# Linux Network Servers

## NFS

A sigla NFS, do inglês Network File System, significa Sistema de Arquivos de Rede. Através de programas específicos no servidor e nas máquinas clientes, podemos fazer com que esses clientes acessem arquivos e diretórios deste servidor de forma transparente, como se fossem locais. NFS pode montar em sua máquina diretórios de outras máquinas ou exportar diretórios de sua maquina para outras maquinas da rede.

Estes diretórios montados localmente podem aplicar qualquer ação, pode transferir arquivos entre outros diretórios da maquina e outros ponto de montagem, respeitadas as permissões pode: ler, editar, gravar, copiar, mover, apagar etc.

Quais são as vantagens de se usar NFS?

Uma das grandes vantagens dessa funcionalidade é o fato de podermos centralizar num servidor os diretórios e arquivos utilizados por vários usuários de diversas estações.

Fica mais seguro centralizar os arquivos dos usuários?

Podemos também centralizar os diretórios pessoais dos usuários, facilitando procedimentos de backup e a administração dos mesmos. Essas vantagens se sobressaem principalmente em grandes redes.

Nessa prática vamos exportar um diretório via NFS e montá-lo.

Exportar diretórios via NFS envolve configurações no servidor e cliente, independe da distribuição, funciona com distribuições diferentes, o que importa é iniciar o serviço e corretamente configurado, também as permissões e configurações de rede.

# Linux Network Servers

Os serviços de rede podem ser divididos em três tipos básicos:

[x]inetd

portmap

standalone

Os serviços tipo [x]inetd são aqueles que dependem do superdaemon de rede inetd – InterNET daemon. A versão mais atual do inetd é o xinetd – eXtended InterNET daemon. A função de um superdaemon é apenas controlar alguns serviços que não terão daemon próprio. Reforçando a parte de processos: um daemon é um processo servidor que roda em segundo plano esperando requisições. Quando falo de inetd e xinetd, estou falando do mesmo superdaemon, mas em algumas distros (como o Debian) ainda é o inetd.

Só que o xinetd possui alguns recursos a mais como controle de acesso, capacidade de fazer logs e determinar horários para que o serviço esteja disponível. Esse superdaemon fica escutando nas portas que os serviços controlados por ele trabalham carregando o programa apropriado quando chega uma requisição na determinada porta.

Exemplos de serviços tipo inetd: telnet, FTP, POP3 etc.

Ao longo do tempo, alguns dos serviços que eram controlados pelo inetd passaram a operar como standalone a fim de contornar problemas associados ao inetd. Os serviços standalone são aqueles em que cada tipo de servidor possui seu daemon próprio. Esta forma de trabalho é preferida hoje em dia pois possibilita um maior controle sobre cada serviço em separado.

Exemplos de serviços que operam dessa forma: SSH, httpd (daemon do Apache), FTP, dentre outros.

Os serviços tipo portmap são aqueles que não possuem porta especifica para operar, como por exemplo, o NIS e o NFS. Estes serviços enviam uma chamada RPC – Remote Procedure Call – para a máquina servidora causando a execução de uma determina subrotina.

# Linux Network Servers

Dessa forma quando um cliente faz a requisição de NFS a um servidor, ele está enviando um RPC tipo NFS e que quando chegar ao servidor será tratada como tal, carregando a subrotina apropriada para enviar a resposta ao cliente.

O NFS, diferente de outros serviços, não trabalha com uma porta padrão convencional (ou seja, ele não possui uma porta própria). Sendo assim, ele utiliza um programa chamado portmapper, que utiliza uma porta fixa (111), que monitora o serviço NFS; ao receber uma conexão referente ao NFS, ele direciona o cliente para as portas certas (normalmente a porta 2049/UDP).

Para que os clientes possam acessar o servidor NFS é necessário que os seguintes daemons estejam rodando no servidor.

Para testar legal o que vamos falar agora, é ideal duas máquinas Linux!

1 - Servidor NFS

2 - Cliente Linux

Para configurar o NFS devemos instalar:

Debian:

- portmap
- nfs-kernel-server
- nfs-common

Instale o pacote nfs-kernel-server:

```
# aptitude install nfs-kernel-server
```

A configuração do NFS é bem simples, bastando apenas configurar o arquivo /etc/exports, onde definimos quais serão os diretórios que deverão ser compartilhados.

# Linux Network Servers

Antes de entrar em NFS, eu preciso explicar como funciona um recurso interessante do Linux que é o TCP Wrappers.

Os TCP Wrappers são utilizados para aplicar regras de acesso aos servidores utilizados em seu sistema, podendo permitir ou negar as conexões a eles.

Eles são controlados por dois arquivos:

/etc/hosts.allow

Configuração de acessos permitidos para determinados Ips.

/etc/hosts.deny

Configuração de acessos negados para determinados Ips.

Exemplos práticos:

Vamos supor que queremos bloquear todos os acessos ao nosso servidor de telnet:

```
# vi /etc/hosts.deny
in.telnetd : ALL
```

Para liberar o nosso servidor de telnet apenas para as máquinas da nossa rede:

```
# vi /etc/hosts.allow
in.telnetd : 192.168.0.0/24
```

Agora, vamos bloquear o acesso SSH para todos, exceto para a rede 192.168.0.0:

```
# vi /etc/hosts.deny
sshd : ALL EXCEPT 192.168.0.0/24
```

# Linux Network Servers

Então, podemos usar os arquivos /etc/hosts.allow e /etc/hosts.deny para limitar quais hosts (máquinas clientes) poderão usar esse recurso no NFS.

Para deixar as coisas mais seguras, podemos negar esse serviço no hosts.deny e no hosts.allow liberar apenas a nossa rede, ou máquinas em específico:

```
# echo "portmap: ALL" >> /etc/hosts.deny
# echo "portmap: 192.168.0.0/24" >> /etc/hosts.allow
```

Ao restringir o acesso garanto um pouco mais de segurança no meu servidor! Sendo que é válido lembrar que não existe um ambiente 100% seguro!

Agora vamos criar um diretório para exportar (compartilhar):

```
# mkdir -p /srv/nfs
```

Feito isso, vamos colocar alguns arquivo nele:

```
# cp /etc/*.conf /srv/nfs
```

Agora podemos pensar nos diretórios que vamos compartilhar, lembrando que os mesmos já devem estar criados no sistema; para isso vamos editar o arquivo /etc/exports. Nesse arquivo podemos definir uma política em cada linha.

A sintaxe desse arquivo é:

```
diretório host(opções)
```

# Linux Network Servers

Onde temos as seguintes opções mais usadas:

ro - compartilhar apenas para leitura

rw - compartilhar para leitura e gravação

root_squash - Para que o usuário root seja um usuário limitado nos privilégios.

no_root_squash - É justamente ao contrário da opção acima.

async - Útil em redes locais pois permite que o NFS transfira arquivos de forma assíncrona, sem precisar esperar pela resposta do cliente a cada pacote enviado, aumenta um pouco a velocidade de transferência de dados.

Agora temos que configurar o que será exportado, temos que editar o arquivo /etc/exports

```
# vim /etc/exports
```

Dentro dele, coloque o seguinte conteúdo:

```
/srv/nfs 127.0.0.1/24(rw)
```

Salve o arquivo.

Agora temos que dizer ao sistema para efetivamente exportar os conteúdo do etc/exports

```
# exportfs -r
```

# Linux Network Servers

Consulte os serviços exportados:

```
rpcinfo -p localhost
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  51682  status
100024    1   tcp  42781  status
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    4   udp   2049  nfs
100021    1   udp  40056  nlockmgr
100021    3   udp  40056  nlockmgr
100021    4   udp  40056  nlockmgr
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100003    4   tcp   2049  nfs
100021    1   tcp  57355  nlockmgr
100021    3   tcp  57355  nlockmgr
100021    4   tcp  57355  nlockmgr
100005    1   udp  47836  mountd
100005    1   tcp  42594  mountd
100005    2   udp  47836  mountd
100005    2   tcp  42594  mountd
100005    3   udp  47836  mountd
100005    3   tcp  42594  mountd
```

# Linux Network Servers

Verifique se o que está exportado está correto:

```
# showmount -e localhost
```

Agora vamos montar:

```
# mount -t nfs localhost:/srv/nfs /mnt
```

Veja o conteúdo de /srv/nfs está em /mnt

```
# ls /mnt
```

**DICA DE SEGURANÇA:** Nunca deixe diretórios exportados de cara para internet. NFS não suporta nenhum tipo de criptografia.

Qualquer alteração feira em /mnt será feita onde?

Todas as ações nestes diretórios montados tem seus efeitos nos diretórios que deram origem a montagem (exportados), quando desmontados as ações praticadas permanecem somente nos diretórios das maquinas remotas, na maquina local (onde montou) eles deixam de existir após a desmontagem.

Toda ação dentro do diretório montado tem efeito somente na maquina remota.

**DICA LPI:** É fundamental saber o funcionamento do no_root_squash.

# Linux Network Servers

As permissões são um detalhe muito importante ao usar NFS.

O servidor NFS acredita na máquina cliente e permite que o usuário logado nela acesse os arquivos no compartilhamento com as mesmas permissões que teria uma usuário local de mesmo ID.

Por exemplo:

Vamos supor que compartilhei o diretório /home/leo/documentos.

Os arquivos desse diretório só podem ser lidos e escritos pelo usuário leo, já que esse diretório é privado, e apenas lidos pelo resto dos usuários do sistema.

Logo, os arquivos em /home/leo/documentos só podem ser alterados se o usuário da máquina cliente tiver o mesmo ID do usuário da máquina servidora.

Ou seja não adianta o compartilhamento ter como opção rw na linha do /etc/exports se não forem ajustadas as permissões.

Por isso, é interessante que você crie diretórios públicos e aplique o que aprendeu nas aulas de administração de usuários e permissões. Compartilhar um diretório que está dentro de um outro privado não é muito interessante.

O único usuário que foge a essa regra é o usuário root.

Por padrão, o NFS não permite que o usuário root (ID = 0) de outra máquina acesse os arquivos do compartilhamento como se fosse o root local.

Para garantir que o root local não terá o mesmo privilégio do que o root do servidor usamos a opção root_squash.

Temos duas opções que são interessantes para comentar: hard x soft.

# Linux Network Servers

Por padrão, os compartilhamentos do NFS são montados com a opção hard.

Essa opção faz com que os clientes fiquem tentando se reconectar ao servidor indefinidamente caso esse saia do ar por algum motivo (queda de energia, manutenção etc) fazendo com que programas que usam os arquivos compartilhados travem ao tentar acessá-los.

Para evitar esse problema, você pode montar os compartilhamentos na máquina cliente usando a opção soft.

Assim, o compartilhamento ficará “escondido” caso o servidor fique fora do ar e programas que tentarem acessá-lo passarão a exibir mensagens de "Não é possível ler o arquivo", ao invés de simplesmente travarem.

Para montar o compartilhamento na máquina cliente ou colocamos uma linha no /etc/fstab ou montamos manualmente.

Exemplos práticos:

Na máquina cliente:

```
# mount -t nfs -o soft 192.168.1.1:/doc /mnt/doc
```

Ou no arquivo /etc/fstab para que a configuração não se perca no próximo boot:

```
192.168.1.1:/doc /mnt/doc nfs defaults,soft 0 0
```

Agora vamos iniciar o serviço, que sempre deverá seguir a ordem:

1º - portmap - Ele deve ser o primeiro, pois como o serviço nfs não possui uma porta padrão convencional, o mesmo precisa de uma porta para direcionar os clientes;

# Linux Network Servers

2º – nfs-kernel-server - Esse é o serviço!

Para iniciar:

Debian:

```
# /etc/init.d/portmap start
# /etc/init.d/nfs-kernel-server start
```

Red Hat:

```
# service portmap start
# service nfs start
```

Depois de configurar e colocar o serviço NFS no ar, é necessário fazer uma checklist para ver se está tudo legal.

Verificar se o portmap está disponível e qual porta esta atuando:

```
#cat /etc/services | grep portmap
111/tcp         portmapper
```

Verifica se a porta está em listen para receber conexões:

```
#netstat -an | grep 111
```

# Linux Network Servers

Verifica quais usuários e processos na porta 111:

```
#fuser -v 111/tcp
```

## HIDS (AIDE)

O que seria detecção de intrusão?

Engloba o processo de monitorar, identificar e notificar a ocorrência de atividades maliciosas, atividades não autorizadas que coloquem em risco e tenham como alvo ativos de tecnologia de uma rede de computadores.

E um Sistema de Detecção de Intrusão ou Intrusion Detection Systems?

(IDSs) é o conjunto de hardware e software cujo objetivo principal é identificar determinados eventos.

Qual seria a consequência de um invasor trocar o executável do sshd?

Ele poderia pegar todas as senhas facilmente, sniffar qualquer conexão. Basicamente fazer o que quiser.

O que podemos fazer para saber se nossa máquina foi comprometida ou não, ou seja, garantir que o nosso sshd está integro?

Usar um HIDS! A sigla HIDS significa Host Intrusion Detection System.

O intuito de termos um sistema de segurança local e proteger a integridade dos nossos dados em nosso sistema, evitando que algo seja roubado, danificado ou modificado para deixar uma porta de entrada em nosso sistema.

# Linux Network Servers

Como funciona um HIDS, no nosso caso o AIDE?

**AIDE (Advanced Intrusion Detection Environment).**

O AIDE é um sistema de detecção de intrusão que trabalha por checagem de integridade dos arquivos.

Quando o AIDE está configurado, é necessário gerar uma base do sistema que contém todas as informações do sistema (assinatura) até o momento em que a base foi criada.

Quando algo estranho é notificado no sistema, pode-se comparar essa base com o sistema atual, caso algo tenha sido modificado, o AIDE vai apontar o estado original do arquivo ou diretório modificado e estado atual.

O AIDE pode apontar se o tamanho do arquivo está diferente, se a localização do arquivo no disco foi modificada, permissões entre outras checagens.

Eu devo manter minha base de assinaturas na mesma máquina?

Não. Pois um invasor pode modificar um arquivo e adulterar a base de assinaturas.

Como fazer então?
Simplesmente fazer uma cópia da base.

Vamos instalar o AIDE:

```
# aptitude install aide
```

Vamos ver o arquivo de configuração:

```
# vim /etc/aide/aide.conf
```

# Linux Network Servers

Parâmetros principais:

```
# vim /etc/aide/aide.conf

database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new
database_new=file:/var/lib/aide/aide.db.new

Checksums                                                    =
md5+sha1+rmd160+haval+gost+crc32+tiger+whirlpool

OwnerMode = p+u+g // permissão + usuario + grupo

Size = s+b // tamanho + blocos

InodeData = OwnerMode+n+i+Size

n = número de links
i = inodes
```

Atualizar a configuração:

```
# update-aide.conf
```

Feito isso, vamos executá-lo:

```
# aideinit
```

Depois de instalado, podemos verificar que o AIDE cria um arquivo de configuração para cada serviço que ele é capaz de verificar.

```
# ls /etc/aide/aide.conf.d
```

Abra um desses arquivos e veja como ele é.

# Linux Network Servers

Agora que terminou a verificação, vamos gerar um  relatório.

Mova o arquivo /bin/rmdir para /root

```
# mv /bin/rmdir /root
```

Trie um arquivo qualquer:

```
# touch /bin/rmdir
```

Agora vamos comparar com a base que foi colhida anteriormente:

```
# aide -C --config=/var/lib/aide/aide.conf.autogenerated | tee
/root/relatorio.txt
```

Abra o relatório e veja o resultado:

```
File: /bin/rmdir

Size     : 13880                    , 0

Bcount   : 32                       , 0

Permissions: -rwxr-xr-x                  , -rw-r--r--

Mtime    : 2007-01-30 16:51:14           , 2008-11-24 21:29:29

Ctime    : 2008-05-19 19:23:40           , 2008-11-24 21:29:29

Inode    : 15985                    , 15926

CRC32    : QSD+GA==                      , AAAAAA==
```