SSH

Uma apresentação sobre como funciona SSH e como proteger.
E exemplo de brute force attack.

Desenvolvido pelos acadêmicos:

- Aline Gemelli
- Bruno Cesca
- Mateus Calza

O que é

- Protocolo de rede
- Garante uma comunicação remota dinâmica e segura entre cliente/servidor.
- Suporte a autenticação, encriptação e a integridade dos dados transmitidos.
- Comumente utilizado para acesso seguro sob redes inseguras.

Um pouco da história

- Tatu Ylonen em 1995
- Desenvolvido após um hacking na rede de uma universidade da Finlândia
- Gerencia mais da metade dos servidores web mundiais

Curiosidades do SSH

- Vem com toda a distribuição Linux, Mac OS X, AIX, Sun Solaris, OpenBSD.
- Criptografa os arquivos enviados ao diretório do servidor.
- Substituto para telnet, rlogin, ftp, rsh, rcp, rdist e programas baseados em r *.

Camadas

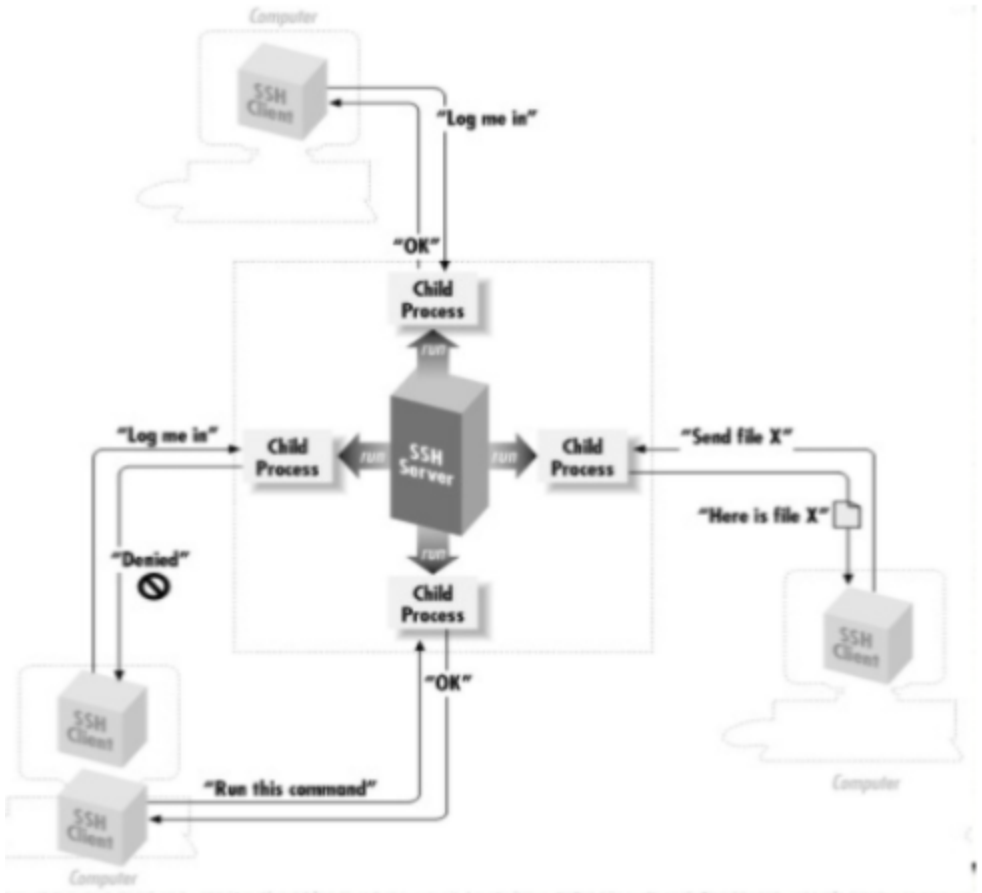
Application Layer	**ssh-connection** Session multiplexing, X11 and port forwarding, remote command execution, SOCKS proxy, etc.
	ssh-userauth User authentication using public key, password, host based, etc.
	ssh-transport Initial key exchange and server authentication, setup encryption
Transport Layer	**TCP**
Internet Layer	**IP**
Network Access Layer	**Ethernet**

Recursos

- Criptografia forte
- Autenticação forte
- Autorização
- Integridade de comunicação
- Encaminhamento ou túnel

Funcionamento

- Um handshake criptografado é feito com o servidor;
- A conexão cliente/servidor remoto é criptografada usando cifra simétrica;
- O cliente se autentica;
- O cliente agora pode interagir com segurança com o servidor


Computer
SSH Client
"Log me in"
"OK"
Child Process
run
"Log me in"
Child Process
run
SSH Server
run
Child Process
"Send file X"
"Here is file X"
"Denied"
Child Process
"OK"
SSH Client
SSH Client
"Run this command"
Computer
SSH Client
Computer

Criptografia usada pelo SSH

- Criptografia Simétrica
- Criptografia Assimétrica
- Hashing

SSH protege contra

- Escuta de dados transmitidos pela rede;
- Manipulação de dados em elementos intermediários da rede;
- Falsificação de endereço IP;
- Falsificação de DNS de nomes de host/endereços de IP confiáveis;
- SSH não irá proteger contra
- Configuração ou uso incorreto;
- Uma conta root comprometida;
- Diretórios pessoais inseguros;

Vantagens

- Tecnologia comprovada
- Criptografia forte
- Existem versões gratuitas e comerciais
- Funciona em várias plataformas
- O encapsulamento de portas funciona bem e pode ser usado para VPNs
- Muitos métodos de autenticação suportados

Desvantagens

- Port ranges e dynamic ports não podem ser encaminhados
- Não é possível restringir quais portas podem ou não ser encaminhadas, por usuário
- Quando um usuário é autenticado por senha, a identidade RSA do cliente não é verificada
- O encaminhamento de portas também pode apresentar problemas de segurança.

Acessando servidores

```
ssh usuario@host
```

Copiando arquivos com SCP

Ao rodar `man scp` em sistemas Linux para a definição do comando, você pode obter a seguinte descrição:

> scp copies files between hosts on a network. It uses ssh for data transfer, and uses the same authentication and provides the same security as ssh. scp will ask for passwords or passphrases if they are needed for authentication.

Em outras palavras, usando a estrutura e segurança existente no SSH, o SCP possibilita tranferência de arquivos. Simples, como o comando nativo `cp`.

```
# Enviando arquivo por SCP/SSH
scp meuarquivo.txt usuario@host:/tmp/meuarquivo.txt

# Recebendo arquivo por SCP/SSH
scp usuario@host:/tmp/meuarquivo.txt meuarquivo.txt
```

Túnel de portas com SSH

É normal você precisar acessar um banco de dados que só esteja disponível localmente no seu servidor. Muitas ferramentas de banco de dados possuem suporte à adicionar uma conexão por SSH, como o DataGrip e DBeaver. O SSH possui um recurso que permite trazer uma porta disponível no destino facilmente para a sua máquina. Uma forma simples de proceder é com a flag `-L`:

```
# ssh usuario@host -L portaLocal:hostNoDestino:portaNoDestino
ssh fulano@200.100.50.25 -L 3307:localhost:3306
```

Se for necessário, o caminho inverso pode ser feito com a flag `-R 3307:localhost:3306`

Configurando acesso seguro sem senha

É possível fazer o acesso usando chaves assimétricas.

Para este procedimento, garanta que há o `ssh-keygen` esteja disponível no seu terminal, em distribuições baseadas no Ubuntu/Debian pode ser instalado com `sudo apt install -y openssh-client`. Em versões mais recentes do Windows já está disponível no terminal.

O primeiro passo é gerar um par seguro de chaves na sua máquina pessoal que irá fazer o acesso, caso ainda não tenha:

```
ssh-keygen -o -a 100 -t ed25519 -f ~/.ssh/id_ed25519 -C "fulano@example.com"
```

Então você deve copiar sua chave pública, se você já possuia uma chave do tipo RSA use o comando `cat ~/.ssh/id_rsa.pub`, se você gerou no passo anterior, apresente sua chave com:

```
cat ~/.ssh/id_ed25519.pub
```

Acesse normalmente seu servidor SSH, e então *com o usuário alvo no destino*, edite o arquivo de chaves públicas autorizadas, e adicione sua chave pública que você copiou no final do arquivo:

```
nano ~/.ssh/authorized_keys
```

Verifique o funcionamento, e desative o acesso por senha no arquivo de configuração `nano /etc/ssh/sshd_config`. Substituia `PasswordAuthentication yes` por `PasswordAuthentication no`. E reinicie o serviço de SSH `systemctl restart ssh*`.

Teste de Brute Force Attack

```
npm install -g mateuscalza/ssh-security
ssh-brute-force user@host
```

Como prevenir ataques

- Configurar acesso seguro sem senha
- Usar uma senha forte, gerada automaticamente
- Bloquear acessos massivos com Fail2ban
- Mudar a porta padrão
- Permitir apenas IPs específicos no arquivo: `nano /etc/ssh/sshd_config`
- Manter apenas usuários SSH necessários: `cat /etc/shadow | grep '^[^:]*:[^\*!]'`

Observações

- Estes exemplos foram aplicados usando Ubuntu 20.04, ainda que estes devam funcionar em outras distribuições Linux
- A ferramenta de testes de brute force attack é compatível com Windows, Linux e MacOS, desde que tenham NodeJS instalado