRE. A Figura 2.2 mostra a cadeia de encapsulamento feita pelo PPTP.

Da mesma forma que outros protocolos de segurança, como o IPSec e o SSL, o PPTP também requer a negociação de parâmetros antes de, efetivamente, proteger um tipo de tráfego entre duas entidades. Porém, o seu procedimento de negociação é feito sem qualquer proteção, permitindo que um atacante obtenha dados como o endereço IP dos extremos do túnel, nome e versão do *software* utilizado, nome do usuário e, em alguns

casos, o *hash* criptográfico da senha do usuário [75].

Outra vulnerabilidade do PPTP é o fato do cliente só precisar se autenticar após a conclusão do processo de estabelecimento de parâmetros. Esta característica permite que atacantes façam com que um servidor inicie diversos processos de negociação falsos, o que pode resultar em negação de serviço (DoS) e até mesmo na total paralisação do servidor.

2.2.1 Considerações sobre a implementação do PPTP pela *Microsoft*

Implementações espefícias de um protocolo podem conter, além dos problemas de segurança detectados na própria especificação, falhas que podem comprometer o uso deste protocolo.

Em relação ao PPTP, a *Microsoft* possui uma implementação com extensões proprietárias incluída na maioria das versões do *Microsoft Windows*. Pelo fato deste sistema operacional ser amplamente utilizado e o seu suporte ao PPTP representar uma solução viável e de baixo custo para a configuração de VPNs, análises de segurança desta implementação do PPTP merecem atenção especial.

Uma primeira fragilidade do PPTP da *Microsoft* está em um dos formatos de armazenamento e transmissão de *hashes* de senhas nas várias versões do *Microsoft Windows*, conhecido como *LanMan*. Senhas do *Windows NT* possuem 14 caracteres de extensão. Porém, quando armazenadas no formato *LanMan*, que é *case-insensitive*, todos os caracteres são convertidos para *uppercase*, diminuindo o número de *hashes* possíveis e, conseqüentemente, facilitando ataques de força-bruta. Porém, a maior vulnerabilidade do *LanMan* é a divisão da cadeia de 14 caracteres em duas de 7 caracteres. *Hashes* para cada uma das cadeias são gerados separadamente. Este procedimento reduz o ataque de força-bruta da senha ao esforço necessário para descobrir colisões para duas senhas curtas de 7 caracteres. Caso não ocorresse a divisão da cadeia original, as senhas seriam mais seguras, dado que *hashes* de cadeias de 14 caracteres possuem, aproxidamente, 6 trilhões de possibilidades a mais em relação a um *hash* gerado a partir de uma cadeia de 7 caracteres.

Uma vez que os dados do processo de negociação de parâmetros do PPTP são transmitidos sem a proteção de qualquer serviço de confidencialidade, um atacante pode obter o *hash* da senha de um usuário armazenada no formato *LanMan* e, com base neste dado, descobrir a senha original. Deve-se ainda levar em consideração o fato de que muitos usuários escolhem senhas previsíveis e sujeitas a ataques de dicionário [36, 57, 21], o que certamente facilita a violação de senhas representadas e transmitidas em *LanMan*.

Apesar do *Windows NT* possuir um novo formato para a manipulação de senhas[1], caso

[1]O formato de geração de *hashes* de senhas no *Windows NT* também possui fragilidades, entre elas o

a compatibilidade com o *LanMan* esteja ativa, as senhas serão manipuladas utilizando este formato. Desta forma, o uso do protocolo PPTP certamente representa uma exposição perigosa da senha dos usuários de um ambiente *Windows*. Vale ressaltar que o ataque a *hashes* de 7 caracteres pode ser realizado através da utilização de computadores pessoais de baixo custo [15, 56, 44].

Outra vulnerabilidade grave da implementação do PPTP em sistemas *Windows* está no tamanho e no processo de geração de chaves criptográficas para o serviço de cifragem. Dois modos de confidencialidade são oferecidos através do algoritmo RC4 [75, 56]: um que utiliza chaves de 40 *bits* e outro com chaves de 128 *bits*. No primeiro caso, além da utilização de chaves pequenas, altamente suscetíveis a ataques força-bruta, as chaves geradas são baseadas nas senhas dos usuários. Em outras palavras, várias sessões de um mesmo usuário irão utilizar a mesma chave, a não ser que este usuário altere o valor de sua senha. Esta fato agrava-se ainda mais se o atacante conseguir a senha de um usuário através da obtenção da sua versão no formato *LanMan*.

No segundo modo de cifragem, que utiliza chaves de 128 *bits*, consideradas atualmente seguras, o valor gerado para uma chave é baseado, novamente, na senha do usuário, porém combinada com um número aleatório específico para cada sessão. Apesar deste procedimento ser mais seguro que o anterior, o uso constante da senha do usuário diminui consideravelmente o número de tentativas que podem compor um ataque.

O uso de ferramentas como o *L0phtcrack*, que automatiza o ataque a senhas, pode obter sucesso em menos de um minuto, considerando o uso de senhas comuns, fragilizando ainda mais o processo de geração de chaves criptográficas baseadas nas senhas dos usuários. No caso de um ataque de força bruta, o *L0phtcrack* pode obter sucesso num período entre 26 e 250 horas, caso a senha seja composta somente por caracteres alfabéticos.

O conjunto de vulnerabilidades do PPTP implementado em sistemas *Windows* fez com que a própria *Microsoft* recomendasse a desabilitação do formato *LanMan* em cenários onde é possível o uso de outras opções [10]. Os resultados de uma análise de segurança detalhada sobre a implementação *Microsoft* do protocolo PPTP pode ser encontrada em [58].

2.3 L2TP

Da mesma forma que o PPTP, o L2TP (*Layer 2 Tunneling Protocol*) [70] é um protocolo baseado no PPP [64], cujo objetivo é criar túneis através do encapsulamento de pacotes IP.

Uma das diferenças entre o L2TP e o PPTP está no protocolo utilizado na camada inferior. Enquanto o PPTP deve ser sempre utilizado acima do IP, o L2TP pode utilizar

uso do algoritmo MD4.

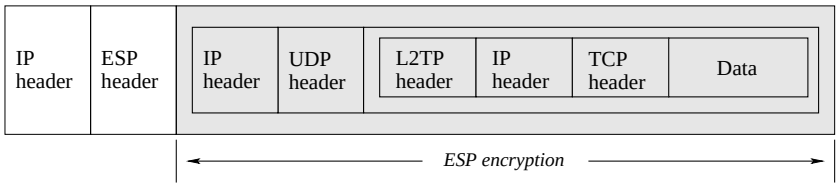


Figura 2.3: Encapsulamento de um pacote IP feito pelo L2TP sob a proteção do cabeçalho ESP do IPSec.

um conjunto de outros protocols inferiores, como o PPP, o IP e o *Frame Relay* [51]. Sob o ponto de vista da segurança da comunicação, o L2TP, diferentemente do PPTP, não possui serviços de cifragem e integridade de dados. Porém as informações iniciais relativas ao processo de autenticação dos dois extremos do túnel são protegidas, enquanto que no PPTP os parâmetros podem ser livremente obtidos.

Dado que o L2TP não foi projetado para a configuração de ambientes seguros, seu uso em cenários onde existe uma rede não-confiável, como a Internet, entre os extremos de um túnel, deve sempre ser combinado com outros protocolos capazes de suprir a sua ausência de serviços de segurança. Um conjunto de propostas tem sido desenvolvido para conciliar o uso do L2TP com o IPSec [66, 49]. Quando executado sobre o IP, o L2TP é transportado através do UDP. Desta forma, a aplicação da proteção do IPSec sobre o L2TP pode basear-se simplesmente no uso de seletores que filtram o tráfego L2TP.

É importante notar que a combinação do tunelamento do IPSec com o L2TP resulta na criação de dois túneis, um para cada protocolo. A Figura 2.3 exibe o encapsulamento de um pacote TCP feito pelo L2TP sendo utilizado sobre o IP protegido pelo ESP. Um problema na integração do L2TP com o IPSec é a impossibilidade do segundo levar em consideração os valores dos campos de pacotes IP encapsulados pelo primeiro.

Outros procedimentos de interação entre os dois protocolos têm sido sugeridos no intuito de prover o desenvolvimento de soluções para aspectos ainda não padronizados do IPSec. Uma das propostas sugere, por exemplo, o L2TP como protocolo para a troca de informações relativas às políticas de segurança entre um *host* e um *gateway* IPSec. Apesar destas soluções serem práticas e de baixo custo, pelo fato do protocolo L2TP já ser um padrão definido, existem críticas severas quanto ao uso de um protocolo que não foi projetado para ambientes seguros na execução de procedimentos vitais para um protocolo de segurança como o IPSec [19].

2.4 SSH

O SSH (*Secure Shell*) [75] foi desenvolvido pela *SSH Communications Security Ltd.* com o intuito de substituir os comandos de acesso remoto originais dos sistemas Unix BSD por versões seguras, previnindo a captura de informações como senhas de usuários e o *hijacking* de conexões. Atualmente, duas especificações do SSH são utilizadas: o SSH 1 e o SSH 2. Ambas as versões suportam o uso do SSL. O SSH 2, adicionalmente, tem suporte ao TLS.

O estabelecimento de conexões seguras é precedido, da mesma forma que no IPSec e no SSL, pela negociação de algoritmos criptográficos. No SSH 2, a escolha dos métodos para a troca de chaves (capazes de evitar ataques de *man-in-the-middle*) e para geração dos MACs fazem parte do processo de negociação de parâmetros. A autenticação do servidor é sempre realizada primeiro na tentativa de evitar que um cliente seja vítima de um servidor forjado por um atacante com o objetivo de capturar os dados de autenticação do cliente. Se esta etapa for concluída com sucesso, a autenticação do cliente é efetuada através de um dos diversos mecanismos suportados pelo SSH. O mais comum é baseado no uso de senhas.

Aproveitando os esforços que vêm sendo dispendidos no desenvolvimento de soluções para a distribuição e verificação de chaves públicas, o SSH 2 suporta a consulta a autoridades certificadoras para a obtenção e validação de chaves públicas. Além disso, o mecanismo original, desenvolvido no SSH 1, onde o próprio servidor envia sua chave pública, caso o cliente ainda não a tenha, também é suportado pelo SSH 2. É importante notar que este mecanismo expõe o cliente a servidores forjados por atacantes na ocasião do primeiro contato entre estes dois *hosts*, onde, certamente, as chaves públicas de ambos não são conhecidas. Por outro lado, caso a chave pública do servidor seja recebida com sucesso, conexões futuras não estarão sujeitas a este tipo de ataque, o que representa um nível de segurança maior do que àquele provido por protocolos como *telnet*, onde cada estabelecimento de conexão é suscetível a ataques como o anterior. Uma das maneiras possíveis de evitar este problema no SSH é obter, de maneira segura, as chaves públicas de servidores onde serão estabelecidas conexões SSH e mantê-las em base de dados locais, evitando que os servidores as enviem de maneira insegura e não-confiável durante o primeiro contato.

2.5 HTTPS e *Secure* HTTP

O aumento do uso de aplicações baseadas na Web e as suas necessidades de transmissão de informações sigilosas culminaram no desenvolvimento de tecnologias capazes de adicionar segurança às mensagens transportadas pelo protocolo HTTP [18], especificado

inicialmente sem quaisquer serviços de autenticação, integridade, confidencialidade, entre outros. As duas principais soluções para tal finalidade é o HTTPS [52] e o *Secure* HTTP [53].

O objetivo principal do HTTPS é proteger o tráfego HTTP entre um cliente e um servidor através dos serviços de segurança providos pelo SSL. Recentemente, o IETF padronizou o HTTPS para ser utilizado também com o TLS [52]. O desempenho satisfatório do SSL e do TLS em relação a proteção de conexões de curta duração e a possibilidade de recuperar contextos de segurança entre *hosts* que se comunicaram recentemente tornaram o uso destes protocolos fundamental para o desenvolvimento do HTTPS. Quando uma determinada sessão está protegida pelo HTTPS, os comandos e dados do HTTP são incluídos na porção de dados do SSL/TLS. Outra solução próxima ao HTTPS foi especificada para proteger o HTTP através do TLS a partir de conexões TCP já estabelecidas [35]. Este procedimento possibilita o uso do mesmo protocolo para o estabelecimento de comunicação comum ou segura.

Desenvolvido pela *Enterprise Integration Technologies*, o *Secure* HTTP, diferentemente do HTTPS, cujo foco de proteção é o canal de comunicação entre um cliente e um servidor, visa proteger objetos específicos. Em outras palavras, páginas Web recebidas por um cliente contêm assinaturas digitais associadas que impedem seu processamento, caso o mecanismo de verificação da assinatura não obtenha sucesso. Formulários enviados por um cliente a um servidor também são protegidos por assinaturas criptográficas. Além dos serviços de autenticação, integridade e confidencialidade, informações enviadas do cliente ao servidor, ou vice-versa, através do *Secure* HTTP, são protegidas pelo serviço de não-repúdio, onde todo objeto é associado a uma única entidade, a menos que sua chave privada seja comprometida.

Dada a diferença de enfoque entre o HTTPS e o *Secure* HTTP, é possível utilizar ambos os protocolos no estabelecimento de conexões seguras para aplicações Web.

2.6 Comparações com o IPSec

Apesar de preservarem semelhanças entre si, como o aumento da quantidade de tráfego transmitido em uma rede e a necessidade de processamento extra em decorrência das operações criptográficas envolvidas, os protocolos de segurança possuem vantagens e desvantagens específicas resultantes da sua estrutura e especificação.

Assumindo o fato de que um modelo que vise proteger o tráfego de aplicações quaisquer em redes de computadores, como é o caso da proposta deste trabalho, deve utilizar soluções que não sejam limitadas a protocolos de alto-nível, utilizar o SSH, o HTTPS ou o *Secure* HTTP é uma decisão inadequada. Estas soluções são voltadas para aplicações específicas e, portanto, são incapazes de proteger o tráfego de outros serviços. Por outro lado, o uso

destes protocolos de segurança de alto-nível deverá continuar sendo útil, mesmo diante da adoção de outras tecnologias mais abrangentes. Determinadas aplicações possuem requisitos de segurança que, muitas vezes, não podem ser contemplados adequadamente somente através da proteção de outros protocolos mais genéricos, como o próprio IPSec.

Protocolos mais abrangentes, como o L2TP e o PPTP, capazes de encapsular dados dos protocolos da camada de rede, transporte e aplicação, também possuem limitações no que diz respeito as suas adoções em modelos como o citado anteriormente. O primeiro pelo fato de não ter sido desenvolvido para vislumbrar características de segurança, não possui, sozinho, serviços suficientes para proteger o tráfego de aplicações. Apesar de garantir o sigilo das informações utilizadas no procedimento de autenticação, a ausência de confidencialidade representa um grave empecilho para sua adoção. Indispensável para preservar informações sensitivas, este serviço é de fundamental importância para qualquer protocolo de segurança.

Adicionar os serviços do IPSec ao L2TP representa uma alternativa para o uso desse protocolo em ambientes seguros. Porém, como nesta composição os mecanismos básicos de proteção são oriundos do IPSec, o L2TP ou a sua combinação com o IPSec que, por sua vez, acrescenta uma camada adicional de processamento, devem ser preteridos em relação ao IPSec sozinho. A solução composta poderia ser utilizada, caso algum tipo de benefício pudesse suprir o custo do encapsulamento feito pelo L2TP. Porém, dado que o próprio IPSec possui um modo de tunelamento (Seção 4.1), a combinação dos dois protocolos restringe-se a ambientes onde o uso dos serviços do L2TP são estritamente necessários.

As limitações em relação à adoção do PPTP concentram-se, em sua maioria, em problemas relacionados a sua estrutura. O fato do processo de negociação de parâmetros ser suscetível a *sniffer* de dados e a possibilidade de clientes forjados provocarem negação de serviço em servidores torna a escolha deste protocolo inadequada, dada a gravidade de suas vulnerabilidades. Os problemas relativos a implementações como a da *Microsoft*, descritos anteriormente, não foram levados em conta no processo de avaliação deste protocolo como possível solução para a construção do modelo proposto, devido ao fato de tratar-se de instâncias específicas do protocolo cujas vulnerabilidades são passíveis de correções. Em contrapartida, os diversos ambientes que utilizam implementações problemáticas do PPTP tornam-se vulneráveis às fragilidades do próprio protocolo e àquelas inerentes a da plataforma utilizada.

Da mesma forma que o L2TP, a análise do PPTP como solução para o modelo a ser apresentado neste trabalho ainda deve levar em consideração o custo do encapsulmento feito por esse protocolo. Em determinadas situações onde não existe pelo menos um *gateway* entre dois *hosts* que pretendem comunicar-se seguramente, o encapsulamento pode ser desnecessário. Com o IPSec, para estas situações, o modo de transporte é mais adequado e, computacionalmente, menos custoso.

SSL e TLS	
Vantagens	**Desvantagens**
• Mais flexível para prover proteção em nível de usuário em relação ao IPSec • Amplamente utilizado no desenvolvimento de aplicações voltadas para a Web • Pode incorporar novos algoritmos criptográficos • É capaz de otimizar o processo de estabelecimento de conexões seguras, diminuindo a quantidade de tráfego na rede, através de um mecanismo opcional de *cache*	• Aplicações precisam ser modificadas para usufruir dos serviços • Repassa muita responsabilidade ao analista e/ou programador que deve estabelecer o momento para a efetivação da comunicação segura • Protege somente aplicações baseadas no TCP • Não suporta o uso de qualquer combinação de algoritmos criptográficos • Permite que ambos os extremos permanecem anônimos durante a conexão

Tabela 2.1: Principais vantagens e desvantagens do SSL e TLS.

Apesar de seus serviços de proteção abrangerem um conjunto distinto de camadas da arquitetura TCP/IP, o IPSec e o SSL[2], por serem independentes de protocolos de aplicação específicos e pelo fato de estarem mais integrados à arquitetura TCP/IP em relação ao L2TP e ao PPTP, representam as soluções mais factíveis ao modelo, dentre aquelas apresentadas. O primeiro protege as principais informações do protocolo IP, e todos os protocolos superiores, da camada de transporte e aplicação. O segundo, por sua vez, protege somente os protocolos de aplicação baseados no TCP.

As principais vantagens e desvantagens do IPSec e do SSL estão resumidas nas Tabelas 2.1 e 2.2, respectivamente. Entre as vantagens do SSL, somente a primeira representa uma ganho direto deste protocolo em relação ao IPSec. Os dois tópicos seguintes possuem contrapesos que os limitam significativamente. Se, por um lado, o SSL é a solução mais utilizada em aplicações Web seguras, a adição de seus serviços requer a alteração no código de todas as implementações de uma determinada aplicação. Além disso, novas aplicações baseadas no SSL requerem que o analista e/ou programador sejam extremamente cuidadosos para que nenhuma informação sensitiva que possa viabilizar ataques

[2]Neste momento, as considerações feitas em relação ao SSL referem-se ao SSL 3.0 e ao TLS.

seja transmitida antes do estabelecimento da conexão segura. Vale ressaltar que falhas no projeto de aplicações podem provocar uma falsa sensação de segurança aos usuários que, por confiarem que uma determinada aplicação é protegida pelo SSL, não temem em utilizá-la, mesmo correndo o risco de ter informações sigilosas sendo transmitidas sem qualquer proteção.

IPSec	
Vantagens	**Desvantagens**
• Proteção é aplicada a qualquer protocolo superior de transporte e aplicação • Aplicações não necessitam ser modificadas para ter seus pacotes protegidos • Configuração flexível, permitindo a criação de diversos tipos de cenários de segurança • Permite a utilização de diversos algoritmos criptográficos e qualquer combinação • Nativo no IPv6	• Descrição complexa e requer conhecimento profundo da estrutura do protocolo, sob pena de comprometer a comunicação e a segurança das informações • Dependente de muitos mecanismos como o protocolo IKE e outros componentes que vêm sendo desenvolvidos • Aumenta o tamanho dos pacotes IP

Tabela 2.2: Principais vantagens e desvantagens do IPSec.

Em relação à possibilidade de utilizar diversos algoritmos criptográficos, o que poderia representar uma importante característica para o desenvolvimento do modelo que será apresentado neste trabalho pelo fato de tornar possível a utilização de níveis de proteção distintos, a limitação do SSL no uso de grupos de algoritmos pré-determinados reduz o número de combinações possíveis e, conseqüentemente, a flexibilidade de interoperação entre plataformas com suporte a conjuntos distintos de algoritmos criptográficos.

O IPSec não possui as limitações descritas anteriormente no SSL. Primeiramente, o tráfego de qualquer aplicação pode ser protegido pelo IPSec sem qualquer alteração em suas implementações. Tal proteção não onera responsabilidade ao analista e/ou ao programador, que podem focar seus esforços somente no projeto das funcionalidades principais de aplicações. Além disso, os algoritmos suportados por uma plataforma podem ser livremente combinados entre si.

O fato do IPSec ser um componente nativo ao IPv6 (Capítulo 3), próxima geração do protocolo IP a ser utilizada na Internet, torna-o um protocolo de segurança obrigatório que deverá estar disponível em todas as plataformas com suporte ao IPv6, o que, certamente, será uma característica muito importante para a popularização do IPSec. Mesmo diante das desvantagens identificadas, o IPSec foi a melhor solução encontrada dentre as possíveis para o desenvolvimento deste trabalho. A abrangência de seus serviços faz deste protocolo uma tecnologia capaz de prover segurança a qualquer tipo de tráfego em uma rede de computadores. Uma de suas desvantagens, a complexidade de seu uso, é, exatamente, um dos problemas que este trabalho procura resolver.

Capítulo 3

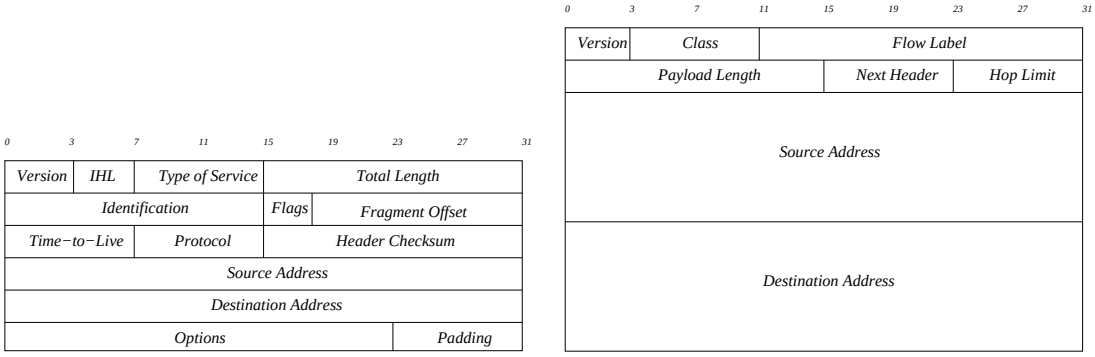
IPv6

O IPv4, versão atual do protocolo IP utilizada na Internet, não contempla as atuais necessidades da rede mundial. O problema mais visível é a iminente exaustão do seu espaço de endereçamento de 32 *bits*. O aumento exponencial no uso da Internet fez com que a alocação de endereços fosse muito além do imaginado há aproximadamente 30 anos pelos seus projetistas. Na tentativa de racionalizá-los, foram desenvolvidas técnicas como o NAT (*Network Address Translation*), que possibilita o compartilhamento de endereços válidos, e o CIDR (*Classless Inter-Domain Routing*), que visa um melhor aproveitamento dos endereços da classe B e a diminuição das tabelas de roteamento, outro grave problema que passou a atingir o IPv4.

Aliados a estes fatores, a ausência de mecanismos de segurança mais robustos, de suporte a aplicações multimídia, e a possibilidade de eliminar problemas e acrescentar novas características, fizeram com que o IETF especificasse uma nova versão para o protocolo IP, o IPv6 [55, 30]. Além de suportar um espaço de endereçamento de 128 *bits* e conter uma estrutura mais simples e eficiente, o IPSec foi incluído como parte integrante e obrigatória na sua estrutura, diferentemente do IPv4, onde sua implementação é opcional e não se integra de maneira natural à estrutura desse protocolo.

3.1 Comparações com o IPv4

No cabeçalho do IPv4, mostrado na Figura 3.1-a, existem 14 campos, onde seis (*Type-of-Service*, *Identification*, *Flags*, *Fragment Offset*, *Options* e *Padding*) não são utilizados freqüentemente, dado que suas funcionalidades estão atreladas a condições anormais, como a fragmentação de pacotes e o uso de opções como *source routing*. Porém, o fato de estarem presentes na estrutura exige que seja gasto tempo com o seu processamento. Além disso, como o campo *Options* não tem tamanho fixo, o cabeçalho IPv4 também não o tem, o que representa outro empecilho no processamento dos seus pacotes. Por outro lado,



(a) IPv4 (b) IPv6

Figura 3.1: Estrutura do cabeçalho de ambas as versões do protocolo IP.

o cabeçalho do IPv6 (Figura 3.1-b) possui tamanho fixo (40 *bytes*) e contém apenas 8 campos, utilizados por todos os tipos de pacotes.

Em relação à estrutura do IPv4, o campo *Version* foi o único que permaneceu com o mesmo tamanho, nome e definição no IPv6. Os campos *Header Length*, *Type of Service*, *Identification*, *Flags*, *Fragment Offset* e *Header Checksum* foram removidos. O primeiro, pelo fato do cabeçalho possuir tamanho fixo, e os demais, por serem utilizados somente por um conjunto reduzido de pacotes. Os campos *Total Length*, *Protocol* e *Time-to-Live* foram renomeados para *Payload Length*, *Next Header* e *Hop Limit*, respectivamente, e redefinidos, conforme a seguir:

- *Payload Length*: representa somente o tamanho da carga útil do pacote desconsiderando o cabeçalho;
- *Next Header*: renomeado para refletir a estrutura dos pacotes IPv6, onde diversos cabeçalhos por ser encadeados em seqüência (Seção 3.2);
- *Hop Limit*: inicialmente, seu correspondente no IPv4, *Time-To-Live*, foi projetado para conter o número de segundos que um pacote poderia permanecer em trânsito durante seu trajeto da origem ao destino. Porém, em seguida, passou a representar o número máximo de *hops* permitido. Desta forma, para adequar o nome ao seu conteúdo, o campo foi renomeado.

Finalmente, para facilitar o tráfego de pacotes de tempo-real, os campos *Class* e *Flow Label* foram criados.

3.2 Cabeçalhos de extensão

Apesar do cabeçalho do IPv6 não contemplar campos para parâmetros adicionais, uma nova estrutura foi elaborada com o intuito de viabilizar o uso de opções. Para tanto, o cabeçalho pode ser seguido de outros denominados cabeçalhos de extensão que oferecem serviços específicos. Por exemplo, caso um pacote necessite utilizar a opção de *source routing*, basta adicionar o cabeçalho de extensão *Routing*, que contempla tal funcionalidade.

Atualmente, estão definidos seis cabeçalhos que podem ser inseridos em pacotes IPv6 para especializar o seu processamento:

- *Hop-by-Hop*: repassa informações a todos os roteadores que processam o pacote no decorrer do seu trajeto entre a origem e o destino;
- *Destination Options*: contém informações a serem processadas pelo destino final de um pacote e, opcionalmente, por todos os roteadores identificados no mecanismo de *source routing*, caso este seja utilizado;
- *Routing*: relaciona uma lista de roteadores pelos quais o pacote deve, obrigatoriamente, passar;
- *Fragment*: provê a estrutura necessária para a implementação do serviço de fragmentação de pacotes que ultrapassam o tamanho da MTU de um trecho da sua rota. É importante notar que no IPv6 pacotes são fragmentados na origem e não durante o seu percurso;
- *AH* e *ESP*: cabeçalhos do IPSec (Capítulo 4).

Quando mais de um cabeçalho de extensão está presente em um pacote, eles devem obedecer uma ordenação pré-definida [55] mostrada na Figura 3.2. *Destination Options* é o único cabeçalho que pode ocorrer duas vezes em um mesmo pacote. Na primeira ocorrência, suas informações são repassadas ao destino final e a todos os roteadores indicados no cabeçalho *Routing*, enquanto que na segunda ocorrência, somente o destino final deve processá-las. Dada que a confidencialidade provida pelo ESP é ponto-a-ponto, a segunda ocorrência será cifrada, caso o uso deste cabeçalho seja necessário.

É importante notar que a necessidade do desenvolvimento de novas funcionalidades não requer a reestruturação do IPv6. Para tal, é suficiente especificar novos cabeçalhos de extensão que suportem as características desejadas. Este mecanismo representa flexibilidade no sentido de adaptar este protocolo a requisitos futuros.

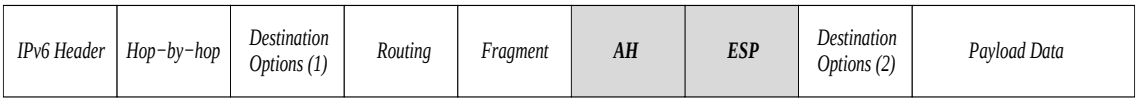


Figura 3.2: Disposição dos cabeçalhos de extensão em um pacote IPv6.

3.3 Considerações sobre o IPv6

O fato do IPSec ser opcional no IPv4 reduz a sua interoperabilidade neste protocolo. Em outras palavras, não é possível utilizar o IPv4 como uma plataforma para a difusão do IPSec enquanto tecnologia para proteger pacotes de serviços em ambientes heterogêneos, pois não se pode garantir que extremos quaisquer terão suporte ao IPSec nas suas implementações de IPv4.

Caso um determinado ambiente possua uma política genérica que estabeleça que toda comunicação com outros ambientes deve se dar através do uso da proteção do IPSec, este certamente não poderá se comunicar com ambientes IPv4 que não contenham o IPSec[1]. Desta forma, o uso deste protocolo sobre o IPv4 deve limitar-se a ambientes pré-definidos onde haja a garantia da existência do suporte ao IPSec.

Por outro lado, a obrigatoriedade de implementação do IPSec e a maneira como está incorporado ao modelo do IPv6 tornam esse protocolo um ambiente nativo para o desenvolvimento de aplicações e mecanismos de segurança que utilizam seus serviços. Respeitadas possíveis incompatibilidades relativas aos parâmetros das políticas de segurança, a comunicação entre dois ambientes baseados em IPv6 que façam uso do IPSec não pode ser inviabilizada pela falta de suporte às suas estruturas. Desta forma, apesar da solução apresentada neste trabalho poder funcionar sobre o IPv4, o seu desenvolvimento e, conseqüentemente, toda sua explanação estão baseados no IPv6. Conforme apresentado em [63], contudo, é preciso ter em mente que a nova versão do protocolo IP traz consigo um conjunto de novos problemas de segurança que devem ser levados em consideração na tentativa de reduzir o impacto na segurança dos sistemas que o estiverem adotando.

[1]Neste exemplo, hipoteticamente, considera-se que esta "política genérica" pudesse ser padrão entre todos os ambientes que utilizam o IPSec.

Capítulo 4

IP Security

Quando o protocolo IP foi especificado, em meados da década de 70, certamente a segurança não era um dos aspectos fundamentais que norteavam seu projeto. Basicamente utilizado em redes limitadas aos meios acadêmico e militar, onde se mantinha uma cooperação harmoniosa entre os usuários, sua pricipal característica era prover o serviço de entrega de pacotes de dados mesmo diante de alterações inesperadas na topologia da rede. Tal característica seria suficiente para, por exemplo, manter a comunicação no caso de uma guerra, onde explosivos poderiam atingir linhas de transmissão no intuito de impedir a troca de informações por parte do inimigo.

Na década seguinte, a Internet formou-se da junção de estruturas de redes já existentes e, desde então, começou rapidamente a ganhar alcance mundial. A indústria, o comércio, organizações de variados portes e ramos de atividades, bem como usuários domésticos, passaram a incorporá-la aos seus cotidianos, transformando-a em um importante elemento social, cultural e econômico. Negócios, lazer, pesquisas e comunicação são apenas alguns dos seus serviços. Porém a "cooperação harmoniosa" não mais se adequava aos novos usuários da rede mundial, que se mostrou extremamente frágil e desarmada diante de ataques e problemas de segurança não explorados anteriormente. O primeiro grande incidente foi o *Internet Worm*, desenvolvido por Robert Tappan Morris, aluno do primeiro ano de graduação na *Cornell University*, em 1988. O programa atingiu, em poucas horas, 6.000 dos 60.000 computadores conectados à Internet àquela época [65, 21].

A necessidade de proteger recursos fez surgirem mecanismos de segurança, como os *firewalls*, e aplicações como o SSH, por exemplo. Além disso, novas características são constantemente incorporadas a tecnologias existentes, resultando em novas soluções, como o HTTPS. Neste contexto, na tentativa de adicionar segurança a pacotes IP, o IETF (*Internet Engineering Task Force*), órgão responsável pela padronização de tecnologias da Internet, resolveu especificar extensões a este protocolo, resultando no IPSec (*IP Security*) [34, 19, 8].

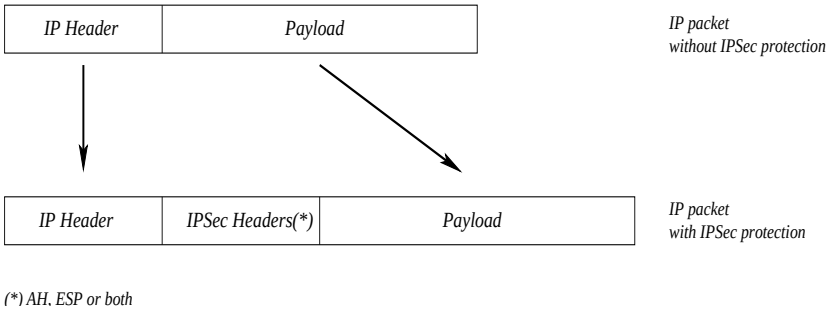


Figura 4.1: Cabeçalhos AH e ESP em um pacote IP.

Neste capítulo, serão apresentados os aspectos básicos do IPSec, visando elucidar suas principais características e estruturas, dado que seu entendimento é de fundamental importância para este trabalho.

4.1 Características gerais

O IPSec é, fundamentalmente, baseado no uso de dois protocolos de segurança: o AH (*Authentication Header*) [32] e o ESP (*Encapsulating Security Payload*) [33], implementados como cabeçalhos adicionais inseridos em datagramas após o cabeçalho IP, conforme mostrado na Figura 4.1. Com o intuito de evitar que o conteúdo de pacotes seja lido, alterado, modificado, ou ainda, reenviado (procedimento utililizado em ataques de *replay* [75]), o IPSec provê o seguinte conjunto de serviços de segurança aos pacotes sob sua proteção:

- *Integridade sem conexão*: garante que o conteúdo de um pacote recebido não foi alterado durante o seu trajeto entre a origem e o destino[1];
- *Autenticação da origem dos dados*: garante a autenticidade do emissor, evitando o processamento de pacotes enviados por terceiros com identidade de origem falsificada (procedimento conhecido como *IP spoofing* [75]);
- *Confidencialidade*: compreende a cifragem da porção de dados de um pacote, impedindo que o seu conteúdo seja lido a partir da sua captura durante o processo de transmissão;

[1]Por tratar-se da camada de rede baseada em IP[69, 9], onde não há qualquer confirmação a respeito do recebimento ou não de um pacote, o serviço de integridade do IPSec é oferecido na ausência de conexão.

Serviços	**AH**	**ESP**[a]	**ESP**[b]
Integridade sem conexão	3		3
Autenticação da origem dos dados	3		3
Confidencialidade		3	3
Proteção contra *replay*[c]	3	3	3
Controle de acesso	3	3	3

[a] Utilizando somente cifragem
[b] Utilizando cifragem, autenticação e integridade
[c] Serviço opcional

Tabela 4.1: Serviços oferecidos pelos protocolos de segurança AH e ESP.

- *Proteção contra replays*: serviço opcional que previne o reenvio de pacotes e, conseqüentemente, impossibilita a prática de determinados ataques que se valem deste procedimento [14];

- *Controle de acesso*: o uso de determinados parâmetros de segurança para o estabelecimento de uma comunicação sob a proteção do IPSec está sujeito à concordância com as regras que compõem as políticas de segurança de ambos os extremos.

Cada um dos cabeçalhos que podem ser utilizados separadamente ou em conjunto em um mesmo pacote oferece um subconjunto distinto dos serviços anteriores, de acordo com a Tabela 4.1. É importante notar que o ESP, cujo principal serviço é a confidencialidade dos dados, pode, opcionalmente, prover autenticação e integridade, dado o risco potencial de ataques como o de *cut-and-paste* [6, 63].

A principal diferença entre os serviços de autenticação e integridade providos pelo AH e pelo ESP está na abragência da proteção, conforme mostrado na Figura 4.2. O AH protege todos os campos de um pacote, excetuando-se aqueles cujos valores são alterados em trânsito, como o *Hop Limit* manipulado pelos roteadores que processam o pacote. Quando oferecidos pelo ESP, estes serviços abrangem somente o próprio cabeçalho do ESP e a porção de dados do pacote.

Ambos os protocolos possuem dois modos de operação: transporte e túnel. No primeiro caso, os cabeçalhos de segurança utilizados por um pacote são inseridos após o cabeçalho IPv6 e os cabeçalhos de extensão que o seguem, conforme mostrado na Figura 3.2 na página 22. O modo de transporte, em geral, é utilizado para a proteção fim-a-fim da comunicação entre dois *hosts* e representa uma solução adequada para auxiliar na segurança de pacotes em redes locais.

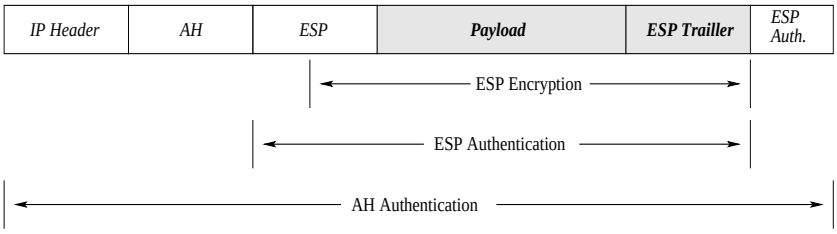


Figura 4.2: Abrangência da proteção provida pelos protocolos AH e ESP.

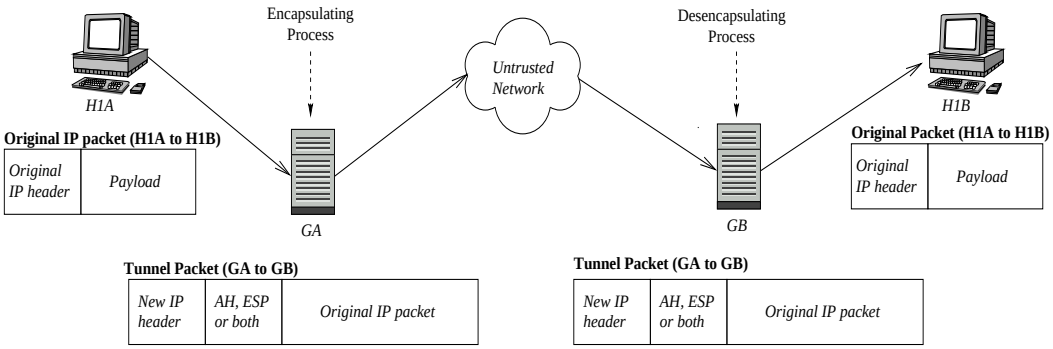


Figura 4.3: Exemplo da aplicação dos cabeçalhos AH e ESP no modo túnel.

O modo de túnel ou tunelamento tem seu uso recomendado na proteção do tráfego entre um *host* e um *gateway* ou entre dois *gateways* [34]. Antes de ser enviado, o pacote original é inserido completamente na porção de dados de um novo pacote que contém, além dos cabeçalhos de segurança, alguns cabeçalhos de extensão que podem ser necessários. Os endereços devem corresponder aos extremos do túnel. Na Figura 4.3 é apresentado um cenário onde um *host* da rede A, $H1_A$, envia um pacote para um *host* da rede B, $H1_B$, através de um túnel configurado entre os *gateways* G_A e G_B das redes A e B, respectivamente. O pacote original, gerado em $H1_A$, ao ser recebido por G_A, é inserido em um novo pacote contendo endereço G_A como origem e o de G_B como destino. Após atravessar uma rede não-confiável como a Internet, por exemplo, o pacote é recebido por G_B, que extrai o pacote original e o entrega para $H1_B$ da maneira como foi gerado por $H1_A$.

4.1.1 Estrutura do *Authentication Header*

O protocolo IP em sua forma original, sem o uso do IPSec, não contém qualquer proteção contra o envio de pacotes com endereço de origem falsificado muito menos contra a modificação do conteúdo de pacotes capturados em trânsito. Estas fragilidades, combinadas com outros fatores, permitiram o desenvolvimento de ataques como o *IP spoonfing*, DNS *spoofing*, reinjeção de pacotes, entre outros.

O mecanismo de *checksum*, existente somente no IPv4 [68, 9], foi projetado apenas para detectar problemas decorrentes da manipulação incorreta dos pacotes por parte de periféricos, sendo incapaz de detectar a manipulação maliciosa de pacotes capturados e posteriormente reenviados. Em outras palavras, após capturar, acessar e modificar o conteúdo de um pacote, é suficiente recomputar o valor do *checksum* para que o destino o valide e o aceite como legítimo. Desta forma, um dos serviços fundamentais para a definição de mecanismos capazes de prover segurança à camada IP é a autenticação e integridade dos dados, provida no IPSec pelo AH.

É importante notar que os dois serviços, apesar de serem oferecidos em conjunto, são distintos entre si. A autenticação garante que um pacote foi realmente enviado pelo endereço indicado. A integridade garante que as informações recebidas não foram alteradas durante seu trajeto entre a origem e o destino.

A Figura 4.4 mostra o formato do AH, composto por cinco campos de tamanho fixo e um de tamanho varíavel. A seguir, é apresentada uma breve descrição sobre o significado de cada um dos seus campos:

- *Next Header*: contém a identificação do próximo cabeçalho que segue o AH, podendo indicar a presença dos protocolos TCP, UDP, ICMP, IP (caso o modo de tunelamento esteja sendo utilizado), ou cabeçalhos de extensão;

- *Payload Length*: representa o tamanho do cabeçalho AH em palavras de 32 *bits* menos 2. Inicialmente, na especificação original [3], este valor representava exatamente a porção de dados do AH, já que o campo *Sequence Number Field* não estava especificado. Com sua inclusão, *Payload Length* pode também ser visto como o tamanho da porção de dados mais um, oriundo da palavra consumida pelo campo *Sequence Number Field*;

- *Reserved*: reservado para uso futuro, sendo preenchido por uma cadeia de zeros;

- *Security Parameters Index* (SPI): índice utilizado para indicar os parâmetros a serem utilizados pelo receptor no processamento do pacote (Seção 4.2);

- *Sequence Number Field*: número incrementado pelo emissor para que o receptor proteja-se do reenvio indevido de pacotes. O funcionamento correto deste serviço é

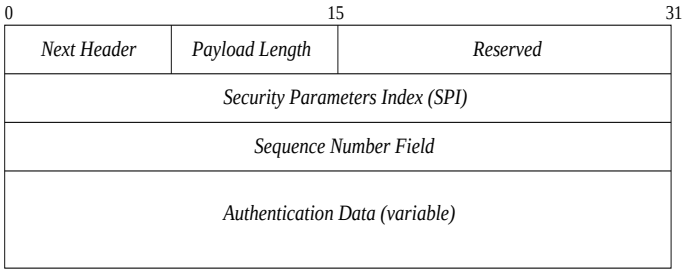


Figura 4.4: Formato do *Authentication Header*.

condicionado ao processamento deste valor por parte do receptor. Caso contrário, mesmo que o emissor incremente este valor a cada pacote enviado, o receptor estará sujeito a este tipo de ataque;

- *Authentication Data*: único campo cujo tamanho não é fixo, *Authentication Data* contém o ICV (*Integrity Check Value*), que corresponde a um MAC (*Message Authentication Code*), utilizado pelo receptor para conferir a autenticidade e integridade do pacote, e *bytes* de *padding*, que podem ser inseridos caso seja necessário ajustar o tamanho deste campo aos limites exigidos pelo algoritmo específico que estiver sendo utilizado.

O cálculo do ICV é precedido da geração de uma versão especial de um pacote cujos campos ou cabeçalhos, alterados em trânsito (e.g. o campo *Hop Limit* ou o cabeçalho *Hop-by-Hop*), são desconsiderados e seus valores zerados. Alguns campos que sofrem modificações durante o trajeto, contudo, podem ter seus valores finais previstos (e.g. o endereço de destino de pacotes que utilizam o cabeçalho *Routing* para o serviço de *source routing*) e, portanto, são inclusos na computação do ICV.

4.1.2 Estrutura do *Encapsulating Security Payload*

Além da ausência de serviços de autenticação e integridade, o protocolo IP também não possuía qualquer mecanismo capaz de prover cifragem ao conteúdo dos seus pacotes. Desta forma, aplicações antigas, como *Telnet* e *FTP*, que não contêm qualquer preocupação com segurança, permitem que suas informações, incluindo *login* e senha, sejam facilmente obtidas através da captura dos pacotes correspondentes, resultando no comprometimento do acesso dos usuários vítimas e, conseqüentemente, representando um risco para toda a rede da qual estes fazem parte. Este fato agrava-se ainda mais considerando-se o fato de que muitos usuários utilizam as mesmas senhas para acessar diversos sistemas (e.g. a

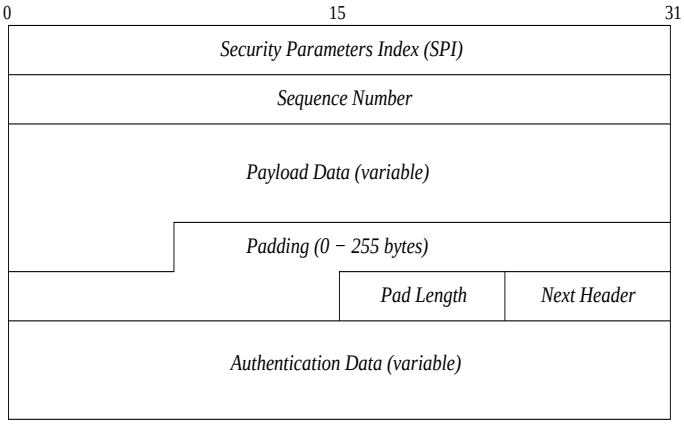


Figura 4.5: Formato do *Encapsulating Security Header*.

senha capturada é a mesma utilizada para acessar o *site* do banco no qual o cliente possui conta corrente) [57].

Para impedir a obtenção de informações sigilosas, foram desenvolvidos protocolos (e.g. SSH e SSL) que passaram a proteger suas informações através do uso de algoritmos criptográficos de cifragem, impedindo que a obtenção dos seus pacotes pudesse comprometer os dados transmitidos. Porém, estas soluções são desenvolvidas para fins específicos e ainda deixavam expostas informações importantes, como os dados do protocolo de transporte. Sendo assim, para prover um mecanismo genérico capaz de garantir o sigilo de dados transmitidos, o IPSec provê o ESP.

Na Figura 4.5 é apresentada a estrutura do cabeçalho ESP, formada por cinco campos de tamanho fixo e dois com tamanhos variáveis:

- *Security Parameters Index* (SPI): índice utilizado para indicar os parâmetros a serem utilizados pelo receptor no processamento do pacote (Seção 4.2);

- *Sequence Number Field*: número incrementado pelo emissor, para que o receptor proteja-se do reenvio indevido de pacotes. O funcionamento correto deste serviço é condicionado ao processamento deste valor por parte do receptor. Caso contrário, mesmo que o emissor incremente este valor a cada pacote enviado, o receptor estará sujeito a este tipo de ataque ;

- *Payload Data*: contém a versão cifrada da porção de dados do pacote original e, opcionalmente, dados não cifrados, e.g. o IV (*Initialization Vector*), necessários para o funcionamento do procedimento de cifragem de alguns algoritmos criptográficos;

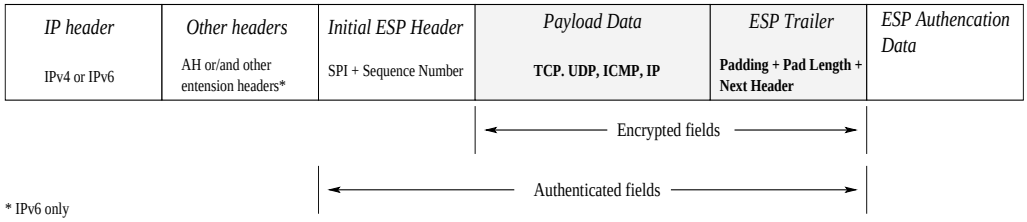


Figura 4.6: Distribuição dos serviços de autenticação, integridade e confidencialidade do ESP.

- *Padding*: campo opcional utilizado para inserção de *bytes* (no máximo 255) de *padding*, para ajustar os blocos dos algoritmos de cifragem e autenticação (caso este serviço seja utilizado) ou para procedimentos de proteção contra análise de tráfego, onde os *bytes* inseridos visam confundir o tamanho exato dos dados válidos do pacote no intuito de dificultar a ação de atacantes;

- *Pad Length*: armazena o número total de *bytes* de *padding* inseridos no campo anterior;

- *Next Header*: contém a identificação do próximo cabeçalho que segue o ESP, podendo indicar a presença dos protocolos TCP, UDP, ICMP, IP (caso o modo de tunelamento esteja sendo utilizado) ou cabeçalhos de extensão;

- *Authentication Data*: contém o ICV relativo aos dados protegidos pelo serviço de autenticação e integridade do ESP. Este campo é utilizado somente quando a autenticação e integridade são requeridas;

Os campos SPI, *Sequence Number* (conhecidos como *Initial ESP Header*) e *Authentication Data*, não são cifrados. Isto se deve ao fato de que seus valores devem estar acessíveis, para que o receptor consiga decifrar suas informações, proteger-se do reenvio de pacotes e validar a autenticidade e integridade dos dados protegidos, respectivamente. Os campos seguintes, *Payload Data*, *Padding*, *Pad Length* e *Next Header* (os três últimos são conhecidos como *ESP Trailer*), são cifrados e, portanto, protegidos contra a leitura por parte de terceiros. A autenticação do ESP abrange as porções: *Initial ESP header*, o *Payload Data* e o *ESP Trailer*. A Figura 4.6 resume a abrangência de aplicação de cada um dos serviços do ESP de acordo com as regiões do cabeçalho.

4.1.3 Algoritmos criptográficos obrigatórios

Apesar dos serviços dos protocolos AH e ESP serem independentes de quaisquer algoritmos criptográficos em particular, sob a justificativa de manter um nível mínimo de compatibilidade entre as diversas implementações, um conjunto básico de algoritmos é obrigatório:

- *HMAC-MD5-96* [40] e *HMAC-SHA-1-96* [41]: algoritmos de autenticação e integridade utilizados pelo AH e, opcionalmente, pelo ESP;
- *DES-CBC* [39]: algoritmo de cifragem utilizado pelo ESP;
- Algoritmos nulos de autenticação e cifragem [22] utilizados pelo ESP.

A presença dos algoritmos de autenticação e cifragem nulos associados ao ESP se dá pelo fato dos serviços de confidencialidade e autenticação/integridade serem opcionais. Porém estes dois algoritmos não podem ser utilizados ao mesmo tempo em um dado pacote [33, 19]. Em outras palavras, o protocolo ESP deve prover, efetivamente, pelo menos um dos seus serviços, quando utilizado na proteção de pacotes.

Os algoritmos de autenticação e integridade obrigatórios, *HMAC-MD5-96* e *HMAC-SHA-1-96*, geram MACs de 128 e 160, *bits*, respectivamente. Porém suas saídas são truncadas por questões de eficiência de processamento para 96 *bits*, o que não prejudica a proteção provida por cada um deles [19, 38].

Considerações sobre a descrição e segurança destes e outros algoritmos utilizados pelo IPSec serão apresentadas no Capítulo 5.

4.2 Associações de segurança

Antes que dois *hosts* possam trocar pacotes protegidos pelos serviços do IPSec, o seguinte conjunto de parâmetros deve ser estabelecido entre as duas entidades: protocolo de segurança (AH ou ESP), modo de operação (transporte ou túnel), algoritmos criptográficos para cada serviço, tamanho e valor das suas chaves, dados para o serviço de proteção contra *replay* e tempo de vida destes parâmetros. Tal conjunto é denominado associação de segurança (AS) [34, 19, 8].

ASs são identificadas em cada *host* por triplas compostas dos seguintes valores: endereço de destino, protocolo de segurança e SPI, índice gerado pelo *host* destino para identificar uma AS dentre outras que podem estar estabelecidas com um mesmo *host* para um protocolo específico. Antes de enviar um pacote, para cada protocolo de segurança utilizado, o IPSec deve indicar, através desta tripla, a AS que deverá ser utilizada para computar os valores necessários para a composição do cabeçalho correspondente.

Para que a entidade de destino consiga validar os serviços de um pacote recebido e dar continuidade ao seu processamento, é necessário que os campos que formam a tripla de identificação de cada AS estejam sempre visíveis. Em vista desta necessidade, o serviço de confidencialidade do ESP não inclui o SPI, conforme explicitado na Subseção 4.1.2.

Além dos dados espefícicos para proteção de pacotes, cada AS contém seletores, informações que permitem a uma entidade verificar sua adequabilidade a um determinado tipo de tráfego. Este procedimento impede que sejam aceitos pacotes enviados por um atacante sob uma proteção com parâmetros menos rígidos do que aqueles que deveriam ter sido, de fato, utilizados. Por exemplo, uma AS estabelecida para proteger o tráfego HTTP entre um cliente e um servidor não deve ser utilizada para processar pacotes referentes a uma transferência de zona de DNS entre dois servidores.

Os seletores definidos na especificação do IPSec englobam:

- Endereços de origem e destino: pode incluir endereços simples ou faixas indicadas através do uso de máscaras de rede;
- Nome: indica o identificador de um usuário ou de um sistema representados por nomes DNS, completamente qualificados no primeiro caso, ou pelo campo *Distinguished Name* de um certificado X.500[2];
- Protocolo de transporte: seleciona o UDP ou TCP;
- Portas de origem e destino: utilizadas para especializar a proteção ao tráfego de um determinado tipo de serviço.

Os seletores podem ser combinados de acordo com a abrangência de cada AS. Aqueles que não são relevantes para a contrução de um cenário específico podem ser descartados. Por exemplo, para formular uma AS que proteja todo o tráfego TCP de um determinado usuário para *hosts* quaisquer, devem ser utilizados somente valores para os seletores de nome e protocolo de transporte, descartando-se os demais. Algumas combinações de seletores possuem classificações específicas, conforme mostrado a seguir:

- Orientado à *host*: quando são utilizados somente endereços de origem e destino, fazendo com que toda a comunicação entre dois *hosts* seja protegida por uma única AS para cada protocolo;
- Orientado à usuário: quando são utilizados somente identificadores de usuários, fazendo com que toda a comunicação entre dois usuários seja protegida por uma única AS para cada protocolo;

[2]No caso do nome representar um sistema é possível ainda utilizar o campo *General Name* de um certificado X.500.

- Orientado à sessão: quando são utilizados endereços de origem e/ou destino, portas de origem e/ou destino e protocolo de transporte. Neste caso, cada AS protege uma sessão do tráfego de um tipo de serviço.

É importante notar que ASs são estruturas unidirecionais e podem conter somente um protocolo de segurança. Ou seja, uma única AS pode proteger um tipo de tráfego entre dois *hosts* em um único sentido através dos serviços do AH ou do ESP. Desta forma, é possível que uma troca de pacotes tenha parâmetros de proteção distintos em ambos os sentidos. Na Figura 4.7 são exibidos alguns possíveis cenários entre dois *hosts*.

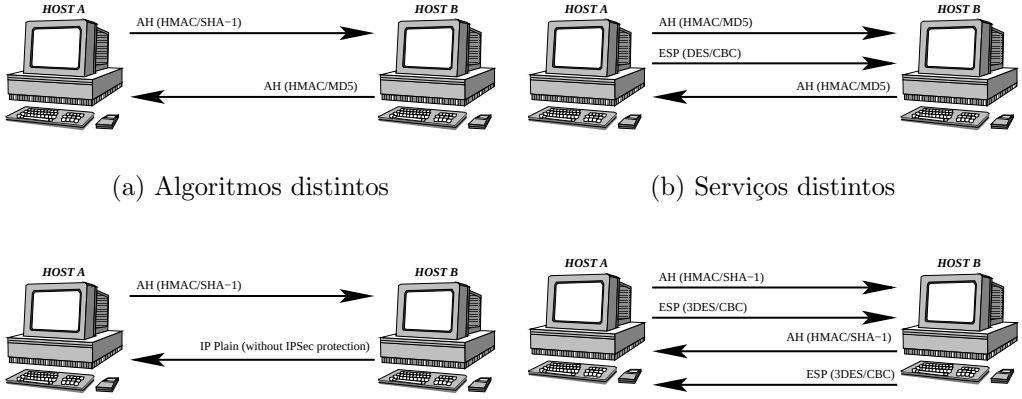
No primeiro exemplo, (a) o tráfego entre os *hosts* A e B é protegido pelo AH em ambos os sentidos, porém o SHA-1, algoritmo utilizado de A para B, é mais seguro que o MD5, utilizado de B para A. De posse de tais informações, um atacante pode concentrar-se somente na exploração dos dados do sentido mais fraco, para tentar desenvolver algum tipo de ataque. Em (b), o tráfego de A para B está protegido por ambos os protocolos. Porém, de B para A, somente o AH, com o mesmo algoritmo do sentido oposto, é utilizado. Note que, se a confidencialidade é fundamental para proteger o tráfego, este sentido representa uma fragilidade grave que pode facilmente ser explorada por atacantes. No cenário (c), somente um dos sentidos da comunicação está protegido. Qualquer que seja o protocolo utilizado, deixar um dos sentidos sem proteção expõe o tráfego a todas as vulnerabilidades conhecidas do protocolo IP. Na situação (d), a proteção é aplicada utilizando os mesmos protocolos e algoritmos, provendo segurança bidirecional ao tráfego. Este cenário é o ideal, pois há um balanceamento perfeito da proteção em ambos os sentidos da comunicação. Vale ressaltar, porém, que todos os cenários anteriores são possíveis de serem construídos com o IPSec, de acordo com a estrutura elaborada pelo administrador do sistema.

4.2.1 Formas de estabelecimento de associações de segurança

O estabelecimento de uma AS pode ser estático ou dinâmico. No primeiro caso, todos os parâmetros devem ser manualmente inseridos pelo administrador, inviabilizando seu uso em ambientes onde a comunicação segura pode ocorrer entre dois *hosts* quaisquer que não possuem parâmetros pré-estabelecidos entre si, a não ser a própria política de segurança.

Além disso, a grande quantidade de intervenção humana pode resultar em erros capazes de impedir o estabelecimento de ASs e, conseqüentemente, a comunicação para o tipo de tráfego que deveria ser protegido. Por exemplo, se em um dos extremos uma chave criptográfica ou o algoritmo de cifragem não está de acordo com o outro extremo, não será possível enviar ou receber pacotes corretamente em ambos os lados.

No segundo caso, os parâmetros de uma AS são estabelecidos automaticamente através do protocolo IKE sem qualquer intervenção por parte do administrador do sistema. Maiores detalhes deste protocolo serão apresentados na Seção 4.3.



(a) Algoritmos distintos (b) Serviços distintos

(c) Proteção unidirecional (d) Mesmos serviços e algoritmos

Figura 4.7: Exemplos de cenários de associações de segurança para a proteção de tráfego entre dois *hosts*.

4.2.2 SAD e SPD

O SAD (*Security Association Database*) é um componente do IPSec utilizado para armazenar as ASs ativas de um *host* num dado momento [34]. Em outras palavras, toda AS estabelecida deve conter uma entrada inserida pelo administrador do sistema ou pelo protocolo IKE, nesta base de dados. Porém, antes de enviar pacotes, o IPSec não consulta as ASs do SAD diretamente para a inclusão dos cabeçalhos de segurança.

ASs são instâncias resultantes das restrições de proteção impostas por regras que formam a política de segurança local armazenadas em uma estrutura denominada SPD (*Security Policy Database*) [34]. Composta pelo arranjo de seletores, cada regra deve definir uma das seguintes ações para os pacotes filtrados:

- Descartar: impede a continuação do processamento do pacote, evitando que este seja transmitido a outro *host* ou recebido pela camada de transporte. Em algumas plataformas, o evento de descarte é registrado;
- Repassar sem proteção: permite que o pacote seja manipulado normalmente pela camada IP, desconsiderando qualquer intervenção por parte do IPSec;
- Aplicar proteção: exige que o pacote seja submetido ao processamento do IPSec.

As regras que determinam a aplicação do IPSec devem indicar os protocolos de segurança e o modo de operação que deverá ser utilizado sobre os pacotes filtrados. É

Rule	Source Address	Dest. Address	Source Port	Dest. Port	Transp.	IPSec	Enc. Alg.	Auth. Alg.	Mode
1	Host A	Host B	any	53	udp	AH	—	MD5	Transp.
2	Host B	Host A	53	any	udp	AH	—	MD5	Transp.
3	Host A	Host C	any	23	tcp	AH	DES	MD5	Transp.
						ESP			Transp.
4	Host C	Host A	23	any	tcp	AH	3DES	SHA1	Transp.
						ESP			Transp.
5	any	Host A	any	22	tcp	ESP	DES	Null	Transp.

Tabela 4.2: Exemplo de regras de um SPD para aplicação da proteção do IPSec.

importante notar que uma única regra pode exigir o uso do AH e do ESP. Além disso, se o modo de tunelamento é exigido, os *gateways* intermediários devem ser especificados.

A Tabela 4.2 apresenta como exemplo regras contidas no SPD de um *host* A, hipotético[3]. Os seletores das duas primeiras regras determinam que todo tráfego UDP enviado ou recebido de qualquer porta de A para o servidor DNS (porta 53) de B deve ser protegido pelo cabeçalho AH utilizando o MD5 como algoritmo de autenticação e integridade, em ambos os sentidos com o modo de transporte. Em outras palavras, todo tráfego relativo a consultas DNS feitas por A a B deve ser submetido à proteção especificada. As regras 3 e 4 filtram todo o tráfego de A destinado ao servidor *telnet* (porta 23) de C e deste servidor para A, indicando a proteção por ambos os cabeçalhos de segurança no modo de transporte. Porém, as direções possuem parâmetros de proteção distintos: de A para C os algoritmos especificados são o MD5 e o DES; no sentido contrário, o SHA-1 e o 3DES devem ser utilizados. É importante notar que, se o serviço de *telnet* de A receber pacotes de C, nenhuma proteção será aplicada segundo as regras apresentadas. A última regra especifica que o tráfego recebido por qualquer *host* destinado ao servidor SSH (porta 22) de A deve ser protegido pelo ESP, no modo de transporte, através do algoritmo DES. Diferentemente dos outros conjuntos de regras, o serviço de SSH está protegido pelo IPSec em somente um dos sentidos. Além disso, o serviço de autenticação e integridade dos dados cifrados não está sendo utilizado.

4.2.3 Interação entre SAD e SPD

O correto funcionamento do IPSec depende da correlação das informações entre o SAD e o SPD. A Figura 4.8 exibe um diagrama elaborado para ilustrar a interação entre estas duas bases de dados no envio de pacotes.

[3]A ação correspondente a todas as regras mostradas na Tabela 4.2 é a aplicação do IPSec.

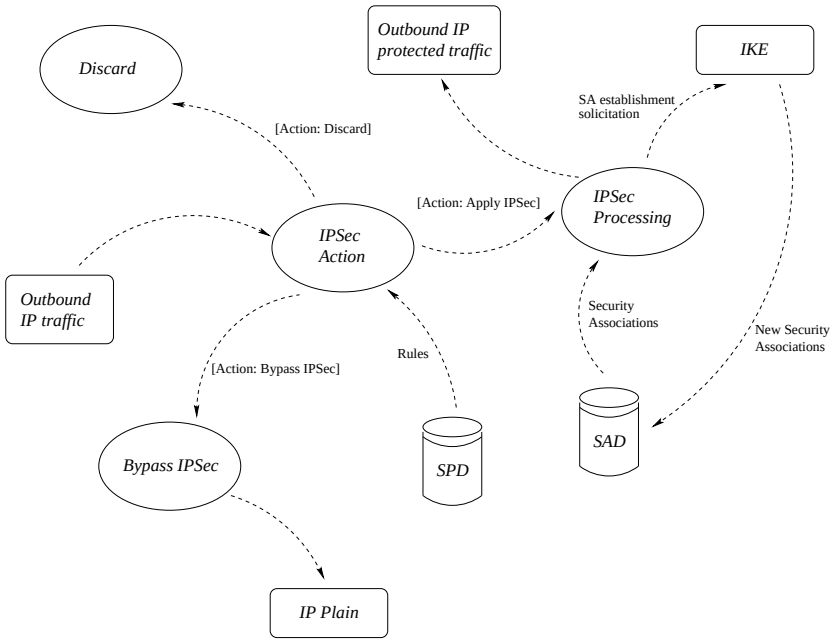


Figura 4.8: Esquema do processamento de pacotes enviados em uma plataforma IPSec.

Primeiramente, os campos de um pacote são verificados de acordo com os seletores das regras presentes no SPD. Após encontrar uma regra que seja compatível, o pacote é encaminhado conforme a ação associada à regra. Caso a aplicação do IPSec seja necessária, as ASs correspondentes àquela regra devem ser recuperadas do SAD para a criação e inserção dos cabeçalhos de segurança correspondentes. Se as ASs necessárias não estiverem presentes no SAD, o protocolo IKE é acionado para que seja dado início ao processo de estabelecimento de novas ASs. Se esta operação ocorrer com sucesso, os parâmetros das ASs recém-criadas são inseridas pelo IKE no SAD e o IPSec dá continuidade à manipulação do pacote. Caso contrário, o pacote deve ser descartado. Apesar da especificação determinar que pacotes que não tenham regras associadas sejam descartados, muitas implementações encaminham-os para o processamento normal da pilha IP [19].

No recebimento de pacotes, o SAD é consultado antes do SPD. Para cada um dos cabeçalhos presentes em um pacote, as triplas SPI, endereço de destino e protocolo de transporte, são utilizadas para a recuperação de cada AS no intuito de validar e processar os serviços utilizados: autenticação, integridade e/ou confidencialidade. Em seguida, é feita uma comparação dos campos do pacote com os seletores que compõem as ASs. Caso tais valores sejam correspondentes, as ASs utilizadas no recebimento do pacote são repassadas à segunda etapa do processo de recebimento. Nesta ocasião, é verificado se as ASs utilizadas no processamento do pacote estão de acordo com a política de segurança

representada pelas regras do SPD. Caso negativo, o pacote deve ser descartado[4].

O nível de granularidade das ASs a serem estabelecidas para um ambiente são de responsabilidade do administrador do sistema. Em outras palavras, da mesma forma que a política de segurança pode ser elaborada com base em ASs orientadas à *host*, é possível elaborá-la para ASs orientadas à sessão. Pode-se ainda defini-la utilizando uma combinação dos diversos tipos de ASs. Utilizar ASs mais específicas aumenta a segurança das informações protegidas, pois uma quantidade menor de dados é transmitida utilizando os parâmetros das mesmas ASs, i.e, o ataque a uma AS não compromete a proteção do tráfego restante. Por outro lado, tal procedimento deve aumentar a quantidade de tráfego na rede, em decorrência da necessidade do estabelecimento freqüente de ASs exigir maior quantidade de processamento e memória.

4.3 Internet Key Exchange

O funcionamento do IPSec está intimamente relacionado ao uso, definição e distribuição de chaves criptográficas, seu tempo de vida e tamanho, algoritmos de autenticação e/ou cifragem, e de um conjunto de outros parâmetros que constituem as associações de segurança.

As associações de segurança podem ser estabelecidas manualmente ou dinamicamente, conforme mostrado na Subseção 4.2.1. O primeiro método é, certamente, inadequado para viabilizar a aplicação do IPSec em redes escaláveis, seja pela dificuldade na manutenção dos parâmetros seja pela redução da segurança provida devido ao uso de chaves criptográficas baseadas em valores fixos.

Especificado pelo IETF como um componente adicional ao IPSec, o IKE (*Internet Key Exchange*) [27, 19] é o protocolo responsável pelo estabelecimento e manutenção dinâmica das associações de segurança. Fundamentalmente, o IKE é baseado nos protocolos OAKLEY [47] e SKEME [37], que provêem mecanismos para a definição e a troca de chaves criptográficas, e ISAKMP [42], um *framework* que define estruturas e procedimentos gerais para a criação, deleção, modificação e negociação de ASs.

Vale ressaltar que o IKE é suficientemente genérico para criar, a partir de ISAKMP SAs, ASs para outros protocolos além do IPSec, como o TLS e o OSPF, por exemplo. Para tal, é suficiente que, para cada protocolo, seja definido um DOI (*Domain of Interpretation*) que especifica o formato e a troca de mensagens, os algoritmos criptográficos permitidos e outros parâmetros que devem ser utilizados na criação de ASs para um protocolo específico [42]. O DOI para o IPSec está definido em [50].

[4]Algumas implementações registram estas ocorrências pelo fato de poderem indicar tentativas de ataque.

4.3.1 Funcionamento básico

O processo de estabelecimento de ASs é composto por duas fases distintas. Num primeiro momento, uma AS denominada ISAKMP SA (*ISAKMP Security Association*) é definida para proteger todo o tráfego subseqüente do protocolo IKE entre dois *hosts*.

A ISAKMP SA pode ser criada através de três métodos que se diferenciam pela quantidade e tipos de mensagens trocadas entre o *initiator*, *host* que dá início ao processo de estabelecimento de uma AS, e o *responder host* ao qual a solicitação do *initiator* destina-se: o *Main Mode*, mais seguro, contendo seis mensagens; o *Aggressive Mode*, menos seguro e contendo somente três mensagens; e o *Base Mode*, que utiliza quatro mensagens e procura manter-se como um meio termo entre a segurança do primeiro modo e a eficiência do segundo. O *Base Mode* foi criado após a especificação original do IPSec e, portanto, muitas plataformas ainda não o contemplam [19].

Dentre as incumbências da ISAKMP SA estão a geração de um segredo compartilhado utilizado na derivação das chaves criptográficas para os algoritmos de proteção desta AS e a autenticação de ambos os *hosts* envolvidos na comunicação. Para realizar este último procedimento, a especificação do IKE oferece três métodos: o segredo pré-compartilhado, onde o valor que será derivado para a criação das chaves criptográficas é inserido manualmente pelo administrador; cifragem com chave pública; e assinatura digital, onde são requeridas operações baseadas no uso das chaves privada e pública que devem estar associadas aos *hosts*. No primeiro método, a verificação de identidade é baseada somente no valor do segredo pré-compartilhado. Desta forma, sua obtenção é suficiente para falsificar uma negociação. Além disso, não há, neste método, um mecanismo definido para a substituição de velhos valores por novos, limitando-o a ambientes pequenos e pouco escaláveis. Os dois outros métodos contemplam a interação com entidades confiáveis para a distribuição de chaves públicas através da estruturação, por exemplo, de uma PKI (*Public Key Infrastructure*) [19, 2, 28].

Definida a ISAKMP SA e sob sua proteção, as ASs específicas do IPSec, denominadas IPSec SAs (*IPSec Security Associations*), são estabelecidas entre o *initiator* e o *responder* através de um único método denominado *Quick Mode*, composto por três mensagens. Cada IPSec SA é gerada a partir da análise das regras que compõem a política de segurança armazenada no SPD com o objetivo de proteger uma determinada espécie de tráfego IP. Em outras palavras, no momento em que um determinado pacote é filtrado por alguma regra e pelo menos uma dentre ASs necessárias para prover os serviços requeridos não está presente no SAD, o IKE é acionado para estabelecer a(s) IPSec SA(s) restantes. Quando um pacote deve ser submetido à proteção de mais de uma AS e duas ou mais ainda não estão estabelecidas, uma única negociação de *Quick Mode* pode ser utilizada para a criação das várias ASs. Este conjunto de ASs, destinado a proteção de um tráfego comum, é denominado na especificação por *SA bundle* [19].

Proposta	Prot.	Transf.	Cifragem	Autent.	Modo	DH	Tempo	
							s	*bytes*
1	ESP	1	DES	–	Transp.	–	150	100
		2	DES	MD5	Transp.	–	400	400
		3	3DES	SHA-1	Transp.	–	3500	1500
		4	Blowfish:128	SHA-1	Transp.	1	3500	2000
		5	Blowfish:192	RIPEMD	Transp.	5	5000	3000

Tabela 4.3: Exemplo de proposta para o estabelecimento de uma associação de segurança para o protocolo ESP.

Na primeira mensagem do *Quick Mode* enviada pelo *initiator* ao *responder* podem estar contidas várias propostas que, por sua vez, podem ser compostas por diversos transformadores, que são agrupamentos de parâmetros como algoritmos criptográficos, modo de operação, tempo de vida[5] e grupo específico para o algoritmo de Diffie-Hellman, utilizado na troca de informações secretas. Cada proposta refere-se a um único protocolo de segurança (AH ou ESP). Por exemplo, a Tabela 4.3 exibe a descrição de uma proposta para o estabelecimento de uma AS para o protocolo ESP.

Ao receber tais dados, o *responder* deve selecionar uma das linhas da Tabela 4.3 contendo o conjunto de parâmetros de sua preferência e informar ao *initiator* a respeito de sua escolha. É importante notar que uma proposta pode conter qualquer tipo de agrupamento de parâmetros. Neste exemplo, algoritmos criptográficos mais frágeis (Capítulo 5) estão associados a tempos de vida menores. Uma outra proposta poderia utilizar tempos de vida iguais para todos os algoritmos listados, reduzindo a segurança de potenciais ASs que pudessem ser estabelecidas com base nos conjuntos de parâmetros contendo algoritmos com chaves criptográficas menores. Além disso, tempos de vida próximos podem não ser suficientes para igualar a diferença de proteção provida pelos algoritmos relacionados em uma proposta.

A diferença entre os graus de proteção de cada transformador também pode ser identificada no parâmetro que indica o grupo Diffie-Hellman a ser utilizado no estabelecimento da IPSec SA. Por *default*, as chaves simétricas das IPSec SAs são resultado da combinação do segredo compartilhado durante a ISAKMP SA e de valores denominados por *nonces*, que devem ser gerados aleatoriamente pelo *initiator* e pelo *responder* para cada IPSec SA estabelecida. Desta forma, para evitar que o segredo compartilhado da ISAKMP SA seja utilizado para a criação de chaves criptográficas para diversas IPSec SAs, o *initiator*

[5]O tempo de vida no IPSec pode ser estabelecido com base em tempo, quantidade de *bytes* submetidos a proteção da associação de segurança ou ambos.

pode exigir que um novo segredo, específico para cada IPSec SA, seja trocado entre as duas entidades através do algoritmo de Diffie-Hellman. Para tal, é suficiente inserir no transformador onde esta troca é desejada um valor para o grupo Diffie-Hellman que deve ser utilizado na troca deste segredo adicional. Esta medida, que garante uma propriedade denominada de PFS (*Perfect Forward Secrecy*) [19], visa evitar que o comprometimento do segredo compartilhado de uma ISAKMP SA resulte na fragilização das chaves geradas para as IPSec SAs resultantes. Ainda no exemplo anterior, os transformadores 3 e 5, diferentemente dos demais, exigem que uma nova troca Diffie-Hellman seja executada, aumentando ainda mais a disparidade da segurança provida entre as opções oferecidas.

Sob ponto de vista da segurança, o perigo de uma proposta conter transformadores com parâmetros tão distintos entre si é a iminência do estabelecimento de uma AS cuja capacidade de proteção pode variar, de maneira transparente ao usuário, significativamente. Por exemplo, se a proposta da Tabela 4.3 fosse utilizada para proteger o tráfego de um serviço que requer alta confidencialidade, haveria a possibilidade dos transformadores 1 ou 2, mais frágeis, serem selecionados pelo *responder*, o que poderia comprometer os requisitos de segurança do serviço em questão.

Propostas para protocolos distintos que contêm a mesma identificação compõem uma *SA bundle* e, portanto, caso seja selecionada uma proposta deste tipo, o *responder* deve escolher um transformador para cada protocolo de segurança da proposta. Na Tabela 4.4 são apresentadas, como exemplo, três propostas onde uma exige a criação de somente uma AS para o ESP, enquanto que as outras duas exigem a criação de duas ASs, uma para o AH e outra para o ESP. Caso a primeira seja selecionada, o seu único transformador, que propõe o uso do ESP somente com o DES, deve, automaticamente, ser utilizado na criação da AS. Caso a segunda proposta seja selecionada, o *responder* deverá, obrigatoriamente, selecionar o MD5 para o AH e escolher entre o DES e o 3DES/MD5 para o ESP. Finalmente, se as duas anteriores são preteridas em relação à terceira, o *responder* deverá optar pelo SHA-1 ou RIPEMD para a criação da AS do AH e pelo Rijndael[6]/SHA-1, Twofish[7]/RIPEMD ou IDEA/RIPEMD para a AS correspondente ao ESP.

Neste exemplo, apesar das propostas 2 e 3 protegerem os pacotes através do AH e do ESP, a primeira oferece somente o ESP e sem qualquer serviço de autenticação e integridade e, caso seja selecionada, certamente a AS resultante será menos segura que qualquer combinação dos transformadores das outras propostas.

As propostas e seus respectivos transformadores devem ser arranjados de acordo com a ordem de preferência preferida pelo *initiator*. Desta forma, é possível que o *responder* escolha, dentro de suas possibilidades, o conjunto de parâmetros mais adequados para ambos os extremos.

[6]Utilizando chaves de 192 *bits*.
[7]Utilizando chaves de 128 *bits*.

Proposta	**Prot.**	**Transf.**	**Cifragem**	**Autent.**	**Modo**	**DH**	**Tempo**	
							s	*bytes*
1	ESP	1	DES	–	Transp.	–	1000	–
2	AH	1	–	MD5	Transp.	–	200	250
	ESP	1	DES	–	Transp.	–	350	500
		2	3DES	MD5	Transp.	2	700	500
3	AH	1	–	SHA-1	Transp.	–	1000	750
		2	–	RIPEMD	Transp.	–	1000	750
	ESP	1	Rijndael:192	SHA-1	Transp.	2	2500	3000
		2	Twofish:128	RIPEMD	Transp.	1	1500	2000
		3	IDEA	RIPEMD	Transp.	5	1500	2000

Tabela 4.4: Exemplo de propostas para o estabelecimento de uma associação de segurança para os protocolos AH e ESP.

A composição das propostas é baseada em filtros que fazem parte da configuração de cada implementação do IKE. No momento em que o IPSec detecta a necessidade de iniciar o processo de estabelecimento de ASs, o IKE é acionado e, por sua vez, consulta suas configurações para determinar os parâmetros que devem ser utilizados na criação das propostas.

Capítulo 5

Algoritmos criptográficos de chave secreta

Apesar do IPSec oferecer mecanismos para a proteção de pacotes IP através da autenticação, integridade e confidencialidade, o grau de segurança desses serviços depende, fundamentalmente, dos parâmetros negociados para as associações de segurança, dentre os quais os algoritmos criptográficos utilizados pelos cabeçalhos AH e ESP.

Prover sigilo ao conteúdo de pacotes de um determinado serviço através do DES, que utiliza chaves de 56 *bits* certamente não representa a mesma segurança, caso o 3DES, com 192 *bits* de chave, fosse utilizado. De outra forma, a confidencialidade de um pacote provida por um algoritmo com chaves de 128 *bits*, com diversas vulnerabilidades conhecidas, pode ser menos seguro que o DES. Para finalizar, o uso de um algoritmo considerado seguro, com chaves resistentes a ataques de força-bruta, não representa muita proteção, caso as chaves criptográficas possam ser obtidas por um atacante.

Os algoritmos de chave pública, também conhecidos como algoritmos de chave assimétrica, são utilizados pelo IKE durante o processo de negociação das ISAKMP SAs para a troca dos segredos compartilhados entre as entidades, quando o método de autenticação é feito através da cifragem com chave pública ou de certificados digitais. Em seguida, para proteger as mensagens restantes do estabelecimento das ISAKMP SAs, as mensagens da negociação das IPSec SAs e os pacotes protegidos pelos cabeçalhos AH e ESP, algoritmos de chave secreta (ou algoritmos de chave simétrica), são negociados.

A principal característica dos algoritmos simétricos, capazes de prover autenticação, integridade e confidencialidade, é a utilização de uma chave criptográfica única que deve ser compartilhada entre as duas entidades protegidas por seus serviços. Em outras palavras, ambos os extremos podem oferecer o mesmo conjunto de serviços ao outro. Em relação aos algoritmos assimétricos, o único serviço que os simétricos não podem oferecer é o não-repúdio, que previne que uma entidade negue o envio de um determinado conjunto

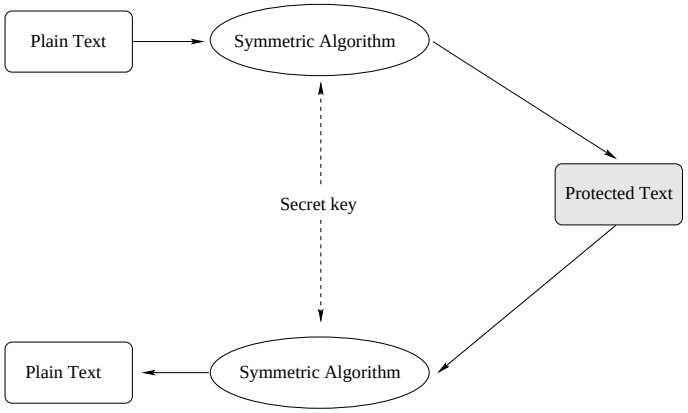


Figura 5.1: Esquema de funcionamento dos algoritmos de cifragem de chave secreta.

de dados, a menos que sua chave criptográfica tenha sido comprometida [43].

Parte deste trabalho concentra-se na racionalização de IPSec SAs para proteger o tráfego de aplicações. Desta forma, dado que este processo envolve o uso de algoritmos de chave secreta, neste capítulo serão apresentadas as características básicas dos principais representantes desta espécie de algoritmos criptográficos que, por sua vez, serão utilizados na composição do modelo a ser apresentado como resultado deste trabalho.

5.1 Algoritmos de cifragem

Representantes mais comuns dos sistemas de chave secreta, os algoritmos simétricos de cifragem têm o objetivo de impedir o acesso à informações por parte de terceiros que não possuem a chave criptográfica pré-estabelecida entre duas entidades. Muitas vezes, este tipo de algoritmo confunde-se com a própria classe de sistemas de chave secreta.

Ao receber como entrada uma cadeia de caracteres e a chave, o algoritmo de confidencialidade produz o texto cifrado. O resultado do processo inverso, onde as entradas do algoritmo são o texto cifrado e a chave, corresponde ao procedimento necessário para a obtenção do texto original. Este processo está ilustrado na Figura 5.1. É importante notar que, de acordo com a definição de algoritmo de chave simétrica, as operações de cifragem e decifragem são realizadas com base na mesma chave. Portanto, utilizar uma outra chave para a operação de decifragem, por exemplo, resultará, provavelmente, em uma cadeia de *bits* sem qualquer significado ou relação com o texto original.

Os algoritmos simétricos de confidencialidade são classificados em dois tipos básicos: os cifradores de bloco (*block ciphers*), que efetuam as operações de cifragem e decifragem com base em porções de *bits* de tamanho fixo (em geral, 64 *bits*) extraídas do texto original;

e os cifradores de cadeia (*stream ciphers*), cuja unidade de cifragem não possui tamanho fixo. Em especial, os algoritmos de confidencialidade utilizados pelo ESP são baseados em blocos, combinados através do modo CBC (*Cipher Block Chaining*), onde a versão cifrada de um bloco é utilizada na cifragem do bloco seguinte [56]. A seguir são mostradas as formulações que representam o processo de cifragem e decifragem no modo CBC:

$$C_i = E_k(P_i \oplus C_{i-1})$$
$$P_i = C_{i-1} \oplus D_k(C_i)$$

onde P_i e C_i são a versão original e cifrada do bloco i, respectivamente, e $E_k(M)$ e $D_k(M)$ indicam a cifragem e decifragem da mensagem M com a chave k, respectivamente. Antes da operação de cifragem propriamente dita do bloco i, ele é submetido a uma operação lógica XOR com a versão cifrada do bloco $i-1$, C_{i-1}. Da mesma maneira, para obter a versão original do bloco i, P_i, após a sua decifragem, este também é submetido, em conjunto com C_{i-1}, à operação XOR.

Dado que o primeiro bloco, P_1, não possui antecessor, ele necessita de um valor para utilizar como sendo um "P_0 virtual". Este valor, que deve ser gerado aleatoriamente, denominado IV (*Initialization Vector*), também tem como função impedir que dados iguais sejam cifrados para cadeias iguais. Por exemplo, o fato de pacotes possuirem campos dos protocolos de transporte UDP e TCP, que são imutáveis ou razoavelmente previsíveis, e a utilização de um valor fixo para composição do primeiro bloco cifrado resultariam na geração de blocos iniciais semelhantes para os pacotes com os mesmos valores iniciais dos protocolos de transporte[1]. Tal fato facilita o processo de criptoanálise de determinados tipos de tráfego. Portanto, a utilização de um valor inicial aleatório pode fazer com que algoritmos de chave secreta no modo CBC, utilizando cadeias e chaves criptográficas iguais, produzam blocos cifrados distintos entre si.

Uma propriedade interessante dos algoritmos em modo CBC é a capacidade de limitar a propagação de erros causados por blocos transmitidos ou recebidos incorretamente. Por exemplo, se um determinado bloco cifrado, C_i, é transmitido incorretamente, somente a recuperação de P_i e P_{i+1}, que dependem de C_i, será prejudicada. A decifragem do próximo bloco, P_{i+2}, não será afetada, dado que a operação depende somente de C_{i+1} e C_{i+2}, transmitidos e recebidos corretamente [56, 44]. Esta propriedade é muito atrativa para protocolos como o IPSec, cujos pacotes estão sujeitos a problemas no decorrer da transmissão.

[1]Tratando-se do modo de tunelamento, os campos do protocolo IP poderiam ser utilizados para a previsão de parte da cadeia de *bytes* não-cifrada.

5.1.1 DES

Surgido da necessidade da indústria em adotar uma solução única para a cifragem de dados, o DES (*Data Encryption Standard*)[2] [56, 44, 43] foi regulamentado em 1974 pelo NIST (*National Institute of Standards and Technology*) como algoritmo padrão do governo norte-americano para prover confidencialidade a informações diversas.

O DES foi desenvolvido pela IBM baseado em um algoritmo anterior conhecido como Lucifer e, antes de efetivamente ser padronizado, mediante solicitação do NIST, foi analisado pela NSA (*National Security Agency*) e levemente modificado. Além de especulações que dão conta de que as alterações feitas pela NSA foram formas de inserir *trapdoors*[3] no algoritmo ou ainda uma garantia de que a própria IBM não teria inserido tais artíficios, a mudança mais significativa foi a redução do tamanho da chave original de 128 *bits* para 56 *bits* [43].

Diversas implementações do DES exigem chaves de 64 *bits*, porém somente os 56 *bits* mais significativos são considerados. Os demais são utilizados como *bits* de paridade. Operando sobre blocos de 64 *bits*, as funções que compõem este algoritmo restringem-se ao uso de operações aritméticas e lógicas simples, o que facilitou o projeto de sua implementação tanto em *hardware* quanto em *software*.

Seja pela sua popularidade, seja pelo fato de ter-se tornado um padrão do governo norte-americano, certamente o DES é um dos algoritmos simétricos de cifragem mais submetidos a estudos por parte de criptoanalistas. Apesar de algumas vulnerabilidades terem sido detectadas, sua principal fragilidade, atualmente, está no tamanho da chave criptográfica. Em outras palavras, o DES pode ser submetido a ataques de força-bruta [56], onde todas as 2^{56} chaves possíveis são testadas até que seja encontrado o valor utilizado na cifragem e decifragem de uma determinada mensagem.

O ataque de força-bruta está intimamente relacionado ao poder computacional vigente. Por exemplo, o tempo necessário para explorar as chaves de algoritmos com 128 *bits* pode ser, atualmente, inviável para a maioria das organizações mundiais. Por outro lado, diferentemente do que se poderia imaginar na década de 70, com investimentos relativamente baixos é possível explorar o DES e violar o conteúdo de mensagens sob sua proteção. A Tabela 5.1 mostra os custos e os tempos estimados para obter a chave criptográfica do DES apresentados em [15].

Nas estimativas que contêm dois tempos, o maior está estimado com base na construção de estruturas baseada em FPGA (*Field Programmable Gate Array*), *chips* programáveis de baixo-custo que podem otimizar ataques de força-bruta. O tempo mínimo

[2]Na literatura, também é comum referenciar o DES por DEA (*Data Encryption Algorithm*), denominação adotada pela ANSI, ou ainda por DEA-1, denominada pela ISO.

[3]Técnicas para a recuperação, a partir do texto cifrado, do texto original sem a necessidade de utilizar a chave criptográfica.

Custo	Tempos para obtenção da chave
U$ 400	38 anos
U$ 10.000	18 meses
U$ 300.000	3 h – 19 dias
U$ 10.000.000	6 min – 13 h
U$ 300.000.000	12 s

Tabela 5.1: Estimativa de 1996 para a recuperação de chaves do DES com 56 *bits*.

é baseado em *hardwares* construídos especificamente para operações de decifragem. Em relação às demais medidas, as duas primeiras são baseadas somente em FPGA e a última somente em *hardware*. É importante notar que as estimativas apresentadas foram coletadas em 1996 e, segundo a lei de Moore [56], que afirma que o poder computacional dobra sua capacidade em, aproximadamente, 18 meses, em poucos anos o DES pode tornar-se um algoritmo de cifragem exposto a ataques de força-bruta advindos de computadores pessoais comuns.

5.1.2 3DES

Tendo em vista a fragilidade do DES em relação a ataques de força-bruta, na tentativa de adotar uma nova solução, a combinação de sucessivas aplicações do DES a uma mesma mensagem foi a maneira encontrada para prolongar a sua vida útil.

A primeira solução, o *Double* DES, previa a proteção de mensagens através de duas cifragens consecutivas do DES utilizando duas chaves criptográficas distintas de 56 *bits* cada. Porém, esta solução mostrou-se vulnerável a um ataque conhecido como *meet-in-the-middle*, através do qual é possível reduzir o número de tentativas para a recuperação de uma chave criptográfica qualquer de 2^{112} para 2^{57}, ou seja, somente o dobro de possibilidades em relação ao DES original [56, 67]. Apesar desta vulnerabilidade ter sido detectada sobre o *Double* DES, ela pode ser generalizada para qualquer algoritmo de cifragem em bloco, reduzindo o espaço de chaves de 2^{2n} para 2^{n+1}, onde n é o número de *bits* da chave no algoritmo original.

Na tentativa de definir uma solução mais segura que a anterior, foi proposto o *Triple* DES (ou simplesmente 3DES), que consistia na aplicação de três operações do DES utilizando-se duas chaves. Contudo, apesar de ser imune ao *meet-in-the-middle* original, foram detectadas maneiras para facilitar a recuperação de chaves, o que culminou na proposição do 3DES utilizando três chaves criptográficas, totalizando uma chave efetiva de 168 *bits*. Apesar de ser computacionalmente mais custoso que o algoritmo original, o 3DES passou a ser o algoritmo padrão *de facto* para a proteção de informações cuja

garantia de sigilo deveria ser superior àquela provida pelo DES.

É importante notar que ambas as versões do 3DES possuem mecanismos que garantem a compatibilidade com o DES original através da combinação dos valores das chaves. No 3DES com duas chaves, é suficiente fazer $K_1 = K_2$. No 3DES com três chaves, a compatibilidade é mantida através da combinação $K_1 = K_2$ ou $K_2 = K_3$ [67].

5.1.3 IDEA

Utilizando chaves criptográficas de 128 *bits* e blocos de 64 *bits*, o IDEA (*International Data Encryption Algorithm*) [56, 44, 67] foi desenvolvido em 1990 como uma alternativa para substituir o DES em uma das suas revisões periódicas.

Avanços significativos na análise diferencial da primeira versão do IDEA, denominada PES (*Proposed Encryption Algorithm*), fizeram com que seus autores elaborassem uma atualização, o IPES (*Improved Proposed Encryption Algorithm*), segura contra este tipo de ataque. Finalmente, em 1992, o nome IDEA foi adotado.

Resultados práticos têm mostrado que o IDEA é, aproximadamente, duas vezes mais veloz que o DES [56, 75]. Aliado a este fato, as chaves de 128 *bits*, mais seguras contra ataques de força-bruta em relação aos 56 *bits* utilizados pelo DES, tornam o IDEA uma alternativa mais segura para a proteção de informações que requerem mais segurança do que o DES é capaz de prover.

Apesar de terem sido obtidos alguns avanços na criptoanálise de sua última versão, a única possibilidade de explorar facilmente o IDEA seria através do uso de um conjunto de chaves criptográficas frágeis que permitem sua rápida recuperação por parte de um atacante. Por outro lado, a possibilidade de escolha de uma chave deste conjunto é de somente $\frac{1}{2^{96}}$ e, além disso, o algoritmo pode ser modificado de forma a evitar a geração destas chaves [56]. Portanto, atualmente, não existe nenhum tipo de ataque mais eficiente que o de força bruta para violar os dados sobre a proteção do IDEA [44].

A exposição a estudos de criptoanálise e a ausência da descoberta de vulnerabilidades significativas contribuíram para popularização do IDEA no decorrer dos anos. Como exemplo, o algoritmo é utilizado, juntamente com o CAST-128 e o 3DES, pelo PGP (*Pretty Good Privacy*) [43, 67], um sistema amplamente utilizado principalmente para a troca segura de mensagens através do serviço de correio eletrônico. Uma desvantagem do IDEA em relação ao DES, por exemplo, é o fato do seu uso ser patenteado e requerer licensa para uso comercial.

5.1.4 Blowfish

Desenvolvido em 1994 por Bruce Schneier, o Blowfish [56, 67] utiliza blocos de 64 *bits* para as operações de cifragem e decifragem. Porém, diferentemente da maioria dos algo-

ritmos criptográficos simétricos anteriores a ele, o Blowfish é capaz de utilizar chaves com tamanhos diversos, de 32 a 448 *bits*, variando em palavras de 32 *bits*, o que pode prover diferentes graus de proteção, de acordo com o comprimento da chave utilizada.

O Blowfish, no decorrer dos anos, vem sendo submetido a diversas criptoanálises. Porém, apesar da detecção de chaves frágeis, que permitem a descoberta das suas presenças e não do valor específico em versões do Blowfish com um número reduzido de iterações (*rounds*)[4], vulnerabilidades que podem resultar na rápida descoberta das informações sob sua proteção não foram, até então, encontradas. Além disso, o fato do desempenho deste algoritmo ser bastante satisfatório em relação a outras soluções como o DES (e conseqüentemente o 3DES), o RC5 e o IDEA, o Blowfish tornou-se popular, sendo, atualmente, amplamente utilizado em diversos produtos e aplicações [56].

No IPSec, o Blowfish pode ser utilizado com diversos tamanhos de chaves. Porém, por *default*, 128 *bits* são utilizados quando nenhum valor específico é indicado [19].

5.1.5 CAST-128

O CAST, cujo nome é derivado das iniciais dos nomes de seus autores, Carlisle Adams e Stafford Tavares, representa, na verdade, um conjunto de algoritmos [67].

Dentre os diversos algoritmos, o CAST-128 [1, 67], mais popular entre os demais, é de interesse deste trabalho, dado o grau de segurança provido por seu serviço de cifragem. Apesar de ser patenteado pela *Entrust Technologies*, a licensa garante a legalidade de sua utilização em qualquer tipo de aplicação. O CAST-128 é baseado em blocos de 64 *bits* e o comprimento de suas chaves pode variar de 40 a 128 *bits* com incrementos de *8 bits* [1].

Embora menos utilizado que outros algoritmos mais populares, como o IDEA e o Blowfish, por exemplo, criptoanálises do CAST-128 não têm constatado vulnerabilidades capazes de por em risco sua segurança, o que faz hoje deste algoritmo uma solução segura, quando utilizado com chaves de tamanhos adequados segundo os requisitos de segurança especificados. Em decorrência disto, alguns produtos e aplicações, dentre as quais o PGP, conforme mostrado na Subseção 5.1.3, utilizam o CAST-128 como opção para a execução do serviço de confidencialidade.

Em relação ao IPSec, o CAST-128 é utilizado com chaves de 128 *bits* e 16 iterações[5] [19].

5.1.6 AES (Rijndael)

O projeto de adoção do DES, datado de 1977, como algoritmo padrão do governo norte-americano, previa a realização de revisões periódicas a cada cinco anos, para que fosse

[4] O número normal de *rounds* do Blowfish é 16.

[5] O número de iterações pode variar de acordo com o tamanho da chave utilizada.

avaliada a possibilidade de sua manutenção ou não como solução padrão para a cifragem de informações.

As revisões de 1983, 1987 e 1993[6] deram continuidade ao uso do DES. Contudo, sua fragilidade em relação aos ataques de força-bruta fez com que o NIST desse início ao processo de adoção de um novo algoritmo, a ser denominado por AES (*Advanced Encryption Standard*), para substituir o DES.

Dentre os requisitos para o novo padrão, o NIST recomendou o uso de chaves de, no mínimo, 128 *bits* e a possibilidade de utilização de chaves maiores, de 192 e 256 *bits*. Após selecionar cinco candidatos como finalistas, o Rijndael [11] foi homologado como o AES, através de critérios baseados em sua segurança, eficiência (em *hardware* e *software*), pré-requisitos de funcionamento e portabilidade.

Apesar do DES ser o algoritmo de cifragem obrigatório a todas as implementações do IPSec, espera-se que, com a definição do AES, o DES seja oficialmente substituído pelo novo padrão [19] em documentações futuras do IPSec, a exemplo do que deverá acontecer com todas as aplicações, protocolos e equipamentos que adotam o DES como solução. Vale ressaltar que, apesar de substituir o DES, o 3DES deverá ainda ser mantido como solução alternativa pelo governo norte-americano [43].

5.1.7 Outros finalistas ao AES

Além do Rijndael, outros quatro algoritmos foram classificados como finalistas ao AES: o MARS, o RC6, o Serpent e o Twofish.

É importante notar que nenhum dos finalistas foi desclassificado por questões de segurança. Em outras palavras, segundo os estudos de criptoanálises realizados, todos os algoritmos "parecem prover segurança adequada ao AES" [43] e suportam, de acordo com as exigências do NIST, chaves de 128, 192 e 256 *bits*. Desta forma, tais algoritmos podem ser considerados alternativas seguras para a cifragem de informações.

Em relação ao desempenho, todos os finalistas são mais rápidos que o DES, com exceção do Serpent, que possui eficiência comparável à do antigo padrão de cifragem. Vale ressaltar, porém, que as medidas de desempenho podem conter variações significativas, de acordo com a plataforma considerada. Isto se deve à facilidade de execução, por parte de cada arquitetura utilizada, das operações básicas contidas nos algoritmos. Um estudo detalhado a respeito da comparação dos tempos de execução dos candidatos ao AES pode ser encontrada em [59].

Apesar de ser possível utilizar qualquer um dos algoritmos anteriores como mecanismo para a cifragem do conteúdo de pacotes através do ESP, nem todos possuem características

[6]Neste ano o uso do DES foi aprovado por mais cinco anos e tinha como previsão que sua vida útil iria expirar no máximo em 1992. Porém, em 1993 seu uso foi novamente autorizado por mais um período de cinco anos.

suficientes para ser amplamente adotados nas diversas implementações do IPSec. Dado que um *gateway* ou roteador com suporte ao IPSec poderá manipular uma grande quantidade de associações de segurança, é necessário que os algoritmos de cifragem utilizados sejam agéis o suficiente para suportar a rápida troca de parâmetros de segurança. Este requisito torna-se ainda mais crítico quando se leva em consideração que a maioria dos pacotes IP são pequenos (e.g. 64 *bytes*), exigindo trocas mais freqüentes de contexto. Dentre os finalistas, o RC6 e o MARS possuem dificuldades para sistemas IPSec de alto-desempenho [73].

Diferentemente de outras tecnologias para a transmissão segura de informações, o desempenho dos algoritmos utilizados com o IPSec ganha maior importância pelo fato de sua proteção poder ser aplicada a todos os tipos de serviços. Outras soluções, em sua maioria, possuem abrangência menor. O SSL, por exemplo, é utilizado, basicamente, para o comércio eletrônico.

5.2 Algoritmos de autenticação e integridade

Atualmente, é comum encontrar a definição de sistemas de chave secreta atrelada somente ao serviço de cifragem. Contudo, é possível utilizar os sistemas simétricos para prover autenticação e integridade. Por exemplo, no IPSec, quando dois extremos de uma comunicação estabelecem uma associação de segurança para o AH ou ESP com os serviços de autenticação e integridade habilitados, uma chave única deverá ser definida para prover tais serviços.

Para garantir integridade, são utilizados algoritmos de *hash* (ou *one-way hash functions*) onde, dado um conjunto de *bytes* de tamanho qualquer como entrada, uma cadeia de *bits* de tamanho fixo (comumente denominada por *hash*), que representa a entrada, é produzida como saída. Como requisito primordial para as funções de *hash*, é necessário que seja computacionalmente inviável recuperar a cadeia original somente com base no seu *hash*. Além disso, a mínima alteração no valor de entrada deve produzir mudanças significativas no *hash* resultante, impedindo o relacionamento do *hash* de entradas muito semelhantes.

Como a cadeia de *bytes* inicial não possui tamanho fixo, é suficiente pensar que há, matematicamente, a possibilidade de que entradas distintas produzam o mesmo *hash*, resultando em uma situação denominada de colisão [56]. A dificuldade de se encontrar métodos eficientes para a descoberta de colisões é um dos principais fatores levados em consideração na análise de segurança de uma função de *hash*. Uma maneira de reduzir a possibilidade de detecção de colisões para uma dada mensagem é concatená-la com o seu tamanho antes de produzir o *hash*, reduzindo a possibilidade do mesmo *hash* ser gerado por duas mensagens de tamanhos diferentes [19].

Função de *hash*	**Megabytes/s**
DES	7
MD5	57
SHA-1	25
RIPEMD-160	24

Tabela 5.2: Comparação entre a velocidade de *bytes* processados pelos principais algoritmos de *hash*.

5.2.1 MD5

Um dos mais populares e utilizados algoritmos para a geração de *hashes*, o MD5 [54, 56, 67, 43], foi desenvolvido por Ron Rivest como uma atualização do algoritmo antecessor, o MD4.

A cadeia de *bytes* de entrada, que pode ser de qualquer extensão, é processada pelo MD5 em blocos de 512 *bits* segregados em 16 sub-blocos de 32 *bits*, gerando como resultado um *hash* de 128 *bits*. Em relação aos outros algoritmos de *hash* utilizados atualmente, o MD5 tem como principal diferencial a sua eficiência. A Tabela 5.2, apresentada em [43], mostra uma comparação entre os principais algoritmos de *hash* utilizados atualmente.

Apesar de ser mais veloz em relação as outras soluções, nos últimos anos, diversas vulnerabilidades foram detectadas no MD5. Dentre as principais fragilidades atuais estão a exploração da função de compressão [13], que permite a detecção de colisão para uma determinada mensagem, e o ataque de força-bruta, que viabiliza ataques de aniversário através de 2^{64} tentativas, passíveis de ser exploradas por sistemas computacionais contemporâneos. Desta forma, caso haja a necessidade de garantir a autenticação e integridade de determinadas informações por um período indeterminado, o uso de outras funções de *hash*, como o SHA-1 e o RIPEMD-160, é mais adequado, tendo em vista os problemas de segurança do MD5. Por outro lado, informações cuja proteção pode ser feita através da utilização dos serviços de autenticação e integridade por períodos curtos de tempo, compatíveis com os tempos necessários para a aplicação de algum dos ataques conhecidos do MD5, podem fazer uso desse algoritmo, dada a sua diferença significativa de desempenho em relação às outras funções de *hash*.

5.2.2 SHA-1

O SHA-1 (*Secure Hash Algorithm*) [56, 67, 43], da mesma forma que o MD5, é baseado na estrutura do MD4. Desenvolvido em 1993 pelo NIST (*National Institute os Standards and Technology*), o SHA-1 tornou-se um padrão federal do governo norte-americano para

a geração de *hashes* criptográficos.

A cadeia de entrada, cujo tamanho não deve exceder $2^{64}-1$ *bits*, é processada em blocos de 512 *bits*, gerando um *hash* de 160 *bits* de comprimento. Atualmente, não foram detectadas vulnerabilidades graves no SHA-1 e, portanto, ele parece representar uma solução segura para a produção de *hashes* criptográficos. Em relação ao ataque de aniversário, 2^{80} tentativas seriam necessárias, o que resulta em um esforço 65.536 ($2^{80-64} = 2^{16}$) maior que aquele dispendido para efetivar o mesmo ataque sobre uma função de *hash* de 128 *bits*, como o MD5, por exemplo.

A especificação do AES fez com que surgisse a necessidade do desenvolvimento de novas soluções, com graus de segurança compatíveis com o AES, para a geração de *hashes* criptográficos. Desta forma, recentemente, o NIST divulgou extensões do SHA-1: o SHA2-256, SHA2-384 e SHA2-512, capazes de produzir *hashes* mais seguros contra ataques de força-bruta, com 256, 384 e 512 *bits*, respectivamente. Os dois últimos podem receber como entrada mensagens de até 2^{128} *bits*.

Apesar de não existir qualquer tipo de restrição para a utilização destes novos algoritmos para prover autenticação e integridade com o IPSec, até o momento, não existem RFCs que documentem suas incorporações a este protocolo[7]. Por outro lado, a ausência de documentação não impede que plataformas contenham suportem a estes algoritmos.

5.2.3 RIPEMD-160

O RIPEMD-160 [67, 19] foi desenvolvido pelo projeto RIPE (*European RACE Integrity Primitives Evaluation*), coordenado por Hans Dobbertin, criptógrafo que conseguiu atacar o MD5 através da sua função de compressão. O RIPEMD-160, da mesma forma que o MD5 e o SHA-1, é, também, baseado no MD4 e pretendia tornar-se uma alternativa mais segura a este algoritmo.

Inicialmente, o RIPEMD-160 foi especificado para *hashes* de 128 *bits* e era denominado simplesmente por RIPEMD. Contudo, após Dobbertin detectar falhas de segurança na versão original, uma atualização foi feita no algoritmo a fim de torná-lo seguro contra os problemas encontrados. O funcionamento básico do RIPEMD consiste, fundamentalmente, em duas aplicações paralelas do MD5, diferindo-se na ordem de aplicação das suas funções internas.

De maneira geral, o RIPEMD-160 possui os mesmos parâmetros do SHA-1, utilizando blocos de 512 *bits* e produzindo *hashes* de 160 *bits*. Além disso, pelo fato de ambos os algoritmos gerarem *hashes* com o mesmo comprimento, as possibilidades de ataques de

[7]Na época da produção escrita deste trabalho, somente o SHA2-256 estava especificado através do *Internet Draft* "The HMAC-SHA-256-96 Algorithm and Its Use With IPsec" `<draft-ietf-ipsec-ciph-sha-256-00.txt>`. Contudo, como este tipo de documento não é definitivo, ele não está relacionado na seção de Referências Bibliográficas.

força-bruta possuem as mesmas características, sendo o RIPEMD-160, portanto, considerado uma alternativa segura no que diz respeito à produção de *hashes* criptográficos.

5.2.4 HMAC

Caso o conteúdo de um pacote seja dado como entrada para um algoritmo de *hash*, todos os seus dados estarão protegidos contra erros de transmissão e problemas decorrentes do processamento incorreto por parte de roteadores, pois se o destinatário recebe um pacote e verifica que o *hash* gerado não corresponde àquele recebido, certamente os dados sofreram algum tipo de alteração em trânsito. Contudo, alterações maliciosas, efetuadas por atacantes capazes de capturar determinados pacotes e reinjetá-los na rede, não podem ser detectadas pelo simples cálculo de um *hash* gerado na entidade de origem.

Ao capturar um pacote no intuito de realizar o *spoofing* de dados, para fazer com que a vítima processe-o como válido, é suficiente que o atacante, após efetivar as alterações desejadas, calcule novamente o *hash* com base no novo conteúdo do pacote antes de reinjetá-lo na rede. Desta forma, se nenhum problema de transmissão ocorrer, a vítima irá computar o *hash* com base no conteúdo do pacote alterado pelo atacante e, ao conferir que o valor produzido é o mesmo que aquele recebido com o pacote adulterado, este irá ser validado e repassado às camadas superiores, resultando no sucesso das ações do atacante.

Para fazer com que o *hash* calculado garanta a detecção de erros ocasionais bem como a alteração maliciosa do conteúdo de pacotes por parte de atacantes, os *hashes* devem ser calculados através do uso de uma chave criptográfica secreta, compartilhada entre as duas entidades que desejam garantir os serviços de autenticação e integridade aos seus pacotes. O *hash* com chave, comumente denominado de MAC (*Message Authentication Code*) [67, 56, 43], exige que um atacante, ao tentar modificar o conteúdo de um pacote, tenha a chave correta para gerar um MAC que seja considerado válido pelo destino.

Inicialmente, os MACs eram criados através do uso de algoritmos de cifragem de chave secreta, como o DES, por exemplo. Porém, por questões de desempenho, outras soluções que utilizam modificações em algoritmos de *hash* simples para que estes passassem a produzir MACs foram desenvolvidas. Em suas primeiras especificações, o AH e o ESP utilizavam versões adaptadas do MD5 e do SHA-1 para a incorporação de chaves, o Keyed-MD5 [45] e o Keyed-SHA [46], respectivamente, como algoritmos de implementação obrigatória. Porém, problemas de segurança encontrados no MD5 [13] que permitiam, através da resolução de algumas operações matemáticas, a descoberta de duas mensagens, diferentes somente em uma palavra, que produzem o mesmo *hash* levaram à adoção de outro mecanismo para a geração de MACs nos protocolos do IPSec, o HMAC (*Hashed MAC*) [38, 19, 67].

Desenvolvido como um *framework* capaz de utilizar implementações quaisquer de

funções de *hash* existentes, como o MD5 e o SHA-1, o HMAC possibilita a rápida substituição de uma função de *hash* com problemas de segurança por outra mais segura. Outra vantagem do HMAC está no fato do seu uso não aumentar significativamente o custo computacional da função de *hash* interna [19, 43].

Em geral, a nomenclatura do algoritmo HMAC indica a função interna de *hash* em uso. Por exemplo, HMAC-MD5 indica a composição do HMAC com a função de *hash* MD5. Em alguns casos, onde o tamanho do MAC produzido não corresponde àquele da função de *hash*, é comum indicar o comprimento da cadeia resultante. HMAC-MD5-96 indica que o MAC produzido é truncado para 96 *bits*, ao invés dos 128 produzidos pelo MD5. Outra nomenclatura usual é indicar o tamanho da chave a ser utilizada pelo algoritmo de *hash*, quando este é capaz de produzi-los em variados comprimentos. O RIPEMD, por exemplo, que foi projetado para gerar *hashes* de comprimentos distintos, ao ser incorporado no HMAC, deve indicar a sua versão. HMAC-RIPEMD-160-96 representa a combinação do HMAC com a especificação do RIPEMD-160, que produz *hashes* de 160 *bits* e cujo resultado será truncado para 96 *bits*.

No IPSec, a especificação exige o uso de chaves de 128 *bits* para o HMAC-MD5-96 e 160 *bits* para o HMAC-SHA1-96, o mesmo comprimento do HMAC-RIPEMD-160-96, que, apesar de não ser de implementação obrigatória, já tem seu uso no IPSec devidamente documentado [48].

Funcionamento básico

A produção de uma MAC pelo HMAC é composta pela incorporação de uma chave secreta à mensagem de entrada e pela dupla aplicação do função de *hash*, o que aumenta a dificuldade dos procedimentos de criptoanálise até mesmo em funções com vulnerabilidades conhecidas como o MD5 [19].

A Figura 5.2 mostra a estrutura de funcionamento do HMAC. Primeiramente, a chave é concatenada com zeros até gerar uma cadeia de *bits* cujo tamanho deve ser igual ao do bloco do algoritmo. Posteriormente, a chave criptográfica é submetida à operação lógica XOR com um valor fixo, *ipad*, resultando em Key 1, que, por sua vez, é concatenada com a mensagem de entrada e, em seguida, tal resultado é processado pela função de *hash* (e.g. MD5 ou SHA-1) específica utilizada, gerando um MAC inicial, *Preliminary keyed hash*. A chave secreta, da mesma forma que na geração de Key 1, é submetida novamente à operação XOR, porém, o segundo operando é um outro valor fixo, denominado *opad*. O resultado de tal operação resulta em Key 2, que é concatenada com *Preliminary keyed hash*, e tal composição é processada pela função interna de *hash*, gerando, enfim, o MAC final.

A definição do HMAC não limita o tamanho da chave secreta. Vale ressaltar, contudo, que, caso o tamanho da chave utilizada exceda o do bloco de operação da função de *hash*,

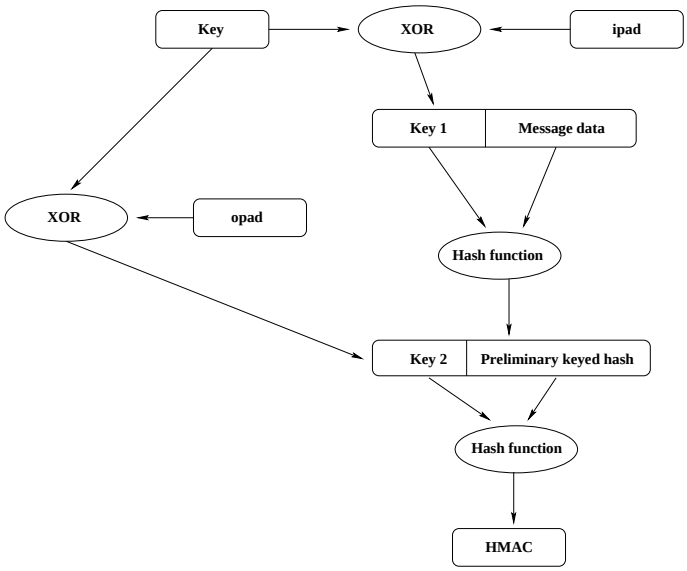


Figura 5.2: Esquema de funcionamento do algoritmo HMAC.

a primeira é submetida ao processamento por parte desta função e o resultado é utilizado como chave criptográfica. Por outro lado, por questões de segurança, chaves com tamanho inferior ao resultado produzido pela função de *hash* não devem ser utilizadas. Além disso, estudos mostram que o uso de chaves com comprimento superior ao da cadeia resultante da função de *hash* não contribui significativamente para aumentar a segurança do MAC resultante [19].

Capítulo 6

Security Level Model

Apesar do IPSec ser capaz de resolver muitos dos problemas detectados no IP ao longo dos anos, dada a ausência de serviços de autenticação, integridade e confidencialidade, seu uso, atualmente, é restrito a ambientes limitados, onde os extremos da comunicação segura são bem conhecidos. Segundo as próprias recomendações de Chapman [75] para a utilização do IPSec:

"IPSec is a good choice for building virtual private networks."

Certamente, a complexidade e a ampla flexibilidade deste protocolo são fatores limitantes no que diz respeito ao seu uso na proteção dos pacotes de variados tipos de serviços entre dois *hosts* quaisquer [17]. Utilizá-lo sem o conhecimento adequado de suas numerosas opções e características pode significar um risco à segurança de uma rede computacional [19]. É possível, contudo, fazer com que o IPSec vá além de um mecanismo para a implementação de VPNs, tornando-o parte efetiva e integrante de pacotes IP de qualquer natureza.

Reunidos os conceitos fundamentais apresentados nos capítulos anteriores, neste capítulo será apresentado o SLM (*Security Level Model*), desenvolvido como resultado deste trabalho. Na oportunidade serão elucidadas a motivação para o seu desenvolvimento, suas características e funcionamento.

6.1 Problemas no uso do IPSec

Variados algoritmos criptográficos e outros parâmetros podem ser utilizados pelo IPSec para prover proteção a pacotes IP. Contudo, para cada tipo de serviço que se deseja proteger, é necessário que se defina uma regra correspondente àquele tráfego na política de

segurança do IPSec, armazenada no SPD. Porém, nenhuma recomendação geral ou padronização determina quais alternativas utilizar para os diversos serviços a serem protegidos, muito menos o nível de granularidade destas regras.

É de responsabilidade do administrador de sistemas determinar como serão aplicados os serviços do IPSec. Da mesma forma que toda a comunicação (ou pelo menos uma parte dela) entre *hosts* quaisquer de uma rede pode ser protegida através de regras genéricas, abrangendo diversos tipos de serviços, em outro cenário, mais especializado, cada um dos serviços pode ser protegido segundo uma regra específica, contendo os parâmetros de segurança adequados aos seus requisitos de proteção, por exemplo.

No primeiro caso, onde ocorre uma simplificação da especificação da política de segurança e, conseqüentemente, da configuração do IPSec, corre-se o risco de sobrecarregar os *hosts* e a rede através do uso de algoritmos criptográficos computacionalmente custosos para todas as aplicações. Em outras palavras, até mesmo serviços cuja segurança poderia ser provida somente pelo AH com um algoritmo de autenticação rápido, mesmo que mais vulnerável que as outras opções, por exemplo, iriam ter seus pacotes "inchados" desnecessariamente pela inserção do cabeçalho ESP e acabariam por consumir mais recursos em conseqüência do processamento extra. Por outro lado, uma política genérica contendo apenas algoritmos atualmente vulneráveis a ataques de força-bruta (e.g. DES ou outro algoritmo com chaves pequenas) poderia colocar em risco as informações de serviços com maiores exigências de segurança. A "falsa sensação de segurança", provocada pelo fato de se ter o tráfego de determinados serviços protegido por algum conjunto de algoritmos, pode fazer com que os usuários, incondicionalmente, confiem na proteção aplicada, mesmo diante da possibilidade de tais escolhas permitirem ataques que podem resultar no *spoofing* de dados ou até mesmo na violação de informações altamente sigilosas.

Ainda sobre a definição de políticas demasiadamente abragentes, a combinação, em uma mesma regra, de algoritmos criptográficos com os mais variados graus de proteção é potencialmente perigosa sob o ponto de vista da segurança das informações. Por exemplo, da mesma maneira que o tráfego de uma determinada aplicação entre dois *hosts* pode estar sendo protegido pelo AH e pelo ESP através do SHA-1 e do Rijndael, a utilização deste mesmo serviço entre um dos *hosts* anteriores e um terceiro poderia ocorrer sob a proteção do MD5 e do DES, reconhecidamente mais vulneráveis que a primeira combinação. Tal anomalia é possível devido ao desnível dos parâmetros contidos na proposta feita pelo *initiator* ao *responder*, que, por não possuir implementada outra alternativa mais segura, ou mesmo por sua política dar preferência a um algoritmo mais fraco, seleciona, dentre as possibilidades, aquelas mais suscetíveis a ataques. De maneira geral, esta alternativa não garante a uniformidade do grau de proteção aplicado ao tráfego dos serviços.

No segundo caso, onde o administrador determina regras com parâmetros específicos de acordo com o seu julgamento dos requisitos de segurança de cada serviço, apesar de poder

contemplar, separadamente, a proteção adequada a cada aplicação, a complexidade para a especificação da política de segurança aumenta, especialmente tratando-se de ambientes onde variadas aplicações são utilizadas. A definição de regras para cada serviço exige, necessariamente, um conhecimento mínimo a respeito dos algoritmos criptográficos e dos outros parâmetros possíveis para o funcionamento do IPSec. Caso contrário, corre-se o risco de, a exemplo da política genérica com muitas opções, mostrada anteriormente, ocorrerem desigualdades no grau de proteção das associações de segurança de alguns serviços. Além disso, aplicações podem não ter seu tráfego protegido adequadamente em decorrência da escolha de opções inadequadas ou, ainda, das associações de segurança serem estabelecidas com graus de proteção e protocolos diferentes em ambos os sentidos da comunicação (ver Seção 4.2).

Outro fator que deve ser levado em consideração neste segundo cenário é que a grande quantidade de regras distintitas pode acarretar problemas para a administração das bases de dados da política de segurança, resultando, por exemplo, na construção de propostas incompatíveis. Dado que o SPD é específico para cada *host*, todos devem conter regras e parâmetros compatíveis com aqueles que compõem a política de segurança estabelecida. Caso contrário, se, por exemplo, um determinado serviço de um *host* estiver associado a uma regra cujos parâmetros diferem da política de segurança dos demais *hosts*, devido a problemas ocorridos na manutenção dos SPDs, seu uso pode ser inviabilizado, caso não existam valores comuns entre a regra deste *host* e a regra presente nos *hosts* restantes.

Existem ainda dificuldades de uso do IPSec para proteger serviços em uma rede no que diz respeito à implementação da política de segurança em plataformas distintas. Atualmente, com a grande quantidade de sistemas operacionais disponíveis e as diferentes características de cada um, é freqüente a construção de ambientes onde há a interação de variadas plataformas através de protocolos comuns. Da mesma maneira, em um ambiente que utilize o IPSec, para que os diferentes sistemas interajam entre si, faz-se necessária a representação da política de segurança em cada plataforma presente no ambiente computacional. Tal tarefa pode ser demasiadamente custosa para o administardor, dado que cada sistema com suporte ao IPSec possui uma linguagem específica para a configuração dos parâmetros deste protocolo.

6.2 Características gerais do SLM

Apresentadas as dificuldades detectadas para a utilização do IPSec como ferramenta de proteção do tráfego de aplicações em redes de computadores, propõe-se, neste trabalho, o SLM (*Security Level Model*), um modelo cujo intuito visa contribuir para a viabilização do uso do IPSec, fazendo-o ir além de uma ferramenta para a configuração de ambientes estáticos, como as VPNs tradicionais.

Os primeiros elementos para a definição do SLM foram apresentados em [60]. Nesta oportunidade definiu-se um mecanismo capaz de proteger o tráfego de serviços através de associações de segurança estabelecidas a partir de parâmetros agrupados em níveis. Com base nestas especificações, os demais componentes do SLM, seu funcionamento básico foram descritos em [61]. Em [62] estão elucidados os aspectos relativos a implementação do SLM e a maneira pela qual ele foi incorporado ao IPSec.

Dentre as principais características do SLM, estão:

- Prover estruturas que permitam ao administrador implementar a política de segurança de maneira rápida e ágil através de uma representação de alto-nível, independente de plataforma, evitando o uso de regras genéricas e baseando-se nos serviços que se almeja proteger;
- Esconder detalhes específicos de configuração do IPSec, tornando seu uso seguro, através do entendimento da descrição de agrupamentos de parâmetros, ao invés de exigir a compreensão do grau de proteção de algoritmos criptográficos, por exemplo;
- Centralizar a especificação da política de segurança, impedindo a existência de SPDs que estejam com parâmetros de configuração em desacordo e, conseqüentemente, mantendo em todos os *hosts* da rede um conjunto de regras compatíveis entre si;
- Impedir o estabelecimento de associações de segurança com disparidades entre os parâmetros de proteção e os protocolos de segurança selecionados para um mesmo serviço. Além disso, a proteção é sempre bidirecional, devido ao estabelecimento de pares de associações de segurança para cada protocolo;
- Permitir maior facilidade para a compatibilização de políticas de segurança de ambientes distintos.

O SLM insere-se como uma camada de interação entre o administrador de sistemas e o IPSec, conforme mostrado na Figura 6.1. Armazenando as informações em bases de dados específicas, o SLM traduz tais informações para o IPSec, que, por sua vez, armazena-as no SPD. Vale ressaltar, contudo, que o SLM não interfere na interação original existente entre o administrador e o IPSec, o que permite o uso deste protocolo de ambas as maneiras: diretamente ou através do SLM.

6.3 Níveis de segurança

Estruturas fundamentais para o SLM, os níveis de segurança [60, 61] são agrupamentos de parâmetros necessários para o estabelecimento de ASs, incluindo: protocolos de segurança,

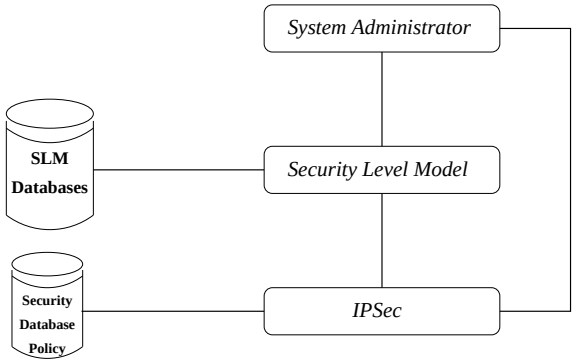


Figura 6.1: Inserção do SLM entre o administrador de sistemas e a implementação de IPSec.

algoritmos criptográficos, tamanho das chaves, grupo *Diffie-Hellman* para a garantia do PFS e tempo de vida das ASs. Os parâmetros de cada nível fazem com que o grau de proteção entre eles seja distinto.

O objetivo dos níveis de segurança é encapsular inúmeros detalhes de configuração do IPSec com graus de proteção semelhantes. Desta forma, a descrição de cada nível deve ser suficiente para o administrador compor as regras de sua política de segurança, abstraindo-se dos detalhes específicos encapsulados.

Quatro níveis, organizados em ordem crescente, conforme seus graus de proteção, foram definidos para o SLM , com base na oferta dos serviços de autenticação, integridade e confidencialidade, através de diferentes algoritmos criptográficos e demais parâmetros oriundos do IPSec. As especificações dos níveis de segurança definidos são apresentadas nas quatro subseções seguintes.

6.3.1 Nível *Unclassified*

Primeiro na hierarquia de níveis de segurança definida para o SLM, o *Unclassified* é o único que oferece somente os serviços de autenticação e integridade, não sendo, portanto, a confidencialidade um item necessário, até mesmo pela possibilidade de afetar negativamente o desempenho dos serviços a serem associados a este nível. Em outras palavras, a obtenção das informações contidas em pacotes sob a proteção deste nível não representam qualquer risco. Em relação ao IPSec, somente o AH é utilizado.

A autenticação e a integridade providas pelo nível *Unclassified* são garantidas por períodos curtos de tempo que devem ser suficientes para a transmissão, recepção e processamento imediato das informações. É importante notar, portanto, que, a proteção do nível pode ser violada após tempo factível dedicado a ataques de força-bruta.

No SLM, os serviços do *Unclassified* representam a mínima proteção a que qualquer tipo de tráfego deve ser submetido, em virtude de qualquer pacote IP sem o suporte do IPSec estar sujeito a *spoofing* de endereços e dados. Por esta razão, o SLM não possui um nível de segurança onde nenhum tipo de serviço é provido.

6.3.2 Nível *Confidential*

O nível *Confidential*, da mesma forma que o *Unclassified*, também provê serviços por períodos curtos de tempo. Contudo, além da autenticação e integridade, o *Confidential* oferece confidencialidade. Em outras palavras, serviços cujo tráfego possui requisitos de segurança compatíveis com o *Unclassified*, porém requerem o mínimo de garantia do sigilo de suas informações, podem ser associados ao *Confidential*.

Vale ressaltar que o sigilo, bem como os demais serviços por ele oferecidos, são vulneráveis a ataques de força-bruta, não sendo, portanto, adequados à proteção de informações com teor altamente sensitivo que podem, por exemplo, comprometer a segurança do ambiente. Por outro lado, informações que perdem sua importância após a transmissão e processamento e cuja violação posterior não oferece qualquer risco potencial podem estar vinculadas ao *Confidential*.

É importante notar que ataques aos dados de *Confidential* não podem ser efetuados de maneira rápida por qualquer tipo de atacante, sendo necessário, pelo menos, um investimento mínimo em algum tipo de tecnologia capaz de violar as informações em períodos de tempo factíveis. Desta forma, somente organizações de porte médio a grande poderiam investir na construção das infra-estruturas necessárias.

6.3.3 Nível *Secret*

Provendo autenticação, integridade e confidencialidade através de algoritmos criptográficos muito menos suscetíveis a ataques de força-bruta em relação aos utilizados por *Confidential*, o nível *Secret* é capaz de proteger informações cuja sensitividade vai além da sua transmissão e processamento.

Secret está posicionado entre o primeiro nível do SLM que provê os três serviços do IPSec (*Confidential*) e o último nível (*Top Secret*), mais seguro dentre os demais. Mais informações são apresentadas na Seção 6.4.

A violação de suas informações pode ser feita somente mediante altos investimentos, possíveis apenas a grandes organizações, como as agências governamentais de inteligência. Por outro lado, apesar de poder violá-los, os dados protegidos não devem possuir grande valor para tais organizações e sim a atacantes que, provavelmente, não terão condições de ter acesso aos mecanismos de exploração das informações.

A transferência de zonas de DNS, por exemplo, é um dos serviços que podem ser protegidos pelos parâmetros do nível *Secret*, pois, apesar de não representar uma informação muito valiosa que justifique altos investimentos, a obtenção das listagens dos *hosts* de um domínio pode ser um ponto de partida importante para a composição de um ataque.

6.3.4 Nível *Top Secret*

O nível *Secret* é o mais seguro em relação aos outros níveis especificados para o SLM. Nele, os serviços de autenticação, integridade e confidencialidade devem ser garantidos por períodos indeterminados de tempo, sendo imunes a tentativas de ataques oriundos de qualquer tipo de organização.

As informações cuja segurança for relacionada ao *Top Secret* devem representar riscos potenciais se violadas a qualquer momento, durante ou após sua transmissão. Aplicações que transferem senhas de usuários, por exemplo, são candidatas naturais a serem protegidas pelo *Top Secret*, que pode incluir, inclusive, aquelas que já possuem algum tipo de proteção através de algoritmos criptográficos. Isto se deve ao fato dos outros protocolos de proteção utilizados por tais aplicações não proverem qualquer segurança ao protocolo IP, o que pode permitir a realização de ataques sobre tais serviços. Mais informações são apresentadas na Seção 6.4.

6.3.5 A base de dados *Security Level Definition*

Todas as especificações referentes aos níveis de segurança do SLM são armazenadas em uma base de dados denominada por *Security Level Definition* [61, 62]. A seguir são apresentados os campos que a compõem:

- *slNumber*: identificador do nível de segurança;
- *secProtocol*: contém qual protocolo de segurança o nível utiliza para prover seus serviços, o AH, o ESP ou ambos;
- *authAlgorithm*: relaciona os algoritmos de autenticação a serem utilizados pelo nível de segurança na composição das propostas do AH ou do serviço opcional de autenticação e integridade do ESP. Algoritmos que suportam chaves de variados comprimentos devem conter o tamanho a ser considerado;
- *encAlgorithm*: relaciona os algoritmos de cifragem a serem utilizados pelo nível de segurança na composição das propostas do ESP. Da mesma forma que o campo anterior, algoritmos com suporte a chaves de comprimentos distintos devem possuir a discriminação do tamanho a ser considerado;

- *ltRule*: especifica a métrica de referência para o tempo de vida de cada associação de segurança estabelecida com base nos parâmetros do nível em questão. Seus valores podem ser *time* ou *byte*, indicando o uso de tempo ou quantidade de dados protegidos pela associação, respectivamente;
- *ltUnit*: a unidade de referência, segundo a métrica definida pelo campo anterior. Caso se trate de tempo, é possível definir como valores *sec*, *min* e *hour*. Por outro lado, tratando-se de *bytes*, pode-se utilizar *B* para *bytes* ou *MB* para *megabytes*;
- *ltValue*: contém o valor do tempo de vida maximo das associações de segurança do nível, segundo a métrica e a unidade especificadas pelos dois campos anteriores;
- *pfsGroup*: identifica o grupo Diffie-Hellman a ser utilizado na geração de um novo segredo para a composição das chaves criptográficas das IPSec SAs, evitando a utilização do segredo estabelecido na ISAKMP SA e garantindo o PFS. Caso algum dos níveis não requeira uma nova troca, é suficiente não inserir valor neste campo. Contudo, para incrementar a segurança, a especificação dos níveis feita neste trabalho exige o PFS para todos;
- *description*: provê uma breve descrição, utilizando-se, em geral, o nome por extenso relacionado ao nível de segurança (e.g. *Confidential.*).

As informações da base de dados *Security Level Definition* em relação às outras contidas no modelo, a serem apresentadas posteriormente, é a que deve possuir menos intervenção por parte do administrador. Isto se deve ao fato de que nela estão encapsulados os dados que o SLM tenta abstrair. Por outro lado, administradores com conhecimento suficiente sobre os parâmetros contidos em *Security Level Definition* podem alterá-lo conforme suas necessidades específicas. É importante notar, contudo, que modificações no *Security Level Definition* irão se refletir sobre todos os serviços associados àqueles níveis cujos parâmetros foram alterados.

Atribuição de algoritmos criptográficos e outros parâmetros aos níveis de segurança

Os principais componentes dos níveis de segurança são os algoritmos criptográficos e os respectivos tamanhos de suas chaves. Apesar do SLM, da mesma forma que o IPSec, não se restringir a um conjunto fixo e limitado de algoritmos, uma atribuição inicial é necessária para o funcionamento do modelo.

A Tabela 6.1 apresenta uma proposta para a alocação de algoritmos para compor os níveis de segurança cuja classificação foi realizada, fundamentalmente, com base em descrições relativas à segurança de cada solução. Neste caso, entende-se como métrica

Nível de Segurança	**Autenticação**		**Cifragem**	
	Algoritmo	**Chave**	**Algoritmo**	**Chave**
Top Secret	HMAC-SHA2-512-96	512	Rijndael Twofish Serpent Cast Blowfish	256 256 256 256 256
Secret	HMAC-SHA2-384-96 HMAC-SHA2-256-96	384 256	Rijndael Twofish Serpent Idea Blowfish Cast 3DES	128/192 128/192 128/192 128 128 128 168
Confidential	HMAC-SHA-1-96 HMAC-RIPEMD160-96	160 160	Rijndael Twofish Serpent Idea Blowfish Cast DES	128 128 128 128 128 128 56
Unclassified	HMAC-MD5-96	128	Não aplicável	

Tabela 6.1: Proposta de distribuição de algoritmos criptográficos nos níveis de segurança.

de segurança o tempo de resistência a ataques de força-bruta, medido conforme o comprimento da chave utilizada. Nenhum dos algoritmos utilizados possui resultados significativos de estudos de criptoanálise que os tornem vulneráveis a outros ataques mais eficientes que o de força-bruta, com exceção do DES. Além disso, não foram encontradas comparações a respeito do grau de proteção de algoritmos considerados seguros utilizando um mesmo tamanho de chave. Em outras palavras, a ausência de vulnerabilidades resulta em indícios de que um determinado algoritmo é seguro. Desta forma, sob o ponto de vista da segurança, dentre os algoritmos selecionados para a composição da proposta, aqueles que possuem o mesmo tamanho de chave são considerados como tendo o mesmo grau de proteção. De qualquer maneira, avanços comprometedores na criptoanálise de um dos algoritmos utilizados podem resultar na sua remoção de alguns níveis, na sua alocação a níveis inferiores ou até mesmo na sua exclusão do modelo.

Os tempos de vida das associações de segurança estabelecidas por cada nível foram definidos como 2 horas para o *Top Secret* e *Secret*, 5 horas para o *Confidential* e 12 horas para o *Unclassified*. Da mesma forma que os parâmetros que compõem cada um dos níveis estes valores também são passíveis de alteração, de acordo com observações futuras que possam vir a identificar ajustes que produzam melhorias significativas na segurança ou desempenho do modelo.

Ainda na proposta apresentada, existem algoritmos de cifragem com o mesmo tamanho de chave distribuídos nos níveis *Secret* e *Confidential*. Observe que chaves de 128 *bits*

Algoritmo de cifragem	**Comprimento da chave**	**Tempo**
DES	56 *bits*	35,64s
3DES	192 *bits*	61,44s
Blowfish	256 *bits*	29,12s
	192 *bits*	27,21s
	128 *bits*	26,44s
Rijndael	256 *bits*	30,04s
	192 *bits*	28,04s
	128 *bits*	26,32s

Tabela 6.2: Tempos para a transferência de um arquivo de 50MB entre dois *hosts* utilizando o ESP com HMAC-MD5-96 e variados algoritmos de cifragem.

provêem garantia de proteção superior à descrição de *Confidential*, e o agrupamento de algoritmos com este parâmetro no mesmo nível que o DES, que utiliza chaves de 56 *bits*, certamente mais vulnerável a ataques de força-bruta, pode, aparentemente, representar uma inconsistência. Contudo, o custo computacional de cifrar mensagens com o DES é, na maioria das implementações, mais alto que a utilização de outros algoritmos com tamanhos de chave superiores a 56 *bits*. Por outro lado, sua permanência se explica pelo fato de sua proteção estar de acordo com os pré-requisitos de *Confidential*. Além disso, implementações que não contenham qualquer um dos algoritmos anteriores ao DES podem utilizá-lo para serviços classificados neste nível.

Em relação ao desempenho dos algoritmos utilizados para compor o modelo, é notória a desvantagem de soluções como o DES e o 3DES. Contudo, em se tratando de algoritmos mais velozes, quando se utiliza o mesmo tamanho de chave a diferença de desempenho, na maioria dos casos, é pequena e levemente variável fazendo com que um algoritmo ora leve desvantagem em relação a outro ora seja um pouco mais veloz. A Tabela 6.2 mostra os tempos médios obtidos pela transferência, através do serviço de FTP, de um arquivo de 50MB entre dois *hosts* utilizando o ESP com o HMAC-MD5-96 como algoritmo de autenticação e integridade, e variando-se o algoritmo de cifragem. Os testes foram realizados com base na implementação IPSec feita pelo projeto KAME (Capítulo 7). Um dos *hosts* é um Pentium III 700MHz com 256MB de RAM e o outro um Pentium II 266MHz com 64MB de RAM, ambos utilizando o FreeBSD 4.5.

Percebe-se facilmente a influência que uma má escolha de algoritmos criptográficos pode fazer sobre o desempenho, cada vez mais importante em um cenário de redes de alta velocidade. Com o SLM, o administrador poderá fazer uma escolha específica adequada, não só em termos de segurança, mas também de desempenho, permitindo assim melhor uso

da banda disponível. Outras estimativas, mais rigorosas, podem ser feitas com o intuito de tentar encontrar diferenças entre o desempenho de algoritmos com tempos de execução próximos. É possível se considerar, por exemplo, a aplicabilidade de cada algoritmo para tráfegos com características distintas: pacotes pequenos, grandes, em rajada, entre outros.

6.4 Distribuição de serviços nos níveis de segurança

Conhecidos os níveis de segurança e suas descrições, é necessário associar cada aplicação que se deseja proteger através do SLM a um nível compatível com seus requisitos de segurança[1]. Esta tarefa representa a maneira pela qual o admnistrador pode construir ou implementar, através do SLM, a política de segurança a ser utilizada em seu ambiente.

A Figura 6.2 exibe um esquema que resume a classificação dos serviços, que deve ser feita para a utilização do SLM. É importante salientar que a proteção do tráfego das aplicações restringe-se a sua distribuição nos níveis de segurança ao invés da necessidade de construção de regras específicas com parâmetros próprios para cada tipo de tráfego. Além disso, serviços que não devem possuir qualquer proteção ou algum tipo de proteção específica através do IPSec podem, perfeitamente, conviver conjuntamente com aqueles sob a proteção do SLM. Por exemplo, para evitar que qualquer tipo de serviço produza tráfego desprotegido, pode-se criar uma política no IPSec, externa ao SLM, capaz de proteger todos os serviços que não possuem regras específicas contempladas no modelo. Desta forma, se algum usuário passa a utilizar alguma aplicação ainda não classificada em um nível de segurança, seu tráfego será automaticamente submetido à proteção desta regra *default*.

Apesar do SLM prover uma maneira mais fácil e rápida para a geração de políticas de segurança, o processo de avaliação dos requisitos de proteção das aplicações e suas conseqüentes classificações nos níveis, realizado pelo administrador, é de fundamental importância para a segurança do tráfego dos serviços associados ao modelo. Por exemplo, se todos os serviços forem vinculados a um único nível, o SLM produzirá como resultado o mesmo efeito de uma política genérica na qual um conjunto de aplicações, com requisitos de segurança distintos, é protegido pelos mesmos parâmetros. De outra maneira, caso aplicações que transmitam informações críticas sejam associadas a um nível com parâmetros que não estejam adequados a sua proteção, tal serviço pode representar um ponto de risco para o ambiente.

Na Tabela 6.3 é apresentado um exemplo de classificação de serviços de rede comuns nos níveis de segurança. Nesta instância de distribuição, algumas considerações se fazem necessárias. O SSH e o HTTPS, por exemplo, apesar de utilizarem o SSL para prover

[1]Neste contexto, requisitos de segurança representam a necessidade ou não de autenticação, integridade e/ou cifragem de pacotes e em que grau de proteção.

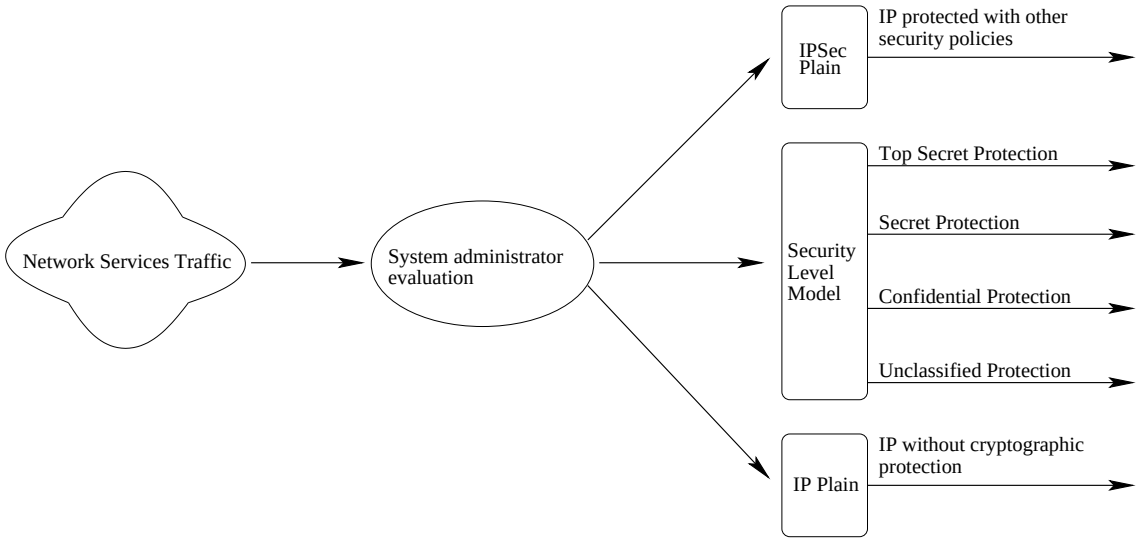


Figura 6.2: Processo de distribuição dos serviços de uma rede nos níveis de segurança do SLM.

Nível de Segurança	**Serviços**
Top Secret	Telnet, SSH, FTP, POP3, HTTPS, SMTP, SNMP, DoIt4Me
Secret	DNS(*zone transfer*), NNTP, Syslog, LDAP
Confidential	FTP-DATA, HTTP, BOOTPC, BOOTPS, TFTP
Unclassified	DNS(*query*), Time, Daytime, Echo, Finger

Tabela 6.3: Exemplo de distribuição de serviços comuns nos níveis de segurança do SLM.

um canal de comunicação seguro, não possuem qualquer tipo de proteção para os campos do protocolo IP, o que representa uma maneira de explorar tais aplicações. Sendo assim, mesmo que a porção de dados da aplicação esteja protegida por outro protocolo que faça uso de algoritmos considerados seguros, a falta de proteção adequada ao protocolo de rede viabiliza a realização de ataques que podem por em risco o correto funcionamento de tais aplicações.

Ainda sobre o exemplo apresentado, as consultas DNS foram associadas ao nível *Unclassified*, sendo, portanto, protegidas somente através dos serviços de autenticação e integridade. Apesar do acesso ao conteúdo de consultas DNS não representar riscos significativos, a transferência de zonas, procedimentos através do qual todo o mapa de um determinado domínio é transmitido do servidor principal a um servidor secundário, pode

prover a um atacante informações preliminares importantes para a elaboração de ataques. Por esta razão, a transferência de zona que, nas regras pode ser diferenciada das consultas pelo protocolo de transporte utilizado, está classificada em *Secret*[2].

Além dos serviços de rede convencionais, qualquer outro tipo de aplicação que possa ter seu tráfego filtrado segundo o uso de seletores, pode ser incorporado ao SLM. Como exemplo, o DoIt4Me [4], desenvolvido para a configuração remota de parâmetros de segurança em redes baseadas em sistemas Windows, foi incluído, no exemplo de distribuição de serviços (ver Tabela 6.3), no nível *Secret*.

Apesar do exemplo de distribuição apresentado poder servir como ponto de partida, a modelagem completa dos serviços nos níveis de segurança pelo administrador deve ser altamente encorajada, incorporando novos serviços, excluindo aqueles que não são utilizados em seu ambiente ou ainda alterando o posicionamento de outros. Feito um primeiro arranjo dos serviços, o administrador poderá, conforme suas observações e necessidades, fazer apenas pequenas modificações na sua alocação de serviços original.

Em relação ao SLM, as informações referentes às associações de serviços aos níveis de segurança são armazenadas em uma base de dados específica denominada *Protected Services* [61, 62], cujos campos estão descritos a seguir:

- *servName*: contém o nome do serviço (e.g. telnet, domain e time);
- *tpProtocol*: identifica o protocolo de transporte para o serviço anteriormente especificado (e.g. TCP ou UDP);
- *slNumber*: identificador do nível de segurança ao qual o serviço deve ser associado.

Comparando com *Security Level Definition*, a base de dados *Protected Services* deve possuir maior manipulação por parte do administrador, dado que este repositório irá conter parte considerável das informações da política de segurança.

[2]Consultas simples a servidores DNS são feitas através do protocolo UDP. Contudo, quando a resposta retornada a uma consulta utilizando UDP excede 512 *bytes*, tamanho máximo considerado seguro dos pacotes desse protocolo na época do projeto do DNS, o *resolver* recebe os primeiros 512 *bytes* da resposta e um sinal indicando o seu trucamento. Com base nesta sinalização, o *resolver* pode efetuar novamente a consulta ao servidor DNS utilizando, a partir de então, o TCP, capaz de, seguramente, transmitir uma cadeia maior de *bytes*. Além disso, transferências de zona entre servidores, por se tratar de uma quantidade maior de informações e pelo fato de requisitar mais segurança na transmissão dos dados, utiliza o TCP como protocolo de transporte [68]. Vale ressaltar, contudo, que a maioria das consultas DNS pode ser efetivada com sucesso somente através do UDP e, portanto, mesmo que uma regra aplique proteção distinta ao tráfego DNS baseado em TCP, capturando transferências de zona e consultas, grande parte das associações de segurança estabelecidas será aplicada às transferências.

Serviço	**Protocolo**	**Modo de interação**
FTP	tcp	*client*
FTP-DATA	tcp	*client*
DNS	udp	*client*, *server*
DNS	tcp	*client*, *server*
LDAP	tcp	*client*, *server*
POP3	tcp	*client*
SMTP	tcp	*client*
SSH	tcp	*client*, *server*
DoIt4Me	tcp	*server*

Tabela 6.4: Exemplo de serviços e modos de interação utilizados por um *host* sob a proteção do SLM

6.5 Incorporação dos *hosts* de uma rede ao SLM

Concluída a associação dos serviços nos níveis de segurança, faz-se necessário incluir na estrutura do SLM os *hosts* e suas respectivas aplicações que serão protegidas.

Gerar regras baseadas somente em *Protected Services* não é suficiente, pois um mesmo conjunto de regras seria utilizado para toda e qualquer entidade. Caso isto ocorresse, *hosts* que não executam um servidor HTTP conteriam em seus SPDs regras para a proteção de conexões a este servidor inexistente, se o protocolo HTTP estivesse associado a um dos níveis do SLM, por exemplo. Desta forma, o modelo provê, através da base de dados *Host Services* [61, 62], uma maneira de armazenar quais serviços, dentre aqueles distribuídos pelo administrador nos níveis de segurança, cada *host* de sua rede utiliza. Conseqüentemente, torna-se possível a criação de regras baseadas em um pacote de serviços adequado a cada tipo *host*. Em outras palavras, cada entidade conterá as regras de SPD necessárias à proteção dos seus serviços específicos.

Cada entrada de *Host Services* corresponde a um *host* protegido pelo SLM e contém, além dos serviços utilizados pelo *host*, incluindo o nome das aplicações e os seus protocolos de transporte, os modos de interação com cada serviço: cliente ou servidor. Ou seja, como as associações de segurança são direcionais (Seção 4.2), a regra para um servidor de um serviço, que recebe a conexão, é distinta da regra de um cliente, que inicia a conexão é necessário indicar qual papel é desempenhado por um *host* em relação a um determinado serviço. É possível ainda que um *host* seja servidor e cliente de uma mesma aplicação. Por exemplo, é comum em determinados ambientes fazer com que a maioria dos *hosts* seja cliente e servidor de SSH. A Tabela 6.4 apresenta a composição dos serviços utilizados por um *host* hipotético e os respectivos modos de interação..

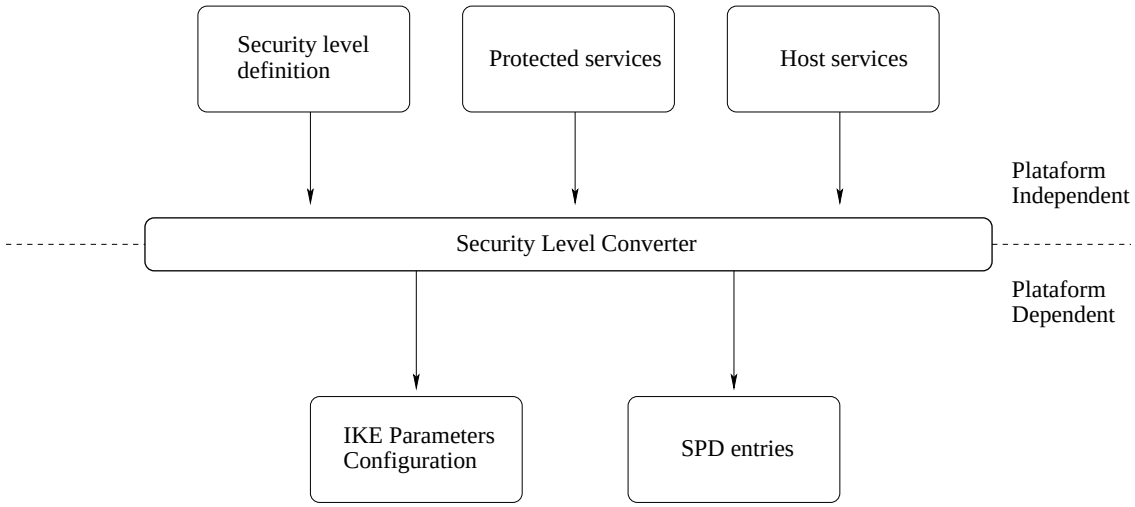


Figura 6.3: Esquema do processamento das informações de alto-nível, contidas nas bases de dados do SLM, pelo SLC.

A seguir são apresentados os campos que compõem a base de dados *Host Services*, necessários para o armazenamento da relação de serviços e seus respectivos modos de interação que cada *host* terá com o SLM:

- *slmHostname*: armazena o nome do *host* associado ao SLM;
- *slmHostServ*: contém os serviços (nome e protocolo de transporte) e o(s) modo(s) de interação a ser(em) utilizado(s) como referência na construção das regras;
- *description*: apresenta uma descrição qualquer do *host*, como por exemplo, alguma característica (e.g. cliente, servidor dedicado a algum serviço, entre outros);

É nesta base de dados que se dará a maior interação do administrador de sistemas com o SLM, pois o acréscimo ou a remoção de *hosts* e a inclusão ou exclusão de um novo serviço para um determinado *host* são modificações que devem ser feitas em *Host Services*. Tais operações, comumente, são mais freqüentes que modificações na alocação dos serviços em *Protected Services*.

6.6 *Security Level Converter*

As três bases de dados do SLM, *Security Level Definition*, *Protected Services* e *Host Services*, apresentadas anteriormente, contêm informações de alto-nível que não são capazes

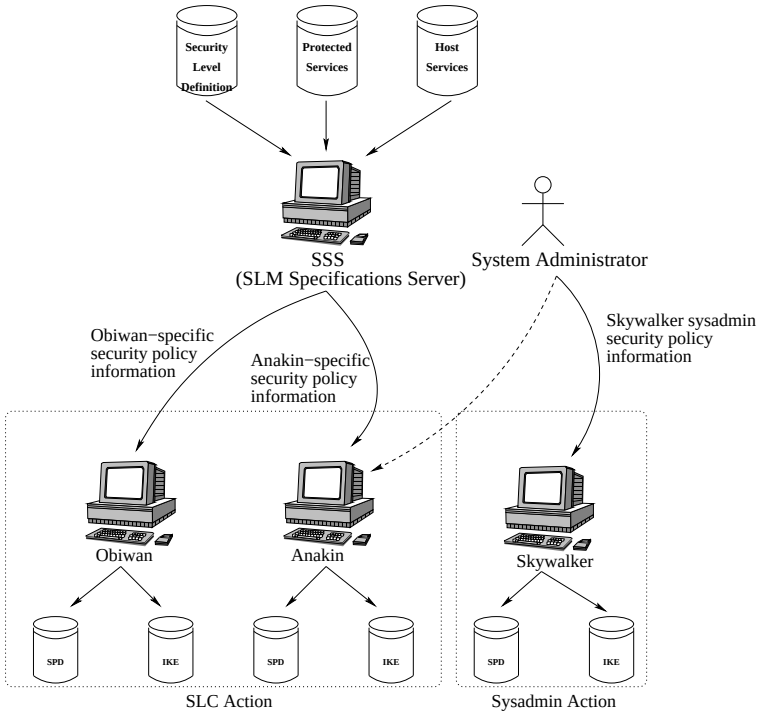


Figura 6.4: Cenário de um ambiente onde dois *hosts* estão protegidos pelo SLM.

de transmitir diretamente qualquer instrução ao IPSec e às suas estruturas. Em outras palavras, é necessário converter tais informações para o formato específico de uma implementação contida em uma determinada plataforma. Esta é, dentro do SLM, a função do SLC (*Security Level Converter*) [61, 62].

De maneira geral, o SLC processa as informações das três bases de dados e, como resultado, produz as regras da política de segurança para o SPD e os parâmetros necessários à elaboração das propostas para o IKE. Na Figura 6.3 é mostrado um diagrama que ilustra o funcionamento geral do SLC, que, através de parâmetros independentes de plataforma, gera, sem qualquer intervenção por parte do administrador de sistemas, as configurações de uma implementação específica do IPSec, dependentes de plataforma, portanto, necessárias à efetiva proteção dos pacotes IP dos serviços vinculados ao SLM.

6.6.1 Funcionamento básico

Com o intuito de facilitar suas manutenções, as bases de dados são centralizadas em um servidor da rede denominado SSS (*SLM Specifications Server*), responsável por prover a todos os *hosts* incluídos na proteção do SLM as informações necessárias à geração de seus

parâmetros de configuração.

No momento da inicialização de um sistema, o SLC é executado e sua primeira providência é fazer uma consulta ao SSS solicitando a entrada relativa ao *host* em *Host Services*, para obter conhecimento a respeito dos serviços e os respectivos modos de interação. Em seguida, são consultados os níveis de segurança associados a cada serviço listado para o *host*, de acordo com a distribuição realizada pelo administrador, armazenada em *Protected Services*. Posteriormente, o SLC solicita ao SSS o recebimento das especificações do nível de segurança de cada um dos serviços associados ao *host*, contidas em *Security Level Definition*. A comunicação inicial do SLC com as bases de dados do SSS devem ser protegidas por uma regra inserida manualmente pelo administrador em cada *host* participante do modelo com o intuito de impedir ataques no momento da passagem de parâmetros do SSS ao SLC para a construção da política de segurança local.

De posse das informações de alto-nível, armazenadas nas bases de dados mantidas no SSS, o SLC, que deve compreender o formato de funcionamento da implementação do IPSec contida na plataforma utilizada, processa os dados recebidos e os traduz para a linguagem de especificação exigida pela plataforma em uso. A partir desse momento, o *host* contendo as configurações necessárias terá o tráfego dos serviços contemplados pelo SLM protegido através do IPSec, conforme os parâmetros de segurança produzidos.

É importante notar ainda que o SLM não interfere mais na segurança da comunicação dos serviços, a não ser que o sistema seja reinicializado ou o administrador altere alguma especificação nas bases de dados e execute novamente o SLC para a atualização das mudanças realizadas. A Figura 6.4 exibe um ambiente onde dois *hosts*, *Obiwan* e *Anakin*, utilizam o SLM para a geração automática das configurações do IPSec e outro, *Skywalker*, cuja configuração é realizada através da intervenção manual do administrador de sistemas, que pode, inclusive, acrescentar outros parâmetros às máquinas protegidas pelo SLM, conforme o indicado através da linha pontilhada.

As regras geradas para cada serviço, seja como cliente, servidor ou ambos, exigem sempre que sejam estabelecidas duas associações de segurança para cada protocolo de segurança, uma para cada sentido da comunicação. Em outras palavras, caso o AH seja utilizado na proteção do tráfego de um determinado serviço, duas associações de segurança para este protocolo - uma do cliente para o servidor e outra do servidor para o cliente - devem ser geradas. Desta forma, a combinação dos dois protocolos requer a definição de quatro associações de segurança onde cada par é definido com base na mesma proposta de algoritmos e parâmetros. Caso um *host* cujas regras do SPD e parâmetros do IKE não tenham sidos gerados pelo SLM tente se comunicar com um dos serviços em um *host* com regras criadas a partir do SLM, dois requisitos básicos fazem-se necessários:

1. No SPD: a concordância quanto à aplicação do IPSec, os protocolos de segurança e a criação de um par de associação de associação de segurança para cada protocolo

(exigido pelo SPD do *host* incluso no SLM);

- No IKE: a definição de pelo menos uma combinação de algoritmos e outros parâmetros comuns entre os dois extremos.

Caso as configurações do IKE (algoritmos criptográficos ou grupo Diffie-Hellman, por exemplo) sejam propositalmente manipuladas para que seja gerada uma proposta composta por parâmetros com garantia de proteção inferior àquela que deveria ser provida segundo o nível de segurança associado ao serviço, o outro extremo da comunicação, contemplado pelo SLM, irá rejeitá-la, devido a sua incompatibilidade com a relação de parâmetros aceitáveis inseridos pelo SLC na configuração do IKE para proteção da aplicação em questão.

Capítulo 7

Implementação

Descritos os componentes e as estruturas que compõem o SLM, neste capítulo serão apresentados os seus detalhes de implementação, incluindo as modelagens desenvolvidas, as tecnologias utilizadas e as decisões tomadas no decorrer do desenvolvimento deste trabalho de pesquisa.

7.1 A plataforma IPSec

Atualmente, há diversas plataformas do IPv6/IPSec em desenvolvimento. Cada implementação, em geral, é voltada para um sistema operacional específico ou uma família de sistemas com características semelhantes, como é o caso dos sistemas BSD.

Desenvolvido pela *Hitachi*, o *Toolnet* é uma implementação do IPv6 voltada para Windows 95/98 e Windows NT 4.0. Contudo, além de não poder ser distribuída com o código-fonte, o suporte ao IPSec é incompleto. Em relação às soluções apresentadas pela própria *Microsoft*, destacam-se três iniciativas: o *MSRIPv6*, cujo código-fonte é disponível e gratuito, voltado para o Windows NT e Windows 2000; o *MSDN Technology Preview*, do qual são distribuídos somente os binários, voltado somente para o Windows 2000; e uma versão específica para o Windows XP. Contudo, o *MSRIPv6*, mais factível dentre as demais opções, está em uma fase de desenvolvimento prematura, não suportando muitas das características fundamentais do IPSec, como o uso de cifragem pelo cabeçalho ESP e a negociação dinâmica de associações de segurança através do IKE.

Em relação aos sistemas Linux, o FreeS/WAN é a principal e mais avançada implementação do IPSec. Por outro lado, no início da codificação deste trabalho, o FreeS/WAN não possuía o IKE, hoje desenvolvido através do *daemon* denominado por *Pluto*. Além disso, mesmo diante do fato do IPSec ser opcional ao IPv4 e obrigatório ao IPv6, o suporte estável é baseado somente na antiga versão do protocolo IP. A adaptação ao IPv6 está em fase de desenvolvimento.

O OpenBSD, além de poder incorporar outras implementações, conforme será mostrado a seguir, possui suporte próprio ao IPSec e contém suas principais funcionalidades, incluindo a negociação dinâmica através de duas opções implementadas nos *daemons photurisd* e *isakmpd*, e a interatividade com o IPv6. Contudo, da mesma maneira que o FreeS/WAN, o estado da implementação do suporte ao IPSec/IPv6 não estava estável na época da implementação do SLM. Além disso, a documentação disponível para o OpenBSD é escassa, o que poderia dificultar o desenvolvimento do projeto.

Voltado para as variantes do sistema BSD, outra alternativa em relação às anteriores era a adoção da implementação de IPv6/IPSec desenvolvida pelo projeto KAME, iniciativa de empresas japonesas para a união de esforços destinados a um suporte unificado e estável aos dois protocolos. Provendo as principais características determinadas na especificação do IPSec, como a possibilidade do seu uso com o IPv6 e a implementação funcional do IKE através do *daemon racoon*, o KAME, dentre as demais opções, era a alternativa mais completa e estável e, portanto, foi adotado por este projeto sobre a plataforma FreeBSD, mais documentada e divulgada em relação ao OpenBSD e ao NetBSD.

7.2 LDAP

Armazenar as bases de dados do SLM, *Security Level Definition*, *Protected Services* e *Host Services*, em arquivos simples e distribuí-los por todos os *hosts* da rede, certamente não seria a maneira mais adequada para gerenciar suas informações.

A grande quantidade de intervenção manual necessária para manter as bases de dados locais consistentes entre si poderia resultar em erros capazes de impedir o uso de determinados serviços entre alguns *hosts* por conta de potenciais incompatibilidades em seus parâmetros de configuração. Tal dificuldade agrave-se de maneira proporcional ao número de *hosts* a ser incluído no SLM. Sendo assim, a exemplo de outros sistemas que adotaram a mesma solução, optou-se pela centralização das bases de dados em uma unidade da rede capaz de repassar as informações requisitadas aos clientes.

De maneira geral, duas soluções foram consideradas: o NIS e o LDAP. O primeiro é um serviço cuja função é armazenar e distribuir mapas que representam arquivos de sistema que devem ser utilizados, na maioria dos casos, por todos os *hosts* da rede. Por exemplo, o arquivo `/etc/passwd` contendo a base de dados de todos os usuários de um sistema pode ser distribuído através do mapa NIS, denominado `passwd`, a todos os *hosts*, eliminando-se a necessidade de manter uma cópia local deste arquivo e, conseqüentemente, facilitando a manutenção de suas informações.

Apesar de amplamente implementado e utilizado em sistemas Unix para a centralização e distribuição de dados, o NIS possui um conjunto de limitações e vulnerabilidades [75] que o tornam inadequado no que diz respeito a sua adoção pelo SLM. Primei-

ramente, o NIS é projetado para redes locais e, mesmo o SLM sendo definido como um modelo baseado nestas redes, expansões para a sua interação entre diversas redes locais certamente seriam limitadas por conta do uso do NIS. Além disso, a flexibilidade disponibilizada por este serviço é limitada pelo fato de não permitir controles de acesso mais refinados aos mapas de dados. Por exemplo, se os mapas `hosts` e o `passwd` são gerados pelo servidor, todos os *hosts* da rede poderão ter acesso ao conteúdo dos mesmos, não sendo possível, desta forma, distribuir o mapa `hosts` e impedir o acesso à `passwd` a um determinado conjunto de *hosts*. O controle de acesso do NIS limita-se à possibilidade de listar o conjunto de endereços IP que podem acessar as informações produzidas pelo servidor.

Por outro lado, mais flexível, robusto e completo em relação ao NIS, o LDAP (*Lightweight Directory Access Protocol*) [72, 29] é um protocolo padronizado pelo IETF capaz de armazenar informações de qualquer espécie, modeladas conforme os requisitos específicos de cada aplicação e arranjadas de maneira hierárquica, facilitando a sua organização e localização. Além disso, várias implementações estáveis estão disponiveis para este protocolo em diversas plataformas. Assim, o LDAP foi adotado como solução para o armazenamento das bases de dados do SLM [62]. A implementação adotada foi o OpenLDAP, uma versão *open source* deste *software* que pode ser utilizada sobre muitas plataformas Unix, incluindo o FreeBSD e o Linux.

7.2.1 Organização das informações no LDAP

A hierarquia das informações armazenadas pelo LDAP, algumas vezes referenciada por diretório, é semelhante a uma árvore. Uma entrada (*entry*) é a unidade básica de armazenamento, sendo composta por um conjunto de atributos cuja função é descrevê-la.

Os atributos contêm um tipo pré-determinado e valores associados. Além disso, um mesmo atributo pode ter várias instâncias em uma única entrada. O tipo do atributo indica o formato dos dados que devem ser armazenados e a forma como o LDAP deve manipulá-los. Por exemplo, caso a especificação de um atributo indique a característica `CaseIgnoreString`, as diferenças entre caracteres minúsculos e maiúsculos serão desconsideradas nos procedimentos de busca efetuados pelo LDAP neste campo.

A Figura 7.1 mostra um exemplo de um diretório LDAP. A maneira como estão dispostas as entradas facilita a busca por informações, mesmo considerando que o nome de alguns dos atributos, neste caso, não são muito representativos (`o`: *organization*, `ou`: *organization unit*, `cn`: *common name*, etc). Cada entrada possui associada a si um DN (*Distinguished Name*) que serve como um identificador único em todo o diretório. O DN de uma entrada é formado pela concatenação dos atributos que identificam a própria entrada e todas as outras entidades superiores até a raiz da hierarquia. Por exemplo, "`cn=Jansen`

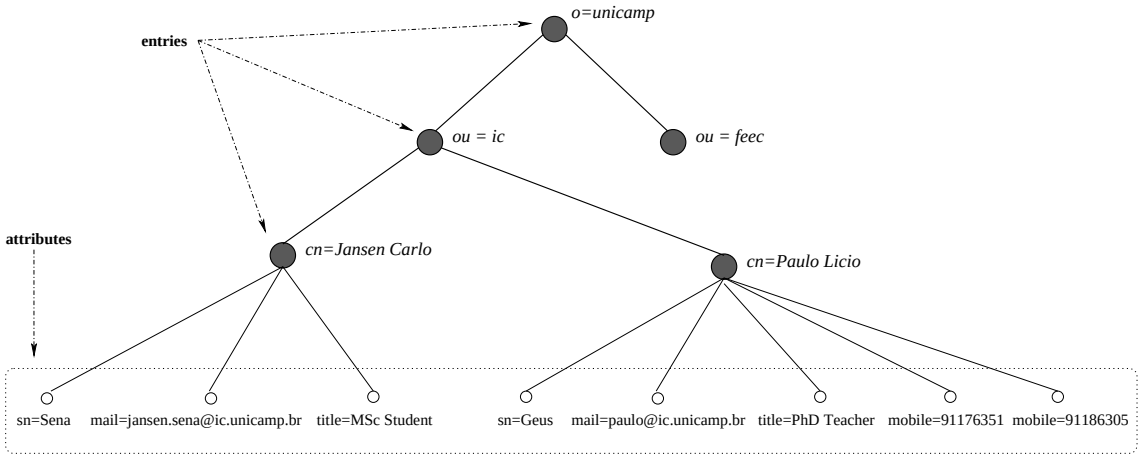


Figura 7.1: Exemplo da disposição de informações em um diretório LDAP.

`Carlo,ou=ic,o=unicamp`" é o DN do nó que contém as informações do aluno Jansen Carlo Sena.

Cada entrada contém um conjunto de classes de dados pré-definidas associadas a si. Os atributos possíveis de uma entrada compreendem a união dos atributos das classes que a compõem. Como exemplo, a seguir é mostrada a especificação da entrada "`cn=Paulo Licio,ou=ic,o=unicamp`" em LDIF [23, 29], formato cujo objetivo é prover uma representação textual para as entradas de um diretório LDAP:

```
dn: cn=Paulo Licio, ou=ic, o=unicamp
cn: Paulo Licio
sn: Geus
objectclass: top
objectclass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
mail: paulo@ic.unicamp.br
mobile: 91176351
mobile: 91188067
title: PhD Teacher
```

Desta forma, os atributos que podem compor esta entrada devem estar relacionados em uma das quatro classes utilizadas: `top`, `person`, `organizationalPerson` e `inetOrgPerson`.

À descrição das classes e seus atributos denomina-se, segundo a especificação do LDAP, esquema (*directory schema*).

7.2.2 O esquema e a estrutura de diretório definidos para o SLM

Apesar do LDAP já conter um conjunto de classes e atributos especificados em esquemas pré-definidos, para armazenar as bases de dados do SLM foi necessária a modelagem e implementação de um novo esquema, em virtude da inexistência de estruturas que pudessem ser utilizadas para a finalidade desejada [62].

Três classes, `secLevel`, `secServ` e `slmHost`, foram definidas para representar *Security Level Definition*, *Protected Services* e *Host Services*, respectivamente. Os atributos contidos em cada uma das classes correspondem aos seus campos, apresentados no Capítulo 6. As especificações destas classes são apresentadas no Apêndice A.

Além das classes e dos atributos foi definida ainda uma hierarquia para dispor as entradas, denominada na especificação do LDAP por *namespace*, conforme mostrado na Figura 7.2. Três classes já existentes no LDAP foram utilizadas: `organization`, para a criação do nó raiz representando o LAS (Laboratório de Administração e Segurança); `organizationalRole`, para a instanciação de uma entrada para determinar o administrador da hierarquia; e `organizationalUnit`, para a criação de duas entradas para o agrupamento das bases de dados. Abaixo da primeira, `gs` (*generic security*), devem ser dispostas as entradas das classes `secLevel` e `secServ`, que contêm parâmetros de segurança genéricos a serem utilizados por todos os *hosts* da rede. Em outras palavras, as entradas posteriores à `gs` representam a descrição da política de segurança elaborada pelo administrador, contendo a definição dos níveis e a distribuição dos serviços nestes níveis, de acordo com os requisitos de proteção desejados.

Inferiores a `ss` (*specific security*) estão as entradas da classe `slmHost`, identificadas pelo nome do *host*, cujo tráfego de serviços será protegido pelo SLM. Cada entrada contém os serviços e modos de interação específicos a serem utilizados na geração das regras referentes à política de segurança local de um determinado *host*. Por exemplo, a entrada relativa ao *host* `obiwan` pode descrever que devem ser criadas regras para o uso do HTTP e DNS (consultas somente) como cliente e SSH como cliente e servidor. Exemplos de entradas em LDIF utilizadas na implementação do modelo para a composição da hierarquia LDAP do SLM são mostradas no Apêndice A.

É importante notar que a hierarquia modelada para o SLM pode ser composta com estruturas definidas para outras finalidades dentro de uma organização. Por exemplo, aliado ao diretório que contém informações pessoais dos usuários da rede e a descrição geral (e.g. *hardware*, serviços e localização) dos *hosts* que a compõem, é possível inserir as estruturas do SLM, criando um único diretório para o armazenamento de diversas

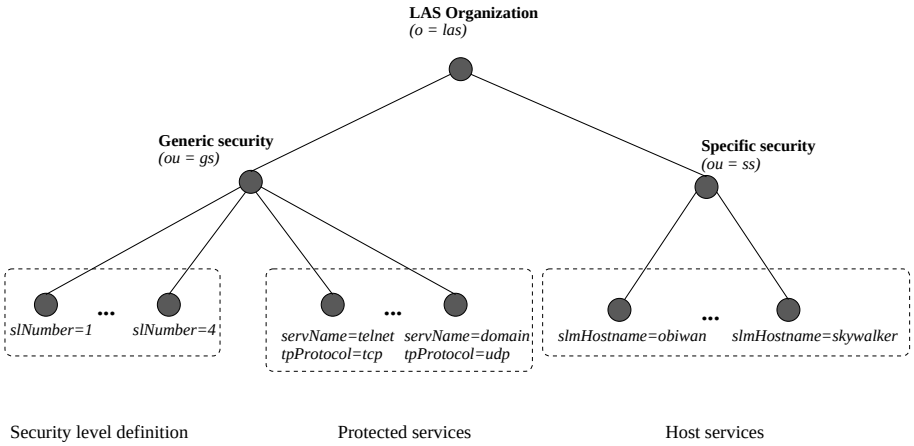


Figura 7.2: Estrutura do diretório LDAP definido para o armazenamento das bases de dados do SLM.

informações úteis para a utilização da rede. Aliada a esta possibilidade, os mecanismos de controle de acesso providos pelo LDAP permitem que porções de uma determinada hierarquia possuam restrições mais rígidas de acesso, limitando, por exemplo, o acesso externo às informações que compõem a política de segurança.

7.3 A implementação do SLC

Para obter as informações específicas dos serviços de um *host*, agregá-las com as descrições do níveis de segurança correspondentes e converter para parâmetros de configuração específicos, um conjunto de procedimentos faz-se necessário.

Na Figura 7.3 é mostrado um DFD (Diagrama de Fluxo de Dados) que exibe as principais funções implementadas no SLC. Outras funções com menor grau de importância são apresentadas no Apêndice B.

O primeiro procedimento do SLC é verificar, através de *Check_parameters*, a validade dos parâmetros recebidos na sua chamada. Caso exista algum problema nos argumentos ou em seus valores, o SLC aborta sua execução, apresentando uma mensagem que indica o problema ocorrido. Caso contrário, *Open_ldap_connection* dá continuidade ao processamento, abrindo uma conexão com o servidor LDAP da rede e passando o fluxo a *Set_security_policies_ldap*, que, por sua vez, consulta `slmHost`[1] para obter as informações referentes aos serviços e modos de interação do *host* que está executando o SLC. Cada

[1] A consulta realizada em `slmHost` toma como base o DN `slmHost=hostname,ou=ss,o=las`, onde `hostname` deve ser substituído pelo nome do *host* em questão.

Figura 7.3: Composição das principais funções do SLC.

um dos serviços, incluindo seu nome, porta e protocolo de transporte, é repassado a *Insert_security_pameters*, que solicita a *Find_service_ldap* uma busca em `secServ`[2] para obter como retorno o nível de segurança associado pelo administrador ao serviço específico. De posse de tal informação, *Insert_security_parameters* repassa a *Find_level_ldap* a identificação do nível de segurança recebida anteriormente para obter os seus parâmetros representados em alto-nível contidos em `secLevel`[3]. Finalmente, os parâmetros necessários para a construção das regras do SPD e dos parâmetros do IKE são repassados a *Insert_spd_policies* e *Insert_ike_statements*, respectivamente, que constroem e inserem nos arquivos de configuração correspondentes os valores adequados.

Encerrada a execução correta do SLC, os arquivos de configuração estão prontos para serem utilizados pelo SPD e pelo IKE. Para automatizar a execução do SLC, foi incluída uma chamada no arquivo `/etc/rc`, executado logo após a inicialização de um *host* FreeBSD. Além disso, o utilitário `setkey` é executado com o arquivo do SPD gerado pelo SLC como parâmetro para inserir no *kernel* as regras necessárias. Para terminar, o *daemon* `racoon` é inicializado considerando as setenças do IKE também produzidas pelo SLC e armazenadas em um arquivo específico[4]. Nesse momento, os serviços de um *host* terá seu tráfego protegido pelas configurações produzidas de maneira transparente ao usuário, que nem precisa, na maioria dos casos, tomar conhecimento da segurança que está sendo aplicada sobre o tráfego IP.

Apesar de ser mais útil se as informações das bases de dados do SLM estiverem centralizadas em um servidor na rede, o SLC também suporta a consulta das bases de dados através de arquivos locais. Esta característica teve como objetivo auxiliar na depuração da fase inicial do seu desenvolvimento e pode contribuir para verificar a funcionalidade do SLM, permitir seu uso em ambientes pequenos ou ainda quando a configuração do servidor LDAP não está concluída ou em produção. Em relação à modelagem desta característica, o fluxo de dados permanece o mesmo que o mostrado na Figura 7.3, sendo que toda função cujo nome termina com "*_ldap*" possui outra correspondente, que efetua os mesmos procedimentos, porém com base em arquivos locais.

[2] A consulta realizada em `secServ` toma como base o DN `servName=nameservice+tpProtocol=protocol,ou=gs,o=las`, onde `nameservice` deve ser substituído pelo nome do serviço em questão, mapeado para um identificador pelo arquivo `/etc/services` e pelo nome do protocolo de transporte.

[3] A consulta em `secLevel` toma como base o DN `slNumber=id,ou=gs,o=las`, onde `id` deve ser substituído pelo identificador do nível de segurança.

[4] É importante notar que o arquivo de configuração do IKE contém somente os parâmetros correspondentes as IPSec SAs. Desta forma, é necessário concatenar este arquivo com outro que contém as especificações relativas ao modelo de estabelecimento de ISAKMP SAs adotado: segredo pré-compartilhado, chave pública ou certificado digital. No caso deste trabalho, o segredo pré-compartilhado foi utilizado, pelo fato de ser mais simples que as demais soluções. Além disso, implementar mecanismos mais seguros para o estabelecimento de parâmetros das ISAKMP SAs está fora do escopo deste trabalho, uma vez que seu objetivo principal está voltado para as IPSec SAs.

7.3.1 A linguagem de programação

Como o SLC é o componente do SLM responsável pela conversão das informações de alto-nível em regras do SPD e parâmetros do IKE, era de fundamental importância que a linguagem de programação utilizada para a sua implementação pudesse interagir com o LDAP.

Dentre as principais possibilidades, tinha-se o C/C++, Java e Perl, todas com APIs voltadas para o tratamento de informações armazenadas em servidores LDAP. Contudo, em virtude da simplicidade das funções de manipulação de dados contidos em repositórios LDAP, a facilidade para lidar com *strings* e arquivos e a popularidade para a implementação de ferramentas de administração e segurança de sistemas, o Perl foi escolhido como linguagem de implementação do SLC [62].

Vários módulos foram desenvolvidos em Perl para a conexão com servidores LDAP. Todavia, o `Net::LDAP` foi utilizado pela facilidade de instalação, conjunto de funções e documentação completa e inteligível. O código completo do SLC é apresentado no Apêndice B.

Capítulo 8

Conclusões

A complexidade e as variadas possibilidades de utilização do IPSec podem inviabilizar sua utilização para a proteção do tráfego de serviços diversos em uma rede de computadores.

Políticas de proteção genéricas podem colocar em risco a segurança de um sistema ou inserir processamento extra desnecessário a pacotes, uma vez que o tráfego de aplicações com requisitos de segurança distintos é protegido com base nos mesmos parâmetros. Por outro lado, políticas muito específicas requerem alto custo de manutenção, sob pena de gerar incompatibilidades nas regras do SPD e ainda nos parâmetros utilizados para compor as propostas do IKE, podendo resultar na impossibilidade de comunicação enter dois *hosts*.

Neste contexto, o SLM foi desenvolvido na tentativa de facilitar o uso do IPSec através do encapsulamento de parâmetros específicos do protocolo IPSec em níveis de segurança que provêem proteção em diferentes escalas e da associação dos serviços de rede utilizados nestes níveis, simplificando o processo de implementação e manutenção das políticas de segurança do IPSec. Baseado em estruturas e definições de alto-nível, independentes de implementação, armazenadas em um servidor, o SLM reduz o custo da aplicação do IPSec, resumindo-a à manipulação dos tipos de tráfegos a serem protegidos em cada *host*. Sendo assim, o uso do SLM permite que o IPSec seja utilizado de maneira fácil e racional, impendindo a efetivação de ataques baseados nas fragilidades impostas pela arquitetura tradicional do protocolo IP.

8.1 Trabalhos Futuros

Algumas extensões do modelo proposto foram detectadas no decorrer do seu desenvolvimento.

Com a disponibilização de versões estáveis do IPSec contendo suas principais funcionalidades, é possível fazer com que o SLM se torne multi-plataforma, sendo capaz de gerar políticas de segurança para variados sistemas operacionais, como o Linux, através

do FreeS/WAN, e o OpenBSD, através de sua implementação própria de IPSec/IPv6. Tal funcionalidade é de especial interesse para a popularização do SLM. Tendo em vista que muitos ambientes computacionais utilizam uma miscelânea de sistemas operacionais, fazer com que a mesma proteção seja aplicada transparentemente a qualquer *host* da rede independente do sistema que está sendo utilizado num dado momento certamente pode contribuir para a utilização do SLM em redes de qualquer natureza, acadêmica ou corporativa.

Apesar de impedir parte considerável dos ataques baseados no protocolo IP, o IPSec não é uma panacéia da segurança de sistemas. Em ambientes como a Internet, onde pacotes podem ser processados por diversos *firewalls* que aplicam o controle de acesso estabelecido, o IPSec não pode se integrar de maneira natural. Isto se deve ao fato de que o uso de cifragem pode impedir o acesso a porções dos pacotes utilizadas pelos *firewalls*. Na tentativa de resolver tal problema, foi desenvolvido o *Multi-Layer IPSec* [74, 31], cuja proposta consiste em dividir um pacote em zonas com associações de segurança distintas. Cada zona pode conter diversas entidades que possuem as chaves criptográficas para obter acesso ao seu conteúdo específico. Em outras palavras, filtros intermediários podem ter as chaves correspondentes à zona que engloba os dados do protocolo de transporte, por exemplo. Desta forma, é possível realizar a integração do *Multi-Layer IPSec* com o SLM, para associar as zonas de cada pacote com os níveis de segurança.

Outra alternativa para incrementar o uso do SLM compreende a incorporação de uma PKI, para a distribuição de chaves públicas ou certificados digitais, tornando o processo de estabelecimento de ISAKMP SAs mais seguro em relação ao segredo pré-compartilhado utilizado no estabelecimento .

Apêndice A

Estruturas do LDAP definidas para o SLM

Conforme descrito no Capítulo 7, a implementação do SLM exigiu a especificação de um esquema LDAP para armazenar as informações das três bases de dados do modelo. A seguir são apresentadas as descrições das classes e os atributos definidos. Posteriormente são apresentadas as entradas utilizadas na implementação do modelo descritas em LDIF.

A.1 Classes e atributos

```
#
# Security Level Definitions schema
#
# Created by.........: Jansen Sena (jansen.sena@ic.unicamp.br)
# Creation date......: 01/12/2001
#
# Last Modification..: 03/12/2001
#
# Standard X.501(93) Operational Attribute Types from RFC2252

attributetype ( 2.5.18.1.11 NAME 'slNumber'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{3} )

attributetype ( 2.5.18.1.12 NAME 'secProtocol'
        EQUALITY caseIgnoreMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{5} )

attributetype ( 2.5.18.1.14 NAME 'authAlgorithm'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{30} )

attributetype ( 2.5.18.1.15 NAME 'encAlgorithm'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{30} )

attributetype ( 2.5.18.1.16 NAME 'ltRule'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{10}
        SINGLE-VALUE )

attributetype ( 2.5.18.1.17 NAME 'ltValue'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

attributetype ( 2.5.18.1.18 NAME 'ltUnit'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{15}
        SINGLE-VALUE )

attributetype ( 2.5.18.1.19 NAME 'pfsGroup'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

objectclass ( 2.5.18.2.11 NAME 'secLevel' SUP top STRUCTURAL
        MUST slNumber
        MAY ( secProtocol $ authAlgorithm $ encAlgorithm $
                ltRule $ ltValue $ ltUnit $ pfsGroup $ description ) )

attributetype ( 2.5.18.1.20 NAME 'servName'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{25}
```

```
        SINGLE-VALUE )

attributetype ( 2.5.18.1.21 NAME 'tpProtocol'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{10}
        SINGLE-VALUE )

objectclass ( 2.5.18.2.12 NAME 'secServ' SUP top STRUCTURAL
        MUST ( servName $ tpProtocol $ slNumber ) )

attributetype ( 2.5.18.1.22 NAME 'slmHostserv'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{30} )

attributetype ( 2.5.18.1.23 NAME 'slmHostname'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{50} )

objectclass ( 2.5.18.2.13 NAME 'slmHost' SUP top STRUCTURAL
        MUST slmHostname
        MAY ( slmHostserv $ description ) )
```

A.2 Entradas na hierarquia LDAP do SLM

```
# ##################################
# Root tree: LAS organization entry
# ##################################

dn: o=las
objectClass: top
objectClass: organization
o: las
telephoneNumber: 55 19 3788-5857
street: Av Albert Eisntein 1251 sala 84
postalCode: 13046-970
st: Campinas SP Brasil
```

```
l: Instituto de Computacao
description: Laboratorio de Administracao e Seguranca de Sistemas

dn: cn=lasadmin,o=las
objectClass: organizationalRole
cn: lasadmin
description: SLM directory service administrator

dn: ou=gs,o=las
ou: gs
objectClass: top
objectClass: organizationalUnit
description: General security parameters for SLM configuration

dn: ou=ss,o=las
ou: ss
objectClass: top
objectClass: organizationalUnit
description: Specific security parameters for SLM configuration

# ############################
# Security levels definition
# ############################

dn: slNumber=1,ou=gs,o=las
objectClass: secLevel
slNumber: 1
secProtocol: AH
secProtocol: ESP
authAlgorithm: hmac_sha1
authAlgorithm: hmac_ripemd
encAlgorithm: rijndael:256
encAlgorithm: twofish:256
encAlgorithm: serpent:256
encAlgorithm: blowfish:256
ltRule: time
ltValue: 2
ltUnit: hours
```

```
pfsGroup: 2
description: Top secret level

dn: slNumber=2,ou=gs,o=las
objectClass: secLevel
slNumber: 2
secProtocol: AH
secProtocol: ESP
authAlgorithm: hmac_sha1
authAlgorithm: hmac_ripemd
encAlgorithm: rijndael:192
encAlgorithm: twofish:192
encAlgorithm: serpent:192
encAlgorithm: rijndael:128
encAlgorithm: twofish:128
encAlgorithm: serpent:128
encAlgorithm: blowfish:128
encAlgorithm: idea
encAlgorithm: cast128
encAlgorithm: 3des
ltRule: time
ltValue: 2
ltUnit: hours
pfsGroup: 1
description: Secret level

dn: slNumber=3,ou=gs,o=las
objectClass: secLevel
slNumber: 3
secProtocol: AH
secProtocol: ESP
authAlgorithm: hmac_md5
encAlgorithm: rijndael:128
encAlgorithm: twofish:128
encAlgorithm: serpent:128
encAlgorithm: blowfish:128
encAlgorithm: idea
encAlgorithm: cast128
```

```
encAlgorithm: des
ltRule: time
ltValue: 5
ltUnit: hours
pfsGroup: 2
description: Confidential Level

dn: slNumber=4,ou=gs,o=las
objectClass: secLevel
slNumber: 4
secProtocol: AH
authAlgorithm: hmac_md5
ltRule: time
ltValue: 12
ltUnit: hours
pfsGroup: 1
description: Unclassified Level

# ##################
# Protected Services
# ##################

dn: servName=telnet+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: telnet
tpProtocol: tcp
slNumber: 1

dn: servName=ssh+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: ssh
tpProtocol: tcp
slNumber: 1

dn: servName=ftp+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: ftp
tpProtocol: tcp
```

```
slNumber: 1

dn: servName=pop3+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: pop3
tpProtocol: tcp
slNumber: 1

dn: servName=smtp+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: smtp
tpProtocol: tcp
slNumber: 1

dn: servName=https+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: https
tpProtocol: tcp
slNumber: 1

dn: servName=snmp+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: snmp
tpProtocol: tcp
slNumber: 1

dn: servName=domain+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: domain
tpProtocol: tcp
slNumber: 2

dn: servName=nntp+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: nntp
tpProtocol: tcp
slNumber: 2
```

```
dn: servName=syslog+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: syslog
tpProtocol: tcp
slNumber: 2

dn: servName=ftp-data+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: ftp-data
tpProtocol: tcp
slNumber: 3

dn: servName=http+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: http
tpProtocol:tcp
slNumber: 3

dn: servName=bootpc+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: bootpc
tpProtocol: tcp
slNumber: 3

dn: servName=bootps+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: bootps
tpProtocol: tcp
slNumber: 3

dn: servName=tftp+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: tftp
tpProtocol: tcp
slNumber: 3

dn: servName=domain+tpProtocol=udp,ou=gs,o=las
objectClass: secServ
```

```
servName: domain
tpProtocol: udp
slNumber: 4

dn: servName=time+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: time
tpProtocol: tcp
slNumber: 4

dn: servName=daytime+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: daytime
tpProtocol: tcp
slNumber: 4

dn: servName=echo+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: echo
tpProtocol: tcp
slNumber: 4

dn: servName=finger+tpProtocol=tcp,ou=gs,o=las
objectClass: secServ
servName: finger
tpProtocol: tcp
slNumber: 4

# ###############
# Hosts services
# ###############

dn: slmHostname=obiwan,ou=ss,o=las
objectClass: slmHost
slmHostname: obiwan
slmHostserv: ssh tcp c,s
slmHostserv: telnet tcp c,s
slmHostserv: pop3 tcp c
```

```
slmHostserv: smtp tcp c
slmHostserv: http tcp c
slmHostserv: https tcp c
slmHostserv: ldap tcp c
slmHostserv: domain tcp c
slmHostserv: domain udp c
slmHostserv: ftp tcp c,s
slmHostserv: ftp-data tcp c,s
slmHostserv: daytime tcp c
description: Common client host

dn: slmHostname=mahul,ou=ss,o=las
objectClass: slmHost
slmHostname: mahul
slmHostserv: ssh tcp s
slmHostserv: telnet tcp s
slmHostserv: pop3 tcp s
slmHostserv: smtp tcp s
slmHostserv: http tcp s
slmHostserv: https tcp s
slmHostserv: ldap tcp s
slmHostserv: domain tcp s
slmHostserv: domain udp c,s
slmHostserv: ftp tcp s
slmHostserv: ftp-data tcp s
slmHostserv: daytime tcp c,s
slmHostserv: time tcp c,s
description: General server

dn: slmHostname=skywalker,ou=ss,o=las
objectClass: slmHost
slmHostname: skywalker
slmHostserv:  ssh tcp c,s
slmHostserv: telnet tcp c,s
slmHostserv: pop3 tcp c,s
slmHostserv: smtp tcp c,s
slmHostserv: http tcp c,s
slmHostserv: https tcp c,s
```

```
slmHostserv: ldap tcp c,s
slmHostserv: domain tcp c,s
slmHostserv: domain udp c,s
slmHostserv: ftp tcp c,s
slmHostserv: ftp-data tcp c,s
slmHostserv: daytime tcp c,s
slmHostserv: time tcp c,s
slmHostserv: echo tcp c,s
slmHostserv: finger tcp c,s
description: Experimental host
```

Apêndice B

Aspectos da utilização e implementação do SLC

Neste apêndice serão apresentadas algumas das características de utilização do SLC, arquivos produzidos como resultado da sua execução e seu código-fonte.

B.1 Manual de utilização do SLC

Para facilitar o uso do SLC, a implementação contém um manual que explica as opções que podem ser utilizadas para especificar o seu comportamento. Esta *man page* pode ser obtida através da chamada do programa `slc` com o parâmetro "–man". Como resultado, o texto a seguir é mostrado ao usuário:

```
NAME
    slc Security Level Converter

SYNOPSIS
    slc [options]

    Options: --help, --man, --spd <file>, --ike <file>, --sl <file>, --ss
    <file>, --mode <files|ldap>, server <IP address|hostname>, --verbose|noverbose

OPTIONS
    --help  Print a brief help message and exit.

    --man   Print the manual page and exit.

    --spd   Target file for security policy database (only for KAME).
            Default: spd.conf
```

```
    --ike   Target file for IKE SA phase 2 parameters (only for RACOON).
            Default: ike.conf

    --sl    Security level definition file (only for files operation mode)

    --ss    Services and security level attribution file (only for files
            operation mode)

    --mode  Choose files or ldap operation mode. Default: ldap

    --server
            IP address or hostname of the LDAP server. Default: localhost

    --verbose|--noverbose
            Prints debug information while processing security level
            definitions. Default: noverbose

DESCRIPTION
    This program generates IPSec/IKE policies and parameters to protect
    network services according to security levels specified in local files
    or on an LDAP server.

EXAMPLES
    slc --verbose --server 10.1.1.46 --spd setkey.conf --ike racoon.conf
    In this example SLC connect to LDAP server 10.1.1.46 in verbose mode.
    The configuration files are created as setkey.conf (spd rules) and
    racoon.conf (ike statements).

    slc --mode files --sl seclevels.conf --ss secserv.conf
    In this example the IPSec security policy definitions are contained in
    the local files seclevels.conf and secserv.conf. The configuration files
    are created with the default output file names.

AUTHORS
    Jansen Carlo Sena (jansen.sena@ic.unicamp.br) and Paulo Lício de Geus
    (paulo@ic.unicamp.br)
```

Para obter uma descrição mais breve em relação a *man page*,o `slc` mostra uma descrição simplificada de suas opções através da execução com o parâmetro "–help".

```
Use --man option to more details
Usage:
    slc [options]
```

```
Options: --help, --man, --spd <file>, --ike <file>, --sl <file>, --ss
<file>, --mode <files|ldap>, server <IP address>, --verbose|noverbose
```

B.2 Arquivos produzidos pelo SLC

O objetivo final do SLC é a criação dos arquivos específicos para uma plataforma IPSec contendo as regras do SPD e os parâmetros do IKE. A seguir são apresentados os arquivos produzidos para a implementação do projeto KAME no FreeBSD após a execução do SLC para o *host* `obiwan`, cujos serviços e modos de iteração estão especificados em LDIF no Apêndice A.

Vale ressaltar que todas as sentenças estão baseadas em dois endereços IPv6 fixos,`fec0::1:250:8bff:fe0f:c66f` e `fec0::1:2a0:c9ff:fee8:548`. No LAS (Laboratório de Administração e Segurança), local onde o SLC foi desenvolvido, estes dois *hosts* foram utilizados para compor uma rede que pudesse servir como ambiente de trabalho. Para a geração de setenças que compreendam o *host* local e outro conjunto de *hosts* pode-se alterar as variáveis `$dest_address` e `$dest_prefix` para conter o endereço da rede local ou mesmo o endereço IPv6 genérico `0:0:0:0` (também representado por "`::`"). Esta segunda opção, contudo, permite que a comunicação segura ocorra entre um *host* local e qualquer outro *host*, mesmo que este esteja externo a rede local[1].

B.2.1 Arquivo `spd.conf`

```
#
# Policies to IPSec SPD
# File generates by converter program copyright 2001 Jansen Sena
# Creation Date: Tue Apr  2 17:36:45 GMT 2002
# Mode creation: ldap
#
#
# Policies to protect ssh (tcp/22) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[22] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[22] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# local host (server) <---> remote host (client)
```

[1] A opção de utilizar o endereço `0:0:0:0` para compor os arquivos de configuração provocava problemas com o *racoon* que estão sendo corrigidos para as versões futuras do *software*.

```
#
spdadd fec0::1:250:8bff:fe0f:c66f[22] fec0::1:2a0:c9ff:fee8:548[any] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[any] fec0::1:250:8bff:fe0f:c66f[22] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect telnet (tcp/23) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[23] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[23] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# local host (server) <---> remote host (client)
#
spdadd fec0::1:250:8bff:fe0f:c66f[23] fec0::1:2a0:c9ff:fee8:548[any] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[any] fec0::1:250:8bff:fe0f:c66f[23] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect pop3 (tcp/110) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[110] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[110] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect smtp (tcp/25) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
```

```
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[25] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[25] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect http (tcp/80) service with security level 3
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[80] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[80] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect https (tcp/443) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[443] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[443] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect ldap (tcp/389) service with security level 2
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[389] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[389] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect domain (tcp/53) service with security level 2
#
#
```

```
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[53] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[53] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect domain (udp/53) service with security level 4
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[53] udp -P out ipsec
        AH/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[53] fec0::1:250:8bff:fe0f:c66f[any] udp -P in ipsec
        AH/tranport//require;
#
# Policies to protect ftp (tcp/21) service with security level 1
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[21] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[21] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# local host (server) <---> remote host (client)
#
spdadd fec0::1:250:8bff:fe0f:c66f[21] fec0::1:2a0:c9ff:fee8:548[any] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[any] fec0::1:250:8bff:fe0f:c66f[21] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect ftp-data (tcp/20) service with security level 3
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[20] tcp -P out ipsec
```

```
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[20] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# local host (server) <---> remote host (client)
#
spdadd fec0::1:250:8bff:fe0f:c66f[20] fec0::1:2a0:c9ff:fee8:548[any] tcp -P out ipsec
        AH/transport//require
        ESP/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[any] fec0::1:250:8bff:fe0f:c66f[20] tcp -P in ipsec
        AH/transport//require
        ESP/tranport//require;
#
# Policies to protect daytime (tcp/13) service with security level 4
#
#
# local host (client) <---> remote host (server)
#
spdadd fec0::1:250:8bff:fe0f:c66f[any] fec0::1:2a0:c9ff:fee8:548[13] tcp -P out ipsec
        AH/tranport//require;
spdadd fec0::1:2a0:c9ff:fee8:548[13] fec0::1:250:8bff:fe0f:c66f[any] tcp -P in ipsec
        AH/tranport//require;
```

B.2.2 Arquivo `ike.conf`

```
#
# Statements to IKE(racoon) daemon
# File generates by converter program copyright 2001 Jansen Sena
# Creation Date: Tue Apr  2 17:36:45 GMT 2002
# Mode creation: ldap
#
# Statements to the phase 2 SA to ssh (tcp/22) service according SL 1 parameters
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[22] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# local host (server) <---> remote host (client)
```

```
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64[22] address fec0::1:2a0:c9ff:fee8:548/64 tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to telnet (tcp/23) service according SL 1 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[23] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# local host (server) <---> remote host (client)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64[23] address fec0::1:2a0:c9ff:fee8:548/64 tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to pop3 (tcp/110) service according SL 1 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[110] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to smtp (tcp/25) service according SL 1 parameters
```

```
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[25] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to http (tcp/80) service according SL 3 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[80] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 128,twofish 128,serpent 128,blowfish 128,\\
        idea,cast128,des;
        authentication_algorithm        hmac_md5;
}
#
# Statements to the phase 2 SA to https (tcp/443) service according SL 1 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[443] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to ldap (tcp/389) service according SL 2 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[389] tcp
{
```

```
        pfs_group                       1;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 192,twofish 192,serpent 192,rijndael 128,\\
        twofish 128,serpent 128,blowfish 128,idea,cast128,3des;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to domain (tcp/53) service according SL 2 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[53] tcp
{
        pfs_group                       1;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 192,twofish 192,serpent 192,rijndael 128,\\
        twofish 128,serpent 128,blowfish 128,idea,cast128,3des;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to domain (udp/53) service according SL 4 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[53] udp
{
        pfs_group                       1;
        lifetime                        time    2       hours;
        authentication_algorithm        hmac_md5;
}
#
# Statements to the phase 2 SA to ftp (tcp/21) service according SL 1 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[21] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
```

```
# local host (server) <---> remote host (client)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64[21] address fec0::1:2a0:c9ff:fee8:548/64 tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 256,twofish 256,serpent 256,blowfish 256;
        authentication_algorithm        hmac_sha1,hmac_ripemd;
}
#
# Statements to the phase 2 SA to ftp-data (tcp/20) service according SL 3 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[20] tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 128,twofish 128,serpent 128,blowfish 128,\\
        idea,cast128,des;
        authentication_algorithm        hmac_md5;
}
#
# local host (server) <---> remote host (client)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64[20] address fec0::1:2a0:c9ff:fee8:548/64 tcp
{
        pfs_group                       2;
        lifetime                        time    2       hours;
        encryption_algorithm            rijndael 128,twofish 128,serpent 128,blowfish 128,\\
        idea,cast128,des;
        authentication_algorithm        hmac_md5;
}
#
# Statements to the phase 2 SA to daytime (tcp/13) service according SL 4 parameters
#
#
# local host (client) <---> remote host (server)
#
sainfo address fec0::1:250:8bff:fe0f:c66f/64 address fec0::1:2a0:c9ff:fee8:548/64[13] tcp
{
        pfs_group                       1;
        lifetime                        time    2       hours;
        authentication_algorithm        hmac_md5;
}
```

B.3 Código-fonte

A seguir é apresentado o código-fonte do SLC implementado em Perl. A licensa Gnu GPL está inserida como maneira de garantir o livre acesso e modificação ao programa respeitados os direitos intelectuais do autor.

O caracter "\\" contido no final de uma linha indica a sua continuidade na linha seguinte.

```
#!/usr/bin/perl
#
# ##############################################################################
# Copyright 2001 Jansen Sena
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
# ##############################################################################
#
# Loading Perl modules
#
use Net::LDAP;                                          # LDAP interaction
use Getopt::Long;                                       # Manipulação de opções
use Pod::Usage;                                         # Impressão de man page
#
$source_address = "fec0::1:250:8bff:fe0f:c66f";
$source_prefix  = 64;
$dest_address   = "fec0::1:2a0:c9ff:fee8:548";
$dest_prefix    = 64;
#
$hostname       = "obiwan";
$ldap;
$ss_base        = "ou=ss,o=las";
```

```
$gs_base        = "ou=gs,o=las";
#
#
$client;
$server;
#
# Default values for command line parameters
#
$spd_file_default       = "./spd.conf";
$ike_file_default       = "./racoon.conf";
$sl_file_default        = "./seclevels.conf";
$ss_file_default        = "./secservices.conf";
$hostserv_file_default  = "./hostservices.conf";
$mode_default           = "ldap";
$verbose_default        = 0;
$ldap_server_default    = "localhost";
#
# Command line parameters initialization
#
$man           = 0;
$help          = 0;
$verbose       = 0;
$spd_file      = "";
$ike_file      = "";
$sl_file       = "";
$ss_file       = "";
$hostserv_file = "";
$log_file      = "";
$mode          = "";
$ldap_server   = "";
#
&Check_parameters();
#
# Open files to spd and racoon
#
open(SPD,">$spd_file") || die "Can't open $spd_file file: $!\n";
open(IKE,">$ike_file") || die "Can't open $ike_file file: $!\n";
&Print_comments;
#
if ( $mode eq ldap ) {
    &Open_ldap_connection($ldap_server);
    &Set_security_policies_ldap();
    $ldap->unbind;
} else {
    #
    # Read file with security levels definitions
```

```
    #
    open(FILE,"<$sl_file") || die "Can't open $sl_file file: $!\n";
    @Seclevels_file = <FILE>;
    close(FILE);
    #
    # Read file with security levels from services
    #
    open(FILE,"<$ss_file") || die "Can't open $ss_file file: $!\n";
    @Secservices_file = <FILE>;
    close(FILE);
    #
    # Read file with specifics host services to protect
    #
    open(FILE,"<$hostserv_file") || die "Can't open $hostserv_file file: $!\n";
    @Hostservices_file = <FILE>;
    close(FILE);
    &Set_security_policies();
}
close(SPD);
close(IKE);
if ( $verbose ) { print "Security policies creates in $spd_file and $ike_file files\n"; }
#
# --- End of program ---------------

#
# Functions
#
#
# Lê cada serviço associado ao host e chama as funções para gerar as entradas do IPSec e IKE
#
sub Set_security_policies_ldap {
    my $port, my $service, my $protocol, my $remainder, my @options;
    my $filter = "(&(objectClass=slmHost)(slmHostname=$hostname))";
    my $result = $ldap->search(base => $ss_base, filter => $filter);
    if ( $result->count > 0 ) {
        my $entry  = $result->entry(0);
        my @values = $entry->get_value(slmHostserv);
        my $qtde_value = @values;
        if ( $qtde_value > 0 ) {
            foreach $values ( @values ) {
                $client = 0;
                $server = 0;
                ($service,$protocol,$remainder) = split(' ',$values);
                @options = split(/,/,$remainder);
                &Check_directions(@options) || die "Invalid flags in $values of the LDAP\\
                server\n";
```

```
                $port = getservbyname($service,$protocol);
                &Insert_security_parameters($port,$service,$protocol);
            }
            1;
        } else {
            print "$hostname host no have services to protect in LDAP server\n";
            &Abort_program;
        }
    } else {
        print "$hostname host entry not found in LDAP server\n";
        &Abort_program;
    }
}

#
# Lê cada serviço associado ao host e chama as funções para gerar as entradas do IPSec e IKE
#
sub Set_security_policies {
    foreach $Hostservices_file (@Hostservices_file) {
        my $port, my $service, my $protocol, my $remainder, my @options;
        if ( ! ($Hostservices_file =~ /^#/)) {
                $client = 0;
                $server = 0;
                ($service,$protocol,$remainder) = split(/:|\n$/,$Hostservices_file);
                @options = split(/,/,$remainder);
                &Check_directions(@options) || die "Invalid flags in $Hostservices_file\n";
                $port = getservbyname($service,$protocol);
                &Insert_security_parameters($port,$service,$protocol);
        }
     }
}

#
# Lê cada servico do arquivo secservices.conf
# @_[0]: port / @_[1]: service / @_[2]: protocol
#
sub Insert_security_parameters {
    &Clear_fields;
    if ($mode eq "ldap") {
        $sl_service = &Find_service_ldap(@_[1],@_[2]);
        &Find_level_ldap($sl_service);
    } else {
        $sl_service = &Find_service(@_[1],@_[2]);
        &Find_level($sl_service);
    }
    &Insert_spd_policies(@_[0],@_[1],@_[2],$sl_service);
```

```
    &Insert_ike_statements(@_[0],@_[1],@_[2],$sl_service);
}

#
# Seleciona o SL de um determinado serviço no servidor LDAP
# @_[0]: service    @_[1]: protocol
#
sub Find_service_ldap {
    my ($service,$protocol) = (@_[0],@_[1]);
    my $security_level;
    my $entry;
    my $filter = "(&(objectClass=secServ)(tpProtocol=$protocol)(servName=$service))";
    my $result = $ldap->search(base => $gs_base, filter => $filter);
    if ( $result->count > 0 ) {
        if ( $result->count > 1 ) {
            print "More than one entry for $service($protocol) in LDAP server\n";
            &Abort_program;
        }
        $entry = $result->entry(0);
        $security_level = $entry->get_value(slNumber);
    } else {
        print "$service($protocol) service is not found in LDAP server\n";
        &Abort_program;
    }
}

#
# Seleciona o SL de um determinado serviço
# @_[0]: service    @_[1]: protocol
#
sub Find_service {
    my ($service,$protocol,$security_level);
    my $found = 0;
    my $count = 0;
    my $lines_number = @Secservices_file;
    while ((! $found) && ($count <= $lines_number)) {
        if ( ! (@Secservices_file[$count] =~ /^#/)) {
                ($service,$protocol,$security_level) = split(/:|\n$/,\\
                @Secservices_file[$count],4);
                if (($service eq @_[0]) && ($protocol eq @_[1])) {
                    $found = 1;
                }
        }
        $count++;
    }
    if ($found == 1) {
```

```
        $security_level;
    } else {
        0;
    }
}

#
# Seleciona parâmetros de um nível de segurança no servidor LDAP
# @_[0]: security level
#
sub Find_level_ldap {
    my $security_level = @_[0];
    my $filter = "(&(objectClass=secLevel)(slNumber=$security_level))";
    my $result = $ldap->search(base => $gs_base, filter => $filter);
    if ( $result->count > 0 ) {
        my $entry = $result->entry(0);
        @security_protocols        = $entry->get_value(secProtocol);
        @pfs_group                 = $entry->get_value(pfsGroup);
        @lifetime[0]               = $entry->get_value(ltRule);
        @lifetime[1]               = $entry->get_value(ltValue);
        @lifetime[2]               = $entry->get_value(ltUnit);
        @encryption_algorithms     = $entry->get_value(encAlgorithm);
        @authentication_algorithms = $entry->get_value(authAlgorithm);
        1;

    } else {
        print "Security level is not found in LDAP server.\n";
        &Abort_program;
    }
}

#
# Seleciona parâmetros de um nivel de seguranca
# @_[0]: security level
#
sub Find_level {
    my $found = 0;
    my $count = 0;
    my $seclevel_load = 1;
    while ( ($seclevel_load == 1) && ( $count <= $#Seclevels_file )) {
        if ( ! (@Seclevels_file[$count] =~ /^#/)) {
           if ( $found == 0) {
              # Secfields eh divido em 4 devido ao \n apos o caractere "{" que confunde\\
              o teste if a seguir
              @Secfields = split(' ', @Seclevels_file[$count],4);
              if (($Secfields[0] eq "security_level") && ($Secfields[1] == $_[0]) &&\\
```

```
            ($Secfields[2] eq "{")){
               $found = 1;
            }
         } else {
             @Secfields = split(' ',@Seclevels_file[$count],2);
             if ( @Secfields[0] eq "security_protocols" ) {
                 @security_protocols = split(/,|;\n$/,@Secfields[1]);
             } elsif ( @Secfields[0] eq "pfs_group") {
                 @pfs_group = split(/,|;\n$/,@Secfields[1]);
             } elsif ( @Secfields[0] eq "lifetime") {
                 @lifetime = split(/\s+|;\n$/,@Secfields[1]);
             } elsif ( @Secfields[0] eq "encryption_algorithms") {
                 @encryption_algorithms = split(/,|;\n$/,@Secfields[1]);
             } elsif ( @Secfields[0] eq "authentication_algorithms") {
                 @authentication_algorithms = split(/,|;\n$/,@Secfields[1]);
             } elsif ( (@Seclevels_file[$count] =~ /^\}$/) ) {
                 $seclevel_load = 0;
             } else {
                 print "Parameter @Secfields[0] is not recognized !\n";
             }
         }
        }
        $count++;
    }
    if ( $count > ($#Seclevels_file) && $found == 0) {
        print "Security Level $_[0] is not defined !\n";
    }
}

#
# Insert_spd_policies: create spd.conf file with security policies for IPSec
# @_[0]: port / @_[1]: service / @_[2]: protocol / @_[3]: security level
#
sub Insert_spd_policies {
    my $port           = @_[0];
    my $service        = @_[1];
    my $protocol       = @_[2];
    my $security_level = @_[3];
    foreach $security_protocols (@security_protocols) {
       if ( $security_protocols eq @security_protocols[$#security_protocols] ) {
          $spd_protocols = $spd_protocols . "\t$security_protocols/tranport//require;\n";
       } else {
          $spd_protocols = $spd_protocols . "\t$security_protocols/transport//require\n";
       }
    }
    print SPD "#\n# Policies to protect $service ($protocol/$port) service with security level\\
```

```
    $security_level\n#\n";
    if ($client) {
        #
        # Policy: local host (client) <-> remote host (server)
        #
        print SPD "#\n# local host (client) <---> remote host (server)\n#\n";
        print SPD "spdadd " . "$source_address\[any\] " . "$dest_address\[$port\] " .\\
        "$protocol -P out ipsec\n";
        print SPD "$spd_protocols";
        print SPD "spdadd " . "$dest_address\[$port\] " . "$source_address\[any\] " .\\
        "$protocol -P in ipsec\n";
        print SPD "$spd_protocols";
    }
    if ($server) {
        #
        # Policy: local host (server) <-> remote host (client)
        #
        print SPD "#\n# local host (server) <---> remote host (client)\n#\n";
        print SPD "spdadd " . "$source_address\[$port\] " . "$dest_address\[any\] " . \\
        "$protocol -P out ipsec\n";
        print SPD "$spd_protocols";
        print SPD "spdadd " . "$dest_address\[any\] " . "$source_address\[$port\] " . \\
        "$protocol -P in ipsec\n";
        print SPD "$spd_protocols";
    }
}

#
# Insert_ike_statements: create racoon.conf file with IKE statements
# @_[0]: port / @_[1]: service / @_[2]: protocol / @_[3]: security level
#
sub Insert_ike_statements {
    my $port           = @_[0];
    my $service        = @_[1];
    my $protocol       = @_[2];
    my $security_level = @_[3];
    print IKE "#\n# Statements to the phase 2 SA to $service ($protocol/$port) service \\
    according SL $security_level parameters\n#\n";
    if ($client) {
print IKE "#\n# local host (client) <---> remote host (server)\n#\n";
print IKE "sainfo address $source_address/$source_prefix address $dest_address/ \\
        $dest_prefix\[$port\] $protocol\n{\n";
        if (scalar @pfs_group) {
         print IKE "\tpfs_group\t\t\t@pfs_group[0];\n";
        }
        if (scalar @lifetime) {
```

```
    print IKE "\tlifetime\t\t\t@lifetime[0]\t@lifetime[1]\t@lifetime[2];\n";
        }
        if (scalar @encryption_algorithms) {
    @cipher_algorithms = split(/:/,join(",",@encryption_algorithms));
    print IKE "\tencryption_algorithm\t\t@cipher_algorithms;\n";
        }
        if (scalar @authentication_algorithms) {
    @auth_algorithms = split(/:/,join(",",@authentication_algorithms));
    print IKE "\tauthentication_algorithm\t@auth_algorithms;\n";
        }
        print IKE "\}\n";
    }
    if ($server) {
        print IKE "#\n# local host (server) <---> remote host (client)\n#\n";
print IKE "sainfo address $source_address/$source_prefix\[$port\] address \\
        $dest_address/$dest_prefix $protocol\n{\n";
        if (scalar @pfs_group) {
    print IKE "\tpfs_group\t\t\t@pfs_group[0];\n";
        }
        if (scalar @lifetime) {
    print IKE "\tlifetime\t\t\t@lifetime[0]\t@lifetime[1]\t@lifetime[2];\n";
        }
        if (scalar @encryption_algorithms) {
    @cipher_algorithms = split(/:/,join(",",@encryption_algorithms));
    print IKE "\tencryption_algorithm\t\t@cipher_algorithms;\n";
        }
        if (scalar @authentication_algorithms) {
    @auth_algorithms = split(/:/,join(",",@authentication_algorithms));
    print IKE "\tauthentication_algorithm\t@auth_algorithms;\n";
        }
        print IKE "\}\n";
    }
}

#
# Clear_fields: empty security level fields
#
sub Clear_fields{
    @security_protocols         = ();
    @pfs_group                  = ();
    @lifetime                   = ();
    @encryption_algorithms      = ();
    @authentication_algorithms  = ();
    $spd_protocols              = "";
}
```

```
#
# Print_comments: insert in spd and racoon files a head of general informations
#
sub Print_comments{
    $date = `date`;
    #
    # Putting general information in spd file
    #
    print SPD "#\n# Policies to IPSec SPD\n";
    print SPD "# File generates by slc program copyright 2001 Jansen Sena\n";
    print SPD "# Creation Date: $date";
    print SPD "# Mode creation: $mode\n#\n";
    #
    # Putting general information in racoon file
    #
    print IKE "#\n# Statements to IKE(racoon) daemon\n";
    print IKE "# File generates by slc program copyright 2001 Jansen Sena\n";
    print IKE "# Creation Date: $date";
    print IKE "# Mode creation: $mode\n#\n";
}

#
# Check_parameters: check the parameters to conversor program
#
sub Check_parameters{
    my @server_entry;
    GetOptions( 'help' => \$help, 'man' => \$man, 'spd=s' => \$spd_file, 'ike=s' \\
    => \$ike_file, 'sl=s' => \$sl_file, 'ss=s' => \$ss_file, 'hostserv=s' => \\
    \$hostserv_file, 'logfile=s' => \$log_file, 'mode=s' => \$mode, 'server=s' => \\
    \$ldap_server, 'verbose!' => \$verbose ) || pod2usage( -message => "Syntax \\
    error. Use --man option to more details", -verbose => 1 );
    pod2usage( -message => "Use --man option to more details", -verbose => 0 ) if ( $help );
    pod2usage( -verbose => 2) if ( $man );
    pod2usage( -message => "Invalid parameter @ARGV. Use --man to more details", \\
    -verbose => 0 ) if ( @ARGV > 0 );
    #
    # Checking possible values to operation mode parameter
    #
    if ( ( $mode ne "files" ) && ( $mode ne "ldap" ) && ( $mode ne "" ) ) {
        pod2usage(-message => "Invalid operation mode. Use --man option to more \\
        details", -verbose => 0);
    }
    #
    # Setting default values to command lines parameters
    #
    if ( ! $verbose ) {
```

```
   $verbose = $verbose_default;
}
if ( ! $mode ) {
    $mode = $mode_default;
    if ( $verbose ) { print "Setting operation mode to $mode_default.\n" };
}
if ( ( ! $ldap_server ) && ( $mode eq "ldap" )) {
    $ldap_server = $ldap_server_default;
    if ( $verbose ) { print "Setting $ldap_server_default as LDAP server.\n"; }
}
if ( ( $mode eq "files" ) && ( ! $sl_file ) ) {
    $sl_file = $sl_file_default;
    if ( $verbose ) { print "Setting security levels file to $sl_file_default.\n"; }
}
if ( ( $mode eq "files" ) && ( ! $ss_file ) ) {
    $ss_file = $ss_file_default;
    if ( $verbose ) { print "Setting security services file to $ss_file_default.\n"; }
}
if ( ( $mode eq "files" ) && ( ! $hostserv_file ) ) {
    $hostserv_file = $hostserv_file_default;
    if ( $verbose ) { print "Setting host services file to $hostserv_file_default.\n"; }
}
if ( ! $spd_file ) {
    $spd_file = $spd_file_default;
    if ( $verbose ) { print "Setting spd file to $spd_file_default.\n"; }
}
if ( ! $ike_file ) {
    $ike_file = $ike_file_default;
    if ( $verbose ) { print "Setting ike file to $ike_file_default.\n"; }
}
#
# Checking ldap server name
#
if ( $mode eq "ldap") {
    @server_entry = gethostbyname($ldap_server);
    if ( ! @server_entry ) {
        die "$ldap_server host no exists.\n";
    }
}
#
# Checking files existence
#
die "$sl_file file no exists.\n"       if ( ! ( -e $sl_file ) && $mode eq "files" );
die "$sl_file is not a plain file.\n"  if ( ! ( -f $sl_file ) && $mode eq "files" );
die "$ss_file file no exists.\n"       if ( ! ( -e $ss_file ) && $mode eq "files" );
die "$ss_file is not a plain file.\n"  if ( ! ( -f $ss_file ) && $mode eq "files" );
```

```
    #
    # Verifying parameters consistent
    #
    if ( ( $mode eq "ldap" ) && ( $sl_file || $ss_file ) ) {
        pod2usage(-message => "Incompatible parameters. Use --man option to more details",\\
        -verbose => 1);
    }
    if ( ( $mode eq "files" ) && ( $ldap_server ) ) {
        pod2usage(-message => "Incompatible parameters. Use --man option to more details",\\
        -verbose => 1);
    }
}

#
# Check_directions: check the directions parameters in hostservices.conf
#
sub Check_directions{
    my $qtde_flags = @_;
    if ($qtde_flags > 2) {
        0;
    } else {
        ($opt1,$opt2) = (@_[0],@_[1]);
        if (($opt1 eq "c") || ($opt2 eq "c")) {
            $client = 1;
        }
        if (($opt1 eq "s") || ($opt2 eq "s")) {
            $server = 1;
        }
        if ( ( ($opt1 eq "c") || ($opt1 eq "s") || ($opt1 eq "") ) && ( ($opt2 eq "c") ||\\
        ($opt2 eq "s") || ($opt2 eq "") ) && ( ($opt1 ne "") || ($opt2 ne "") ) ) {
            1;
        } else {
            0;
        }
    }
}

#
# Open_ldap_connection: open a connection with the LDAP server
# @_[0]: LDAP server
#
sub Open_ldap_connection{
    if ( $verbose ) { print "Connecting to LDAP server..."; }
    $ldap = Net::LDAP->new(@_[0]) || die "Problems to connect to LDAP server: $@\n";
    $ldap->bind;
    if ( $verbose ) { print "done.\n" };
```

```
    1;
}

#
# Abort_program: Generates fail message, close and delete the configuration files
#
sub Abort_program{
    print SPD "-- Program fail --\n";
    print IKE "-- Program fail --\n";
    close(SPD);
    close(IKE);
    die "Security policies is not generates. Program fail.\n";
}

__END__

=head1 NAME

slc Security Level Converter

=head1 SYNOPSIS

slc [options]

Options:
--help, --man, --spd <file>, --ike <file>, --sl <file>, --ss <file>, --mode <files|ldap>,\\
server <IP address|hostname>, --verbose|noverbose

=head1 OPTIONS

=over 8

=item B<--help>

Print a brief help message and exit.

=item B<--man>

Print the manual page and exit.

=item B<--spd>

Target file for security policy database (only for KAME). Default: spd.conf

=item B<--ike>
```

```
Target file for IKE SA phase 2 parameters (only for RACOON). Default: ike.conf

=item B<--sl>

Security level definition file (only for files operation mode)

=item B<--ss>

Services and security level attribution file (only for files operation mode)

=item B<--mode>

Choose files or ldap operation mode. Default: ldap

=item B<--server>

IP address or hostname of the LDAP server. Default: localhost

=item B<--verbose|--noverbose>

Prints debug information while processing security level definitions. Default: noverbose

=back

=head1 DESCRIPTION

B<This program> generates IPSec/IKE policies and parameters to protect network services\\
according to security levels specified in local files or on an LDAP server.

=head1 EXAMPLES

=item B<slc --verbose --server 10.1.1.46 --spd setkey.conf --ike racoon.conf>

In this example SLC connect to LDAP server 10.1.1.46 in verbose mode. The configuration\\
files are created as setkey.conf (spd rules) and racoon.conf (ike statements).

=item B<slc --mode files --sl seclevels.conf --ss secserv.conf>

In this example the IPSec security policy definitions are contained in the local files\\
seclevels.conf and secserv.conf. The configuration files are created with the default\\
output file names.

=head1 AUTHORS

Jansen Carlo Sena (jansen.sena@ic.unicamp.br) and Paulo Lício de Geus (paulo@ic.unicamp.br)
=cut
```

Referências Bibliográficas

[1] C. Adams. *The CAST-128 Encryption Algorithm*. Internet Engineering Task Force, RFC 2144, 1997.

[2] Carlisle Adams e Steve Lloyd. *Understanding Public-Key Infrastructure (PKI)*. New-Riders Publishing, 1999.

[3] R. Atkinson. *IP Authentication Header*. Internet Engineering Task Force, RFC 1826, 1995.

[4] Alessandro Augusto, Jansen Carlo Sena, e Paulo Lício de Geus. Security Management of Windows 2000 Networks with DoIt4Me and IPSec. Em *III Simpósio Segurança em Informática*, pp. 11–15, São José dos Campos, São Paulo, Brasil, 2001.

[5] Mihir Bellare, Ran Canetti, e Hugo Krawczyk. Keying Hash Functions for Message Authentication. Em *Advances in Cryptology - CRYPTO '96 Proceedings*, pp. 1–15, Springer-Verlag, 1996.

[6] Steven M. Bellovin. Problem Areas for the IP Security Protocols. Em *Proceedings of the Sixth Usenix UNIX Security Symposium*, San Jose, California, 1996.

[7] Steven M. Bellovin. Probable Plaintext Cryptanalysis of the IP Security Protocols. Em *Proceedings of the Symposium on Network and Distributed System Security*, pp. 155–160, San Diego, California, 1997.

[8] Uyless Black. *Internet Security Protocols*. Prentice Hall, Upper Saddle River, New Jersey, 2 edição, 2000.

[9] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, Upper Saddle River, New Jersey, 3 edição, 1995.

[10] Microsoft Corporation. New Registry Key to Remove LM Hashes from Active Directory and Security Account Manager (Q299656). In URL: http://www.microsoft.com/security, Agosto de 2001.

[11] Joan Daemen e Vicent Rijmen. The Rijndael Block Cipher. In URL: http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, Setembro de 1999.

[12] T. Dierks e C. Allen. *The TLS Protocol Version 1.0.* Internet Engineering Task Force, RFC 2246, 1999.

[13] Hans Dobbertin. The Status of MD5 After a Recent Attack. *RSA Laboratories' Criptobytes*, 2(2), 1996.

[14] Marcin Dobrucki. The Effects of the Transition to IPv6 on Internet Security. Nixu Ltda, Dezembro de 1999.

[15] Matt Blaze et al. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security. A Report by an Ad Hoc Group of Cryptographers and Computer Scientists. In URL: http://www.counterpane.com/keylength.pdf, Janeiro de 1996.

[16] D. Farinacci, T. Li, S. Hanks, D. Meyer, e P. Traina. *Generic Routing Encapsulation (GRE).* Internet Engineering Task Force, RFC 2784, 2000.

[17] Niels Ferguson e Bruce Schneier. A Cryptographic Evaluation of IPsec. Relatório técnico, Counterpane Internet Security, Inc., San Jose, CA, USA, 2000.

[18] R. Fielding e et al. *Hypertext Transfer Protocol – HTTP/1.1.* Internet Engineering Task Force, RFC 2616, 1999.

[19] Sheila Frankel. *Demystifying the IPsec Puzzle.* Artech House, Norwood, Massachusetts, 2001.

[20] A. Frier, P. Karlton, e P. Kocher. The SSL 3.0 Protocol. Netscape Communications Corporation, Novembro de 1996.

[21] Simson Garfinkel e Gene Spafford. *Practical UNIX and Internet Security.* O'Reilly Associates, Sebastopol, California, 2 edição, 1996.

[22] R. Glenn e S. Kent. *The NULL Encryption Algorithm and its Use with IPsec.* Internet Engineering Task Force, RFC 2410, 1998.

[23] G. Good. *The LDAP Data Interchange Format (LDIF) - Technical Specification.* Internet Engineering Task Force, RFC 2849, 2000.

[24] K. Hamzeh. *Point-to-Point Tunneling Protocol (PPTP).* Internet Engineering Task Force, RFC 2637, 1999.

[25] S. Hanks, T. Li, D. Farinacci, e P. Traina. *Generic Routing Encapsulation (GRE).* Internet Engineering Task Force, RFC 1701, 1994.

[26] S. Hanks, T. Li, D. Farinacci, e P. Traina. *Generic Routing Encapsulation over IPv4 Networks.* Internet Engineering Task Force, RFC 1702, 1994.

[27] D. Harkins e D. Carrel. *The Internet Key Exchange (IKE).* Internet Engineering Task Force, RFC 2409, 1998.

[28] John Housley e Tim Polk. *Planning for PKI.* John Wiley Sons, New York, 2001.

[29] Tim Howes, Mark Smith, e Gordon Good. *Understanding and Deploying LDAP Directory Services.* NewRiders Publishing, 1998.

[30] Christian Huitema. *IPv6 The New Internet Protocol.* Prentice Hall, Upper Saddle River, New Jersey, 2 edição, 1997.

[31] M. Karir. IPSEC and the Internet. Tese de Mestrado, University of Maryland, Dezembro de 1999.

[32] S. Kent e R. Atkinson. *IP Authentication Header (AH).* Internet Engineering Task Force, RFC 2402, 1998.

[33] S. Kent e R. Atkinson. *IP Encapsulating Security Payload (ESP).* Internet Engineering Task Force, RFC 2406, 1998.

[34] S. Kent e R. Atkinson. *Security Architecture for the Internet Protocol.* Internet Engineering Task Force, RFC 2401, 1998.

[35] R. Khare e S. Lawrence. *Upgrading to TLS Within HTTP/1.1.* Internet Engineering Task Force, RFC 2817, 2000.

[36] D. V. Klein. Foiling the Cracker: A Security of, and Implications to, Password Security. Em *2nd USENIX Workshop on Security*, pp. 5–14, 1990.

[37] H. Krawczyk. Skeme: A versatile secure key exchange mechanism for internet. Em *IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security*, 1996.

[38] H. Krawczyk, M. Bellare, e R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* Internet Engineering Task Force, RFC 2104, 1997.

[39] C. Madson e N. Doraswamy. *The ESP DES-CBC Cipher Algorithm with Explicit IVx.* Internet Engineering Task Force, RFC 2405, 1998.

[40] C. Madson e R. Glenn. *The Use of HMAC-MD5-96 within ESP and AH.* Internet Engineering Task Force, RFC 2403, 1998.

[41] C. Madson e R. Glenn. *The Use of HMAC-SHA-1-96 within ESP and AH.* Internet Engineering Task Force, RFC 2404, 1998.

[42] D. Maughan e et al. M. Schertler. *Internet Security and Key Management Protocol (ISAKMP).* Internet Engineering Task Force, RFC 2408, 1998.

[43] H. X. Mel e Doris M. Baker. *Cryptography Decrypted.* Addison-Wesley, Reading, Massachusetts, 2000.

[44] A. Menezes, P. van Oorschot, e S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Confirmar, 2 edição, 1996.

[45] P. Metzger e W. Simpson. *IP Authentication using Keyed MD5.* Internet Engineering Task Force, RFC 1828, 1995.

[46] P. Metzger e W. Simpson. *IP Authentication using Keyed SHA.* Internet Engineering Task Force, RFC 1852, 1995.

[47] H. Orman. *The OAKLEY Key Determination Protocol.* Internet Engineering Task Force, RFC 2412, 1998.

[48] H. Orman. *The Use of HMAC-RIPEMD-160-96 within ESP and AH.* Internet Engineering Task Force, RFC 2857, 2000.

[49] B. Patel, B. Aboba, W. Dixon, G. Zorn, e S. Booth. *Securing L2TP using IPsec.* Internet Engineering Task Force, RFC 3193, 2001.

[50] D. Piper. *The Internet IP Security Domain of Interpretation for ISAKMP.* Internet Engineering Task Force, RFC 2407, 1998.

[51] V. Rawat, R. Tio, S. Nanji, e R. Verma. *Layer Two Tunneling Protocol (L2TP) over Frame Relay.* Internet Engineering Task Force, RFC 3070, 2001.

[52] E. Rescorla. *HTTP Over TLS.* Internet Engineering Task Force, RFC 2818, 2000.

[53] E. Rescorla e A. Schiffman. *The Secure HyperText Transfer Protocol.* Internet Engineering Task Force, RFC 2660, 1999.

[54] R. Rivest. *The MD5 Message-Digest Algorithm.* Internet Engineering Task Force, RFC 1321, 1992.

[55] R. Hinden S. Deering. *Internet Protocol Version 6*. Internet Engineering Task Force, RFC 2460, 1998.

[56] Bruce Schneier. *Applied Cryptography*. John Wiley Sons, New York, 2 edição, 1996.

[57] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley Sons, New York, 2000.

[58] Bruce Schneier e P. Mudge. Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP). Em *5th ACM Conference on Computer and Communications Security*, pp. 132–141, San Francisco, California, 1998.

[59] Bruce Scnheier e et al. Performance Comparision of the AES Submissions. In URL: http://www.counterpane.com/aes-performance.pdf, Fevereiro de 1999.

[60] Jansen Carlo Sena e Paulo Lício de Geus. Um Mecanismo para Estabelecimento de Associações de Segurança Baseado em Categorias de Serviço. Em *III Simpósio Segurança em Informática*, pp. 193–202, São José dos Campos, São Paulo, Brasil, 2001.

[61] Jansen Carlo Sena e Paulo Lício de Geus. A Protection Model for Network Communications Based on Security Levels. Em *2002 International Conference on Security and Management (SAM'02)*, Las Vegas, Nevada, USA, 2002.

[62] Jansen Carlo Sena e Paulo Lício de Geus. Uma Ferramenta para Proteção do Tráfego de Serviços Utilizando o IPSec. Em 20^o *Simpósio Brasileiro de Redes de Computadores, WSeg2002 (Workshop em Segurança de Sistemas Computacionais)*, Búzios, Rio de Janeiro, Brasil, 2002.

[63] Jansen Carlo Sena, Paulo Lício de Geus, e Alessandro Augusto. Impactos da Transição e Utilização do IPv6 sobre a Segurança de Ambientes Computacionais. Em 20^o *Simpósio Brasileiro de Redes de Computadores, WSeg2002 (Workshop em Segurança de Sistemas Computacionais)*, Búzios, Rio de Janeiro, Brasil, 2002.

[64] W. Simpson. *The Point-to-Point Protocol (PPP)*. Internet Engineering Task Force, RFC 1661, 1994.

[65] Eugene H. Spafford. The Internet Worm Incident. Relatório Técnico CSD-TR-933, Department of Computer Sciences, Purdue University, West Lafayette, IN USA 47907-2004, 1991.

[66] P. Srisurech. *Secure Remote Access with L2TP*. Internet Engineering Task Force, RFC 2888, 2000.

[67] William Stallings. *Cryptography and Network Security*. Prentice Hall, Upper Saddle River, New Jersey, 2 edição, 1998.

[68] W. Richard Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley, Reading, Massachusetts, 2 edição, 1996.

[69] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, New Jersey, 3 edição, 1996.

[70] W. Townsley. *Layer Two Tunneling Protocol (L2TP)*. Internet Engineering Task Force, RFC 2661, 1999.

[71] David Wagner e Bruce Schneier. Analysis of the SSL 3.0 Protocol. In URL: http://www.cs.berkeley.edu/ daw/ssl3.0.ps, 1996.

[72] M. Wahl e T. Howes. *Lightweight Directory Access Protocol (v3)*. Internet Engineering Task Force, RFC 2251, 1997.

[73] Doug Whiting, Bruce Schneier, e Steve Bellovin. AES Key Agility Issues in High-Speed IPsec Implementations. In URL: http://citeseer.nj.nec.com/313958.html.

[74] Yongguang Zhang e Bikramjit Singh. A Multi-Layer IPsec Protocol. Em *9th USENIX Security Symposium*, Denver, Colorado, 2000.

[75] Elizabeth D. Zwicky, Simon Cooper, e D. Brent Chapman. *Building Internet Firewalls*. O'Reilly Associates, Sebastopol, California, 2 edição, 2000.