Networking, SSH

Mikhail Dozmorov

Summer 2018

What is SSH?

- "SSH is a protocol for secure remote login and other secure network services over an insecure network." – RFC 4251
- Secure channel between two computers
  - Uses strong encryption and authentication to provide confidentiality and authenticity of the data
- Many uses other than remote shell

https://en.wikipedia.org/wiki/Secure_Shell

Some implementations

- `OpenSSH` – common on UNIX systems, open implementation of the last free SSH release, now supports SSH2 protocol
- `PuTTY` – client only, Windows
- `MobaXterm` - Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more

https://www.openssh.com/

https://www.chiark.greenend.org.uk/~sgtatham/putty/

https://mobaxterm.mobatek.net/

Layering of SSH Protocols

- **Transport Layer Protocol**
  - Provides server authentication, confidentiality, and integrity
- **User Authentication Protocol**
  - Authenticates the client-side user to the server
- **Connection Protocol**
  - Multiplexes the tunnel into logical channels
- New protocols can coexist with the existing ones

Basic use

```
ssh <ssh_server_name>
ssh -l <username> <ssh_server_name>
ssh <ssh_server_name> <command_to_run>
```

User Configuration Files (OpenSSH)

- `~/.ssh/`
  - `id_*` - private authentication keys
  - `id_*.pub` – public authentication keys
  - `known_hosts` – list of known public host keys
  - `authorized_keys` – list of allowed public authentication keys

Encryption concepts

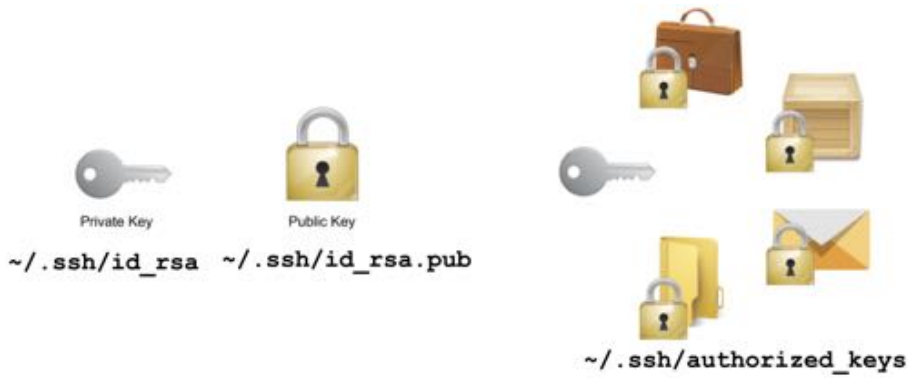
Public and private keys

- Both public and private keys are generated by one individual – they are yours
- A public key is a “lock” that can be opened with the corresponding private key
- Public key can be placed on any other computer you want to connect to
- Private key stays private on any machine you’ll be connecting from
- Only your private key can “open” your public key

Getting public and private keys

Generate your public and private keys

- First, check if you already have them, `ls -al ~/.ssh`
- If not, generate, `ssh-keygen -t rsa -b 4096 -C your_email@example.com`



Add public key to any machine

- Copy your public key `~/.ssh/id_dsa.pub` to a remote machine
- Add the content of your public key to `~/.ssh/authorized_keys` on the remote machine
- Make sure the `~/.ssh/authorized_keys` has the right permissions (`read` + `write` for user, `nothing` for group and all)

```
cat ~/.ssh/id_dsa.pub | ssh user@remote.machine.com 'mkdir
-p .ssh; cat >> .ssh/authorized_keys; chmod 600
authorized_keys'
```

http://mah.everybody.org/docs/ssh

Password-less login

- When you ssh to a remote machine that has your public key, you may skip login if your private key is visible to your terminal session. Need to start `ssh-agent`
  - Remembers your private key(s)
  - Other applications can ask `ssh-agent` to authenticate you automatically
  - Unattended remote sessions.
  - Should already be running in the background
  - `ssh-add [KeyName]`

http://mah.everybody.org/docs/ssh

https://gist.github.com/rezlam/850855

Password-less login

- Automate `ssh-agent` start by adding the auto-start function in your `~/.bashrc`

```
# Start ssh-agent
 SSH_ENV=$HOME/.ssh/environment
 function start_agent {
    echo "Initializing new SSH agent..."
    # spawn ssh-agent
    /usr/bin/ssh-agent | sed 's/^echo/#echo/' > "${SSH_ENV}"
    echo succeeded
    chmod 600 "${SSH_ENV}"
    . "${SSH_ENV}" > /dev/null
    /usr/bin/ssh-add
 }

 if [ -f "${SSH_ENV}" ]; then
    . "${SSH_ENV}" > /dev/null
    ps -ef | grep ${SSH_AGENT_PID} | grep ssh-agent$ > /dev/null || {
        start_agent;
    }
 else
    start_agent;
 fi
```

Advantages

- **Password Exposure**: SSH eliminates the risk of password exposure because. It doesn't transmit passwords in plaintext format, therefore making it impossible to "sniff" the passwords.
- **Data Eavesdropping**: SSH uses strong encryption and authentication when transmitting data. SSH guarantees that only the recipient can read the transmitted data.
- **Man-in-the-Middle Attack**: The SSH protocol applies server authentication and cryptographic integrity checks to ensure that the data cannot be modified undetected while sent through a network.

Drawbacks

- **Password Cracking**: SSH improves password security through encryption, but it's still a weak form of authentication, because it can be lost, given away, or guesses
- **IP and TCP Attacks**: SSH operates on top of TCP, therefore some of its weaknesses come from TCP/IP problems
- **Traffic Analysis**: Traffic patterns can be an important source of information for a hacker. Sudden increase or decrease in traffic can indicate important transactions or unguarded networks
- **Covert Channels**: SSH doesn't attempt to eliminate covert channels, because their analysis is usually performed by other security applications on a system
- **Carelessness**: SSH is an effective tool, but it can't take over every security aspect and its effectiveness depends on the user.

Secure copy (`scp`) files over the network

scp: securely copy a file from one computer to another

- Use `scp` to securely transfer files between two Unix computers
- Replaces `ftp`, `rcp`, file sharing
- The `scp` command uses SSH to transfer data, so it requires a password or passphrase for authentication
- `scp` encrypts both the file and any passwords exchanged
- Alternatively, use `rsync`

The syntax for the scp command is:

```
scp [options] username1@source_host:directory1/filename1 \
              username2@destination_host:directory2/filename2
```

scp examples

- Copy remote file locally (in the current folder)

```
scp mdozmorov@merlot.bis.vcu.edu:hg38.Ensembl.gtf .
```

- Copy local file to the remote home folder of the user

```
scp hg38.Ensembl.gtf mdozmorov@merlot.bis.vcu.edu:~
```

- Use `-r` (recursive) option to copy a directory
- Explore `rsync` command as an alternative to `scp`
  - It copies a tree of files from a master out to a copy on another machine
  - Can use `ssh` as its transport
  - `rsync -azv -e ssh /home/wstearns/webtree/ mirror.stearns.org/home/web/`