

# Extra Topics - SSH

OPS102 Week 13 Class 1

---

John Sellens

April 1, 2024

Seneca Polytechnic

SSH – The Connectivity Swiss Army Knife

What is Public-Key Cryptography?

SSH Basics

I Don't Want To Type My Password All The Time

SSH Configuration

SSH Summary

# SSH – The Connectivity Swiss Army Knife

---

## SSH – The Connectivity Swiss Army Knife

- An open standard for securely connecting between machines over the network
- Uses public-key cryptography
  - As does HTTPS for web sites
- Originally intended to replace "**rlogin**" and "**rcp**"
- Now also used as a transport layer under lots of protocols
  - Such as SFTP, rsync, git server access, etc.
- Can forward (proxy) ports over a connection
- And easily make multi-hop connections across multiple machines

# SSH History

- First released in 1995 by Tatu Ylönen, then at Helsinki University of Finland
- Initially open source, then proprietary versions.
- Last open source release was forked, eventually becoming OpenSSH
  - Originally as part of the OpenBSD operating system
- IETF standardized SSH-2 in 2006 in RFC 4253
- It is now everywhere
- [https://en.wikipedia.org/wiki/Secure\\_Shell](https://en.wikipedia.org/wiki/Secure_Shell)
- <https://en.wikipedia.org/wiki/OpenSSH>

# What is Public-Key Cryptography?

---

# Symmetric vs Asymmetric Encryption

- Encryption is used to encode/obscure/hide the real contents of a message.
- Symmetric encryption relies on a single key, which is used to encode and decode the message.
- Asymmetric encryption uses a matched pair of keys – encode with one, decode with the other.

# What is Public-Key Cryptography?

- Public-key cryptography uses a private key and a matching public key.
- A message encrypted with one key can only be unencrypted with the other key.
- If you know my public key, you can send a message that only I can read.
- And I can encrypt a message with my private key that you can decode with my public key, and thus know that the message came from me.



# Web of Trust, Certificates, Signatures

- Public-key cryptography can be used to “sign” data
- Keys can be used as identification – certificates
- A chain of signatures can vouch for a certificate or an identity
- If A knows (trusts) B, and B knows (trusts) C
  - The B can vouch for (sign) C’s key (certificate)
  - A trusts B, so A can trust that it’s actually C’s key
  - This is how HTTPS web certificates work
    - Your web browser has a built in list of certificate authorities (signers) that it trusts

## How Does It All Work?

- Big numbers, sophisticated arithmetic
- Key exchange, session keys, etc.
- I don't know any details
- But I trust the experts

## How Does SSH Use Public-Key Cryptography?

- Users (and machines) create public/private key pairs
- Public keys are exchanged – usually manually
  - Remember the first login to matrix? You were asked to accept matrix's host key?
- We don't typically use certificate signing chains with SSH
- A machine can allow a user in with no password based on a list of authorized public keys
- I don't know (or care much about) all the details

# SSH Basics

---

## SSH Basics

Login to another machine:

```
ssh machine.example.com
```

```
ssh user@machine.example.com
```

```
ssh -l user machine.example.com
```

Run a command on another machine – anything after the destination machine is treated as a command:

```
ssh machine.example.com ps aux
```

When you connect with `ssh` your current directory on the remote machine is that remote user's home directory (whatever that happens to be).

Many (many) options available, but this is often all you need – see all the details in the man page `ssh(1)`.

## Copying Files Between Machines – scp

- The `scp` command is very similar to the `cp` command, except either the source or destination must refer to a different machine:

```
scp myfile machine.example.com:backups/myfile.bak
```

or to copy from there to here:

```
scp -p -r machine.example.com:backups .
```

- Or, if you just want to copy to the remote home directory:

```
scp myfile user@machine.example.com:
```

- Remote relative paths are relative to the remote user's home directory.
- Common mistake: If you leave out the colon, `scp` thinks it's a local path (not remote), and behaves just like a local `cp` command.
  - Everyone ends up with a file called "`machine.example.com`" at some point.

## Copying Files Between Machines – sftp

- `sftp` is another remote file copy command, similar to `ftp`, but secure!
- Session oriented – open a connection, move files back and forth, close the connection.

```
% sftp user@machine.example.com
```

```
sftp> ls
```

```
myfile.bak
```

```
sftp> get myfile.bak backups/
```

```
Fetching /home/user/myfile.bak to backups/myfile.bak
```

```
sftp> exit
```

- Lots of internal `sftp` session commands – `mkdir`, `ls`, `cd` – see `sftp(1)`.

I Don't Want To Type My Password All  
The Time

---



# I Don't Want To Type My Password All The Time

- Normally, you will be prompted to the remote user's password every time you use SSH to connect.
- That can get boring very quickly.
- An SSH public/private key pair (and a little configuration) to the rescue!
- Put your public key in the remote user's ".ssh" directory, and you're halfway there.

## Create a Key Pair

- The `ssh-keygen` command is used to create a public/private key pair.
- By default it will create an “RSA” (type of encryption) key pair.
  - That’s the best (most widely supported) choice in most cases.
- You will be encouraged to put a secret “passphrase” on your private key.
  - Using a passphrase is the most secure choice.
  - Most user environments will make it easy to only unlock your key once per session.
- Your key pair will be stored in your "`/.ssh`" directory.

Link: [Key-based authentication in OpenSSH for Windows](#)

## Make Your Private Key Available

- If you have no passphrase on your key pair, typically nothing to do here.
- If you do have a passphrase, you don't want to type that all the time.
- Your computer makes an “SSH agent” available to you, which will store your unlocked private key during a session, for SSH to use.
- It may “do the right thing” transparently (and prompt you once for your passphrase when required).
- You might need to “encourage” it.
  - For Windows, see the link on the previous page
  - For MacOS, see link: [Adding your ssh keys to MacOS Keychain \(which seems reasonable?\)](#)
  - For Linux, it will likely “just work”

## Tell Remote Machines and Services to Trust You

- If you copy your public key to remote machines and services, they will likely allow you to connect with SSH without a password prompt.
- In most cases, copy your `"id_rsa.pub"` file to `".ssh/authorized_keys"` on the remote machine e.g.

```
ssh user@matrix.senecapolytechnic.ca mkdir -p .ssh
scp .ssh/id_rsa.pub \
    user@matrix.senecapolytechnic.ca:~/.ssh/authorized_keys
```
- For services like github, copy and paste your public key from your `"id_rsa.pub"` file into your account settings in the web interface.
  - Make sure it copies as a single (long) line.

# SSH Configuration

---

# SSH Configuration

- SSH has many options – user, network, ports, algorithms, proxying, etc.
- Your ".ssh/config" file can have global and per-destination-host settings.
- For example, you could set up a "host" called matrix which has the appropriate settings for you to easily connect to matrix e.g.

```
Host matrix matrix.senecapolytechnic.ca
    Hostname matrix.senecapolytechnic.ca
    User yoursenecausername
```

- See the man page `ssh_config(5)`.

- You can set default configuration for every host by using an asterisk (“\*”) as the hostname e.g.

```
Host *  
    ForwardAgent yes  
    KeepAlive yes  
    TCPKeepAlive yes  
    ServerAliveInterval 30  
    User jsellens
```

## Proxying Through Intermediate Hosts

- You can (obviously) SSH to host A, then SSH from there to host B, and so on
  - Often necessary when an organization has a single “gateway” or “jump” host.
- SSH makes this easy – e.g. to get to a specific maxtrix cluster host:

```
Host matrix1
```

```
ProxyJump yoursenecausername@matrix.senecapolytechnic.ca
```

```
Host mtrx-node01pd.dcm.senecapolytechnic.ca
```

```
User yoursenecausername
```

That doesn't actually work – see next page.

- Older system that doesn't have "ProxyJump"? Use e.g.

```
ProxyCommand ssh -q -W %h:%p -l user gateway.example.com
```



## Why Didn't "ProxyJump" Work to matrix?

- System administrators can turn off SSH functionality in the system's "sshd\_config" file.
- I think matrix has "AllowTcpForwarding no" set.
- So, use an old style "ProxyCommand" (assuming you have the "matrix" config from 2 slides back):

```
Host matrix1
```

```
Host mtrx-node01pd.dcm.senecapolytechnic.ca
```

```
User yoursenecausername
```

```
ProxyCommand ssh -q matrix nc %h %p
```

## Proxying Ports

- Sometimes you need to get to a remote network port, or allow a remote machine to get to a local network port on your machine.
- For example, you might want to get to a web server inside a remote network.  
`ssh -L8443:intweb.example.com:443 gateway.example.com`  
and then point your web browser to  
`https://localhost:8443/`  
and ignore SSL certificate warnings about hostname mis-match.
- Or use the "`LocalForward port remotehost:port`" setting in your "`.ssh/config`" file.
- Similarly, to forward a remote port back here, use "`-R`" or the "`RemoteForward port localhost:port`" setting.

## SSH Summary

---

## SSH Summary

- SSH has turned into a hugely powerful and ubiquitous tool.
- We have only scratched the surface of what is possible e.g.
  - "**autossh**" can automatically proxy ports bi-directionally between systems.
  - An "**.ssh/authorized\_keys**" file can include restrictions on what commands are allowed after authenticating with a key.
  - Parallel SSH ("**pssh**") and cluster SSH ("**cssh**") make it easy to use SSH to run the same commands and multiple (or many) machines.
- You can likely make your network life simpler if you're familiar with what SSH can do to help you.
- The web has, of course, all sorts of "how tos" available.