Marcelo Giovani Mota Souza

Alternativa Simples e Segura ao NFS e Samba

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Dra. Marluce Rodrigues Pereira

Lavras
Minas Gerais - Brasil
2009

Marcelo Giovani Mota Souza

Alternativa Simples e Segura ao NFS e Samba

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *20 de Abril de 2009*

Prof. Herlon Ayres Camargo

Prof. Denilson Vedoveto Martins

Dra. Marluce Rodrigues Pereira
(Orientadora)

Lavras
Minas Gerais - Brasil
2009

Dedico esta monografia à minha esposa e aos meus companheiros de trabalho.

Dedico também a todos que contribuem de alguma forma com o movimento do software livre, seja criando, ensinando ou simplesmente usando, afinal, "todos moramos sob o mesmo sol; todos caminhamos sob a mesma lua; todos vivemos sob o mesmo céu; todos olhamos para as mesmas estrelas".

(Klaus Meine e Bruce Fairbairn)

Agradecimentos

Agradeço a todos que me apoiaram, especialmente aos meus eternos orientadores que me guiam pelas jornadas acadêmicas. Ao Prof. Nilton Alves Jr., pela compreensão, disposição de equipamentos e de laboratórios.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A utilização de sistemas de arquivos compartilhados pela rede via NFS e Samba apresenta problemas de segurança, não garantindo, por exemplo, a privacidade dos arquivos dos usuários. Este trabalho apresenta uma proposta de solução para o problema de segurança no uso de sistemas de arquivos compartilhados através de um conjunto de *scripts* em *bash* e configurações do sistema. Este pacote de *scripts* e configurações é denominado SSHomeFS e utiliza ferramentas *open source* nativas na maioria das distribuições Linux (SSH e SSHFS), além de recursos de *shell script* e modificações simples no sistema operacional. Mais especificamente, neste trabalho é proposto o compartilhamento seguro dos diretórios HOME dos usuários de uma rede do CBPF (Centro Brasileiro de Pesquisas Físicas), propondo um uso ainda não comum para o serviço SSH.

Palavras-Chave: Sistema de arquivos compartilhado; Linux; Open Sources; SSHFS; SSHomeFS

Capítulo 1

Introdução

O modelo de rede de computadores, composto por estações de trabalho, que se autenticam em servidores para fornecer aos usuários os recursos computacionais de uma rede, requer basicamente dois serviços: em primeiro lugar um serviço de autenticação remota de usuários; em segundo lugar um serviço de compartilhamento de arquivos pela rede.

Com o avanço da tecnologia de *hardwares* e *softwares*, aumentam exponencialmente a facilidade e as formas de se acessar uma rede indevidamente. Como exemplos de tecnologias recentes que facilitam o acesso indevido a uma rede podem ser citados BIOS que permite boot pela rede ou USB, Hds e CDROMs com interface USB, gavetas USB que suportam Hds e CDROMs IDE/SATA, sistemas operacionais completos capazes de serem executados direto de uma imagem de CD, interconectividade com tecnologias de redes de menor segurança (wireless), dentre outras.

O sistema de autenticação de usuários mais comumente encontrado nas distribuições Linux, o *Network Information Service* (NIS) (KUKUK, 2007) possui algumas diretivas de segurança. Este serviço possui nativamente suporte a senhas do tipo *shadow*, permite consulta somente para o usuário administrador da estação e impõe o conhecimento do "domínio NIS" da rede para consultas. Ainda que as restrições de segurança do serviço NIS não sejam muitas, outras opções à este serviço se encontram disponíveis facilmente, como o NIS+ (KUKUK, 2006) e o *Lightweight Directory Access Protocol* (LDAP) (OPENLDAP, 2008) que, além de oferecerem maiores recursos de segurança, são largamente usados e conhecidos atualmente.

O compartilhamento de arquivos mais simples e mais comumente usado, nativo da grande maioria das distribuições Linux, é o *Network File System* (NFS) (SMITH, 2006b). Este protocolo oferece uma forma rápida e simples de compartilhar arquivos, priorizando desempenho e aproveitamento da banda. A segurança do NFS é baseada na confiabilidade da máquina que solicita dados e não no usuário, podendo ser facilmente burlada.

O serviço de compartilhamento de arquivos SAMBA (SAMBA, 2009) também oferece uma forma de compartilhar arquivos pela rede com algumas opções de configuração de segurança. Por se tratar de um protocolo fechado, o SMB recebe poucas atualizações por parte do desenvolvedor e brechas de segurança podem permanecer sem correções por longos períodos de tempo. Sob estas circunstâncias, uso de um protocolo aberto é amplamente recomendado por fornecer a possibilidade de correções de segurança à comunidade desenvolvedora.

Neste cenário, este trabalho pretende oferecer um recurso alternativo, de fácil implementação e manutenção simplificada, para o compartilhamento dos diretórios HOME dos usuários de uma rede com autenticação remota de usuários. O uso de ferramentas open sources foi privilegiado neste trabalho e a absoluta maioria das ferramentas aqui citadas são nativas das grandes distribuições Linux.

1.1 Motivação

A realização deste trabalho foi motivada por dois principais fatores. Em primeiro lugar, a necessidade de implementação de segurança nas trocas de arquivos entre estações clientes e servidores da rede da pós-graduação do Centro Brasileiro de Pesquisas Físicas (CBPF), para assegurar a privacidade dos arquivos de teses e dados de pesquisas dos usuários, aproveitando a estrutura e os serviços já presentes na rede. A rede da pós-graduação apresenta um ambiente misto onde alguns usuários possuem máquinas clientes que necessitam de autenticação para uso e outros usuários são administradores de suas próprias máquinas. O segundo ponto motivador foi a baixa oferta de uma solução simples e segura para o problema de compartilhamento dos diretórios HOME em uma rede com ferramentas que fossem do uso cotidiano de administradores Linux.

1.2 Objetivos

O objetivo principal deste trabalho é oferecer uma forma segura de compartilhamento dos diretórios HOME dos usuários, fazendo uso apenas de *softwares open sources*, tendo quase sua totalidade presente na maioria das grandes distribuições Linux, além de avaliar a possibilidade do uso do protocolo SSH como provedor de acesso remoto aos diretórios HOME de usuários.

A solução para alcançar este objetivo foi a criação de um conjunto de aplicativos e configurações que possibilitassem a identificação do usuário durante o processo de *login* para o sistema de arquivos remoto, automatizando o processo de "liberação" de seu diretório HOME para a estação solicitante mediante *login*/senha. Esse conjunto de aplicativos e configurações foi estruturado e desenvolvido pelo autor, utilizando programação em *scripts* de *Bash*, e será chamado e referenciado em todo esse trabalho por SSHomeFS.

Foram descartadas as alternativas de compartilhamento de arquivos, como a integração do NFS4 com o *Kerberos*(KERBEROS, 2009) e o *Andrew File System* – AFS(CENTER, 2009), por se tratarem de implementações de serviços e *software* menos difundidos que o serviço de SSH, não possuírem nativamente todos os pacotes necessários na instalação padrão das distribuiões Linux, além de exigirem mais trabalho e maiores conhecimentos dos administradores para as implementações.

Para garantir a segurança dos dados trafegados pela rede foi escolhido o serviço de SSH para servir arquivos, por ser um serviço do cotidiano de administradores Linux e de fácil manuseio. De acordo com a seção 9 da RFC 4251 (YLONEN, 2006), o serviço de transporte de arquivos do protocolo SSH oferece um canal confidencial sob uma rede insegura.

Os *softwares* usados e configurações necessárias no sistema SSHomeFS estão enumerados abaixo.

a) *Software* OpenSSH (YLONEN, 2009a) – Open Source; Presente de forma nativa em quase a totalidade das grandes distribuições Linux.

b) *Software* SSHFS (SZEREDI, 2008) – Open Source; De fácil instalação, com dependências nativamente encontradas em quase a totalidade das

grandes distribuições Linux.

c) Recursos de *Shell Script* em *Bash* (GNU, 2006) – Open Source; Interpretador *shell* padrão das distribuições Linux, presente de forma nativa nas distribuições Linux.

d) Ajustes no sistema – Modificações em arquivos textos, feitas com qualquer editor de texto, alterando e adicionando comportamentos aos sistemas Linux servidor e cliente.

Os resultados obtidos podem ser vistos no capítulo 5 deste trabalho, bem como dados relevantes a desempenho e particularidades encontradas.

1.3 Estrutura do trabalho

O presente trabalho está dividido em seis capítulos incluindo esta introdução, que expõe o problema, o cenário onde este problema se aplica e também os objetivos propostos. Adicionalmente a estes capítulos, estão presentes dois apêndices. Um apêndice contem o código fonte dos *softwares* do pacote SSHomeFS. O outro apêndice é dedicado a exibir de forma condensada os testes do sistema para a implementação do SSHomeFS e procedimentos a serem executados em caso de necessidade de reparos ou ajustes gerais.

O Capítulo 2 exibe informações sobre sistemas de arquivos, local e de rede, apresentando os sistemas de arquivos de rede mais usados. Neste capítulo é exposto também as fragilidades de segurança dos sistemas de arquivos em questão e formas de exploração das mesmas, bem como exemplos e procedimentos para tal exploração.

O Capítulo 3 aborda o serviço SSH, mais especificamente o serviço prestado pelo *software* OpenSSH, suas características, configurações e recursos. Aborda também o *software* SSHFS, exibindo informações para instalação, configuração e recursos avançados do mesmo.

O Capítulo 4 apresenta o sistema SSHomeFS. Exibe tecnicamente os objetivos do sistema, as dificuldades encontradas e as formas de superá-las, as modificações necessárias no sistema, a instalação dos scripts, além de informações individuais

dos três componentes do pacote SSHomeFS.

No Capítulo 5 são expostos os resultados obtidos, índices quantitativos de testes de desempenho e informações técnicas de problemas encontrados.

O Capítulo 6 apresenta uma análise final e as conclusões do trabalho, além de apresentar sugestões para trabalhos futuros.

Capítulo 2

Sistemas de arquivos

Este Capítulo apresenta as características dos sistemas de arquivos local e de rede, destacando o NFS e Samba.

Pela definição da Conectiva Linux, “Sistema de Arquivos - É o método de armazenamento das informações no disco rígido. Diferentes sistemas operacionais usam diferentes sistemas de arquivo, dificultando um pouco o compartilhamento do conteúdo. Em sua maioria os sistemas de arquivo são estáticos, mas outros são dinâmicos, como o sistema de arquivos */proc*, que produz os dados dinamicamente a partir dos dados disponíveis no kernel.” (CONECTIVA, 2004)

Para Smith (SMITH, 2004), sistema de arquivos é “um conjunto de estruturas de dados que permite a Linux localizar e manipular arquivos”, incluindo em “conjunto de estruturas de dados” os aplicativos fornecidos pelo sistema operacional para a manipulação dos arquivos.

De acordo com o Guia Foca Linux, um sistema de arquivos “É criado durante a formatação da partição de disco... Após a formatação toda a estrutura para leitura/gravação de arquivos e diretórios pelo sistema operacional estará pronta para ser usada... Cada sistema de arquivos tem uma característica em particular mas seu propósito é o mesmo: Oferecer ao sistema operacional a estrutura necessária para ler/gravar os arquivos/diretórios.” (SILVA, 2007)

Pode-se entender um sistema de arquivo, então, como sendo o conjunto de regras usadas para acessar (leitura ou escrita) um dado ou conjunto deles em uma unidade de armazenamento.

2.1 Sistema de arquivos local

Existem inúmeras formas de manipular leitura e gravação de dados em uma unidade de armazenamento. As unidades de armazenamento mais comumente encontradas são os Hds, floppy disk, CDROM e pen-drives.

Independente da forma com que a unidade de armazenamento se comunica com o computador, por meio de interface USB, IDE, SATA, SCSI ou outra, o sistema operacional precisa saber todo o procedimento para resgatar ou gravar uma informação neste local.

Sistemas de arquivos locais são conjuntos de regras para acesso de leitura ou escrita a dados em uma unidade de armazenamento local. O sistema operacional manipula os dados diretamente no local de armazenamento.

O conjunto de regras para manipular dados em uma unidade local pode tornar os sistemas de arquivos mais complexos ou menos. Isso pode refletir diretamente no desempenho do sistema de arquivo, na confiabilidade de resgate dos dados gravados, na confiabilidade da gravação de dados na unidade armazenadora ou até mesmo nas possibilidades de inserir restrições ao acesso/gravação de dados. Sistemas de arquivos mais modernos suportam nativamente cota de disco e ACLs.

Exemplos de sistemas de arquivos locais são: FAT32, FAT16, FAT12 (NETWORK, 2009b), Ext2 (CARD; TS'O; TWEEDIE, 2009), Ext3 (TWEEDIE, 2001), Ext4 (CALLEJA, 2009), ReiserFS (CALLEJA, 2003), Reiser4 (MACDONALD; REISER; ZAROCHENTCEV, 2002), XFS (SGI, 2009), HFS, JFS (L.P, 2008),NTFS (NETWORK, 2009a), ZFS (MICROSYSTEMS, 2009), BtrFS (MASON, 2007), Tux3 (PHILLIPS *et al.*, 2007).

2.2 Sistema de arquivos de rede

Dando continuidade ao conceito de sistema de arquivos exposto no início deste capítulo, pode-se entender sistemas de arquivos de rede como sendo uma forma de abstração, emulação ou virtualização de um sistema de arquivos, acessado por meio de um dispositivo de rede, possibilitando ao usuário formas de manuseio como se os arquivos estivessem armazenados localmente.

O acesso pela rede a arquivos localizados em outros hosts pode ocorrer de inúmeras formas. As solicitações aos arquivos podem ser encapsuladas em pacotes

TCP, em pacotes UDP, pode-se usar chamadas de processos remotos ou outras formas quaisquer de solicitação de um arquivo ao sistema.

Tecnologias mais recentes como o AoE (HOPKINS; COILE, 2009) e o iSCSI (SATRAN *et al.*, 2004) permitem um nível mais baixo de acesso ao *hardware* remoto. Com o uso destes protocolos é possível ter acesso a uma partição ou até mesmo a um disco inteiro remotamente, podendo ser formatado com sistemas de arquivos locais como o Ext4, Reiser4, BtrFS ou outros. Estas tecnologias possibilitam o acesso remoto direto ao *hardware* e não aos arquivos. Tais tecnologias possuem características especiais e são recomendadas a situações específicas de uso. O acesso remoto de baixo nível a dispositivos de armazenamento não pertence ao escopo desta obra e não será aprofundada em nenhum dos capítulos deste trabalho.

2.2.1 Características gerais

Um sistema de arquivos de rede proporciona ao usuário a forma mais simples de acesso a arquivos por meio de um compartilhamento de rede, podendo manipular estes arquivos com as operações mais básicas do sistema operacional como se estes arquivos estivessem presentes localmente.

Como qualquer serviço de rede, a inserção de mecanismos de segurança é penalizada com perdas de desempenho, quer sejam em tempo de processamento ou em uso do canal de comunicação. Sistemas de arquivos remotos com diretivas de segurança, em geral, não proporcionam o mesmo desempenho dos sistemas de arquivos de rede mais básicos. Nestes sistemas, em geral, as transferências de arquivos são mais lentas.

2.2.2 Os mais populares

Os sistemas de arquivos de rede mais populares e possivelmente os mais usados são os prestados pelo SAMBA (SAMBA, 2009) e NFS (SMITH, 2006b).

O NFS é, possivelmente, um dos sistemas de arquivos de rede mais antigos e mais usados. Criado pela Sun Microsystems em 1984 (SLADKEY, 1993), o *Network File System* possui suas especificações na RFC1094 para a versão 2 deste protocolo, na RFC1813 para a versão 3 deste protocolo e na RFC3010 para a versão 4 deste protocolo. De acordo com Hunt (HUNT, 2004), “O NFS é o *software*

de compartilhamento de arquivos mais comum na rede Unix".

Criado pelo australiano e estudante de Ciências da Computação da Universidade Nacional Australiana em Camberra (Australian National University in Canberra) Andrew Tridgell em 1992, o SAMBA foi fruto de engenharia reversa e da necessidade de uso de um aplicativo que requeria uma interface NetBIOS para operar.

O nome SAMBA foi dado ao *software* pelo criador, consultando um verificador ortográfico buscando por palavras com as letras do protocolo usado pela Microsoft para o compartilhamento de arquivos e recursos entre seus computadores, o SMB (Server Message Block). (FERREIRA, 2007)

Possibilitando a integração e compartilhamento de arquivos entre ambientes Microsoft Windows e ambientes Unix-Like, o SAMBA é extremamente difundido e é sempre uma opção quando se necessita de um sistema de arquivos de rede, independente se o ambiente é inteiramente Unix-Like ou misto.

2.3 *Network File System* – NFS

2.3.1 Características gerais

O NFS permite que diretórios e arquivos sejam compartilhados através da rede. Através do NFS, usuários e programas acessam arquivos localizados em sistemas remotos como se eles fossem arquivos locais.

O uso do NFS implica em operar no sistema cliente/servidor. O cliente usa os diretórios remotos como se estes fizessem parte de seu sistema de arquivos local. Independente do real sistema de arquivos presente na partição remota, o cliente irá fazer acesso aos dados na partição remota como sendo um sistema de arquivos do tipo NFS.

O servidor possibilita o acesso aos diretórios pelos usuários. Possibilitar este acesso é chamado, em sistemas Unix-Like, de exportar o diretório. "Conectar" os diretórios remotos ao sistema de arquivos local é chamado, em sistemas Unix-Like, de montar o diretório.

"O NFS é um protocolo de chamada de processo remoto (Remote Procedure Call - RPC) que continua no topo do UDP e IP. Uma chamada de processo remoto é simplesmente uma chamada de sistema que é processada por um servidor remoto. Quando um programa faz uma chamada I/O para um arquivo NFS, a chamada é interceptada pelo sistema de arquivo NFS e enviada através da rede ao servidor remoto por processo." (HUNT, 2000)

Para processar as solicitações do NFS, um serviço chamado de portmap (KUKUK, 2003) é necessário. O portmap desígna números de *port* dinamicamente. Algumas distribuições tratam o portmap como rpc.portmap, outras apenas como portmap. O comando *rpcinfo -p*, exibe os números de *port* de cada serviço vinculado ao RPC que estiver rodando.

O aproveitamento do canal de comunicação pelo NFS é muito alto, atingindo taxas efetivas de transferências próximas às taxas máximas de comunicação do canal usado. Usando o protocolo UDP em boa parte da comunicação o NFS consegue atingir taxas de transferência de arquivos dificilmente igualadas por outro protocolo. (SMITH, 2006a)

2.3.2 Segurança

Em uma rede onde os usuários necessitam de um diretório HOME dinâmico, ou seja, os usuários necessitam que seus diretórios HOME estejam presentes em várias estações da rede, inúmeros problemas de segurança podem ocorrer. O compartilhamento de um diretório, principalmente do diretório HOME, por uma rede implica no aumento substancial de possibilidades de acesos indevidos às informações alheias.

Para fornecer serviço semelhante ao de "perfil remoto"[1], é de fundamental importância que todas as configurações e arquivos pessoais dos usuários estejam acessíveis nas estações remotas como se estivessem locais. Somente assim um usuário teria acesso a todos os seus documentos e arquivos de configuração de sistema/ambiente, possibilitando um ambiente de trabalho ajustado conforme as necessidades do próprio usuário.

[1] "Perfil Remoto" é o nome conhecido em redes Windows onde todas as configurações do ambiente dos usuários, como papel de parede, menus, idiomas, fontes e estilos, estão disponíveis ao se logar nas estações clientes.

Em uma rede com um número elevado de usuários, o processo de "montagem" de cada diretórios HOME individualmente exigiria uma carga de trabalho grande na configuração da máquina servidora, necessitando da exportação dos diretórios HOME separadamente. Este processo seria necessário pois não se sabe qual usuário faria *login* em qual estação cliente em um determinado momento, exigindo a disponibilidade conforme mencionado acima.

Para se evitar este cenário, a estratégia usada é exportar todo o diretório */home* da servidora para as estações clientes. Desta forma, qualquer usuário que fizer *login* em uma estação terá seu diretório HOME acessível localmente por um sistema de arquivos remoto.

A segurança de um ambiente como o mencionado se baseia no fato de que um usuário já foi autenticado pela estação cliente, podendo então ter acesso a seus arquivos conforme permissões de UID e GID do sistema de arquivos da estação cliente.

Note que para o funcionamento do cenário de rede comentado, a estação cliente necessita de autonomia para acessar dados em QUALQUER subdiretório de usuário do diretório */home*. Note também que todo processo de permissão/negação de acesso à dados é feito na estação CLIENTE.

A permissão ou negação de acesso a arquivos baseada na estação que faz as requisições é uma "autenticação a nível de máquina". Esta forma de autenticar baseia-se no simples fato de que se o usuário já se autenticou na estação e esta estação cliente é um objeto de confiança, então este usuário terá acesso aos dados requisitados desde que a estação confiável permita isso.

A segurança de estações clientes está cada vez mais fragilizada com a chegada de novos recursos de *hardwares* e *software*. Confiar na requisição de um arquivo com base simplesmente em uma estação cliente não oferece mais um nível de segurança aceitável hoje.

O serviço NFS oferece restrições de acesso a dados por parte do root da estação cliente, restrições de redes/hosts, recurso de acesso síncronos ou não síncronos, recurso de acesso read-only ou read-write dentre outros. Tais recursos são insuficientes para promover alguma segurança aos dados dos usuários por parte de uma leitura indevida e até mesmo insuficiente para proteção da integridade destes arquivos, pois um diretório HOME deve obrigatoriamente ser exportado com per-

missão de gravação (acesso read-write) para que os arquivos temporários possam ser criados e manipulados pelo ambiente.

2.3.3 Burlando a segurança do NFS

Uma forma de segurança implementada no servidor NFS para impossibilitar o acesso indevido a dados é o parâmetro "root_squash". Com este parâmetro configurado no compartilhamento, requisições feitas pelo usuário de UID=0 serão remapeadas para o usuário "nobody".

Este recurso faz com que o superusuário das estações clientes não consiga acesso a arquivos no servidor, exceto em casos onde isso é desejado. É extremamente recomendado o uso do parâmetro "root_squash" na grande maioria dos sistemas.

Exemplo:

/home 192.168.1.(rw,root_squash,async) 192.168.2.31(rw,root_squash,sync)*

Na página de manual do servidor NFS existem diversas outras opções de "squash" que podem ser usadas para aumentar a segurança do serviço. Existem opções para se evitar o uso de faixas ou de qualquer UID ou GID que se deseje, como "squash_uids=0-50" e "squash_gids=500-600."(LANGFELDT, 1999)

Apesar do uso de recursos como o "root_squash" ser uma interessante ferramenta de segurança, muitos administradores não o conhecem ou o usam.

O uso do "root_squash" impede que o usuário root da estação cliente tenha direitos de root na servidora, o que não impede que ele tenha acesso a todos os arquivos de usuários na servidora.

Considere o seguinte cenário de rede.

Uma rede com NFS exportando o diretório HOME dos usuários através da exportação do diretório */home* da servidora, usando a diretiva "root_squash". Um atacante, que não deve ter muitos problemas para conseguir acesso de root em uma estação cliente usando um live-cd, por exemplo, poderia simplesmente criar um usuário local com o mesmo UID dos arquivos que ele deseja ter acesso.

Se na máquina servidora existir um usuário “eduardo” com UID=171, o atacante poderia criar um usuário, de mesmo nome ou não, com o UID=171 e, usando o comando “su”, se tornar este usuário e ter acesso a todos os arquivos do usuário “eduardo” na servidora, inclusive com direitos de alterar e/ou deletar os dados deste usuário.

A figura 2.1 exibe o resultado da execução dos comandos *exportfs* e *ls*, que correspondem a informações do compartilhamento NFS na máquina servidora. Informações semelhantes também podem ser conseguidas remotamente pelo comando *#showmount -e 192.168.1.2* , considerando que o IP do servidor seja este.

```
root@server:~# exportfs -v

/usr/local      192.168.1.*(ro,async,wdelay,root_squash)
/home           192.168.1.*(rw,async,wdelay,root_squash)

root@server:~# ls -lh /home

drwxr-xr-x  2 root root    48 2006-08-07 08:15 ftp/
drwx--x--x 45 eduardo  users 7.6K 2009-02-27 09:21 eduardo/
drwx--x--x  2 novo users  112 2008-12-04 17:02 novo/
```

Figura 2.1: Server NFS

Nas figuras 2.2 e 2.3, são expostos todos os passos para se burlar o servidor NFS e assumir o controle dos arquivos dos usuários. Todo o procedimento é feito na estação cliente.

O primeiro comando da figura 2.2 configura a rede da estação cliente com um endereço IP da mesma rede da máquina servidora. É interessante manter o mesmo endereço IP original desta estação cliente em funcionamento normal para se evitar conflitos de IP, minimizando as chances de detecção. Os diretórios */home* e */usr/-local* são exportados para toda a rede 192.168.1.0/24 com a diretiva “root_squash” habilitada 2.1. O diretório */usr/local* possui permissão somente para leitura, ou seja, independente de quem solicite os dados lá contidos, estes dados não poderão ser deletados e/ou alterados. O diretório */home* deve, na maioria absoluta das ocasiões, ser exportado com permissão de leitura/escrita, possibilitando alterações em seu conteúdo.

```
root@client:~# ifconfig eth0 192.168.0.3 netmask 255.255.255.0

root@client:~# mount -t nfs 192.168.1.2:/home /home

root@client:~# mount

/dev/sda1 on / type reiserfs (rw,relatime,notail,user_xattr)
none on /proc type proc (rw)
none on /tmp type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
192.168.1.2:/home on /home type nfs (rw,addr=192.168.1.2)

root@client:~# ls -lh /home

drwxr-xr-x  2 root root    48 2006-08-07 08:15 ftp/
drwx--x--x 45 171  100 7.6K 2009-02-27 09:21 eduardo/
drwx--x--x  2 172 100  112 2008-12-04 17:02 novo/

root@client:~# useradd -u 171 -d /tmp/pirata pirata

root@client:~# ls -lh /home

drwxr-xr-x  2 root root    48 2006-08-07 08:15 ftp/
drwx--x--x 45 pirata  100 7.6K 2009-02-27 09:21 eduardo/
drwx--x--x  2 172 100  112 2008-12-04 17:02 novo/
```

Figura 2.2: Burlando NFS parte 01

O conteúdo do diretório */home* na figura 2.1 aparece sendo listado no próprio servidor. Observe que o diretório *eduardo* pertence ao usuário “eduardo” (UID=171) e ao grupo “users”, pois estes existem localmente na servidora.

O privilégio de root em uma estação cliente pode ser obtido de inúmeras formas: iniciando a máquina cliente por um Live-cd, ataque de força bruta, iniciando a máquina por um pen-drive USB, dentre outras formas. Presumindo que nesse momento o atacante já possui privilégios de root na estação cliente e já configurou a rede, visto na figura 2.2, os passos seguintes exibem uma forma para acesso a dados alheios.

```
root@client:~# su - pirata

pirata@client:~$ cd /home/eduardo
pirata@client:$ ls -lha

drwx------   37 pirata  100 2.3K Dec 16 16:46 .
drwxr-xr-x  236 root root  5.9K Feb 13 15:04 ..
-rw-------    1 pirata  100  103 Dec 16 16:46 .Xauthority
-rw-------    1 pirata  100 1.1K Jan 13 15:52 .bash_history
-rw-r--r--    1 pirata  100 1.3K Aug 10  2005 .bashrc
-rw-r--r--    1 pirata  100 2.1M Aug 02  2005 documento01.odt
-rw-r--r--    1 pirata  100 800K Aug 09  2005 financeiro.ods
-rw-r--r--    1 pirata  100 1.2M Aug 11  2005 namorada_01.jpg
-rw-r--r--    1 pirata  100 1.1M Aug 11  2005 namorada_02.jpg
-rw-r--r--    1 pirata  100 1.4M Aug 11  2005 namorada_03.jpg
-rw-r--r--    1 pirata  100 975k Aug 11  2005 namorada_04.jpg
```

Figura 2.3: Burlando NFS parte 02

Próximo ao início da figura 2.2, após a configuração da rede, o atacante monta o diretório compartilhado pela servidora em um diretório local.

Após montar o compartilhamento, o comando *mount* na figura 2.2 exibe a comprovação do diretório remoto montado via NFS na última de sua saída.

Logo após, tem-se a listagem do conteúdo do diretório montado */home*. Note que, ao meio da figura 2.2, o diretório *eduardo* pertence a um usuário de UID=171 mas tal usuário ainda não existe no sistema (caso o atacante tenha feito boot por um Live-cd) e o UID é exibido na saída do comando *ls*.

Cria-se um usuário de nome “pirata” com o UID=171, ao final da figura 2.2.

Novamente é listado o conteúdo do diretório montado */home*. Note no último comando exibido na figura 2.2, o diretório “eduardo” é mostrado como sendo de propriedade do usuário “pirata”.

Neste ponto o atacante se torna o usuário de UID=171, neste caso “pirata”, este fato pode ser visto no início da figura 2.3. Como usuário “pirata” na máquina cliente, o atacante tem acesso irrestrito aos arquivos pertencentes ao usuário “eduardo” da máquina servidora, podendo resgatar, alterar e até mesmo deletar dados. Note que após entrar no diretório */home/eduardo*, o comando *ls -lha* exibe os ar-

quivos desse diretório tendo como dono o usuário local “pirata”.

Todo o procedimento para acesso indevido aos arquivos do usuário "eduardo"foi feito com a servidora configurada com o parâmetro “root_squash” para o compartilhamento.

O não uso do parâmetro “root_squash” pode acarretar em falhas de segurança ainda maiores. Uma vez que o atacante tem acesso aos arquivos de um diretório da servidora como root na estação cliente, o atacante pode habilitar o SUID de um arquivo executável dentro desse diretório de usuário exportado pelo NFS. Ao se conectar na máquina servidora (via ssh, por exemplo) como usuário comum, o atacante pode executar o arquivo preparado por ele e ganhar privilégios de superusuário máquina servidora, colocando em risco não somente os arquivos dos usuários mas todo o sistema em produção.

Deve-se ter em mente que está cada vez mais difícil evitar acessos indevidos às estações clientes. Muitas placas-mãe possuem um recurso novo em sua BIOS que permite ao usuário selecionar o dispositivo inicial de boot sem entrar nas configurações da BIOS. Com esse recurso, o administrador pode inserir senhas administrativas na BIOS das estações cliente, habilitar somente boot pelo HD e desabilitar boot secundário e terciário que ainda assim o usuário mal intencionado conseguirá obrigar a estação cliente a buscar o boot no CDROM ou USB sem a necessidade de digitar a senha de segurança da BIOS. Este recurso é acessado por uma tecla durante o boot, normalmente “F12”, está presente em placas mãe como a GigaByte GA-G31M (HUANG, 2007) e NÃO possui forma de desabilitá-lo (não para esta placa).

2.4 Samba

2.4.1 Características gerais

Além de ser um servidor de arquivos e recursos, o Samba possui um bom conjunto de ferramentas para controle, monitoração, manutenção e gerenciamento da rede.

Com o Samba é possível:

a) implementar um servidor de impressão;

b) implementação de um servidor de data e hora;

c) compartilhar arquivos entre máquinas Windows e Linux ou entre máquinas Linux e qualquer outro sistema operacional que possua cliente NetBEUI, como Macintosh, OS/2 e outros;

d) implementação de uma impressora PDF com possibilidade de determinar o destino do documento gerado no sistema de arquivos;

e) controle de acesso dos recursos compartilhados pelo servidor por métodos diferentes, tais como: compartilhamentos, usuários, domínio, serviços;

f) possibilidade de permitir contas com autenticação sem senha;

g) controle de acesso r/w por compartilhamento;

h) controle de acesso por usuário autenticado;

i) suporte a formas diferentes de controle de senhas (/etc/passwd; LDAP; PAM);

j) suporte a cache;

k) suporte a “diretórios ocultos” no compartilhamento;

l) suporte a aliases de identificação da máquina na rede;

m) suporte a ajustes avançados para melhoria de desempenho na comunicação TCP/IP;

n) suporte ao uso de gerenciadores de mensagens Linux (Linpopup) para troca de mensagens entre estações Windows e Linux, permitindo o redirecionamento das mensagens recebidas por email ou pager;

o) suporte a serviço WINS;

p) suporte a auditoria de arquivos;

q) suporte ao controle de domínio PDC;

r) suporte a backup de controle de domínio BDC;

s) suporte a montagem de unidades mapeadas em sistemas Windows como diretórios Linux;

t) configurações via *softwares* configuradores, manualmente ou pela web;

u) suporte a execução de scripts ao se mapear um compartilhamento ou ao se encerrar um mapeamento;

v) suporte a auditoria em tempo real;

w) suporte a "lixeira" com possibilidade de implementação de "lixeiras" separadas por compartilhamento;

x) suporte a bloqueio de arquivos por extensão, por compartilhamento.

2.4.2 Segurança

Devido a popularização dos sistemas operacionais da Microsoft, ferramentas e protocolos usados nestes sistemas operacionais são foco de ataques com mais freqüência. O simples fato de um protocolo não possuir boa compatibilidade ou nenhuma compatibilidade com os sistemas da Micrsoft já pode ser considerado um ponto positivo no quesito segurança.

Recursos de segurança são oferecidos pelo SAMBA como, encriptação e autenticação por usuário. Apesar da presença destes recursos, o protocolo SMB apresenta pontos negativos relevantes na escolha de um sistema de compartilhamento de arquivos.

A facilidade do uso de ferramentas e a grande variedade de ferramentas disponíveis para o sistema operacional Windows obriga os administradores a tomarem medidas permanentes para promover um ambiente mais seguro aos usuários.

Em ambientes de rede que possuem serviços executados sob um protocolo compatível com estações Windows, alguns cuidados especiais se fazem necessários. Monitorações e outras medidas preventivas assumem um grau elevado de importância .

Ferramentas como o *NetNeighborhood* (WINDOWS..., 2005), figura 2.4, *NetBrute* (NETWORK..., 2005), figura 2.5, *OEConnect* (OECONNECT, 2007), figura 2.6, *Gahnomen* (GAHNOMEN, 2005), figura 2.7 e o *Brutus* (INC, 2000), visto na figura 2.8, são capazes de fazer todo o trabalho para os usuários mal intencionados. Estas ferramentas não exigem maiores conhecimentos técnicos por parte do atacante, sendo um atrativo a atitudes indesejadas, tornando o ambiente de rede inseguro para os arquivos dos usuários.

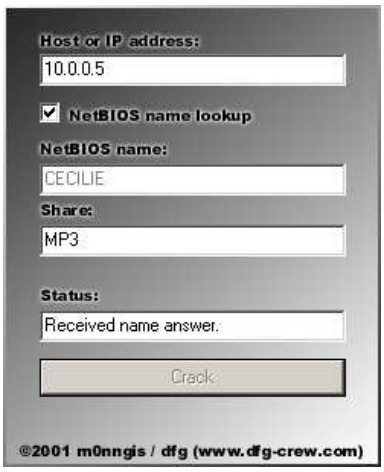

Figura 2.4: *Software* NetNeighborhood

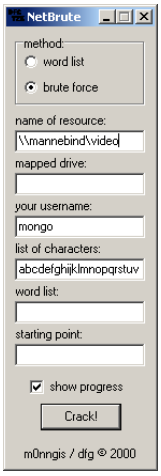

Figura 2.5: *Software* NetBrute

Além das ferramentas citadas, inúmeras outras ferramentas surgem freqüentemente, automatizando cada vez mais os processo de exploração de vulnerabilidades dos protocolos que interagem com estações Windows e exigindo cada vez menos conhecimento técnico para o uso destas ferramentas.

Frente a este cenário, o uso do protocolo de compartilhamento da Microsoft deve ser evitado quando possível. Apesar de possuir fácil instalação e configuração, o uso do SAMBA deve ser considerado somente para os ambientes onde se faz indispensável tal serviço. Protocolos alternativos que ofereçam maior segurança à

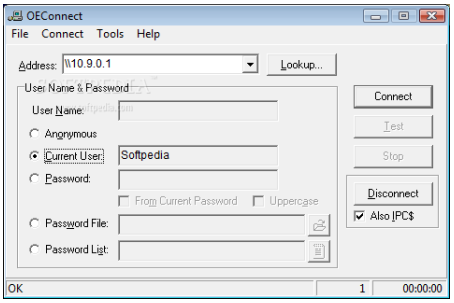

Figura 2.6: *Software* OEConnect

Figura 2.7: *Software* Gahnomen

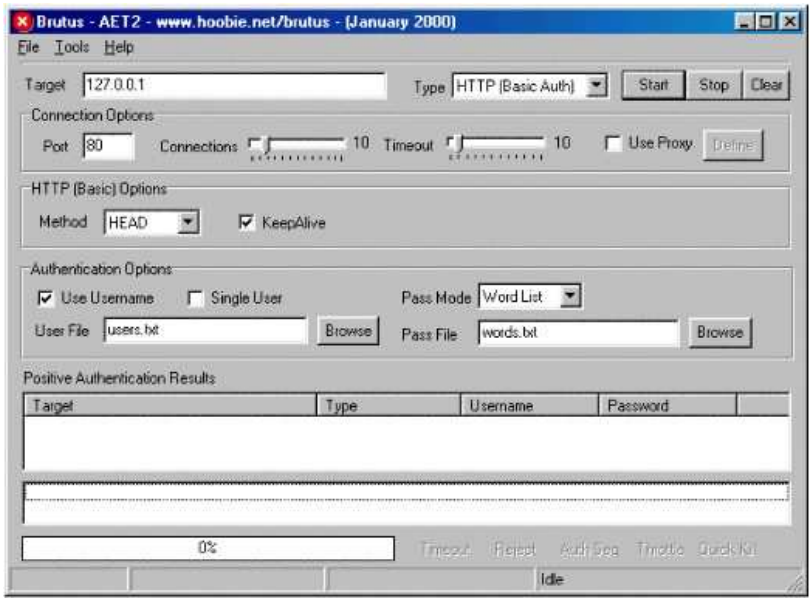

Figura 2.8: *Software* Brutus

rede devem ter preferência na hora da confecção da rede.

Capítulo 3

Serviço SSH

Este capítulo apresenta o *software* OpenSSH. Por ser um ponto fundamental no projeto do SSHomeFS, serão expostas as principais características do OpenSSH, bem como suas configurações e recursos avançados pertinentes ao projeto.

O pacote OpenSSH é uma implementação livre do protocolo SSH e vigora sob a licença BSD (YLONEN, 2009a). Nativo na maioria absoluta das distribuições Linux, o OpenSSH é na verdade uma suite de *softwares* incluindo um substituto seguro para o telnet (ssh), um substituto seguro para o rcp (scp), um substituto seguro para ftp (sftp) e um servidor (sshd) que, dentre outras funcionalidades, fornece serviço sftp-server.

Outras características do OpenSSH são: (YLONEN, 2009b)

a) forte encriptação (3DES, Blowfish, AES, Arcfour);

b) forte mecanismo de autenticação (Public Key, One-Time Password and Kerberos Authentication);

c) interoperabilidade (compatível com os protocolos SSHv1.3, SSHv1.5, SSHv2.0);

d) cliente e servidor sftp suportam protocolos SSHv1 e SSHv2;

e) suporte a compressão de dados;

f) interoperabilidade com tickets kerberos.

As configurações pertinentes ao serviço sshd se encontram no arquivo “/etc/ssh/sshd_config”. Neste arquivo podem ser configurados os mais variados comportamentos do serviço, tais como:(YLONEN *et al.*, 1999b)

a) limitar os usuários permitidos a se conectar no servidor – Parâmetro “AllowUsers”;

b) seleção da porta de trabalho do servidor – Parâmetro “Port”;

c) limite de hosts/redes permitidas para conexão Parâmetro “AddressFamily” e “ListenAddress”;

d) negar conexão de root – Parâmetro “PermitRootLogin”;

e) negar redirecionamentos – Parâmetro “AllowTcpForwarding”;

f) negar redirecionamento do X – Parâmetro “X11Forwarding”;

g) negar tunelamento – Parâmetro “PermitTunnel”.

Além dos comportamento citados, inúmeros outros parâmetros podem ser ajustados para alterar/adicionar comportamentos específicos ao servidor SSH. Mais informações podem ser obtidas no manual do serviço (YLONEN *et al.*, 1999b) ou no site do desenvolvedor (YLONEN, 2009a).

Um ponto importante no projeto SSHomeFS é a não necessidade de configurações especias no servidor SSH para o funcionamento dos HOME remotos. As configurações padrões do serviço são suficientes para um bom funcionamento do sistema SSHomeFS, porém, configurações preferenciais dos administradores poderão ser aplicadas mantendo o funcionamento do sistema com pouco ou nenhuma alteração nos clientes.

3.1 Serviço de transporte de arquivos do OpenSSH

A parte do protocolo SSH referente ao transporte é citado na seção 9 da RFC 4251(YLONEN, 2006), que define os padrões para este protocolo, bem como refe-

rências a integridade dos dados transferidos.

Usando o OpenSSH pode-se transferir arquivos em uma rede de duas formas basicamente: através da versão encriptada do rcp, o scp, ou através da versão segura do ftp o sftp. Em ambos os casos pode-se enviar um arquivo ou conjunto deles da máquina remota para o servidor e/ou buscar um arquivo ou conjunto deles do servidor para a máquina remota.

O servidor sshd suporta nativamente transferência de arquivos por scp ou sftp, não necessitando de nenhuma alteração por parte do administrador de configuração ou adição de funcionalidade extra.

O serviço de transporte de arquivos do OpenSSH também possui suporte a compressão de dados (YLONEN *et al.*, 1999a). O algoritmo de compressão usado é o mesmo usado pelo *software* gzip e também suporta a seleção de níveis de compressão, também chamado de "densidade" de compressão.

O serviço de sftp-server vem habilitado por padrão no sshd da maioria das distribuições Linux, podendo ter essa compatibilidade desabilitada comentando o parâmetro mostrado do arquivo de configuração sshd_config, como exposto na figura 3.1. Este arquivo vem, normalmente, dentro do diretório */etc/ssh*.

```
# override default of no subsystems
Subsystem       sftp    /usr/libexec/sftp-server
```

Figura 3.1: Ativando a compatibilidade com serviço SFTP

3.2 Chaves de autenticação

A parte do protocolo SSH referente a autenticação é citado na seção 9 da RFC 4251 (YLONEN, 2006), que define os padões para este protocolo, bem como referências às chaves de autenticação e autenticação baseada em hosts.

Um recurso suportado pelo OpenSSH, especificado no protocolo SSH, é o uso de autenticação por meio de chaves. Uma chave pública é alocada no servidor e uma chave privada, que teoricamente nunca sairá da máquina cliente, é mantida protegida por meio de uma "passphrase". Neste cenário temos dois níveis de segurança. Para realizar uma conexão é preciso ter a chave privada além de saber a

"passphrase" (MORIMOTO, 2008).

Este recurso de autenticação por par de chaves é usado com frequência para que máquinas clientes acessem servidores SSH e executem comandos, por *shell script*, sem a necessidade de digitação de senhas.

Apesar deste recurso oferecer uma forma de autenticação possível de ser usada em *shell script*, a autenticação por meio de par de chaves é inviável no cenário proposto pelo trabalho. Os motivos para o não uso de autenticação por par de chaves são expostos no capítulo 4, onde também serão apresentadas outras formas de autenticação.

3.3 SSHFS

Para que as estações clientes consigam montar os diretórios HOME dos usuários localmente usando o protocolo SSH, faz-se necessário o uso do *software* SSHFS (SZEREDI, 2008).

O SSHFS é um *software* cliente do protocolo de transferência de arquivos do SSH. Possui simples instalação e é executado em "*userspace*"[1]. A criação de um sistema de arquivos em userspace ocorre pelo fato do SSHFS fazer uso do módulo FUSE.

O módulo FUSE é um módulo de kernel criado especialmente para possibilitar o acesso a alguns recursos do sistema a usuários não administrativos. Com ele é possível a criação de um sistema de arquivos inteiro em *userspace*.

Tanto o SSHFS quanto o FUSE vigoram sob licença GPL/LGPL, possuindo seu código fonte aberto a comunidade.

Um dos motivadores do uso do SSHFS foi o fato de que a maioria das grandes distribuições Linux já possuirem o módulo FUSE habilitado por padrão, não necessitando de nenhuma instalação adicional, atualização de kernel ou recompi-

[1] *userspace*: *"User Space* é o nome dado a programação e execução de instruções no ambiente que conhecemos e vimos (*shell*). Para que o *User Space* possa interagir com o Kernel Space, um recurso conhecido como SystemCalls em ambientes Linux é usado"; Disponível em `http://www.linuxsecurity.com.br/sections.php?op=imprime&artid=12`

lação do mesmo.

3.3.1 Principais características do *software* SSHFS

Nesta seção são expostas as principais características que o *software* SSHFS pode oferecer, com o foco na montagem dos diretórios HOME em estações clientes.

a) encriptação dos dados trafegados;

b) suporte a compressão de dados;

c) suporte aos protocolos 1 (não recomendado) e 2 do ssh;

d) suporte a montagem com escrita síncrona ou assíncrona;

e) suporte a montagem com leitura síncrona ou assíncrona;

f) suporte a reconexão automática em caso de falha da rede;

g) suporte a cache de arquivos;

h) controle das variáveis de tempo dos caches de “r”, “w” e “delete”;

i) suporte a conversão de links e seguidor de links;

j) bloqueio/permissão das pastas montadas para o root no cliente;

k) controle de uid, gid e umask nos arquivos montados;

l) controle de quota de escrita e leitura;

m) multithreading: mais de uma requisição pode ser controlada com o servidor.

A capacidade de fazer cache local nas estações clientes é um recurso de extrema importância para o uso de um HOME inteiramente exportado. Se comparado ao SAMBA ou NFS, a transferência de arquivos de tamanho elevado tem seu

desempenho comprometido pelo uso de um forte mecanismo de encriptação, porém, as variáveis de ambiente e a grande maioria dos arquivos acessados para a execução de um ambiente gráfico de janelas são de tamanho reduzido, proporcionando um desempenho otimizado pelo recurso de cache do SSHFS.

Capítulo 4

SSHomeFS

Neste capítulo é exposto o sistema criado para o uso do protocolo SSH no compartilhamento dos diretórios HOME de usuários de uma rede.

4.1 Informações gerais

O pacote SSHomeFS é um conjunto de *scripts* em *Bash* e configurações do sistema, desenvolvido pelo autor, que permite o uso do serviço de compartilhamento de arquivos do SSH, para que usuários de uma rede tenham seus diretórios HOME da máquina servidora exportados.

Os *softwares* desenvolvidos para o pacote SSHomeFS não possuem o intuito de substituir o Samba em redes *Windows*, mas sim substituir o compartilhamento de arquivos via Samba ou NFS em redes Linux.

Com os diretórios HOME exportados pelo SSH, a segurança da rede aumenta de várias formas.

As estações clientes não terão mais autonomia para acessar arquivos na servidora. Esta autonomia, presente em rede com compartilhamento via NFS, possibilitava o acesso irrestrito aos arquivos dos usuários se um atacante assumir o controle da estação cliente, como visto no capítulo 2.

Estações clientes são potenciais pontos de vulnerabilidade de segurança de uma rede. Uma vez que é mais difícil controlar o acesso físico a elas, as estações clientes se tornam os principais alvos de ataques que fazem uso de presença física.

Com o SSHomeFS, a implementação da segurança se encontra presente mais na servidora que nas estações, minimizando a possibilidade de falha de segurança na rede.

Cada diretório HOME de usuário é montado no ato de fazer *login* na estação cliente e desmontado durante o logout. Com essa característica, ataques às estações clientes não resultam em acesso à arquivos de usuários, previamente montados durante o processo de boot.

Para que a estação cliente consiga montar o diretório HOME de um usuário durante o processo de *login*, é necessário que o usuário forneça sua senha pessoal. A senha do usuário é usada no processo de montagem do HOME e descartada imediatamente após a tentativa de montagem, não deixando rastros em arquivos temporário do sistema, arquivos deletados no HD ou em variáveis de ambiente.

Em caso de falha na montagem do HOME, o usuário é solicitado a fornecer novamente sua senha para uma segunda tentativa. O número de tentativas de montagem do diretório HOME antes de forçar o logout do usuário é definido na variável "Limite_tentativas", no início do programa *autosshfs.sh*.

A escolha dos ambientes oferecidos pelo SSHomeFS ao usuário, logo após a montagem de seu diretório HOME, visto na figura 4.6, pode ser modificada no início do programa *autosshfs.sh*. Para isso, a função "Funcao_escolher_gui" deve ser ajustada, bem como o endereço da variável do ambiente gráfico que deverá receber o caminho para seu "arranque".

Os códigos fontes atualizados do pacote SSHomeFS podem ser emcontrados diretamente no site do autor, em http://www.cbpf.br/∼mgm

Documentações e maiores informações sobre o SSHomeFS também podem ser encontradas no site do autor, bem como *screenshots* e vídeos demonstrativos.

4.1.1 Dependências

Alguns *softwares* são necessários para o funcionamento do SSHomeFS, dentre eles, apenas o SSHFS (SZEREDI, 2008) ainda não está presente de forma nativa nas grandes distribuições Linux. Além do SSHFS, o pacote SSHomeFS possui como dependência o *software* "dialog"(LAM, 2002), nativo na maioria das grandes dis-

tribuições Linux.

Além dos recursos do SSHFS visto em 3.3.1, a opção *password_stdin* deve ser suportada por este programa. Este recurso foi incorporado a partir de versão 2.0 do *software*, necessitando obrigatoriamente do uso de uma versãs igual ou superior a está para o funcionamento do SSHomeFS.

Para a escrita deste documento, o autor desconsiderou a hipótese de ausência de alguns recursos básicos como o próprio *Bash*, test, if, trap e outros. Estes recursos estão presentes praticamente na totalidade dos conjuntos básicos de pacotes das instalações das distribuições Linux.

O Apêndice B oferece um conjunto de testes e procedimentos para verificar, de forma manual, as dependências do pacote SSHomeFS no sistema da estação cliente, bem como a verificação de forma prática e simples dos ajustes configurações necessários. A leitura do Apêndice B é altamente recomendada no momento de implementação do SSHomeFS.

4.2 Principais barreiras

A preocupação com a segurança e as verificações de ambiente foram as principais barreiras a superar para o SSHomeFS. A manutenção remota das estações clientes também foi uma preocupação e implicou em códigos de verificações exclusivo para esse fim.

Para implementar segurança nas operações do SSHomeFS, um conjunto de normas se fez necessário:

a) os usuários não devem ter privilégio para encerrar a execução do programa de *login* do SSHomeFS, o *autosshfs.sh*;

b) para evitar algum registro da senha do usuário ou outra informação na saída padrão (monitor), o teclado não deveria “ecoar” caracteres no terminal de *login* mesmo em caso de falha da execução da janela de *login*;

c) os logs de erro da autenticação do sistema SSHomeFS devem ser individuais por usuários;

d) as senhas dos usuários deverão ser armazenadas pelos menores períodos de tempo possíveis e em locais que só permitam um único acesso pelo proprietário;

e) alterações de características de arquivos temporários (permissões, propriedade) deverão ser verificadas a cada chamada ao *autosshfs.sh*, negando o *login* em caso de anomalias.

Além de verificações de segurança, verificações de ambiente também se fazem necessárias uma vez que inúmeras situações devem ser tratadas pelo sistema de *login* e ações diferentes devem ser tomadas nestas situações. As principais verificações tratadas pelo SSHomeFS são:

a) rotinas para verificar se o usuário que chama o *autosshfs.sh* já não está logado em outro terminal;

b) rotinas para verificar e excluir o usuário root dos mecanismos do *autosshfs.sh*;

c) rotinas para verificar e excluir um usuário predeterminado para manutenção do sistema dos mecanismos do *autosshfs.sh*;

d) rotinas para verificar erros do diretório local do usuário, como permissões, cota, propriedade;

e) rotinas para verificar a existência de um diretório local com as devidas permissões e propriedades, para ser montado o HOME remoto;

f) rotinas para ajustar variáveis necessárias de ambiente.

Inúmeras verificações são necessárias para o funcionamento de um programa que monta e desmonta diretórios durante o *login* e o logout de usuários. Além das verificações citadas, outras variáveis podem ser inseridas no processo, aprimorando o comportamento do pacote proposto.

Faz parte do protocolo SSH a criação/transferência de chaves de identificação no ato do primeiro *login* de um usuário a uma máquina servidora. Quando um usuário se conecta pela primeira vez a uma máquina servidora, é questionado se deseja armazenar permanentemente a chave fornecida pelo servidor. Esta chave

será necessária para criptografia do protocolo e acusará erro caso o servidor acessado sofra adulterações.

Assim como o cliente SSH, o SSHFS também exibe a mensagem de criação da chave no primeiro *login* de um usuário e aquarda a interação do usuário aceitando a chave.

O uso do SSHFS em *scripts* de *shell* obriga que o usuário já tenha feito ao menos uma conexão anteriormente para aceitar a chave fornecida pelo servidor SSH, caso contrário a mensagem sobre a chave não aparecerá para o usuário e a conexão não se completará até que a chave seja aceita. A solução para este problema é vista no subcapítulo 4.4.2, onde as configurações das estações clientes são expostas.

4.3 O pacote SSHomeFS

Para cumprir com os objetivos do trabalho, o pacote SSHomeFS atua principalmente em três pontos distintos nas estações cliente, no processo de *login* propriamente dito, no processo de logout e no processo de boot da estação cliente. Por este motivo, o pacote SSHomeFS é respectivamente dividido em três *scripts* de *shell* que executam tais tarefas separadamente, o *rc.sshfs* (Seção 4.3.1), o *autosshfs.sh* (Seção 4.3.2) e o *.bash_logout* (Seção 4.3.3).

O serviço de autenticação de usuários remotos não pertence ao escopo deste trabalho. Durante o desenvolvimento do SSHomeFS o serviço NIS foi usado para autenticar os usuários remotamente. Qualquer serviço de autenticação de rede poderá ser usado em parceria com o SSHomeFS, desde que seja possível para as estações clientes criar uma lista de usuários durante o boot. Para o uso de um serviço de autenticação de rede diferente do NIS, é necessário ajustar a função "Funcao_criar_lista" no início do programa *rc.sshfs*.

Os três componentes do SSHomeFS preparam o sistema das estações clientes para operar com o tipo de compartilhamento oferecido pela máquina servidora. Estes componentes são inseridos nas estações clientes para que sejam executados durante o boot (*rc.sshfs*), durante processo de *login* (*autosshfs.sh*) e durante o processo de logout (*.bash_logout*).

Para atuar no processo de *login*, o programa *autosshfs.sh* deverá ser inserido em cada estação cliente, uma vez que neste momento os diretórios HOME dos

usuários não se encontram montados. O processo de logout é favorecido pelo diretório HOME do usuário se encontrar montado neste momento, o que possibilita colocar o programa *.bash_logout* dentro dos diretórios HOME de cada usuário na máquina servidora. Este processo pode ser automatizado facilmente durante o processo de criação de novos usuários.

4.3.1 O programa *rc.sshfs*

As tarefas do programa *rc.sshfs*, atuante nas estações remotas, é detalhada nesta subseção

Para que um diretórios remotos seja usado montado sobre um diretório local, usando SSHFS, o diretório local deve atender certos requisitos de propriedade. Para que o usuário tenha permissão e posse dos arquivos internos ao diretório montado, o proprietário do diretório local deve ser o mesmo que executa a montagem. Ajustes do tipo "umask" podem ser feitos no ato da montagem mas a propriedade do diretório e as permissões locais devem atender ao SSHFS.

Um diretório HOME remoto pertencente a um usuário, deve ser montado em um diretório local com permissão de leitura e gravação para o usuário. Por questões de segurança,o diretório local deve permitir leitura e gravação somente para o usuário em questão, forçando que tal diretório tenha permissões do tipo 700 e como dono o próprio usuário.

Para atender a estas necessidades, um diretório HOME local deve existir em cada estação cliente, para cada usuário, para aguardar a montagem dos respectivos diretórios HOME remoto.

O programa *rc.sshfs*, que é executado nas estações clientes a cada boot do sistema, prepara a estação para atender às necessidades do programa *autosshfs.sh*. Isso permite ao administrador criar novos usuários na máquina servidora e estes usuários entrarão em funcionamento automaticamente nas estações clientes após um simples reboot da estação.

O programa *rc.sshfs* deve ser colocado nas estações CLIENTES e ser executado, seguido do parâmetro "start", a cada boot da máquina. O processo de instalação de todo o pacote SSHomeFS é visto na seção 4.4, incluindo o programa *rc.sshfs*.

Durante a execução do *rc.sshfs* é feito todo o ambiente necessário para que os usuários possam fazer *login* usando o programa *autosshfs.sh*. Variáveis e diretórios são criados, permissões, donos e grupos são ajustados.

O programa *autosshfs.sh* recusará o *login* de um usuário se alguma permissão de algum arquivo local não estiver de acordo com os padrões de segurança predeterminados. O programa *rc.sshfs* também incorpora um recurso de verificação de erros.

A figura 4.1 mostra um exemplo de falha de segurança forçada, onde diretórios de usuários poderiam permitir acesso a outros usuários ou até mesmo o sequestro de senhas com as permissões erradas em */tmp/fs*.

Para verificar quais necessidades não foram atendidas para um usuário no momento do *login*, o programa *rc.sshfs* pode ser chamado com o parâmetro "check". O uso deste parâmetro fará o programa retornar as incompatibilidades encontradas (erros) de cada usuário, como exibido ma figura 4.2. Para corrigir tais erros, o programa *rc.sshfs* pode ser chamado manualmente pelo root da estação cliente com o parâmetro "start" e depois verificado novamente com o parâmetro "check".

```
root@virt_client1:~# chmod 777 /home/cbpf
root@virt_client1:~# chmod 707 /home/cara40
root@virt_client1:~# chmod 707 /tmp/fs
root@virt_client1:~# chgrp users /tmp/fs
root@virt_client1:~# chown cbpf /tmp/fs
root@virt_client1:~# chown cara10 /tmp/fs/cara50
```

Figura 4.1: Provocando falhas de segurança

Após a correção dos erros, de forma automático pelo comando */etc/rc.d/rc.sshfs start*, a figura 4.3 mostra uma saída do comando sem erros no sistema.

4.3.2 O programa *autosshfs.sh*

Como exposto em 4.3.1, o programa *rc.sshfs* prepara todo o ambiente da estação cliente necessário para fazer a montagem dos diretórios HOME remotos. O programa responsável pela montagem dos HOME propriamente dito é o *autosshfs.sh*.

```
root@virt_client1:~# /etc/rc.d/rc.sshfs check

Below is shown the erro to TMP:
Description     Parameter       ‘‘Should be’’
-------------------------------------------------------------
Permission        drwx---rwx        drwxr-xr-x
Owner                cbpf                root
Group                users                root


Below is show the erro to FIFOs or HOMEs:
Descrip         Parameter       ‘‘Should be’’
-------------------------------------------------------------
/tmp/fs/cbpf        prw-------        prw-------
/tmp/fs/cara50        cara10               cara50

/home/cbpf        drwxrwxrwx        drwx------
/home/cbpf        cbpf                cbpf

/home/cara40        drwx---rwx        drwx------
/home/cara40        cara40               cara40

 If there was errors, exec ‘‘/etc/rc.d/rc.sshfs start’’ to correct
the errors!
```

Figura 4.2: Erros na estação cliente

O aplicativo SSHFS é usado pelo *autosshfs.sh* para montar o diretório HOME dos usuários no ato do *login*. O *autosshfs.sh* executa as verificações de segurança e de ambiente propostas em 4.2, além de ser responsável pela interação com o usuário.

Informações da rede são necessárias para o funcionamento do *autosshfs.sh*. Com a responsabilidade de montar os HOME, o IP da máquina servidora de arquivos deve ser configurado dentro do *autosshfs.sh*. Informações adicionais também devem ser configuradas de acordo com a particularidade da rede e dos sistemas usados. Os ambientes gráficos e os locais onde estes ambientes são iniciados também devem ser configurados no *autosshfs.sh*.

A interface usada pelo *autosshfs.sh* é promovida pelo *software dialog*(LAM, 2002). Durante o processo de *login*, uma confirmação de senha é solicitada ao

```
root@virt_client1:~# /etc/rc.d/rc.sshfs check

Below is shown the erro to TMP:
Description     Parameter       ‘‘Should be’’
------------------------------------------------------------


Below is show the erro to FIFOs or HOMEs:
Descrip         Parameter       ‘‘Should be’’
------------------------------------------------------------


 If there was errors, exec ‘‘/etc/rc.d/rc.sshfs start’’ to correct
the errors!
```

Figura 4.3: Estação cliente sem erros

usuário e usada para montar seu HOME, como visto na figura 4.4.

Caso a senha digitada não esteja correta, o *autosshfs.sh* exibirá uma mensagem de erro e solicitará novamente a senha. Na persistência de erro o usuário será forçado a um logout e receberá informações de ajuda para a correção do problema, a figura 4.5 exibe um exemplo de mensagem de ajuda ao usuário em caso de persistência de erro. O número de tentativas pode ser modificado na variável “Limite_tentativas”, no início do programa, o padrão é de duas tentativas antes de forçar o logout.

Após montar o diretório HOME, o programa *autosshfs.sh* oferece alternativas de ambientes ao usuário, figura 4.6. O teclado pode ser usado como atalho na seleção do ambiente, digitando a letra em destaque em cada linha dentre as opções oferecidas. No cenário da figura 4.6, pode-se selecionar o ambiente gráfico “FLUBOX” pelas setas do teclado ou simplesmente digitando a letra “F”, uma confirmação se faz necessário com a tecla “Enter”. Caso haja coincidência entre as primeiras letras dos ambientes oferecidos, pode-se selecionar a segunda opção apertando outra vez a tecla com a letra em destaque, seguida de um “Enter”.

O programa *autosshfs.sh* também é responsável por verificar se o usuário já não possui seção ativa no sistema. Em caso afirmativo, o diretório HOME do usuário já estará montado e nenhuma interação adicional se faz necessária, entre-

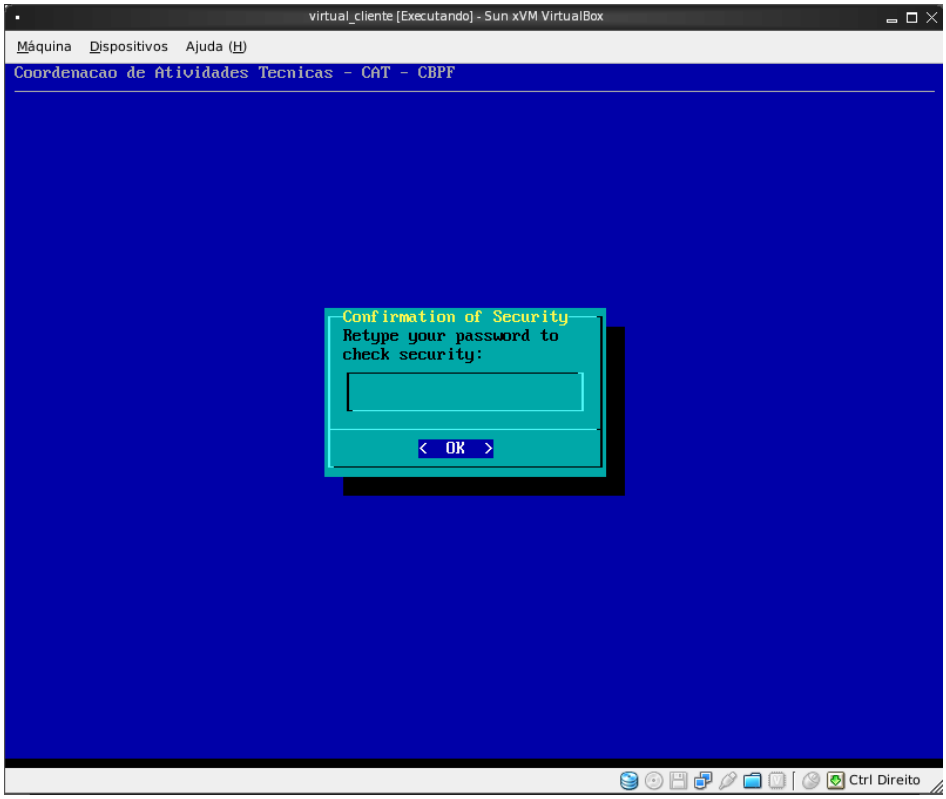

Figura 4.4: Confirmação de senha

gando ao usuário um terminal para uso.

Outra responsabilidade do *autosshfs.sh* é verificar diretrizes de segurança. Erros na execução do programa *rc.sshfs*, como exemplificado na figura 4.2, ou permissões mal configuradas forçarão o *autosshfs* a negar o *login* do usuário . Para estes casos, um log é gerado para análises futuras dos erros e pesquisa de vulnerabilidades de segurança.

Para a administração remota das estações clientes, um usuário é excluído do processo de verificação do *autosshfs.sh*. Isso se mostra necessário quando o programa *autosshfs.sh* apresenta algum erro ou ajuste indevido e o *login* de usuários não se completam. Este usuário "liberado" não apresentará erros no *login* caso o *autosshfs.sh* tenha sido configurado indevidamente ou a estação cliente não atenda a alguma dependência do pacote SSHomeFS. Para configurar o usuário que não

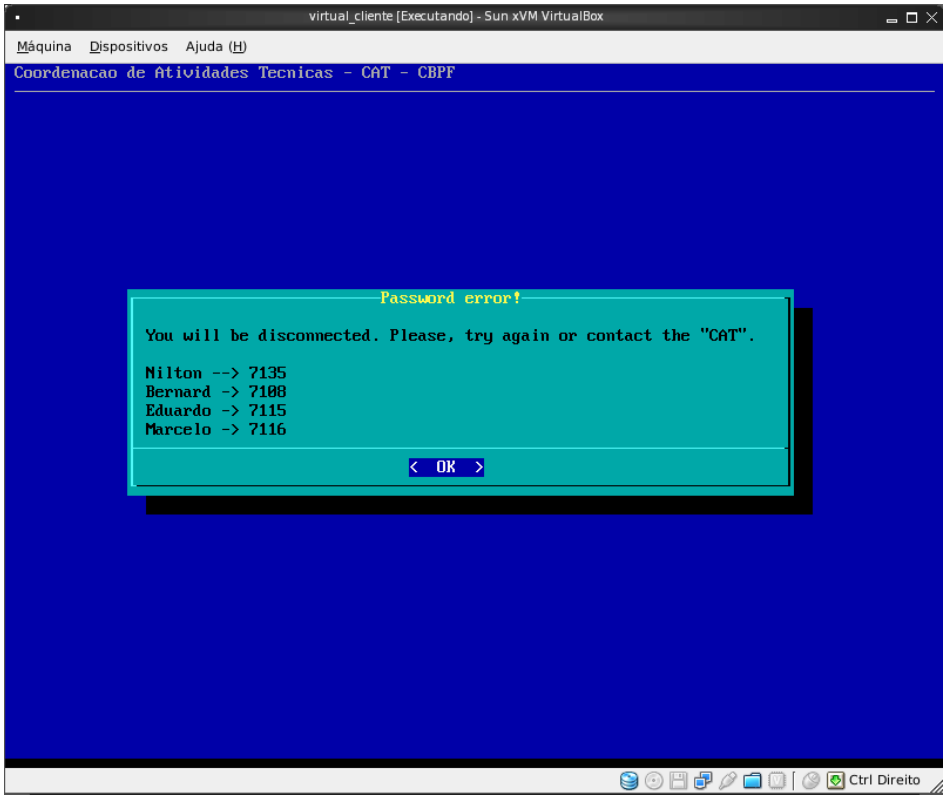

Figura 4.5: Mensagem final de erro

fará parte das verificações do *autosshfs.sh*, a variável "Liberado" deve ser ajustada com o UID do usuário desejado.

O uso de uma versão do SSHFS superior à 2.0 é vital para o funcionamento do pacote SSHomeFS. Como exposto na seção 4.2, o uso do SSHFS em *scripts* de *shell* requer certos cuidados. A necessidade do uso de versões do SSHFS superiores à 2.0 se justifica pelo recurso *passwod_stdin*, o qual torna possível interagir a senha pessoal para conexão em uma linha de comando do script. Para inserir a senha do usuário em *scripts* de *shell*, o uso do programa *sshpass* (SHEMESH, 2008) até a versão 1.04 se mostrou ineficaz em conjunto com o SSHFS.

A solicitação de autorização da chave de encriptação durante o primeiro *login* de um usuário poderia se tornar um grande problema para o administrador, tendo seus *logins* rejeitados nas estações clientes. No subcapítulo 4.4.2 é exposto

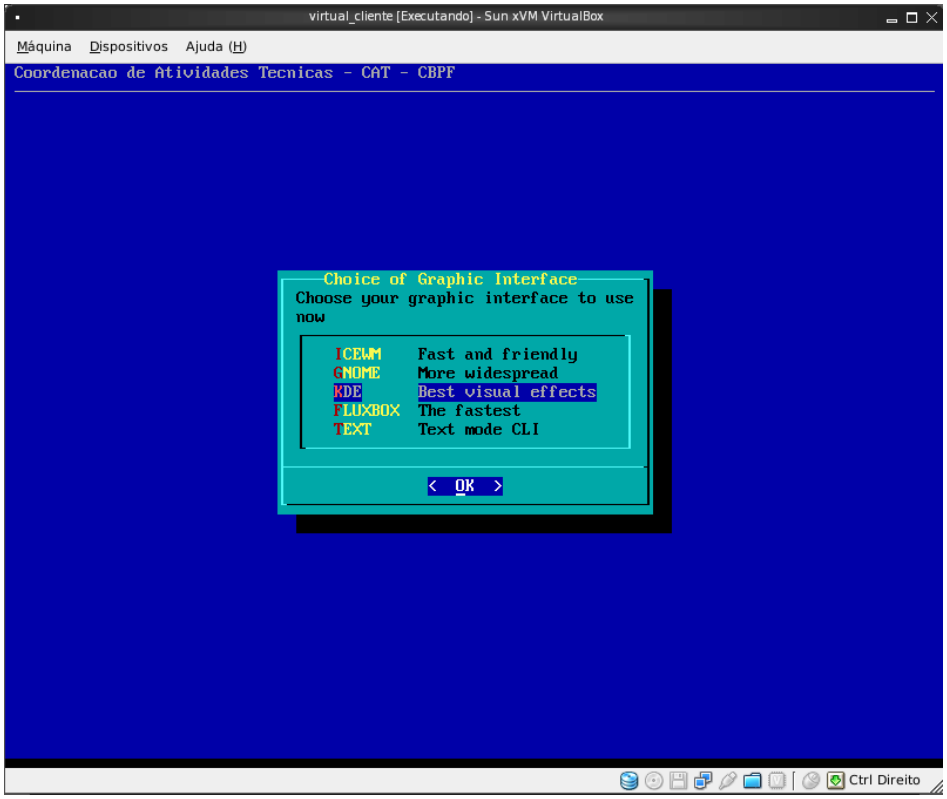

Figura 4.6: Escolha de ambiente de trabalho

o recurso para se evitar este problema, que impossibilitaria o funcionamento do SSHomeFS.

Para solucionar eventuais problemas de implementação ou de falha no *autosshfs.sh*, o Apêndice B exibe procedimentos e verificações a serem feitas manualmente para a identificação do problema.

4.3.3 O programa *.bash_logout*

O programa *.bash_logout* é responsável por desmontar o diretório HOME de um usuário quando este não é mais necessário.

Para usuários que fazem uso do interpretador de comandos *Bash* como *shell* padrão, o arquivo */home/USUARIO/.bash_logout* é executado a cada logout, onde

“USUARIO” é o nome do usuário em uso no sistema. Para que o pacote SSHomeFS execute suas funções, em especial a de desmontar o HOME do usuário no momento do logout, é necessário que os usuários tenham como *shell* padrão o *Bash*.

Quando solicitado, o programa *.bash_logout* verifica se o usuário possui mais de uma seção iniciada no sistema e, em caso negativo, o programa desmonta o diretório HOME do usuário. Esta verificação se faz necessária pois um usuário poderia estar fazendo logout de um segundo terminal (ou terceiro...) em uso mas precisa manter seu HOME montado para acesso a dados de outro terminal. O processo de desmontar o HOME somente deverá atuar quando a última seção do usuário estiver sendo finalizada na estação cliente.

Em sistemas onde o arquivo */home/USUARIO/.bash_logout* já possui outras atribuições, o código do programa *.bash_logout* deverá ser integrado ao código nativo do */home/USUARIO/.bash_logout*.

4.4 Instalação

O processo de “instalação” é exposto dividido em duas seções. Adição de arquivos e configurações do sistema na máquina servidora e adição de arquivos e configurações do sistema nas estações clientes. Este processo também é descrito no arquivo *INSTALL*, dentro do pacote SSHomeFS.

4.4.1 Máquina servidora

Obviamente, além do serviço SSH funcionando, são necessários alguns procedimentos para que o SSHomeFS funcione. Os procedimentos “a” e “b”, aqui citados, são obrigatórios. O procedimento “c” terá sua utilidade dependente da distribuição usada nas estações clientes. O procedimento “d” é apenas uma recomendação.

a) Colocar o programa *bash_logout* dentro dos diretórios HOME de cada usuário, renomeando-os para “.bash_logout”

b) Criar, dentro dos diretórios HOME de cada usuário, o diretório oculto[1] ".ssh". Dentro deste diretório deverá haver uma cópia do arquivo */etc/ssh/ssh_config*, renomeado para "config", com a diretiva "StricHostKeyChecking" ajustada para "ask". Tanto o diretório ".ssh" quanto seu conteúdo devem pertencer ao dono do diretório HOME onde eles se encontram

c) Para se evitar alguns erros de alguns ambientes gráficos nas estações clientes, é necessário um link simbólico de nome "tmp" apontando para o diretório */tmp* dentro de cada diretório HOME dos usuários

d) É recomendado a criação do arquivo ".bashrc", dentro de cada diretório HOME de usuário, contendo a linha alias *ssh="ssh -F /.ssh/config"*. Esse arquivo deve pertencer ao próprio dono do diretório onde está localizado

Todos os processo citados neta seção podem ser automatizado alterando o script de criação de usuários da máquina servidora.

4.4.2 Máquinas clientes

As estações clientes requerem uma atenção especial. A necessidade de alteração ou adição de algum recurso irá requerer mudanças em todos as estações clientes da rede. O preparo das estações clientes deve tentar minimizar o trabalho do administrador ao se adicionar novos usuários no sistema.

Para a "instalação" do pacote SSHomeFS nas estações clientes o autor considera que todas as dependências e recursos de sistema necessários ao pacote estão presentes nas estações. Para verificações e testes, verifique o Apêndice B.

O programa *rc.sshfs* deve ser iniciado durante o boot das estações clientes. Este programa deve ser colocado dentro do diretório */etc/rc.d* ou */etc/init.d* com permissão de execução e ser iniciado pelo "rc.local" com o argumento "start", como mostrado na figura 4.7.

Por se tratar de um programa que deve ser executado a cada *login* de usuário, o programa *autosshfs.sh* deve ser colocado dentro do diretório */etc/profile.d*, quando este está presente. Algumas distribuições Linux iniciam os scripts presentes neste

[1] Em sistemas Unix-like, diretórios que possuem nomes iniciados com "." são ocultos. Para se exibir arquivos ocultos, a opção "-a" do comando *ls* pode ser usada.

```
#!/bin/sh
#
# /etc/rc.d/rc.local:  Local system initialization script.
if [ -x /etc/rc.d/rc.sshfs ] ; then
     /etc/rc.d/rc.sshfs start
fi
```

Figura 4.7: Arquivo *rc.local* das estações clientes

diretório a cada *login*. Caso a distribuição Linux escolhida para as estações clientes não possua este recurso, o conteúdo do programa *autosshfs.sh* deverá ser copiado para dentro do script */etc/profile*, com exceção da primeira linha.

O conteúdo do programa *autosshfs.sh* pode ser adicionado ao conteúdo do script */etc/profile* facilmente, excluindo a primeira linha, como mostrado na figura 4.8.

```
root@client:~#tail -n +2 autosshfs.sh >> /etc/profile
```

Figura 4.8: Adicionando *autosshfs.sh* ao arquivo *profile*

Para se evitar a solicitação de autorização para a criação da chave de encriptação durante o primeiro *login* de um usuário, impossibilitando o funcionamento do SSHomeFS, o arquivo */etc/ssh/ssh_config* das estações clientes devem ser editados, tendo sua diretiva “StricHostKeyChecking” alterada de “ask” para “no”.

É recomendado também a inclusão da linha *exit 1* ao final dos arquivos *.bashrc* no HOME de cada usuário. Esta linha impede o acesso a um terminal em caso de erro na montagem do diretório remoto. Os arquivos *.bashrc* nos HOME das estações clientes devem pertencer ao root e possuir permissão 755 (ítem “e” do subcapítulo B.2).

Capítulo 5

Metodologia e resultados

Este capítulo apresenta os resultados obtidos com o uso do SSHomeFS, erros encontrados, dados comparativos e a metodologia usada para a obtenção destes dados.

5.1 Metodologia

Para quantificar e avaliar a nova estrutura de rede usando o SSHomeFS, foram realizados testes nos laboratórios do Centro Brasileiro de Pesquisas Físicas - CBPF usando a estrutura de redes do edifício César Lattes e computadores da Coordenação de Formação Científica – CFC e da Coordenação de Atividades Técnicas – CAT. O uso do material e das dependências do instituto teve a supervisão do tecnologista sênior Prof. Nilton Alves Júnior.

Os *hardwares* utilizados são expostos na tabela 5.1. A ligação entre estações clientes e máquina servidora foi realizada por switches Extreme da família Summit X450e ligados em "pilha". Durante os testes, as estações clientes se mantiveram com a conexão estabelecida em 100Mbps Full Duplex.

Tabela 5.1: Especificações das máquinas - *Hardware*

	Servidora Física	Servidora Virtual	Estações Clientes
CPU	Intel Core2 Quad 2.4GHz	2 núcleos	Intel Core2 Duo 2.5GHz
Memória	8GB DDR2 800MHz	virtual 512MB	2GB DDR2 800MHz
Disco	2xSATAII - RAID1	virtual, dinâmico	1xSATAII

Os *softwares* utilizados são expostos na tabela 5.2, bem como suas respectivas versões.

Tabela 5.2: Especificações das máquinas - *Software*

	Servidora Virtual	Estações Clientes
S.O.	Mandriva 2009 32b	Mandriva 2009 32b
KDE	4.1.3	4.1.3
GNOME	2.24	2.24
ICEWM	1.3.1	1.3.1
Fluxbox	1.1.1	1.1.1
Firefox	3.0.4	3.0.4
OpenOffice	-	3.0.0
Gimp	-	2.4.7
Scilab	-	4.1.2

Os testes realizados dividem-se em dois principais focos. Um deles, avaliava o uso de diferentes ambientes gráficos, ferramentas nativas e *software* não nativos dos ambientes avaliados. O outro principal foco foi o desempenho da estação cliente executando aplicativos sob um ambiente que use o protocolo SSH para acessar os diretórios HOME na máquina servidora.

Durante os testes dos ambientes gráficos, foram observados os comportamentos dos *softwares* em execução com o SSHomeFS em busca de erros de execução, lentidão excessiva e travamentos.

Para as avaliações de desempenho, o procedimento assumido nas tomadas de tempos consistiam de repetições sucessivas de medições onde os valores apresentados neste capítulo são médias diretas de 10 medições com um mínimo de 3 algarismos significativos por medida, dadas em segundos.

5.2 Ambientes gráficos

Alguns ambientes gráficos apresentaram erros quando usados em conjunto com o SSHomeFS.

Nos testes feitos, o ambiente gráfico GNOME apresentou um erro relacionado ao arquivo “.Xauthority”. Este erro é reportado ao usuário todas as vezes que o

GNOME se inicia.

Após um “Ok”, o ambiente gráfico GNOME se inicia normalmente e o uso das ferramentas do ambiente não apresentou nenhum erro. Esse comportamento se repetiu tendo como sistema da estação cliente o Linux Mandriva 2009 e o Linux Ubuntu 8.10.

Como exposto na figura 5.1, o ambiente gráfico KDE da família 3.x apresentou um erro no aplicativo “DCOPserver”. O DCOPserver é uma dependência para as ferramentas do KDE3, sem ele os programas não conseguem ser executados, nem mesmo o próprio ambiente gráfico do KDE3.

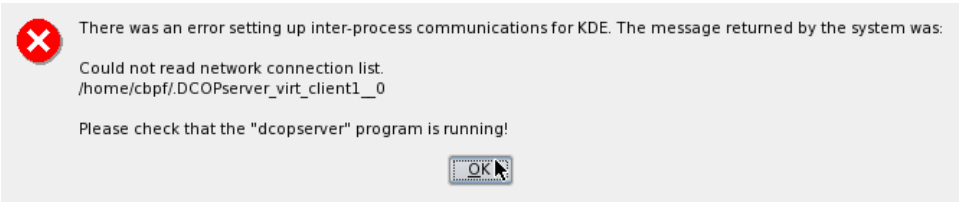

Figura 5.1: Erro DCOPserver

É importante salientar que o ambiente gráfico KDE4 não apresentou erros. Ferramentas do KDE3 poderiam ser usadas no KDE4, porém, tal ato não se torna possível devido erro ao no aplicativo DCOPserver. Demais ambientes não apresentaram erros, como o XFCE4, Fluxbox, ICEWM e FVWM .

5.3 Desempenho

5.3.1 Usabilidade

Para avaliar de usabilidade das estações clientes, a performance destas estações foram testadas observando os tempos de boot, *login* e abertura de programas.

Com o intuito de se aproximar de situações reais, os programas escolhidos foram os aplicativos mais comumentes usados em estações de trabalho. Para uma melhor avaliação, alguns valores obtidos em testes com o SSHomeFS são comparados com valores da mesma estação cliente fazendo uso do NFS.

O NFS foi escolhido como referência por se tratar de um protocolo bem conhecido, por ser destaque em desempenho, por ser um dos mais antigos protocolos de compartilhamento de arquivos e por ser o mais usado em redes Linux (HUNT, 2004).

Os recursos de sincronismo *sync* do NFS não foi ativado, fazendo uso do sistema de arquivos NFS montados como *async*. Apesar de "mascarar" o verdadeiro desempenho do NFS, o *sync* não foi ativado para que o protocolo pudesse oferecer o máximo ao usuário, permitindo assim um comparativo com o SSHFS que faz uso de *cache* para melhorar seu desempenho.

Os tempos das tarefas foram os pontos mais relevantes dos testes. Taxas efetivas de transferência de arquivos assumem uma importância secundária, visto que ambos os protocolos fazem uso de recursos de *cache* de memória para melhorar sua performance aparente para o usuário.

O primeiro ponto avaliado foi o boot do sistema. Nas estações testadas, o boot não apresentou modificações de tempo significativas ao fazer uso do NFS ou do SSHomeFS.

Para a avaliação do *login*, foi desconsiderado o processo de autenticação propriamente dito. Durante a avaliação do SSHomeFS, os tempos medidos tomaram como base o momento da escolha do ambiente gráfico até o seu carregamento por completo, aguardando que o último ícone fosse carregado na tela. Os mesmos critérios foram usados para o NFS, um usuário era autenticado em um console e o tempo medido se iniciava ao chamar o comando para carregar o ambiente gráfico e cessava ao ser carregado o último ícone na tela .

A tabela 5.3 apresenta as médias dos tempos encontrados para carregar os ambientes gráficos por inteiro. A esquerda, os valores em segundos obtidos com o uso do SSHomeFS. A direita, um comparativo dos mesmos testes fazendo uso do NFS.

Devido o fato dos ambientes gráficos mais completos, como o KDE e GNOME, fazerem um uso mais intenso de acessos a informações localizadas no HOME dos usuários, os tempos para o carregamento dos mesmos foram sensivelmente afetados pelo SSHomeFS, aumentando o tempo de *login* do GNOME em aproximadamente 4 vezes e do KDE em aproximadamente 3,5 vezes .

Para os ambientes ICEWM e Fluxbox , os tempos apresentados com o uso do SSHomeFS representam uma espera de aproximadamente 1,5 segundos além do que o usuário esperaria se estivesse usando o NFS.

Tabela 5.3: Tempos para carregar os ambientes gráficos

	SSHomeFS	NFS
ICEWM	3,1	1,9
GNOME	46	11
KDE	41	12
Fluxbox	2,5	1,7

Na tabela 5.4 é exposto as médias dos tempos necessários para se carregar os programas citados, fazendo uso de gerenciadores de ambientes diferentes. Pode-se notar que as médias dos tempos de carregamento dos programas não sofreram grandes alterações com o uso do SSHomeFS em relação ao NFS.

Na medição do programa *Firefox*, foi considerada o tempo de abertura somado ao tempo de exibição da página configurada como padrão. Para reduzir o tempo de comunicação de rede, foi adotado como página padrão o site do CBPF por ser hospedado internamente na própria rede onde ocorreram os testes.

Tabela 5.4: Carregamento de programas de usuário

		Firefox	OpenOffice	Gimp	Scilab
SSHomeFS	ICEWM	5,1	2,1	3,3	2,0
	GNOME	5,6	2,2	3,4	2,1
	KDE4	6,5	2,2	3,5	2,2
	Fluxbox	4,9	1,9	3,2	2,0
NFS	ICEWM	4,9	2,1	3,3	2,0
	GNOME	5,3	2,1	3,4	2,1
	KDE4	6,1	2,1	3,4	2,2
	Fluxbox	4,8	1,9	3,2	2,0

5.3.2 Cache do sistema de arquivos

Para simular um uso da estação cliente e avaliar o compartilhamento do diretório HOME via SSH, um segundo teste foi realizado simulando a cópia de um arquivo

de dentro do diretório HOME do usuário para um diretório local na estação cliente. Este teste teve como principal objetivo avaliar e comparar o recurso de *cache* dos dois sistemas de arquivos.

Na máquina servidora foram criados dois arquivos. Um arquivo pequeno, representando os inúmeros arquivos que são acessados no diretório HOME de um usuário durante o uso da estação cliente com ambiente gráfico ativo, com 2MB de tamanho. Um arquivo "grande", representando um acesso a um documento pesado como um vídeo ou um conjunto de arquivos compactados, com 250MB de tamanho.

Os arquivos de testes foram gerados na máquina servidora com os comandos exibidos na figura 5.2 e colocados no HOME do usuário usado nos testes.

```
root@server:~# dd if=/dev/urandom of=pequeno.bin bs=1M count=2
root@server:~# dd if=/dev/urandom of=grande.bin bs=1M count=250
```

Figura 5.2: Criação de arquivos para teste

O teste de desempenho do *cache* dos sistemas de arquivos consistia de copiar os arquivos criados um primeira vez para um diretório local na estação cliente e, após a primeira cópia, refazer a cópia 1000 vezes para o arquivo *pequeno.bin* e 40 vezes para o arquivo *grande.bin*.

Para automatizar as cópias, o script da figura 5.3 foi usado. Os tempos expostos na tabela 5.4 são médias da repetição de todo oprocesso 10 vezes.

```
root@server:~#cat copiar.sh
#!/bin/bash
_ori="$1" ; _dest="$2" ; _vezes="$3"
for i in ‘seq 1 "$_vezes"‘ ; do
 #       echo $i
        cp -f $_ori $_dest
done
```

Figura 5.3: Script auxiliar no teste do cache

A figura 5.4 exibe um exemplo de saída de um dos testes de desempenho do sistema de arquivos.

```
cbpf@client:~$time ./copiar.sh pequeno.bin /tmp 1000
0.86user 3.99system 0:04.47elapsed 60%CPU (0avgtext+0avgdata 0maxresident)k
64inputs+0outputs (1major+259839minor)pagefaults 0swaps
```

Figura 5.4: Medição de tempos de transferências de arquivos

Para o teste do NFS, o diretório HOME do usuário "cbpf" foi desmontado, deixando de fazer uso do protocolo SSH, e remontado usando NFS. Exceto pelo sistema de arquivos usado, todo o cenário do teste com o SSHomeFS se manteve inalterado.

A tabela 5.5 exibe os tempos encontrados nos testes dos sistemas de arquivos.

Tabela 5.5: Desempenho dos sistemas de arquivos a Fast-Ethernet

Estação cliente a 100Mbps Full Duplex			
pequeno.bin	1ª vez	2,61	2,63
	1000x	4,14	3,95
grande.bin	1ª vez	22,49	22,98
	40x	15,98	15,17
		SSH	**NFS**

Para uma melhor avaliação dos dados da tabela 5.5, os testes de desempenho dos sistemas de arquivos foram refeitos sob uma comunicação cliente-servidor de 1Gbps. A tabela 5.6 exibe os tempos encontrados nos testes dos sistemas de arquivos com conexão padrão Gigabit-Ethernet. Nota-se que fazendo uso do SSHomeFS o arquivo *pequeno.bin* teve uma variação do tempo do primeiro acesso inferior à 20% , e a variação da criação de mais 1000 arquivos como este foi praticamente desprezível.

Tabela 5.6: Desempenho dos sistemas de arquivos a Gigabit Ethernet

Estação cliente a 1Gbps Full Duplex			
pequeno.bin	1ª vez	2,151	2,09
	1000x	4.05	4.09
grande.bin	1ª vez	13	6,88
	40x	15,19	14,21
		SSH	**NFS**

Capítulo 6

Conclusões

Neste trabalho foi proposto uma forma não comum de compartilhamento dos diretórios HOME em uma rede. Para isso, ferramentas *open sources* de uso comum em administração Linux foram usadas para construir um ambiente de tráfego seguro de arquivos entre estações clientes e máquina servidora, com autenticação por usuário.

Com o uso do SSHomeFS, os tempos de *login* aumentaram significativamente para os ambiente gráficos GNOME e KDE4 (tabela 5.3), nas respectivas proporções de 4 e 3,5 vezes o tempo necessário para o *login* usando NFS. Os tempos de *login* dos ambientes gráficos mais leves (tabela 5.3) não apresentaram mudanças significativas, estes tempos sofreram um pequeno aumento de aproximadamente 1,5 segundos em relação ao tempo proporcionado pelo NFS.

Os tempos de execuções de programas com o diretório HOME montado sob o SSHomeFS (tabela 5.4) também não apresentaram significativas mudanças. Este comportamento torna possível o uso do SSHomeFS em ambiente de produção, sem penalizar severamente o desempenho das estações clientes, independente do gerenciador de ambientes escolhido.

Com exceção dos testes da tabela **??**, todos os demais testes foram realizados em um ambiente Fast Ethernet. Conforme apresentado nesta tabela e na tabela 5.4, o desempenho do SSHomeFS sofre alterações extremamente pequenas em acesso a arquivos pequenos, variações inferiores à 20% no primeiro acesso e desprezíveis nos demais acessos. Este comportamento possibilita o uso do SSHomeFS em redes de 100Mbps, ou em redes 1000Mbps com um pequeno ganho de desempenho

para arquivos de tamanhos maiores.

Como exposto no Capítulo 5, o uso do protocolo SSH permite ao pacote SSHomeFS executar sua tarefa, superando as principais barreiras encontradas. Apesar do aplicativo DCOPserver do KDE3 ter apresentado problema, o pacote SSHomeFS fornece um ambiente seguro com totais condições de trabalho, tanto em desempenho quanto em variedades de *softwares* capazes de serem executados sob seu compartilhamento.

Conforme os objetivos propostos na seção 1.2, pode-se concluir que o pacote SSHomeFS cumpre com o proposto, oferecendo uma forma segura de compartilhamento dos diretórios HOME dos usuários, fazendo uso de *softwares* livres presentes na grande maioria das distribuições Linux.

Além de atingir os objetivos propostos, o pacote SSHomeFS abre possibilidades de criação de ferramentas que facilite o uso de outros protocolos no compartilhamento dos HOME dos usuários.

Na possibilidade de interoperar diferentes protocolos no compartilhamento dos diretórios HOME, o pacote SSHomeFS pode ser a primeira de muitas etapas. Tomando como base os conhecimentos expostos na criação e uso dessa ferramenta, o autor sugere como trabalhos futuros as seguintes continuações:

a) o pacote SSHomeFS trabalha como simples aplicativos independentes, onde cada um executa suas atribuições. Ainda usando *shell script*, uma reestruturação poderia ser feita para transformar o pacote SSHomeFS em aplicação cliente/servidor, onde o usuário informaria a cada *login* o local da rede onde se encontra seu diretório HOME e o protocolo usado para acessar este diretório. A aplicação cliente, o substituto do *autosshfs.sh*, se encarregaria de informar os dados necessários à montagem do HOME e demais informações. A aplicação servidora, o substituto do *rc.sshfs*, que desta vez será executada pelo sistema, executaria a montagem propriamente dita do diretório HOME, anteriormente feita pelo *autosshfs*;

b) para aumentar a segurança, diminuindo os riscos de alterações indevidas dos aplicativos criados por pessoas não autorizadas, o autor sugere a criação de um pacote similar ao SSHomeFS em uma linguagem compilada, como C, GTK ou QT por exemplo;

c) o melhoramento da interface com o usuário também seria de grande importância. Uma ferramenta com melhores suportes à bibliotecas gráfica poderia ser criada em substituição ao KDM, GDM ou XDM. Para isso, linguagens como Phyton, GTK ou QT poderiam ser usadas. Tal aplicativo deveria suportar a comunicação com a aplicativo servidor criado na sugestão “a” deste capítulo.

Referências Bibliográficas

CALLEJA, D. *Three reasons why ReiserFS is great for you*. Kernelnewbies.org, 2003. Acessado em 13/03/2009. Disponível em: <http://web.archive.org/web/20030208005816% -/http://namesys.com/content_table.html>.

CALLEJA, D. *Ext4*. Kernelnewbies.org, 2009. Acessado em 13/03/2009. Disponível em: <http://kernelnewbies.org/Ext4>.

CARD, R.; TS'O, T.; TWEEDIE, S. *Design and Implementation of the Second Extended Filesystem*. Massachusetts Institute of Technology, 2009. Acessado em 13/03/2009. Disponível em: <http://web.mit.edu/tytso/www/linux/ext2intro.html>.

CENTER, P. S. *The Andrew File System*. Pittsburgh Supercomputing Center, Carnegie Mellon University, University of Pittsburgh, 2009. Acessado em 15/03/2009. Disponível em: <http://www.psc.edu/general/filesys/afs/afs.php>.

CONECTIVA, t. *Conectiva 10.0*. Conectiva.com, 2004. Acessado em 27/12/2009. Disponível em: <http://www.conectiva.com/doc/livros/online/10.0/usuario/pt_BR/apa.html>.

FERREIRA, S. R. *Sambando com Linux*. 1. ed. Rio de Janeiro: Alta Books, 2007.

GAHNOMEN. Dfg-Crew.com, 2005. Acessado em 13/03/2009. Disponível em: <http://www.dfg-crew.com/index.php?entry=entry050613-191152>.

GNU, c. *Bash Reference Manual*. The GNU Bash Reference Manual, 2006. Acessado em 30/10/2009. Disponível em: <http://www.gnu.org/software/bash/manual% -/bashref.html>.

HOPKINS, S.; COILE, B. *AoE (ATA over Ethernet)*. http://www.coraid.com, 2009. Acessado em 06/03/2009. Disponível em: <http://support.coraid.com/documents/AoEr11.pdf>.

HUANG, T. *GA-G31M-S2L/S2C User's Manual*. rev.1102. Gigabyte.com, 2007. 34 p. Acessado em 06/03/2009. Disponível em: <http://www.gigabyte.com.tw/Support/Motherboard/Manual_Model.aspx?ProductID=2693>.

HUNT, C. *Servidores de Redes com Linux*. 1. ed. São Paulo: Ciência Moderna, 2000. 426 p.

HUNT, C. *Linux Servidores de Rede*. 1. ed. Rio de Janeiro: Ciência Moderna, 2004. 296 p.

INC, H. *Brutus*. Hoobie.net, 2000. Acessado em 13/03/2009. Disponível em: <http://www.hoobie.net/brutus/brutus-download.html>.

KERBEROS, M. *Kerberos: The Network Authentication Protocol*. MIT.edu, 2009. Acessado em 15/03/2009. Disponível em: <http://web.mit.edu/Kerberos/>.

KUKUK, T. *The RPC Portmapper*. The Linux Documentation Progect, 2003. Acessado em 31/02/2009. Disponível em: <http://tldp.org/HOWTO/NIS-HOWTO/portmapper.html>.

KUKUK, T. *NIS+*. Linux-Nis.org, 2006. Acessado em 18/12/2008. Disponível em: <http://www.linux-nis.org/nisplus/>.

KUKUK, T. *NIS*. Linux-Nis.org, 2007. Acessado em 22/12/2008. Disponível em: <http://www.linux-nis.org/>.

LAM, S. http://hightek.org/dialog/, 2002. Acessado em 10/01/2009. Disponível em: <http://hightek.org/dialog/manual-0.7.html>.

LANGFELDT, N. *NFS How-To*. The Linux Documentation Progect, 1999. Acessado em 22/01/2009. Disponível em: <http://br.tldp.org/projetos/howto/arquivos/html/nfs.howto/nfs.howto.pt_BR-636.html>.

L.P, H.-P. D. C. *HP-UX System Administrator's Guide*. HP.com, 2008. Acessado em 13/03/2009. Disponível em: <http://docs.hp.com/en/5992-3387/ch06.html>.

MACDONALD, J.; REISER, H.; ZAROCHENTCEV, A. *Reiser4 Transaction Design Document*. Archive.org, 2002. Acessado em 13/03/2009. Disponível em: <http://web.archive.org/web/20070923032838% -/www.namesys.com/txn-doc.html>.

MASON, C. *The Btrfs Filesystem*. Oracle.com, 2007. Acessado em 13/03/2009. Disponível em: <http://oss.oracle.com/projects/btrfs/dist/documentation/btrfs-ukuug.pdf>.

MICROSYSTEMS, S. *ZFS Documentation*. OpenSolaris.org, 2009. Acessado em 13/03/2009. Disponível em: <http://opensolaris.org/os/community/zfs/docs/>.

MORIMOTO, C. E. *Servidores Linux, Guia Prático*. 1. ed. Porto Alegre: Sul Editores, 2008.

NETWORK, M. D. *NTFS Basics*. NTFS.com, 2009. Acessado em 13/03/2009. Disponível em: <http://www.ntfs.com/ntfs_basics.htm>.

NETWORK, M. D. *THE FAT FILE SYSTEMS. FAT32 FAT16 FAT12*. NTFS.com, 2009. Acessado em 13/03/2009. Disponível em: <http://www.ntfs.com/fat-systems.htm>.

NETWORK Share Brute Forcer. Dfg-Crew.com, 2005. Acessado em 13/03/2009. Disponível em: <http://www.dfg-crew.com/index.php?entry=entry050613-191152>.

OECONNECT. *OEConnect*. Projektas, 2007. Acessado em 13/03/2009. Disponível em: <http://openeyes.projektas.lt/projects/OETools/OEConnect/index.htm>.

OPENLDAP, F. T. *The OpenLDAP Projec*. OpenLDAP.org, 2008. Acessado em 04/01/2009. Disponível em: <http://www.openldap.org/>.

PHILLIPS, D.; NAGHIBZADEH, S.; ZENCZYKOWSKI, M.; MEYER, C.; ROPONEN, T.; STUHL, B. K.; HIROFUMI, O.; HUBER, T.; KUMAR, P.; FIETZ, J.; STUDENTS, I. of C. T. *Tux3 Versioning Filesystem*. Tux3.org, 2007. Acessado em 13/03/2009. Disponível em: <http://tux3.org/shapor-tux3/doc/design.html>.

SAMBA, D. *Samba*. Samba.org, 2009. Acessado em 27/02/2009. Disponível em: <http://samba.org/>.

SATRAN, J.; METH, K.; SAPUNTZAKIS, C.; SYSTEMS, C.; CHADALAPAKA, M.; CO., H.-P.; ZEIDNER, E.; IBM. *AoE (ATA over Ethernet)*. The Internet Engineering Task Force, 2004. Acessado em 06/03/2009. Disponível em: <http://www.ietf.org/rfc/rfc3720.txt>.

SGI, C. O. of. *XFS: A high-performance journaling filesystem*. SGI.com, 2009. Acessado em 13/03/2009. Disponível em: <http://oss.sgi.com/projects/xfs/datasheet.pdf>.

SHEMESH, S. *sshpass*. Sourceforge.net, 2008. Acessado em 13/03/2009. Disponível em: <http://sourceforge.net/projects/sshpass/>.

SILVA, G. M. da. *Guia Foca Linux*. 5.60. ed. Guiafoca.org, 2007. Acessado em 17/01/2009. Disponível em: <http://www.guiafoca.org/guia/intermediario/ch-disc.html>.

SLADKEY, R. *nfs and nfs4 fstab format and options*. 0.99. ed. [S.l.], nov. 1993. Acessado em 27/12/2008. Disponível em: </usr/man/man5/nfs.5.gz>.

SMITH, C. *Linux NFS-HOWTO*. Sourceforge.net, 2006. Acessado em 28/02/2009. Disponível em: <http://nfs.sourceforge.net/nfs-howto/ar01s05.html>.

SMITH, C. *NFS*. SourceForge.net, 2006. Acessado em 30/01/2009. Disponível em: <http://nfs.sourceforge.net/nfs-howto/>.

SMITH, R. W. *Linux Ferramentas Poderosas*. 1. ed. São Paulo: Ciência Moderna, 2004. 303-327 p.

SZEREDI, M. *SSH Filesystem*. SourceForge.net, 2008. Acessado em 20/12/2008. Disponível em: <http://fuse.sourceforge.net/sshfs.html>.

TWEEDIE, S. *Journaling for ext2f*. Kernel.org, 2001. Acessado em 13/03/2009. Disponível em: <ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3/README>.

WINDOWS Network Neighborhood Shared Resource Password Recovery. Dfg-Crew.com, 2005. Acessado em 13/03/2009. Disponível em: <http://www.dfg-crew.com/index.php?entry=entry050613-191152>.

YLONEN, T. *Request for Comments*. IETF, The Internet Engineering Task Force, 2006. Acessado em 02/01/2009. Disponível em: <http://www.ietf.org/rfc/rfc4251.txt>.

YLONEN, T. *OpenSSH*. OpenSSH.org, 2009. Acessado em 26/02/2009. Disponível em: <http://www.openssh.org>.

YLONEN, T. *OpenSSH*. OpenSSH.org, 2009. Acessado em 26/02/2009. Disponível em: <http://www.openssh.org/features.html>.

YLONEN, T.; CAMPBELL, A.; BECK, B.; FRIEDL, M.; PROVOS, N.; RAADT, T. de; SONG, D.; FRIEDL, M.; PROVOS, N. *OpenSSH SSH client*. [S.l.], 1999. Acessado em 03/12/2008. Disponível em: </usr/man/man/ssh.1.gz>.

YLONEN, T.; CAMPBELL, A.; BECK, B.; FRIEDL, M.; PROVOS, N.; RAADT, T. de; SONG, D.; FRIEDL, M.; PROVOS, N. *OpenSSH SSH daemon configuration file*. [S.l.], 1999. Acessado em 15/12/2008. Disponível em: </usr/man/man5/sshd_config.5.gz>.

Apêndice A

Código fonte do pacote *SSHomeFS*

Documentação sobre o SSHomeFS e os códigos fontes atualizados podem ser encontradas em http://www.cbpf.br/∼mgm

A.1 Fonte do *rc.sshfs*

```
#!/bin/bash
#####################################################
#                                                   #
#   Check and Umount ssh_file_system for users      #
#                                                   #
# This program is part of the package:              #
# .bash_logout <-Umount ssh_file_system             #
# rc.sshfs <-Itself                                 #
# autosshfs.sh <--Check and mount ssh_file_system   #
#                                                   #
```

```
# Create by:                                          #
# CAT - Coordenacao de Atividades Tecnicas            #
# CBPF - Brazilian Center For Physics Research        #
# Marcelo giovani - mgm@cbpf.br                       #
# October 30 2008                                     #
#                                                     #
# Last Modified:                                      #
# Marcelo Giovani - mgm@cbpf.br                       #
# November 02 2008                                    #
#                                                     #
#######################################################
#                                                     #
# This file must have 755 permission and its owner#
#must be "root".                                      #
# This file must be in the /etc/rc.d (Slackware       #
#and system based in "init" start), and referenced#
# to it in rc.local. In case of systems based in      #
# "system V" start, it have be in                     #
#/etc/init.d directory and a symbolic link in the     #
#their runlevels (/etc/rc3.d, rc4.d...).              #
#                                                     #
#######################################################


Tmp=/tmp/fs
Data=‘date "+%Y_%M_%d_%H:%M:%S"‘
Quota=no          #Habilita quota nos "HOMEs locais" [yes|no]
User_quota_default="default"     #Usuario cuja quota sera copiada para outros


#----- Cria lista de usuarios validos ----------------------------------------------------#
Funcao_cria_lista(){
```

```
ypcat passwd | cut -d":" -f1 > $Tmp/list  #Cria lista de usuarios em $Tmp/list
}
#-------------------------------------------------------------------------------#

#----- Verifica pasta temporaria e cria lista de usuarios validos ------------------#
Funcao_verifica_tmp_e_cria_lista(){
Permissao_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f1‘
Dono_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f3‘
Grupo_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f4‘
if [ "$Permissao_tmp" != ’drwxr-xr-x’ -o "$Dono_tmp" != "root" -o "$Grupo_tmp" != "root" ] ; then
     rm -rf $Tmp
     mkdir -m 755 $Tmp
     touch $Tmp/list
     chmod 600 $Tmp/list
     echo "$Tmp was necessary to adjust"

fi
Funcao_cria_lista
}
#-------------------------------------------------------------------------------#

#----- Verifica/Criando HOMEs e FIFOs ----------------------------------------#
Funcao_verifica_fifos(){
while read User_valido ; do

     #----- Verifica/Criando FIFOs -----#
     Permissao_fifo=‘ls -l $Tmp/$User_valido 2> /dev/null | cut -d" " -f1‘
     Dono_fifo=‘ls -l $Tmp/$User_valido 2> /dev/null | cut -d" " -f3‘
     if [ "$Permissao_fifo" != ’prw-------’ -o "$Dono_fifo" != "$User_valido" ] ; then
          if [ -n "$Permissao_fifo" ] ; then
               rm -rf $Tmp/$User_valido
               mkfifo -m 600 $Tmp/$User_valido
```

```
                chown $User_valido $Tmp/$User_valido
                echo "$Tmp/$User_valido was necessary to adjust"
           else
                mkfifo -m 600 $Tmp/$User_valido
                chown $User_valido $Tmp/$User_valido
                echo "$Tmp/$User_valido was necessary to create"
           fi
      fi

      #----- Verifica/Criando HOMEs -----#
      Permissao_home=‘ls -ld /home/$User_valido 2> /dev/null | cut -d" " -f1‘
      Dono_home=‘ls -ld /home/$User_valido 2> /dev/null | cut -d" " -f3‘
      if [ "$Permissao_home" != ’drwx------’ -o "$Dono_home" != "$User_valido" ] ; then
           if [ -e "/home/$User_valido" ] ; then
                mv -f /home/$User_valido /home/$User_valido-$Data"-Contac-TheAdm"
                chmod 700 /home/$User_valido-$Data
                chown root /home/$User_valido-$Data
                cp -Rfp /etc/skel /home/$User_valido
                chown -R -f $User_valido /home/$User_valido
                chmod -R 700 /home/$User_valido
                echo "/home/$User_valido was necessary to adjust"
           else
                cp -Rfp /etc/skel /home/$User_valido
                chown -R -f $User_valido /home/$User_valido
                chmod -R 700 /home/$User_valido
                echo "/home/$User_valido was necessary to create"
           fi
           if [ "$Quota" = "yes" ] ; then
                edquota -p $User_quota_default $User_valido
           fi
      fi
done < $Tmp/list
```

```
rm -f $Tmp/list
}
#--------------------------------------------------------------------------------#


#----- Verifica pasta temporaria e cria lista de usuarios validos "CHECK" -----------#
Funcao_verifica_tmp_e_cria_lista_check(){
Permissao_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f1‘
Dono_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f3‘
Grupo_tmp=‘ls -ld $Tmp 2> /dev/null | cut -d" " -f4‘
echo
echo "Below is shown the erro to TMP:"
echo -e "Description\t\tParameter\t\"Should be\""
echo "-------------------------------------------------"
if [ "$Permissao_tmp" != ’drwxr-xr-x’ -o "$Dono_tmp" != "root" -o "$Grupo_tmp" != "root" ] ; then
     echo -e "Permission\t$Permissao_tmp\tdrwxr-xr-x"
     echo -e "Owner\t\t$Dono_tmp\t\troot"
     echo -e "Group\t\t$Grupo_tmp\t\troot"
     echo
fi
Funcao_cria_lista
}
#--------------------------------------------------------------------------------#

#----- Verifica/Criando HOMEs e FIFOs "CHECK" -------------------------------------#
Funcao_verifica_fifos_check(){
echo
echo "Below is shown the erro to FIFOs or HOMEs:"
echo -e "Descip\t\tParameter\t\"Should be\""
echo "-------------------------------------------------"
while read User_valido ; do
```

```
#----- Verifica/Criando FIFOs -----#
Permissao_fifo=‘ls -l $Tmp/$User_valido 2> /dev/null | cut -d" " -f1‘
Dono_fifo=‘ls -l $Tmp/$User_valido 2> /dev/null | cut -d" " -f3‘
if [ -z "$Permissao_fifo" ] ; then
     Permissao_fifo=’?’
fi
if [ -z "$Dono_fifo" ] ; then
     Dono_fifo=’?’
fi
if [ "$Permissao_fifo" != ’prw-------’ -o "$Dono_fifo" != "$User_valido" ] ; then
     echo -e "$Tmp/$User_valido\t$Permissao_fifo\tprw-------"
     echo -e "$Tmp/$User_valido\t$Dono_fifo  \t$User_valido"
     echo
fi

#----- Verifica/Criando HOMEs -----#
Permissao_home=‘ls -ld /home/$User_valido 2> /dev/null | cut -d" " -f1‘
Dono_home=‘ls -ld /home/$User_valido 2> /dev/null | cut -d" " -f3‘
if [ -z "$Permissao_home" ] ; then
     Permissao_home=’?’
fi
if [ -z "$Dono_home" ] ; then
     Dono_home=’?’
fi
if [ "$Permissao_home" != ’drwx------’ -o "$Dono_home" != "$User_valido" ] ; then
     echo -e "/home/$User_valido\t$Permissao_home\tdrwx------"
     echo -e "/home/$User_valido\t$Dono_home   \t$User_valido"
     echo
     if [ "$Quota" = "yes" ] ; then
          edquota -p $User_quota_default $User_valido
     fi
fi
```

```
done < $Tmp/list
rm -f $Tmp/list
echo
echo "If there was errors, exec \"$0 start\" to correct the errors!"
echo
}
#-------------------------------------------------------------------------------#


#----- Main ------------------------------------------------------------------#
if [ "$1" = "start" -o "$1" = "restart" ] ; then
     Funcao_verifica_tmp_e_cria_lista
     Funcao_verifica_fifos
     exit 0
elif [ "$1" = "check" ] ; then
     Funcao_verifica_tmp_e_cria_lista_check
     Funcao_verifica_fifos_check
     exit 0
else
     echo "Please, usage $0 {start|check|restart}"
     exit 1
fi
#-------------------------------------------------------------------------------#
exit 1
```

A.2 Fonte do *autosshfs*

```
#!/bin/bash
#####################################################
#                                                   #
#   Check and mount ssh_file_system for users       #
#                                                   #
```

```
# This program is part of the package:            #
# .bash_logout <-Umount ssh_file_system           #
# rc.sshfs <-Prepare the enviroment for sshfs.sh  #
# autosshfs.sh <--Itself                          #
#                                                 #
# Create by:                                      #
# CAT - Coordenacao de Atividades Tecnicas        #
# CBPF - Brazilian Center For Physics Research    #
# Marcelo giovani - mgm@cbpf.br                   #
# October 22 2008                                 #
#                                                 #
# Last Modified:                                  #
# Marcelo Giovani - mgm@cbpf.br                   #
# December 04 2008                                #
#                                                 #
###################################################
#                                                 #
# This file must have 755 permission and its owner#
#must be "root".                                  #
# This file must be in the etc/profile.d directory#
#in clients. (can be placed inside of the file    #
#/etc/profile )                                   #
#                                                 #
###################################################

trap "" 2 3 15 20    #Proibir "ctrl+c" e outros
stty -echo
File_server="IP_do_servidor_SSH" #Servidora de arquivos SSHFS, onde estah o home.
Liberado="500"       #UID de um user local liberado do sshfs nos clientes
Limite_tentativas="2"      #Num de tentativas de montagens antes de deslogar o user

Log=/tmp/$User"_sshomefs".log #Local de log de erro. O usuario tem que ter permissao de escrita
```

```
Titulo_janelas="Coordenacao de Atividades Tecnicas - CAT - CBPF"
Mensagem_erro_padrao="#> Mensagem \"PERSONALIZADA\" de ... \n\n \"ERRO\" "

Dialog=‘which dialog‘      ####Path dos comando
User=‘whoami‘        ####usados
Sshfs=‘which sshfs‘     ####
Arq_pass=/tmp/fs/$User     #Permissao 755 pata a pasta; 600 dono $User para arquivo FIFOs $User

#----- Primeiras mensagens ----------------------------------------------------------#
if [ "$UID" != "0" -a "$UID" != "$Liberado" ] ; then
     if [ "$File_server" = "IP_do_servidor_SSH" ] ; then
          echo ""
          echo " Configure as variaveis do programa, como"
          echo "o IP do servidor de arquivos, por exemplo."
          exit 1
     fi
fi
#------------------------------------------------------------------------------------#

#----- Escolher gerenciador grafico -------------------------------------------------#
Icewm="/usr/bin/starticewm"   #Caminha para iniciar o kde3   # \
Kde="/usr/bin/startkde"    #Caminha para iniciar o kde4   #  > SE ALTERAR, EDITE A "Funcao_escolher_gui"
Gnome="/usr/bin/startgnome"   #Caminho para iniciar o gnome  # /
Fluxbox="/usr/bin/startfluxbox"  #Caminho para iniciar o fluxbox#
Funcao_escolher_gui(){
Gui=‘$Dialog --stdout --nocancel \
--backtitle "$Titulo_janelas" \
--title "Choice of Graphic Interface" \
--menu "Choose your graphic interface to use now" 0 0 0 \
ICEWM "Fast and friendly" \
GNOME "More widespread" \
```

```
KDE "Best visual effects" \
FLUXBOX "The fastest" \
TEXT "Text mode CLI" ‘

if [ "$Gui" = "ICEWM" ] ; then
     xinit $Icewm
fi
if [ "$Gui" = "GNOME" ] ; then
     xinit $Gnome
fi
if [ "$Gui" = "KDE" ] ; then
     xinit $Kde
fi
if [ "$Gui" = "FLUXBOX" ] ; then
     xinit $Fluxbox
fi
}
#-------------------------------------------------------------------------------#

#----- Sair do programa se a pasta onde ficarao os FIFOs nao existir ----------------#
if [ "$UID" != "0" -a "$UID" != "$Liberado" -a ! -e "$Arq_pass" ] ; then
     echo "‘date‘ Erro nos FIFOs ou na pasta \"fs\" " >> $Log
     echo "Error, there isn’t $Arq_pass"
     echo "$Mensagem_erro_padrao"
     exit 1
fi
#-------------------------------------------------------------------------------#


#----- Sair do programa se as permissoes, donos e local dos FIFOs forem erradas -----#
if [ "$UID" != "0" -a "$UID" != "$Liberado" ] ; then
     Arq_pass_local=‘dirname 2> /dev/null $Arq_pass‘
```

```
     Permissao_arq_pass=‘ls -l $Arq_pass 2> /dev/null | cut -d" " -f1‘
     Permissao_arq_pass_local=‘ls -ld $Arq_pass_local 2> /dev/null | cut -d" " -f1‘
     Dono_arq_pass=‘ls -l $Arq_pass 2> /dev/null | cut -d" " -f3‘
     if [ "$Permissao_arq_pass_local" != ’drwxr-xr-x’ -o "$Permissao_arq_pass" != ’prw-------’ -o \
"$Dono_arq_pass" != "$User" ] ; then
          echo "‘date‘ Erro nos FIFOs \(Permissao,dono\) ou na pasta \"fs\" " >> $Log
          echo "Security error.!!!"
          echo "Please, $Mensagem_erro_padrao"
          exit 1
     fi
fi
#-------------------------------------------------------------------------------------#

#----- Sair do programa em caso de erro no HOME local -------------------------------#
if [ "$UID" != "0" -a "$UID" != "$Liberado" ] ; then
     Permissao_home_local=‘ls -ld /home/$User 2> /dev/null | cut -d" " -f1 | cut -c4-10 | grep ’w’‘
     Dono_home_local=‘ls -ld /home/$User 2> /dev/null | cut -d" " -f3‘
#     if [ "$Permissao_home_local" != ’drwx------’ -o "$Dono_home_local" != "$User" ] ; then
     if [ -n "$Permissao_home_local" -o "$Dono_home_local" != "$User" ] ; then
          echo "‘date‘ Erro no HOME local" >> $Log
          echo "Error, problem with local HOME!"
          echo "$Mensagem_erro_padrao"
          exit 1
     fi
fi
#-------------------------------------------------------------------------------------#


#----- Main -------------------------------------------------------------------------#
if [ "$UID" = "0" -o "$UID" = "$Liberado" ] ; then  #Libera user "500" para acesso sem sshfs
     Sair="OK" ; echo "Sair=$Sair" >> $Log
else
```

```
     Sair="Executar" ; echo "Sair=$Sair" >> $Log
fi
if [ "$Sair" != "OK" ] ; then
     Num_logins=‘expr $(w $User | wc -l) - 2‘   #Quantidade de secoes ativas do usuario
     if [ "$UID" != "0" -a "$Num_logins" -le "1" ] ; then #Verifica se o usuario NAO for root NEM estiver logado
          export HOME="/home/$User"
          Mount=‘mount | grep "$User@$File_server:$HOME on $HOME"‘
          if [ -z "$Mount" ] ; then          #Se o home ja estiver montado -> nada afazer
               Tentativas=0 ; Erro_montagem=1
               while [ "$Erro_montagem" != "0" ] ; do
                    Pass=‘$Dialog --stdout --nocancel \
                    --backtitle "$Titulo_janelas" \
                    --title "Confirmation of Security" \
                    --passwordbox ’Retype your password to check security: ’ 0 0 ‘
                    echo $Pass > "$Arq_pass" &
                    unset $Pass
                    $Sshfs -o reconnect -o nonempty -o password_stdin $User@$File_server:/home/$User /home/$User \
< "$Arq_pass" && Erro_montagem=0 || echo "‘date‘ Erro na montagem do sshfs" >> $Log

#Para erros de acesso de arquivos pelo root, use a linha abaixo no lougar de linha de cima.
#Nao esquecer de criar o arquivo /etc/fuse.conf contendo "user_allow_other" para o parametro "-o allow_root"
                    #$Sshfs -o allow_root -o nonempty -o password_stdin $User@$File_server:/home/$User /home/$User <
# "$Arq_pass" && Erro_montagem=0 || echo "‘date‘ Erro na montagem do sshfs" >> $Log

                    if [ "$Erro_montagem" = "0" ] ; then
                         cd /home/$User
                         Funcao_escolher_gui
                    fi
                    Tentativas=$((++Tentativas))
                    if [ "$Erro_montagem" != "0" -a "$Tentativas" -lt "$Limite_tentativas" ] ; then
                         $Dialog --backtitle "$Titulo_janelas" \
                         --title "Password Error!" \
```

```
                         --sleep 2 \
                         --infobox "\nPlease, type again your password" 5 45
                    fi
                    if [ "$Erro_montagem" != "0" -a "$Tentativas" -ge "$Limite_tentativas" ] ; then
                         $Dialog --backtitle "$Titulo_janelas" \
                         --title "Password error!" \
                         --msgbox "\nYou will be disconnected. Please, try again or: \n\n$Mensagem_erro_padrao" 0 0
                         stty echo
                         exit 1
                    fi
               done
          fi
     fi
fi
alias ssh="ssh -F ~/.ssh/config"
stty echo
#-------------------------------------------------------------------------------#
```

A.3 Fonte do *.bash_logout*

```
#!/bin/bash
#####################################################
#                                                   #
#   Check and Umount ssh_file_system for users      #
#                                                   #
# This program is part of the package:              #
# .bash_logout <-Itself                             #
# rc.sshfs <-Prepare the enviroment for sshfs.sh    #
# autosshfs.sh <--Check and mount ssh_file_system   #
#                                                   #
# Create by:                                        #
# CAT - Coordenacao de Atividades Tecnicas          #
```

```
# CBPF - Brazilian Center For Physics Research    #
# Marcelo giovani - mgm@cbpf.br                   #
# October 30 2008                                 #
#                                                 #
# Last Modified:                                  #
# Marcelo Giovani - mgm@cbpf.br                   #
# November 02 2008                                #
#                                                 #
###################################################
#                                                 #
# This file must have 755 permission and its owner#
#must be "root".                                  #
# This file must be in the HOME directory in      #
#server_files. (can be placed in the /etc/skel)   #
#                                                 #
###################################################


User=‘whoami‘
Num_logins=‘expr $(w $User | wc -l) - 2‘  #Quantidade de secoes ativas do usuario
if [ "$UID" != "0" -a "$Num_logins" -le "1" ] ; then #Executa se o usuario NAO for root NEM ja estiver logado
     fusermount -u -z /home/$User
fi
```

Apêndice B

Procedimentos e testes

Nesta seção é apresentado de forma condensada os passos para a configuração e teste da máquina servidora e das estações clientes.

B.1 Máquina servidora

Para a configuração da máquina servidora,o autor considera que o serviço de autenticação remota já esteja devidamente configurado.

O serviço de autenticação remota usado neste trabalho foi o NIS. As configurações do serviço NIS não serão abordadas na parte da máquina servidora nem nas estações clientes.

Para a implementação do pacote SSHomeFS, basicamente deve-se seguir os seguintes passos.

Baixar e descompactar o pacote SSHomeFS. Inserir os arquivos em seus respectivos lugares.

```
root@server:~#wget http://www.cbpf.br/~mgm/sshomefs/files \
/SSHomeFS.tar.gz
root@server:~#tar -xzvf SSHomeFS.tar.gz
root@server:~#cd SSHomeFS
root@server:#for i in ‘ls /home‘ ; do  cp -v bash_logout /home \
/$i/.bash_logout ; done
root@server:#for i in ‘ls /home‘ ; do  mkdir /home/$i/.ssh; done
root@server:#for i in ‘ls /home‘ ; do  cp -v /etc/ssh/ssh_config \
```

```
/home/$i/.ssh/config ; done
root@server:#for i in ‘ls /home‘ ; do  cd /home/$i/ ; ln -s \
/tmp tmp ; done
root@server:#for i in ‘ls /home‘ ; do  chown -R $i /home/$i/ ; done
```

O diretório *.ssh*, os arquivos *.bash_logout* e *config* e o link *tmp* poderão ser inseridos no diretório */etc/skel* para que os futuros usuários criados mantenham este padrão. Em caso de distribuição que não suporte tal recurso, o script de criação de usuário da máquina servidora pode ser modificado para criar estes arquivos automaticaente.

B.2 Estação cliente

Para a configuração das estações clientes, o autor considera que o cliente de autenticação remota já esteja devidamente configurado. Considera também a presença de alguns ítens básicos de sistema como os compiladores “C/C++” e as demais dependências vistas no subcapítulo 4.1.1.

A configuração das estações clientes devem seguir, basicamente, os passos expostos nesta seção.

a) Baixar e instalar o SSHFS(SZEREDI, 2008). Caso necessário, instalar o módulo FUSE antes para resolver a dependência do SSHFS.

```
root@client:~#wget http://heanet.dl.sourceforge.net/sourceforge\
/fuse/sshfs-fuse-2.2.tar.gz
root@client:~#tar -xzvf  sshfs-fuse-2.2.tar.gz
root@client:~#cd sshfs-fuse-2.2
root@client:#./configure
root@client:#make
root@client:#make install
```

b) Baixar e descompactar o pacote SSHomeFS. Inserir os arquivos em seus respectivos lugares.

```
root@server:~#wget http://www.cbpf.br/~mgm/sshomefs/files\
/SSHomeFS.tar.gz
root@client:~#tar -xzvf SSHomeFS.tar.gz
root@client:~#cd sshomefs
root@client:#cp -v rc.sshfs /etc/rc.d/
root@client:#chmod 700 /etc/rc.d/rc.sshfs
root@client:#cp -v autosshfs.sh /etc/profile.d/
```

```
root@client:#chmod 755 /etc/profile.d/autosshfs.sh
```

c) Ajustar o *rc.local* (para esta finalidade, a distribuição Suse usa o arquivo */etc/rc.d/boot.local*), inserindo neste arquivo o trecho de código mostrado na figura 4.7 devidamente ajustado à distribuição usada.

d) Ajustar o parâmetro "StricHostKeyChecking" para "no" do arquivo */etc/ssh/ssh_config*.

e) Adicionar *exit 1* ao final dos arquivos *.bashrc* de cada usuário. ATENÇÃO, esta linha NÃO deve aparecer no usuário de teste do SSHomeFS.

```
root@client:~#for i in `ls /home` ; do  echo "exit 1" >> \
/home/$i/.bashrc; done
root@client:~#rm /home/"isento"/.bashrc
```

f) Editar o arquivo */etc/profile.d/autosshfs.sh*. Neste arquivo deve ser configurado o endereço IP da máquina servidor de arquivos, o UID de um usuário exportado pela máquinas servidora isento do SSHomeFS, o título das janelas da interface com o usuário, uma mensagem personalizada de erro e as interfaces gráficas desejadas para uso na estação cliente.

B.3 Testes

Uma vez instalados todos os *softwares*, alguns procedimentos e testes são recomendados. Estes procedimentos é a forma mais fácil de se identificar falhas e corrigí-las. Para testar manualmente o pacote SSHomeFS, deve-se seguir os procedimentos citados.

a) logar na estação cliente como root;

b) executar */etc/rc.d/rc.sshfs check*, e na presença de erros, executar */etc/rc.d/rc.sshfs start* ou corrigir manualmente as falhas;

c) em outro terminal, logar com o usuário com o UID liberado do SSHomeFS;

d) montar o HOME desse usuário manualmente. Para isso, crie um arquivo do tipo FIFO em */tmp*, coloque a senha desse usuário neste arquivo,

monte com o SSHFS o diretório HOME, entre nesse diretório, verifique seu conteúdo e o desmonte. Esse processo é mostrado abaixo usando um usuário “isento” de senha “GrandeSenhaDOisento” e a máquina servidora possui IP 192.168.0.1, como exemplo. O programa *dialog* também deve ser verificado.

```
isento@client:~$which dialog
isento@client:~$cd /tmp
isento@client:$mkfifo -m 600 user_isento
isento@client:$echo "GrandeSenhaDOisento" > user_isento &
isento@client:$sshfs -o nonempty isento@192.168.0.1:/home/isento \
/home/isento -o reconnect -o password_stdin < /tmp/user_isento
isento@client:$cd /home/isento
isento@client:~$ls -lh
isento@client:~$mount
isento@client:~$fusermount -u -z /home/isento
```

Caso não tenha ocorrido nenhum erro nos passos acima, mude para outro terminal e faça login com outro usuário. Todo o processo feito manualmente será executado pelo SSHomeFS.

O teste do SSHomeFS deve ser feito por um terminal, não se deve testar outros usuários executando *su – outrousuario*, o processo de autenticação não irá funcionar corretamente mesmo que o pacote esteja perfeitamente configurado.

Durante o teste manual de SSHomeFS, deve se atentar para a solicitação de senha no primeiro uso do SSHFS. Se o sistema solicitar confirmação para a chave de encriptação o arquivo */etc/ssh/sshd_config* da estação cliente não foi configurado corretamente, isso causará falha nas primeiras conexões de cada usuário.