UMA ANÁLISE DE SEGURANÇA DAS VERSÕES DO PROTOCOLO NFS

Guilherme César Soares Ruppert
Instituto de Computação
Universidade Estadual de Campinas
13084-971 Campinas - SP
guilherme.ruppert@ic.unicamp.br

Paulo Lício de Geus
Instituto de Computação
Universidade Estadual de Campinas
13084-971 Campinas - SP
paulo@ic.unicamp.br

RESUMO

Este trabalho tem a finalidade de levantar os recursos e as falhas de segurança dos protocolos NFSv2 e NFSv3, bem como apresentar os mecanismos de segurança propostos na versão 4, a fim de avaliar se o NFSv4 aborda todas as falhas de segurança presentes nas versões anteriores. O trabalho conclui que o NFSv4 apresenta mecanismos de criptografia forte que garantem autenticação, privacidade e integridade dos dados, porém algumas deficiências de menor perigo ainda permanecem.

ABSTRACT

The aim of this work is to point out resources and security vulnerabilities of the NFSv2 and NFSv3 protocols, as well as to present the security mechanisms proposed in version 4, so as to evaluate whether NFSv4 properly deals with all security vulnerabilities of the previous versions. The conclusion is that NFSv4 presents strong cryptographic mechanisms which provide authentication, privacy and data integrity, however a few issues remain open.

1 INTRODUÇÃO

O NFS – *Network File System* (Sistema de Arquivos em Rede) foi originalmente projetado e implementado pela *Sun Microsystems* e endossado por companhias como: IBM, *Hewlett-Packard*, *Apollo Computer*, *Apple Computer* e muitos outros fornecedores de sistemas Unix.

Trata-se de um protocolo para acesso a arquivos remotos. Esse protocolo é certamente o mais utilizado para se obter compartilhamento de arquivos através da rede em ambientes Unix, sendo que todos os sabores de Unix possuem implementação desse protocolo.

A idéia principal do projeto do NFS é prover acesso transparente para o usuário aos arquivos remotos, ou seja, as aplicações acessam arquivos que estão remotos utilizando a mesma semântica Unix de acesso a arquivos locais. Para alcançar esse objetivo, o sistema de arquivos remoto é montado em algum ponto da hierarquia local de arquivos e, a partir daí, é tratado como um arquivo local.

Existem atualmente 4 versões do protocolo NFS. A primeira versão do NFS foi um protótipo que existiu apenas internamente à *Sun Microsystems* e nunca foi disponibilizada publicamente. A segunda versão (NFSv2) desse protocolo foi implementada e disponibilizada inicialmente no SunOS 3.2 em 1985. O NFSv2 foi amplamente utilizado, no entanto, essa versão apresentava algumas limitações, que só poderiam ser contornadas com a especificação de uma nova versão. Problemas como: limitação do tamanho do arquivo para 4GB, performance inadequada devido à escrita síncrona e falta de garantia de consistência levaram à necessidade de uma nova versão.

Em Junho de 1995 foi concluída a especificação do NFSv3 e, em alguns anos, todas as implementações de Unix suportariam esta nova versão do protocolo. Porém grande parte das implementações atuais do NFSv3 não implementa toda a RFC. Um exemplo disso é a implementação para o Linux, cujo suporte de NFS sobre TCP ainda encontra-se em fase experimental. Além disso, a especificação da versão 3 do NFS já contemplava alguns mecanismos de segurança como autenticação via Kerberos e Diffie-Hellman, porém sem caráter mandatório. O resultado disso é que a grande maioria das implementação não colocou esses quesitos de segurança em prática.

O NFSv3 é a versão existente nos sistemas em produção atualmente, incluindo Linux, FreeBSD e Solaris.

Devido à necessidade de uma melhor performance através da Internet, e também de segurança baseada em esquemas de criptografia forte, um comitê de pesquisadores e empresas começou a trabalhar na especificação da versão 4 do NFS. A especificação final foi concluída em dezembro de 2000 [RFC3010], mas passou por uma revisão [RFC3530] em abril de 2003, apresentando mudanças radicais no protocolo principalmente no aspecto da segurança. Algumas RFCs correlatas ainda estão em desenvolvimento para padronizar assuntos como padronização de ACLs dentro do NFSv4, mecanismo de *cache* de credeciais, migração de hierarquias entre servidores etc.

Existem atualmente apenas 4 projetos de implementações deste protocolo, sendo que um deles é considerado a implementação de referência. Esta implementação foi incluída no kernel oficial do Linux a partir da versão 2.6.

2 OBJETIVOS

Este trabalho tem a finalidade de levantar os recursos e as falhas de segurança do NFSv2 e NFSv3, bem como apresentar os mecanismos de segurança propostos na versão 4, a fim de avaliar se o NFSv4 aborda todas as falhas de segurança presentes nas versões anteriores.

3 NFSv2 E NFSv3

Tanto a versão 3 como a versão 2 do NFS são totalmente baseadas nos protocolos RPC (*Remote Procedure Call*) e XDR (*eXternal Data Representation*) da *Sun Microsystems*. Essas duas versões do NFS são bastante similares porque, na terceira versão, poucas modificações foram feitas por receio de alterar radicalmente um protocolo já consagrado que vinha funcionando satisfatoriamente.

O NFS é na verdade dividido em dois protocolos: o Mount, e o NFS propriamente dito. O primeiro é responsável pela negociação inicial entre cliente e servidor. Através do mount, o cliente pode determinar quais sistemas de arquivos estão disponíveis para montagem e obter um descritor de arquivos, que é utilizado para referenciar esses arquivos remotamente. O segundo protocolo permite que o cliente liste o conteúdo dos sistemas de arquivos exportados e obtenha recursivamente os descritores dos outros arquivos, podendo assim criar, modificar e apagar arquivos.

O NFS baseia-se no modelo de segurança do RPC para garantir a segurança de seus serviços. O servidor NFS checa as permissões utilizando as credenciais da informação de segurança do RPC contida em cada requisição remota.

Iremos agora analisar essas duas versões do NFS levando em conta cada um dos seguintes tópicos de segurança: integridade, privacidade, não-repúdio, controle de acesso e autenticação.

3.1 Integridade, Privacidade e Não-repúdio

Os protocolos NFSv2 e NFSv3, da forma em que foram especificados, utilizam o protocolo RPC versão 2, que não possui qualquer mecanismo que possibilite a garantia da integridade, privacidade ou não-repúdio dos dados. Existem dois campos (*credential* e *verifier*) no cabeçalho do RPC que são dedicados à autenticação, porém todos os dados trafegam pela rede de forma não cifrada.

Portanto, nas versões 2 e 3 do NFS, um atacante pode realizar um ataque conhecido como *sniffing*, que consistem simplesmente em capturar todo o tráfego no cabo de rede e, dessa forma, conseguirá ter acesso de leitura a todos os dados (arquivos e diretórios) acessados naquele instante. De posse das requisições e respostas capturadas em texto claro, o atacante terá o trabalho de interpretar toda a comunicação e remontar os arquivos transferidos. Porém, esse é um ataque bastante viável principalmente para arquivos texto, que são arquivos de maior inteligibilidade.

O atacante também poderá realizar um ataque à integridade da informação. Utilizando um conhecido ataque chamado *man-in-the-middle*, o adulterador poderia interceptar uma mensagem RPC e dessa forma alterar o conteúdo das requisições e respostas do NFS. Com esse ataque pode-se, por exemplo, alterar o conteúdo das chamadas WRITE() em andamento naquele instante e, desse modo, alterar o conteúdo de arquivos gravados no servidor. Algoritmos de detecção de erro como CRC-32, por exemplo, não são úteis nesse caso, pois o adulterador pode refazer o *checksum* com o pacote modificado. Seria necessário fazer uso de algum protocolo criptográfico para autenticar uma assinatura da mensagem. Dessa forma, ter-se-ia certeza de que aquela assinatura foi gerada pelo remetente e, portanto, a integridade da mensagem poderia ser garantida.

O outro problema de segurança apontado é a falta de garantia do não-repúdio. Nas versões 2 e 3 do NFS, um usuário pode negar que tenha realizado certas operações no servidor. Uma situação emblemática desse problema é o caso de um usuário que deleta arquivos importantes do servidor e que poderá negar que o tenha feito. Isso é causado por diversos motivos: falta de registros de log no servidor; mecanismos fracos de autenticação, como será visto mais adiante, e possibilidade de ataques à integridade dos dados.

3.2 Controle de Acesso

O NFSv2 possui um mecanismo simples de controle de acesso baseado nos bits de permissões (atributos) do arquivo. Verificando os bits do arquivo que está sendo acessado, o cliente consegue saber se poderá ou não efetuar a operação. Este método permite um nível bastante razoável de controle de acesso, porém apresenta algumas limitações indesejáveis. Este método não suporta o controle de permissões através de ACLs e dificulta a interoperabilidade com sistemas não-Unix que possuem atributos de permissões diferentes.

No NFS versão 2 a única forma confiável de se checar se uma operação é realmente permitida é tentar realizá-la e checar se a operação tem sucesso. A versão 3 do NFS introduz o procedimento ACCESS() que permite que o cliente possa perguntar ao servidor se uma dada operação é permitida, antes de executá-la. Isso é útil em sistemas operacionais, como o Unix, onde as permissões são checadas quando um arquivo é aberto. Com esse procedimento o servidor pode utilizar qualquer política de permissões (inclusive

ACL´s) pois agora a tarefa de checar as permissões são de responsabilidade do servidor.

3.3 Autenticação

Conforme já foi dito, o NFS utiliza o protocolo RPC e existem dois campos (*credential* e *verifier*) no cabeçalho do RPC que são dedicados à autenticação. É através desses campos que o NFS provê mecanismos de autenticação tanto para o cliente quanto para o servidor.

Diversos protocolos de autenticação podem ser suportados pelo RPC, e um campo no cabeçalho RPC chamado esquema de autenticação (*authentication flavour*) indica qual protocolo será utilizado. A tabela 3.1 mostra os protocolos existentes:

Tabela 3.1 – *Authentication Flavours*

#	Nome	Descrição
0	AUTH_NONE	Ausência de autenticação. Acesso anônimo.
1	AUTH_SYS [1]	O servidor simplesmente confia no UID e GID fornecidos pelo cliente.
3	AUTH_DH [2]	Protocolo baseado no DES e Diffie-Hellman para troca de chaves.
4	AUTH_KERB	Baseado nos protocolos Kerberos 4 e DES para troca de chaves.
6	RPCSEC_GSS	Baseado na GSS_API

[1] Também conhecido como AUTH_UNIX
[2] Também conhecido como AUTH_DES

A única implementação que suporta todos esses esquemas é o NFS da *Sun*. Todas as outras implementações estáveis (inclusive a do Linux) implementam somente AUTH_NONE e AUTH_SYS.

Os esquemas AUTH_DH e AUTH_KERB apresentam características de segurança bastante satisfatórias, mas não foram muito difundidos por uma série de razões, como: problemas com exportação de criptografia americana; dificuldade de configuração do sistema para funcionar com esses esquemas; e grande impacto na performance para os equipamentos da época. Por causa de todos esses problemas e também pelo fato da RFC não especificar esses esquemas mais seguros com caráter mandatório, quase nenhum outro sistema operacional implementou os mesmos.

O último esquema da tabela é o RPCSEC_GSS, que é um esquema que surgiu posteriormente aos demais. Este esquema é utilizado pelo NFSv4 e será visto mais adiante.

Portanto, atualmente o único esquema de autenticação utilizado e presente em todas as implementações de NFS é o AUTH_SYS (também conhecido como AUTH_UNIX).

Iremos agora analisar as falhas desse esquema para autenticar cada um dos personagens ligados ao serviço NFS: o cliente, o servidor e o usuário.

3.3.1 Autenticação do Cliente

O esquema AUTH_SYS provê um mecanismo extremamente fraco para autenticar a máquina cliente. Neste esquema, o servidor autentica o cliente apenas comparando o endereço IP da máquina que fez a requisição com uma lista de máquinas permitidas.

Vamos primeiramente dar uma olhada no funcionamento do NFS, mais especificamente no protocolo Mount. Suponha que o cliente deseja, por exemplo, montar o diretório /home da máquina aracaju (servidor) em /users na máquina blumenau (cliente), o usuário poderá executar o seguinte comando na máquina blumenau:

```
# mount -t nfs aracaju:/home /users
```

O programa mount irá conectar-se com o programa servidor remoto chamado mountd na máquina aracaju via RPC. O servidor irá verificar em /etc/exports se o IP da máquina blumenau tem permissão para montar o diretório em questão, e caso tenha, retorna a ela um descritor de arquivos. Este descritor de arquivos será usado em todas as requisições posteriores aos arquivos sob o diretório users.

Nesta etapa do protocolo existe um problema de segurança referente à autenticação da máquina cliente. A autenticação é baseada apenas no endereço IP do cliente mas sabemos que uma máquina pode facilmente se passar por outra bastando reconfigurar o endereço IP (com o endereços de alguma máquina desligada) ou então realizando um ataque de *spoofing* (falsificação do endereço IP). Outras técnicas de ataque (como negação de serviço, alteração de rotas etc) podem ser necessárias para que o *IP spoofing* tenha sucesso. Isso é bastante inseguro, por exemplo, em um cenário onde um atacante pode conectar seu *laptop* à rede. Basta ele configurar seu endereço IP para o endereço de uma máquina desligada para poder montar qualquer *filesystem* que a máquina impersonada teria acesso.

3.3.2 Autenticação do Servidor

O NFSv3 utilizado atualmente não provê nenhum mecanismo que permita que o cliente autentique o servidor. O cliente não tem como descobrir se o servidor é, de fato, o servidor legítimo, ou se alguma outra máquina está se passando pelo servidor.

Existem diversas formas que o atacante poderia utilizar para impersonar o servidor e explorar a falta de autenticação do NFS. Técnicas como *IP spoofing*, *DNS spoofing* e *man-in-the-middle*, são comprovadamente eficientes para tal objetivo. O atacante poderia utilizar um servidor falso, por exemplo, para receber arquivos ou para disponibilizar arquivos falsos.

3.3.3 Autenticação do Usuário

Quando alguém acessa um arquivo remoto através do NFS, o *kernel* envia uma chamada RPC para o *daemon* nfsd na máquina servidora. Esta chamada leva como parâmetros: o descritor de arquivos, o identificador de usuário (UID) e de grupo (GID). Eles são usados para controlar os direitos de acesso sobre um determinado arquivo. Nessa etapa existe outro problema de segurança: a autenticação do usuário. O servidor não se certifica se o UID recebido na requisição é realmente o UID do usuário que está requisitando a operação. O servidor simplesmente confia na informação contida na requisição e executa a operação solicitada. Existem dois cenários bastante comuns onde essa falta de autenticação representa muita dor de cabeça para os administradores.

O primeiro cenário é uma rede onde os usuários podem ter máquinas pessoais com privilégios de administrador. O servidor NFS possui os *homedirs* dos usuário A e B. O usuário A na máquina A cria localmente um usuário B e utiliza o comando su para trocar para o usuário B. Nesse momento toda requisição ao NFS será identificada como usuário B e, dessa forma, o usuário A terá acesso a todos os arquivos do usuário B.

O segundo cenário é mais alarmante ainda. Em qualquer rede que utilize o serviço NFS, qualquer usuário pode ter acesso total aos arquivos de qualquer outro usuário. Note que neste cenário não há necessidade de ser root (super-usuário). Para realizar isso basta que o usuário A escreva um programa em nível de usuário que envie as mesmas chamadas RPC que o cliente NFS envia, só que se identificando com o UID do usuário B. Como o servidor confia cegamente nas mensagens que recebe, ele aceitará a mensagem e executará a requisição com as credenciais falsas. Dessa forma, o usuário A terá acesso a todos os arquivos do usuário B.[1]

3.4 Conclusões sobre NFSv2 e NFSv3

Existe ainda um outro problema de segurança, que não se encaixa em nenhum dos quesitos anteriores e diz respeito à integração do NFS com *firewalls*. O servidor NFS não utiliza um número de porta fixo, ou seja, não se pode saber de antemão em qual porta o servidor aguarda requisições. Apesar desse serviço estar sempre executando na porta 2049, é possível que ele esteja aguardando em outra porta. Logo, é necessário consultar um outro serviço chamado *portmapper* para se descobrir em qual porta está registrado o serviço NFS. O grande problema disso é que, como o número da porta é variável, fica extremamente difícil projetar regras de filtragem do tráfego de NFS. Sistemas auxiliares seriam necessários para se permitir tráfego NFS através de filtros, como o módulo desenvolvido por Marcelo Lima para incorporação ao framework netfilter/iptables o kernel 2.4 do Linux [LIMA].

Como pudemos notar pela análise dos quesitos, a situação atual do NFSv3 é extremamente preocupante, pois a segurança desse protocolo é praticamente inexistente. O serviço depende de uma série de outros protocolos (NFS, Mount, Portmap, RPC etc), o que facilita o aparecimento de vulnerabilidades, além de não apresentar mecanismos de segurança adequados e mandatórios.

4 NFSv4

Apesar do NFSv3 ter apresentado algumas melhorias em relação à versão 2, o protocolo continuou voltado para redes locais Unix. Em 1993, quando o NFSv3 foi desenvolvido, a Internet estava muito longe se ser o que ela é hoje. Não havia *World Wide Web* e nem comercio eletrônico. Muita coisa mudou de lá pra cá e o NFSv3 não acompanhou esse novo cenário. Algumas modificações para tornar o protocolo mais compatível com a Internet foram introduzidas pelo protocolo WebNFS, que foi uma tentativa de utilizar o NFS para transferir arquivos na Web, através do próprio navegador.

Em 1998 foi formado um grupo de trabalho IETF para trabalhar na versão 4 do NFS e a especificação desse protocolo foi reconhecida como padrão em dezembro de 2000 [RFC3010], mas passou por uma revisão [RFC3530] em abril de 2003.

O NFSv4 apresenta uma ruptura com as versões anteriores. O protocolo foi extremamente modificado, inclusive em questões tidas como alicerces da antiga versão. Exemplo disso é o fato do novo protocolo manter estados das operações no servidor. Na versão antiga, o servidor não mantinha estados (*stateless*). As operações eram independentes e por isso não se fazia necessário qualquer mecanismo de recuperação de falhas. Dentre as principais mudanças no protocolo destacam-se:

- O uso do TCP é mandatório, embora continue suportando o UDP.
- Permite chamadas de RPC compostas, ou seja, várias operações em uma mesma mensagem.
- O NFSv4 combina diferentes protocolos (Mount, NLM, etc) em único protocolo.
- Introduz delegação de arquivos, o que permite mecanismos eficientes de cache.
- Diversos mecanismos de segurança usando criptografia forte em caráter mandatório, com suporte à negociação do nível de segurança.

[1] Este ataque foi primeiramente identificado por Marcos Aguilera durante sua iniciação científica em 1991, no IC-Unicamp

Neste trabalho estaremos analisando apenas as novidades do novo protocolo referentes à segurança.

4.1 RPCSEC_GSS

O NFSv4, assim como as outras versões, baseia-se no modelo de segurança do RPC para garantir a segurança de seus serviços, ou seja, utiliza os esquemas de autenticação da tabela 3.1. Porém, o NFSv4 obriga a implementação do esquema RPCSEC_GSS que é baseado na *Generic Security Services API* (GSS-API) .

A GSS-API é uma interface de programação genérica, que tem por objetivo oferecer serviços para construção de aplicações e serviços de segurança. Um dos objetivos desta interface é obter isolamento entre a camada de aplicação e os serviços de segurança propriamente ditos. Desta forma, é possível trocar os algoritmos de criptografia e autenticação de uma aplicação sem alterar seu código. A figura 4.1 mostra que a GSS-API torna o esquema de segurança transparente para a aplicação, no caso, o RPCSEC_GSS.

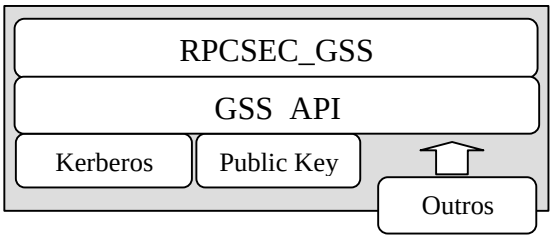

Figura 4.1 – O Esquema RPCSEC_GSS

Diferentemente dos outros esquemas (*flavours*) que só provêm autenticação, o RPCSEC_GSS pode garantir integridade e privacidade do corpo inteiro de uma mensagem RPC. Por esse motivo ele é chamado de um esquema de segurança (*security flavour*) ao invés de um esquema de autenticação (*authentication flavor*) como é chamado o tradicional AUTH_SYS.

Uma implementação compatível com o NFSv4 obrigatoriamente deve implementar dois mecanismos de segurança sob a interface GSS. São eles: Kerberos V5 e LIPKEY.

4.2 Kerberos Version 5

O Kerberos é um protocolo de autenticação que faz uso apenas de criptografia simétrica e consegue garantir o mesmo nível de segurança dos algoritmos de chave pública. Maiores detalhes sobre o funcionamento interno desse algoritmo podem ser encontrados em [RFC1510].

O protocolo Kerberos funciona muito bem dentro de uma Intranet para redes locais, mas não se aplica no caso da Internet, pois exige controle centralizado. Cada domínio deve possuir um KDC (*Key Distribution Center*), que fornece os tickets para os clientes acessarem os serviços dentro do mesmo domínio. O KDC autentica os usuários e a aplicação que fornece o serviço. Para isso ele deve possuir uma base de dados com as chaves secretas de todos os usuário e serviços. Como normalmente os clientes NFS são implementados em nível de *kernel*, os *tickets* ficam armazenados no *kernel* e, toda vez que um usuário desejar acessar o serviço, seu respectivo ticket será utilizado.

O mecanismo do Kerberos usado na GSS-API [RFC1964] especifica 3 modos de garantir integridade. No NFSv4 foi escolhido o algoritmo DES/MAC/MD5 que consiste em computar o DES-CBC MAC em paralelo com o MD5. Desse modo o NFSv4 consegue garantir robustamente a integridade das mensagens.

Quanto à privacidade, o mecanismo Kerberos V5 GSS_API prevê a utilização do DES (56 bits). A própria RFC do NFSv4 alerta para o fato deste algoritmo não ser robusto a ataques de força bruta nos dias atuais. Portanto, o Kerberos v5 no NFS provê uma garantia fraca quanto à privacidade dos dados.

O Kerberos não autentica o *host* servidor, mas apenas a aplicação que oferece o serviço. Porém isso não é preocupante, pois, para impersonar o serviço em outro *host*, o atacante precisaria possuir a chave secreta do serviço legítimo para assim poder decifrar o ticket e obter a chave de sessão. Como a chave secreta do serviço é um segredo de responsabilidade do administrador do sistema, pode-se considerar pouco relevante esse problema.

Um ponto fraco do Kerberos para sua utilização no NFS é o fato dele autenticar apenas o usuário e não autenticar o *host* cliente. As implementações de NFS atualmente possuem filtros para selecionar quais máquinas terão acesso aos sistemas de arquivos exportados. Esses filtros continuarão baseando-se apenas no endereço IP para autenticar o *host* cliente. Portanto, o uso de Kerberos não resolve o problema da falta de autenticação do cliente apontado no item 3.3.3. O administrador não teria como garantir, por exemplo, que certas máquinas não confiáveis (com possíveis vírus, segurança frágil, máquinas de acesso público) sejam impossibilitadas de acessar o serviço NFS.

4.3 LIPKEY

O LIPKEY (*Low Infrastructure Public Key*) é um outro mecanismo de segurança que deve ser fornecido numa implementação NFSv4-compatível. Este mecanismo provê um modelo e segurança equivalente ao SSL (*Secure Socket Layer*) para utilização do NFS através da Internet.

O cliente recebe o certificado do servidor e compara com uma lista de autoridades certificadoras pré-definida. Caso o certificado tenha sido assinado por uma autoridade confiável então o serviço é considerado legítimo. Então o cliente

envia a senha do usuário e uma chave de sessão simétrica (gerada aleatoriamente), ambas cifradas com a chave pública do servidor. O servidor pode então autenticar o usuário baseado em sua senha de acesso ao serviço.

Pode também garantir privacidade dos dados, pois a partir dessa etapa todos os dados serão cifrados através da chave simétrica fornecida. O NFSv4 obriga a implementação do algoritmo cast5CBC para garantir a confidencialidade, embora seja possível negociar outros algoritmos. Este algoritmo é considerado bastante robusto para o estado da arte atual portanto no aspecto da privacidade o LIPKEY é bastante adequado.

Quanto à integridade dos dados, a especificação do NFSv4 permite que sejam negociados diversos algoritmos, mas requer que o algoritmo HMAC-MD5 esteja sempre disponível. Este também é um algoritmo considerado robusto para garantir integridade dos dados, portanto nesse aspecto o LIPKEY é seguro.

O LIPKEY apresenta o mesmo problema de autenticação que o Kerberos levantado anteriormente. O LIPKEY autentica apenas o usuário e o serviço (do lado do servidor), mas as máquinas cliente e servidora não o são.

4.4 Negociação da Segurança

Tendo em vista que o NFSv4 pode oferecer diversos mecanismos de segurança, o cliente precisa de algum método para negociar qual o mecanismo de segurança a ser usado na comunicação com o servidor. Um servidor que exporta diversos pontos de montagem pode especificar quais os algoritmos adequados para cada uma dessas entradas.

Para realizar essa negociação, foi introduzida uma nova operação no NFS chamada SECINFO, que permite que o cliente pergunte qual a segurança que o servidor requer para acessar determinado diretório. Os argumentos e resultados dessa operação devem ser segurados utilizando um dos esquemas de segurança mandatórios. Isso é para que o atacante não consiga interceptar a negociação e forçar o uso de níveis mais baixos de segurança.

A operação SECINFO retorna a lista (ordenada por prioridades) de esquemas de segurança suportados pelo servidor e também, dependendo do esquema, qualquer informação adicional necessária. Por exemplo, no caso do RPCSEC_GSS o servidor e o cliente precisam negociar o mecanismo a ser utilizado e também se será usado integridade ou privacidade.

A partir da resposta do SECINFO o cliente escolhe um esquema dentro da lista dos possíveis e inicia a execução do procedimento remoto. Note que o NFSv4 obriga que toda implementação NFSv4-compatível suporte determinados algoritmos de criptografia forte e que os mesmo tenham prioridade sobre os algoritmos de baixa ou nenhuma segurança. Portanto, o cliente não poderá escolher um esquema inseguro, como o AUTH_SYS pode exemplo, a mesmo que o servidor só permita este, pois haverá sempre um esquema mais seguro com maior prioridade disponível no servidor.

4.5 Controle de Acesso

No NFSv4 os atributos de um arquivo são variáveis. Existem atributos mandatórios e recomendados. Dentre os atributos recomendados há quatro que são referentes ao controle de acesso. São eles: User, Group, Unix mode bits e ACL.

Os três primeiros atributos são os atributos já existentes nas versões antigas do NFS. Com a diferença que agora o usuário e o grupo são definidos por uma string e não por um inteiro. Isso implica no fato de que o mapeamento de UID e GID para nomes não precisa ser o mesmo nas máquinas cliente e servidor.

O último atributo refere-se à utilização de ACLs – *Access Control Lists* (Listas de Controle de Acesso). Uma ACL é simplesmente uma lista que descreve quais os usuários e grupos que têm acesso a determinados serviços, sob certas restrições. O NFS versão 4 introduz suporte a ACL baseado no modelo do Windows NT e não no padrão POSIX, pois o modelo das ACLs do NT é muito mais completo e mais difundido.

Cada entrada na ACL é chamada de ACE (*Access Control Entry*) e pode definir quatro ações diferentes: ALLOW, DENY, AUDIT, or ALARM. As ações ALLOW e DENY permitem ou negam respectivamente o acesso ao arquivo e são suportadas pela POSIX ACL. A operação AUDIT faz com que seja gerado um log da tentativa de acesso ao objeto. ALARM um alarme no sistema caso seja feito um acesso a esse objeto.

Existe um grande diferença entre o modelo NT e o POSIX que representa uma potencial incompatibilidade entre os dois modelos. No modelo NT, a primeira ACE que permite (ou barra) o acesso correspondente a um usuário ou grupo decide se o acesso será permitido ou negado. No modelo POSIX existem dois tipos de ACLs: entradas de usuário e entradas de grupo. Primeiramente são checadas todas as entradas de usuários e posteriormente a dos grupos. Porém todas as ACE´s são verificadas. Por essa e outras diferenças, a compatibilidade dos sistemas que utilizam POSIX ACL com o NFSv4 será problemática. Por esse motivo, já está sendo desenvolvida uma RFC exclusiva para especificar como deve ser feito o mapeamento de ACLs entre POSIX e NFSv4.

Portanto, o mecanismo de ACLs do NFSv4 é um método bastante adequado para controle de acesso porém a utilização deste recurso é problemática, tendo em vista que os diferentes sistemas operacionais adotaram padrões diferentes e incompatíveis de ACLs. Para facilitar essa situação

está sendo desenvolvida uma RFC especificamente para tratar deste problema de compatibilidade.

4.6 Outros aspectos de segurança

O NFSv4 eliminou a necessidade do *port mapper* para descobrir a porta onde o servidor aguarda conexões. O novo protocolo fixa a porta 2049 como a porta padrão do NFS versão 4. Essa mudança, aliada ao uso do TCP, favorece o uso do NFS na Internet e facilita a filtragem desse protocolo através de *firewalls*.

5 CONCLUSÃO

O NFSv4, sob o ponto de vista da segurança, representa um avanço extraordinário em relação às versões anteriores.

Há mecanismos para garantir autenticação, integridade e privacidade, de forma bastante confiável. Apesar de não haver autenticação da máquina cliente e servidora, isso não é tão preocupante, pois há a autenticação do usuário e da aplicação servidora. Dessa forma é possível que um usuário legítimo obtenha acesso a partir de uma máquina não permitida, mas isso pode ser impedido com o uso de *firewalls*.

Há avanços significativos também no aspecto de controle de acesso aos arquivos utilizando ACLs, porém a ampla compatibilidade desse recurso ainda é duvidosa.

Um aspecto bastante negativo do NFSv4 é a sua complexidade. A especificação do protocolo é bastante extensa e incompleta e, além disso, muitas informações são referenciadas de outras RFCs, como, por exemplo, informações sobre os mecanismos e protocolos de segurança. Diversos assuntos como replicação e migração de servidores estão em aberto. Muitos algoritmos e mecanismos de segurança são especificados como mandatórios para que a implementação seja considerada NFSv4-compatível. A grande complexidade do protocolo pode dificultar bastante a sua ampla utilização, tendo em vista que torna as implementações muito caras e demoradas.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ADAMSON, William A.; SMITH, Kendrick M. *Linux NFS Version 4: Implementation and Administration.* In: Proceedings of the 2001 Linux Symposium, Ottawa, 2001.

ADAMSON, William A. *Report on progress of Linux & FreeBSD NFSv4 work done at CITI*. In: 10th Connectathon Conference, 2004

CALLAGHAN, Brent. *NFS Illustrated*, Addison-Wesley, 2000.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. *Distributed Systems - Concepts and Design*, 3rd Edition. Addison-Wesley, 2001.

EISLER, Mike; SCHEMERS, Roland; SRINIVASAN, Raj. *Security Mechanism Independence in ONC RPC.* In: Sixth USENIX Security Symposium, Pp. 51–66 of the Proceedings, 1996

LIMA, Marcelo; GEUS, Paulo Lício. *Filtragem com estados de Serviços Baseados em RPM no Kernel do Linux*, In: 2º Simpósio Segurança em Informática, 2000

NEUMAN, Clifford; TSO, Theodore. *Kerberos: An Authentication Service for Computer Networks*, In: IEEE Communications 32(9), 1994

PAWLOWSKI, Brian; SHEPLER, Spencer; CALLAGHAN, Brent; et al; *The NFS Version 4 Protocol.* In: 2nd International SANE Conference, Maastricht, Países Baixos, 2000

PAWLOWSKI, Brian; JUSZCZAK, Chet ; STAUBACH, Peter; et al. *NFS Version 3 Design and Implementation.* In Proceedings of the Summer USENIX Technical Conference, p. 137-52, June 1994.

POW, Keesje Duarte. *Segurança na arquitetura TCP/IP: de firewalls à canais seguros.* Dissertação de Mestrado, IC/Unicamp, 1999

RFC1094, *NFS: Network File System Protocol Specification*, 1989

RFC1510, *The Kerberos Network Authentication Service (V5)*, 1993

RFC1813, *NFS Version 3 Protocol Specification*, 1995

RFC2624, *NFS Version 4 Design Considerations*, 1999

RFC1831, *RPC: Remote Procedure Call Protocol Specification Version 2*, 1995

RFC1964, *The Kerberos Version 5 GSS-API Mechanism*, 1996

RFC2025, *The Simple Public-Key GSS-API Mechanism (SPKM)*, 1996

RFC2203, *RPCSEC_GSS Protocol Specification*, 1997

RFC2623, *NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5*, 1999

RFC2847, *LIPKEY - A Low Infrastructure Public Key Mechanism Using SPKM*, 2000

RFC3530, *NFS version 4 Protocol*, 2003

SCHNEIER, Bruce. *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd Edition, Wiley, 1996

SMITH, Richard. *Authentication: From Passwords to Public Keys*, 1st Ed., Addison-Wesley, 2002

SPADAVECHIA, Joseph; ZADOK, Erez *Enhancing NFS Cross-Administrative Domain Access*, In: USENIX Annual Technical Conference, 2002

TRAEGER, Avishay; WRIGHT, Charlers; ZADOK, Erez; RAI, Abhishek. *NFS File Handle Security*, Technical Report, Stony Brooks University, 2003