Section 6: More Wireshark, advanced SSH

CSE 461 Computer Networks

Wireshark

- https://courses.cs.washington.edu/courses/cse461/20au/section-data/461-demo.pcap
  - Open this file in wireshark
- https://courses.cs.washington.edu/courses/cse461/20au/section-data/pcap-demo.md

Wireshark Filters

- ip
  - ip.addr == <address>
- icmp
- ipv6
- icmpv6
- tcp
  - tcp.port == 80
- udp
- dns
  - dns.qry.name == website.com
- http
- tls (https)

Combine filters with “&&”, “||”, “^^”, “!”
Compare values with “==”, “<”, “>”, “matches”, “contains”, and more
https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html

(Not that) advanced SSH

```
ssh user@server -p port
```

SSH Keys

SSH Encryption

- SSH uses symmetrical encryption
- The session key is negotiated securely under asymmetrical encryption, upon each connection
- SSH “keys” (or passwords) are used for key negotiation
- We will learn more about cryptography in lecture
  - Take CSE 484 (Security) and CSE 490C (Cryptography) if you are interested
- We will focus on the more practical side of SSH

Why keys over passwords?

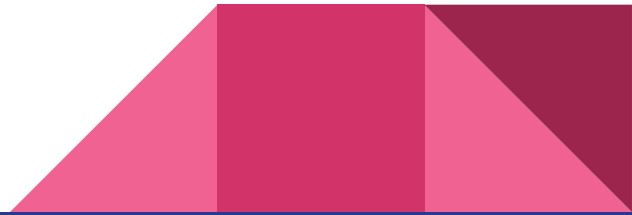
- More secure than passwords
  - Keys have completely (?) random bits
  - Passwords are vulnerable to dictionary attacks
- Easier to manage
  - Keys are kept locally and supplied automatically when you need them
  - Remembering passwords can be a pain
  - Keys can be revoked easily

Generating an SSH key pair

- To generate a key pair (RSA, by default): `ssh-keygen [-t type]`
  - We recommend using Ed25519 over RSA: `ssh-keygen -t ed25519`
  - Ed25519 is faster and more secure, but a lot of people are still using RSA
  - You probably have these already if you have used the CSE Gitlab
- By default, generates keys under ~/.ssh/
  - Public key: id_{rsa|ed25519|...}.pub
  - Private key: id_{rsa|ed25519|...}
  - Keep your private keys private
- Optional passphrase to protect your private keys
  - Additional passphrase-based encryption, so adversaries can't get your private keys even if your machine is compromised
  - Can be skipped by not typing in a password and pressing Enter

Authenticating with your SSH key

- Before you can use your keys, you need to install them on the server
  - i.e. Add your public key as a single line to ~/.ssh/authorized_keys on the server
    - <protocol> <public key text> <annotation>
      - ssh-ed25519 <text from ssh-ed25519.pub> starikov@desktop
  - You can edit the file manually by logging in with your password
  - Or use `ssh-copy-id [-i path/to/private/key] someserver` (on macOS and Linux)
- Use `-i path/to/private/key` to specify a key when SSHing
  - Your id_{rsa|ed25519|dsa|...} key under ~/.ssh/ is used by default
  - Or use the `IdentityFile` option in SSH config
- When you log in, the server looks up your public key in authorized_keys and lets you in if there is a match



Server Verification (Known hosts)

- The client stores the key of every server it knows under ~/.ssh/known_hosts
- SSH stops you from connecting to a server if the server's key doesn't match the one in known_hosts
  - This is to prevent someone from impersonating the server you have previously used
    - Will occur if you install a new OS at the same IP address
    - Or if the ssh server keys are changed
  - If you trust the new server identity, simply delete its key from known_hosts
    - Can be done by deleting the appropriate line manually
    - ssh-keygen -R "hostname"

ssh-agent

- Like a password manager for SSH keys
- `eval `ssh-agent``
  - Starts ssh agent
  - To automatically start, place this in .bashrc:

```
if [ -z "$SSH_AUTH_SOCK" ] ; then
    eval `ssh-agent -s`
    ssh-add
fi
```

- `ssh-add [path/to/private/key]` to add key to ssh-agent
  - By default adds your id_{rsa|ed25519|dsa|...}
- The passphrase is remembered for the entire session
- The ssh agent can be forwarded over SSH
  - ssh -A
  - SSH config file:
    - ForwardAgent yes
    - AddKeysToAgent yes

SSH Config File

SSH Config File

- Per user config at ~/.ssh/config (create if doesn’t exist)
- Allows you to define hosts aliases with configurations

```
Host attu attu? recycle bicycle tricycle
  Hostname %h.cs.washington.edu
  Port 22
  User starikov
  IdentityFile ~/.ssh/id_ed25519
```

Simple host configs

```
Host attu
  Hostname attu.cs.washington.edu
  Port 22
  User starikov
  IdentityFile ~/.ssh/id_ed25519
```

```
Host mininet
  Hostname localhost
  Port 2222
  User mininet
```

With the config above, I can just run `ssh attu` to connect to attu.

Equivalent to

```
ssh starikov@attu.cs.washington.edu -p 22 -i ~/.ssh/id_ed25519
```

`Hostname` also works with IP addresses

A slightly more complicated config

```
Host attu attu? recycle bicycle tricycle
  Hostname %h.cs.washington.edu
  Port 22
  User starikov
  IdentityFile ~/.ssh/id_ed25519
```

This config defines many hosts at the same time, including a wildcard (`attu?`). Note that `%h` will be replaced by the actual value of “Host.”

With this config, I can do `ssh attu8` to connect to attu8.cs.washington.edu.

SSH Port Forwarding/Tunneling

Local Forwarding (-L)

- Opens a local port that forwards to a remote port
- Syntax: `-L port:host:hostport`
- Use case
  - I have a service running on the server but it's bound to localhost only on the remote server
    - `ssh -L 8888:localhost:8888 server`
  - Service is on a private network that the server can reach, but my local computer cannot
    - I can ssh into the server and connect to a service running on privateServer
    - `ssh -L 8888:privateServer:8888 server`
- SSH Config:
  - LocalForward 8888 privateServer:8888
- VSCode's Remote SSH extension provides this feature
  - Ctrl+Shift+P and search for "Forward a Port"

Remote Forwarding (-R)

- Opens a port on remote that forwards to a local port
- Syntax: `-R port:host:hostport`
- Requires "`GatewayPorts yes`" to be enabled on SSH server (sshd_config)
- Use case
  - Access desktop ssh (localhost:22) from publicserver.com:2222
    - `ssh -R 2222:localhost:22 publicserver.com`
  - Access local mininet VM from publicserver.com:2222
    - `ssh -R 2222:192.168.56.101:22 publicserver.com`
      - Port Forwarded Mininet: `ssh -R 2222:localhost:2222 publicserver.com`
- SSH Config:
  - RemoteForward 2222 192.168.56.101:22

Dynamic Forwarding (`-D`)

- Uses SSH as a SOCKS proxy
- Syntax: `-D port`
- Use case
  - Use as a proxy server for accessing hosts from the SSH server's connection
    - Can be used to access multiple hosts that are on an internal network
    - Can also be used to access websites from the IP address of the SSH server
      - Libraries allow access without a paywall/login when using a UW IP address
      - Firefox allows you to connect to a SOCKS proxy
  - `ssh -D 1080 attu`
    - Sets up a SOCKS proxy on localhost:1080 that proxies connections through attu
  - SSH Config:
    - DynamicForward localhost:1080

SSH Jump Host

Jump Host (-J)

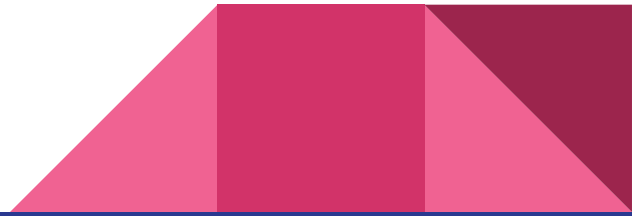
- Jump through intermediate hosts to the final SSH destination
- Syntax: `-J jumphost`
- Use case
  - You want to connect to a host over SSH behind a LAN externally, but only have SSH access to another server in that network
  - `ssh -J attu1 attu2`
    - Equivalent to:
      1. ssh -L 2200:attu2:22 attu1
      2. ssh -p 2200 localhost
  - `ssh -J attu1,attu2,attu3,attu4 attu5`
    - Jumps from attu1 to attu2 to attu3 to attu4 and finally attu5.

SSH Config for Jump Host Proxy

```
### First jumphost. Directly reachable
Host alphajump
  HostName jumphost1.example.org

### Second jumphost. Only reachable via jumphost1.example.org
Host betajump
  HostName jumphost2.example.org
  ProxyJump alphajump

### Host only reachable via alphajump and betajump
Host behindalphabeta
  HostName behindalphabeta.example.org
  ProxyJump betajump
```



X11 Forwarding

X11 Forwarding (`-X`)

- Lets you run GUI apps over SSH
- Syntax: `-X`
- Needs "`X11Forwarding yes`" enabled on server (sshd_config)
- You might need to install an "X server" on the client if you are on Windows or macOS
  - XQuartz for macOS (and add `XAuthLocation /usr/X11/bin/xauth` to your SSH config)
  - Xming or vcxsrv for Windows
- `ssh -X attu`
- SSH Config:
  - `ForwardX11 yes`

Other useful SSH tricks

- VS Code Remote SSH
  - A lot of you have been using it
  - Super useful for debugging code on remote machine
- tmux
  - Keep sessions running even if you disconnect
    - `tmux attach` will reopen a running tmux session
  - Split the terminal into smaller panels and create multiple windows
  - Very configurable: customizable hotkeys, mouse mode, and more!
- See `man ssh` or `tldr ssh` to learn more about advanced SSH features!