ADVANCED SSH: IT’S NOT JUST FOR LINUX ANYMORE

Mac, Windows/WSL, Docker, GitHub, VPN Lite, Mobile, but still mostly Linux & Unix

SCOTT CORZINE

- "Guiding your team in building a Cloud/Enterprise platform that's secure, stable, and scalable for you to host your business applications or SaaS products"
- I've worked in most of the infrastructure fields: Cloud, Database, Storage, Systems, Networking, and always with a strong emphasis on Security.
- I grew up on hard-copy terminals on time-sharing systems and followed a minicomputer path.
- Both as a consultant and in-house staff, I've provided architecture and operations experience across many industries: SaaS Provider, Backbone Provider, ISP, VAR, Finance, Global Manufacturing, Healthcare, Education, State & Local Agencies, Retail, Non-Profits, etc.
- scott@corzine.com
https://www.linkedin.com/in/srcorzine/
https://github.com/bear-ice/presentations-advanced-ssh

SSH BASICS

- SSH has two basic functions:
  - Connecting to a remote system as a terminal session
  - Executing a remote command (non-terminal)
- OpenSSH (ssh) is the main SSH client for Linux, Mac, Unix, and Windows Subsystem for Linux (WSL)
- SSH **does not** provide terminal emulation
  - This is done only by the original workstation, later chains of ssh just pass terminal control back to the originator
  - For Mac this emulation is natively provided by Terminal.app, but add-ons like iTerm are popular
  - Linux GUI emulators like xterm, GNOME terminal, and Konsole are in most distributions and many others exist
  - Windows has the biggest issue, the default command window doesn't provide real terminal emulation
  - Programs like PuTTY act as both terminal emulator and SSH client—once on the remote system they are the same as the rest
  - OpenSSH can be installed for command line on Windows via methods like CygWin, Git for Windows, package managers

SSH BASICS—REMOTE COMMANDS

- If ssh is invoked with just a remote hostname (or user@hostname) it treats the connection as an interactive terminal session
  - This enables certain unseen functions of the protocol and informs the remote to setup accordingly
- If ssh is invoked with additional arguments, it tells the remote to execute those arguments as a command and to exit afterwards
- ssh will set its exit/status code to match the remote commands, this makes it easy to do Linux/Unix shell scripting as normal, almost as if everything was local
- In command mode the special terminal support will not be enabled (unless “ssh –t” is included)—this means that only Linux/Unix stdin, stdout, and stderr will be passed between the caller of the ssh command and the remote command

SSH BASICS—CLIENT SERVER PROTOCOL

- Like many protocols, SSH/ssh is the name of both the protocol and the most common client
- SSH is a client-server TCP protocol between the local client (“ssh”, PuTTY, etc.) and a remote server program, normally named “sshd” on a remote Linux/Unix system
  - Note that this doesn’t restrict using SSH to connect to some “Client” systems like Mac, Linux, or WSL laptops or even Docker containers **if incoming SSH connections are enabled (“remote login” for Mac)**
- The protocol is built around the traditional Unix (& Linux) terminal/command model
  - This mostly matters for the server, other server systems (i.e. non-WSL Windows) may need to adapt

SSH BASICS—CLIENT SERVER PROTOCOL (CONT)

- The protocol has advanced features like multiple channels permitting several independent streams of data within the same SSH session (we'll use this with port forwarding)
- SSH **is unrelated** to other protocols like TLS/SSL, RDP, IPSec, X.509/PKI, S/MIME, PGP/GnuPG
  - (it is possible to use X.509 or PGP keys with SSH but this is *very* unusual)
- This means that when vulnerabilities are found in the others SSH is generally unaffected
  - As we'll see, this can allow them to be **carefully** used together for extra protection such as tunnelling RDP in SSH
- The SSH protocols are open, well-established & proven, and under the control of the Internet Engineering Task Force (IETF)
- SSH is generally accepted as the most secure protocol of its type and has been extensively tested
  - As far as I know there has not been a quantum-cryptology resistant version yet—but most SSH traffic has short lifespan

SSH ESSENTIALS—PASSWORD AUTHENTICATION

- Basic SSH authentication is left to the remote OS and therefore usually falls back to OS passwords
  - These passwords are transmitted within SSH's encryption and are protected to that degree
  - The remote system receives the password unmodified—**with a Man-in-the-Middle (MitM) attack this is serious**
  - All the well-known problems with passwords exist, **especially weak/reused passwords**
  - Having to repeatedly enter these for every command makes many uses of remote commands impractical
- "sshd" can be configured to allow several related risks like direct root login, permitting login without passwords, permitting use of passwords, etc.
  - These are historical
  - Hopefully, most distributions have disabled these except maybe password authentication
  - These are normally configured in the file "/etc/ssh/sshd.conf" (sometimes "/etc/sshd.conf")
- These serious issues should not be accepted in **any** modern system, especially Internet facing ones

SSH ESSENTIALS—USER PUBLIC/PRIVATE KEYS

- The solution to many of these problems lies in User Public/Private Keys (as opposed to Host Public/Private Keys)
    - Most of the rest will be covered with SSH Agent & SSH Agent Forwarding which follows this
- These use standard Public/Private Key algorithms like most cryptographic protocols
- They normally use a SSH specific type of key which isn't compatible with TLS/SSL, X.509, PGP/GnuPG, or S/MIME
- SSH keys normally don't have certificates or any type of PKI—they are considered for "internal use" within an organization and are simpler & more lightweight (enhancing security)
- Special, unusual uses do have SSH certificates/authorities or allow use of X.509 or PGP keys

SSH ESSENTIALS (QUICK)

- SSH Agent/Pagent — only typing that passphrase once (SSO like)
- Agent Forwarding and $SSH_AUTH_SOCK — keys on only your workstations
- Almost never create a private key without a strong passphrase
- Disabling password login
- Now you have a much stronger system/network
- Host key management & Man-in-the-Middle summary

SSH BUILDING BLOCKS (QUICK)

- Tunnelling — One building block in the wall
- SSH Channels
- TCP Port Forwarding/Tunnelling
- VPN-Lite with tunnelling
- Protecting otherwise open ports / risking uncertain encryption
- Potentially have just 1 open port with strong security
- Things to avoid aka Don't make life difficult for Security, Networking, or Production

SSH INTERMEDIATES (QUICK)

- Non-SSH commands with built-in use of SSH behind the scenes
- Rsync
- Git & Github
  - Git SSH signing
- SFTP
  - Winscp
  - SFTPD Sites

JUMP BOXES/BASTION HOSTS

- Provides a chokepoint to control, secure, and administer policy to chunks of your network
- Cutting off a terminated employee should cut off access into the protected network
- Only needed for incoming traffic and shouldn't be critical for production
- If you can make them a disposable image then it can be cloned, brought up & down, and replaced & the old taken out of service if suspect or failed
- See Key/Account distribution

WINDOWS—ADDING PROTECTION TO RDP/PORT 3389

- Other tunnelling (unprotected dev ports, places, TBD)
- Use for controlling

MACS

- you have it just need to turn it on/enable it
- You can tunnel VNC through it providing a free, secure, rough remote desktop

DOCKER, KUBERNETES, & CONTAINERS

- Installing in each container vs the hosts
- Protecting the docker/kubectl port

MORE

- SSH with MFA devices—PAM
- Ways to weaken your keys — weak storage
- Storing your SSH keys on Yubikeys or similar devices (maybe)
- Storing your SSH keys in Keychain, Vault, BitWarden, or other places\
- Host Key distribution\
- User Public Key/Account distribution
- SSH Certificates & Certificate Authorities
- AWS's hidden backdoor (System Manager)

EVEN MORE

- DNS SSH Keys

HOLDER

- Mobile devices — Should you trust your critical resources to someone offering a free ride?