OpenSSH

OpenSSH:

Introdução:

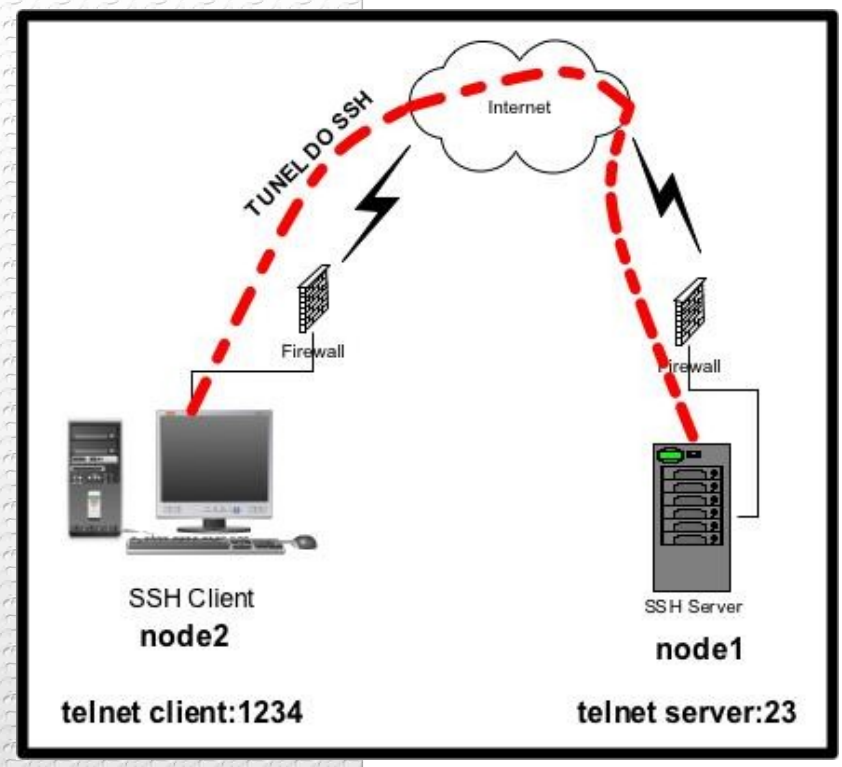
O projeto OpenSSH veio para substituir antigos métodos inseguros de comunicação e autenticação, podemos dizer que o OpenSSH é um substituto direto do *rlogin, rsh* e *telnet.* O OpenSSH também incorpora características próprias e novas funções extremamente úteis nos dias atuais fazendo o uso das rotinas e bibliotecas do OpenSSL que é pré-requisito obrigatório.

A *autenticação segura*, *execução de comandos remota segura* e *login shell remoto seguro* são as principais utilidades do popular chamado SSH. Existem diversas formas de autenticação disponíveis, podemos estabelecer listas de acesso (acls), criar padrões de senhas "fortes" e até a autenticação automática por meio de *chaves públicas criptografadas RSA ou DSA*, método extremamente útil para ser usado em shell scripts ou em eventos de manutenções automatizados.

TCP port forward:

TCP port forward é um poderoso recurso do SSH que muitos administradores ainda não fazem uso. O usuário root pode arbitrariamente fazer *TCP port forwarding* para qualquer porta/socket, isto é, criar um túnel criptografado onde qualquer aplicação ou protocolo possa trafegar. Veja o exemplo:

```
[root@node2 ~]# ssh -f -L 1234:localhost:23 10.0.0.200 sleep 30
root@10.0.0.200's password:
[root@node2 ~]# telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Red Hat Enterprise Linux Server release 5.2 (Tikanga)
Kernel 2.6.18-92.1.22.el5 on an i686
login: linuxclass
Password:
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$
[linuxclass@node1 ~]$ _
```



No exemplo sugerido foi criado um túnel na maquina local *(node2)* na porta *1234* que esta conectado a maquina *(node1) (10.0.0.200)* na porta *23*. Para testar o túnel foi utilizado o aplicativo *cliente de telnet* que esta conectando localmente na porta *1234* porém todo e qualquer tráfego nesta porta esta sendo re-direcionado para a máquina *10.0.0.200* na porta *23* onde existe um *servidor de telnet* a espera de conexões.

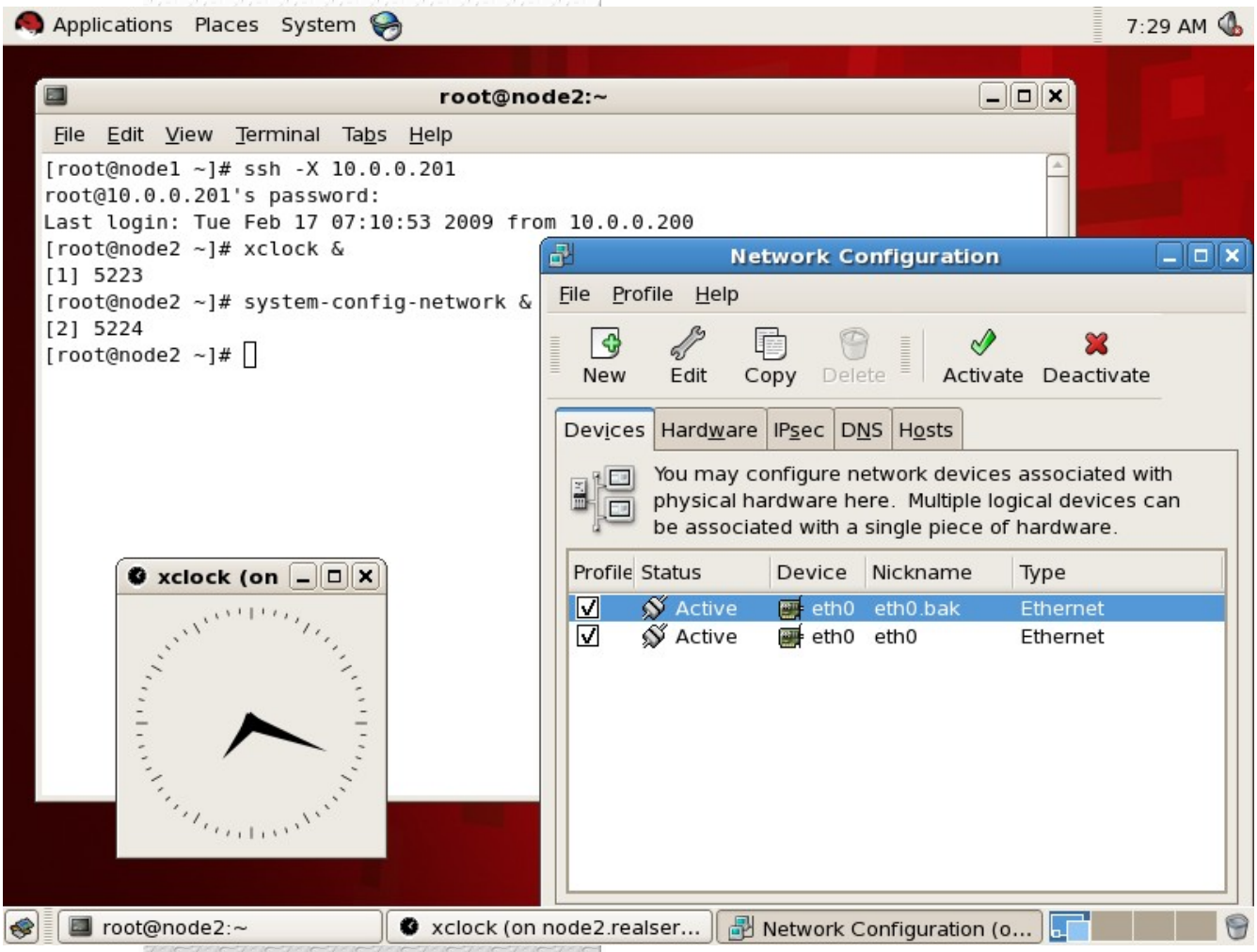
Outra possível utilização do TCP forward é estabelecer conexões através de firewalls que bloqueiam justamente as portas dos protocolos que você necessita utilizar.

X Forward:

Chamamos de *X Forwarding* quando o SSH redireciona pelo túnel criptografado a conexão ao servidor X da máquina remota para o servidor X da máquina local, desta forma, as aplicações ou programas gráficos executados pelo shell remoto irão ser mostradas no display local. Não é necessário que o servidor X da máquina remota esteja rodando. A seguir visualize o screenshot que demonstra a utilização do X forwarding.

Você chama a capacidade de X-Forwarding através dos seguintes parâmetros, veja os exemplos:

- ***ssh -X 10.0.0.200*** (Faz o uso das extensões de restrição de segurança do X11)
- ***ssh -Y 10.0.0.200*** (Não faz o uso das extensões de segurança do X11, chamado também de X11 Forward confiado pois assume-se que ambos cliente e servidor são seguros)



Instalando Pacotes/Softwares Necessários:

Em quase todas as distribuições de Linux, todos os pacotes já estarão instalados, pois o OpenSSH é considerado um software padrão e versátil de manutenção em sistemas Linux/Unix. Veja abaixo os pacotes necessários e suas descrições:

- **openssh**
  Pacote principal que provê capacidades “core” para ambos, ssh-cliente e ssh-servidor.
- **openssl**
  Bibliotecas e rotinas de criptografia e ferramentas para gerar chaves e certificados.
- **openssh-clients**
  Este pacote provê os programas clientes necessários para utilização do OpenSSH.
- **openssh-server**
  Este pacote provê o daemon sshd necessário para disponibilizar o acesso remoto de seu sistema.
- **openssh-askpass**
  Este pacote provê a capacidade de dialogo X11 do passphrase para o OpenSSH.

Configuração do Serviço:

Veremos agora a configuração do OpenSSH servidor (*sshd*), os arquivos de configuração de ambos OpenSSH Servidor e OpenSSH Cliente podem ser localizados em */etc/ssh*. O principal arquivo de configuração do Servidor esta localizado em */etc/ssh/sshd_config*. Veja abaixo as principais opções de configuração deste arquivo:

- **AllowUsers**
  Este parâmetro é a forma mais simples de especificar arbitrariamente uma lista de usuários (acls) permitidos a utilizar o sshd. Por padrão o sshd permite todos os usuários.
- **PasswordAuthentication**
  Especifica se é permitido ou não o login utilizando passwords. O padrão é permitido. O mais comum aqui é quando mudar este parâmetro para impedir o login através de senhas é criar as chaves publicas e privadas para garantir seu acesso.
- **PermitRootLogin**
  Este parâmetro controla o acesso ao super-usuário no sshd. Os possíveis valores deste parâmetro são: *"yes"* valor padrão, *"without-password"* bloqueia a autenticação do super-usuário por password. *"forced-commands-only"* Aceita autenticação do super usuário por chave publica somente se especificado algum comando (esta opção é útil para a execução de algum comando no sistema remoto pois não permite o login shell) e *"no"* desliga o acesso do super-usuário.
- **Protocol**
  Especifica as versões de protocolo suportadas pelo sshd. Lembrando que é fortemente aconselhável permitir somente o protocolo versão SSH2. O protocolo versão SSH1 ainda existe apenas para manter compatibilidade com sistemas muito antigos.
- **Port**
  Especifica a porta que o sshd irá aguardar por conexões.
- **X11Forwarding**
  Os possíveis valores são *"yes"* ou *"no"*, este parâmetro controla o redirecionamento das conexões ao servidor X. O valor padrão padrão é "yes".
- **PermitTunnel**
  Este parâmetro controla o comportamento do dispositivo TUN utilizado para estabelecer VPN. O valor padrão é desligado *"no"*.

Arquivos de configuração envolvidos:

- /etc/ssh/*
- /etc/ssh/sshd_config

Gerando chaves públicas e privadas:

Para gerar o par de *chaves* na implementação de autenticação automática siga os passos abaixo:

CHAVE PÚBLICA	CHAVE PRIVADA
Servidor	**Cliente**
~/.ssh/authorized_keys	***~/.ssh/id_rsa***

O caracter " ~ " significa o path do diretório home do usuário corrente no sistema

1. No diretório home do usuário execute o comando: ***ssh-keygen***
2. Tecle ***ENTER*** para aceitar o valor padrão para o nome da chave pública
3. Tecle ***ENTER*** para ignorar a utilização de passphrase
4. Tecle ***ENTER*** novamente para confirmar a não utilização do passphrase
5. Execute o comando: ***cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys***

Nota: Quando especificado algum passphrase esta "senha frase" será utilizada para encriptar a chave privada do usuário com um algoritmo 3DES. Sempre que for utilizada a chave privada terá que ser fornecido o passphrase para decriptar a chave antes de seu uso. Isso pode gerar alguns problemas para programas que iniciam automaticamente em tempo de boot causando o travamento do sistema a espera da senha.

Nota: O arquivo *~/.ssh/authorized_keys* guarda a lista das chaves públicas permitidas no servidor. É possível adicionar mais chaves públicas a esse mesmo arquivo apenas concatenando seu conteúdo.

Autenticação baseada em hosts (TCP-WRAPPERS):

O sshd foi compilado com suporte a biblioteca *libwrap.so*, desta forma é possível fazer uso dos arquivos */etc/hosts.allow* e */etc/hosts.deny* em ordem a implementar a autenticação baseada em hosts, isto é, em números IPs ou nomes de maquinas e domínios. Para configurar o acesso ao sshd através dos tcp-wrappers siga o exemplo abaixo, lembrando que por padrão ou omissão o acesso é sempre liberado.

sshd: <IPs ou nomes de máquinas/domínios permitidos ou bloqueados>

Exemplo:

/etc/hosts.allow	/etc/hosts.deny
sshd: 192.168.200.	***sshd: ALL***

No exemplo acima é liberado o acesso ao serviço SSH apenas para o grupo de máquina dentro da rede 192.168.200.0, para todos os outros hosts inclusive o próprio localhost será bloqueado o acesso ao servidor de sshd. Use este tipo de autenticação com cautela.

Ativando o Serviço na Inicialização do Sistema:

chkconfig sshd on ; service sshd start

Para verificar se o daemon esta sendo executado, use: *service sshd status* ou *ps -aux |grep sshd*

A verificação das portas/sockets abertos pelo sshd (*porta 22*) se dá através do comando:
netstat -al |grep ssh

Documentação disponível no sistema:

man ssh
man sshd_config

/usr/share/doc/openssh*