Secure Shell

Linux System Administration
Spring 2026

Remote System Access

- The goal
  - Execute commands on a remote system
    - Remote shell or single command
  - Transfer files between systems
- Requirements
  - Invisible - terminal window just like local
  - Secure - the network cannot be trusted

In the beginning

- telnet (**tel**etype **net**work)
  - Original TCP/IP based remote terminal (RFC 15)
    - Plain text with lightweight escape sequences (0xff=IAC)
    - Port 23
  - Unsafe for untrusted networks
    - Still useful in testing TCP/IP connections
    - Sometimes enabled in legacy hardware
- Replaced by **s**ecure **sh**ell (ssh)

BSD r-commands

- Added in BSD 4.1 (1981)
  - Often remapped to ssh, scp, etc on many systems

Client	Function	Daemon	Port	Protocol
rexec	command	rexecd	512	TCP
rlogin	shell	rlogind	513	TCP
rsh	command	rshd	514	TCP
rcp	copy			
rstat	stat	rstatd		UDP
ruptime	uptime	rwhod	513	UDP
rwho	who			

Secure Shell (SSH)

- Designed by Tatu Ylönen 1995
  - OpenSSH - Implementation by OpenBSD (1999)
  - dropbear - Tiny footprint ssh for embedded devices
- Secure login to remote system
  - Public-key encryption over TCP port 22
  - Basis for scp, sftp, etc
  - Supports tunneling (X11, port forwarding)
  - Allows password-less logins

The importance of ssh

- Goal of exams are in large part establishing an ssh connection to a remote server
  - Basic machine functions are up
- Tunneling allow access to network services as if they are local

Using ssh

- ssh user@host

  -p port (default 22)

  -v verbose (-vv or -vvv for even more verbose)

  -L local:host:remote (tunnel)

  -R remote:host:local (reverse tunnel)

  -X Forward X11 (-Y for secure)

  -J Jump host

ssh examples

```
ssh www.dm.com
ssh -p 2222 root@broken.dm.com
ssh -v root@unreachable.dm.com

ssh -J root@100.64.0.43 root@10.21.32.2

ssh -L 2222:10.21.32.2:22 machinea
ssh -p 2222 localhost
scp -Cp -P 2222 localhost:foo .
```

ssh files

- /etc/ssh/ssh_config
  - Global configuration
    - Can be by host or *
  - Override options with ssh -o
- /etc/ssh/sshd_config
  - Incoming connections

- ~/.ssh/
  - `known_hosts`
    - signature
  - `authorized_keys`
    - public key(s)
  - `id_rsa`, `id_ed25519`, ...
    - private key

Important *sshd_config* options

- PermitRootLogin
  - yes = allowed
  - prohibit-password = must use key pair
  - forced-commands-only = key pair only, no shell
  - no = never
- UseDNS (should be yes)
- UsePAM (should be yes)

Copying files with scp

- scp *localfiles host:remotedir*
  scp *host:remotefiles localdir*
  - -p preserve file times and modes
  - -r recursive
  - -P Port (default 22)
  - -C Compress

rsync

- Fast tool for copying files
  - Modern versions transparently run over ssh
  - Conditionally copies files based on time and size
  - Clever block checksum for updating files
  - Can preserve permissions, links and holey files
- Best tool to mirror directory trees

rsync options

-a archive

- -p preserve permissions
- -t preserve times
- -r recursive
- -l preserve symbolic links
- -o preserve owner
- -g preserve group

-H preserve hard links

-A preserve ACLs

-S write holey files

-n dry run

-v verbose

-z compress

--delete-excluded

--existing (no new files)

--ignore-existing (only new files)

--exclude=*pattern*

--bwlimit=

rsync tips

- Watch out for a trailing / on the source
- Use -n to do a dry run before big transfers
  - Especially with --delete-excluded
- *rsync* can be used locally as a clever *cp*

Copy trees with tar

`ssh user@host tar cf - -C dir|tar xf -`

- **ssh** to *host* and run **tar**
  - **-C** does a chdir to *dir*
  - **-cf -** creates and archive to stdout (-)
- **tar xf -**
  - unpack archive from stdin (-)
- archive is piped over ssh

Generating keys

- Generate key pair
  - ssh-keygen
    - -t key type (rsa, rsa-sha2-512, dsa, ecdsa. ed25519, etc.)
    - -b key length (e.g. 4096)
    - -C comment
  - Default output to ~/.ssh/
  - Empty passphrase is OK
- `ssh-keygen -t rsa -b 4096 -C saclass@colorado.edu`
  - Private key `~/.ssh/id_rsa`
  - Public key `~/.ssh/id_rsa.pub`

Distributing keys

- By hand

```
scp ~/.ssh/id_rsa.pub remotehost:
ssh remotehost
cat id_rsa.pub >> ~/.ssh/authorized_keys
rm id_rsa.pub
```

- ssh-copy-id *remotehost*

Using ssh in scripts

- `ssh` will only accept a password interactively
  - Requires password-less authentication
- `sshpass` allows `ssh` to be run non-interactively but with a password
  - `read -s -p "Enter password: " PW`
    `...`
    `sshpass -p $PW ssh` *`host command`*

Notes about key pairs

- The local key must be in `$HOME/.ssh/`
  - What is $HOME when you do sudo?
- The remote key must be in `$HOME/.ssh/` for the remote user you are login in as

Pause for Paranoia

- Password-less logins extends the domain of trust to machines with the private key
  - Allows tunnels through firewalls
  - DMZ hosts should never have private keys for the LAN
    - Set up access from the LAN side
  - root password-less access should be minimized
  - Agonize over laptops

mosh

- mosh is an alternative to ssh that works better over unreliable connections
  - ssh takes a long time to recover from lost packets
  - mosh uses UDP
  - mosh tolerates changing the source IP address
- Good choice for mobile (cellular)
  - Example implementation: blink for iOS