SSH Boot Camp

JJ Quinlivan

October 3, 2019

SSH Boot Camp Agenda

- Security Warning
- Encryption 101
- SSH Basics
  - SSH Encryption
  - SSH Keys
- SSH Configuration
- SSH Fun!
  - SSH tunnels, X11 forwarding and SOCKS proxy

Not Covered Today

- PuTTY
- MOSH
- SSHFS
- Bastion Hosts
- Enterprise SSH Key Management/Certificates

About Me

- Computer Consultant for 25 years
- Disabled 12 years ago
- Linux Enthusiast
- Professional Noob!

Security Warnings

- SSH is a tool
- Tools can be used for good or evil
- SSH can help you save your company
- SSH can help you destroy your company
- JJ is not responsible for reasonable, unreasonable, or ridiculous damages caused by your use/abuse of SSH

SSH Basics

- SSH = Secure Shell
- Standard network protocol and service (TCP port 22)
- Most popular implementation: OpenSSH
  - Developed by OpenBSD
  - Portable OpenSSH for other operating systems
  - Now works in Windows with Linux Service Layer
- Relies on cryptography/encryption

SSH Usage

- Unix command `ssh`; server-side: `sshd`
- Establishes a secure communication channel between two machines
- Gets a shell (terminal) on a remote machine
- Advanced Usages:
  - Data transfer (`scp`, `sftp`, `rsync`)
  - Connect to specific services (i.e. Git)
  - Dig secure tunnels through the public internet

SSH Usage

- `ssh user@remote-hostname` - shell on remote server
  - `-p port` - to change port number
  - `ssh user@remote-hostname ls /etc` - to execute remote command
- Copying data
  - `scp local-file user@remote-hostname:remote-dir/` - scp (can reverse)
  - `sftp user@remote-hostname` - sftp (can use standard ftp commands)
- SSH works with many programs
  - rsync - `rsync -avzP localdir user@remote-hostname:remote-dir/`
  - Git - `git clone ssh://user@github.com/user/repository.git`
  - ZFS replication

Encryption 101

Encryption is transforming plain text to cipher text and back again using an algorithm and key

Encryption Algorithms

- Symmetric
  - same method & key used to encrypt and decrypt
  - Fast
- Asymmetric
  - different methods to encrypt or decrypt
  - one key for encryptions
  - different key for decryption
  - slow
  - many algorithms - RSA, Blowfish, etc

How SSH Uses Encryption

- Public key for initial session setup
  - Both user and server keys
- Agree on temporary symmetric session setup
- symmetric for most of session
- occasional rekeys

SSH Keys

- Use Public/Private SSH key authentication instead of passwords
- Create RSA 4096 key pair with `ssh-keygen -lb 4096`
- Create ed25519 key pair with `ssh-keygen -lt ed25519`
- Copy public key to server with `ssh-copy-id user@remote-hostname`
- Do not share private keys between devices, create new one for each
- Create new keys every couple years
- May want to create separate keys for different purposes
  - Different clients or (home/work)
  - Different key for Git

SSH Keys

- Always use a passphrase!
- Can use `ssh-agent` in each terminal session so only have to enter passphrase once
  - Add keys manually with `ssh-add key-filename`
- You can add `ssh-agent` to your `~/.profile` for whole login session
  - https://stackoverflow.com/questions/18880024/start-ssh-agent-on-login

SSH Configuration

- Configration files in `/etc/ssh`
  - `ssh_config` - host-wide client configuration
  - `ssh_host_*_key.pub` - public keys
  - `ssh_host_*_key` - private keys
  - `sshd_config` - server configuration

Testing SSH Server

- Copy `/etc/sshd_config` to another file (i.e. `sshd_config_test`)
- `sshd -f sshd_config_test -p 2222`
  - test alternative configuation on odd port
- `sshd -f sshd_config_test -p 2222 -ddd`
  - run in foreground
  - one connection only
  - useful for weird error debugging

SSH User Configuration

- `~/.ssh` - Local user SSH directory (`chmod 0600`)
  - `config` - user's individual configuration
    - Documented in `man ssh_config(5)`
    - Use alternate with `-f filename`
    - All configuration options work same
  - `known_hosts` - all host (server) public keys
  - `id_rsa` - default private key name
  - `id_rsa.pub` - default public key name
- Test client with `ssh -vvv -f filename user@remote-hostname`

SSH User Configuration Example (~/.ssh/config)

```
Host mail # alias/shortcut - ssh foo
    Hostname mail.acme.com`
    User jjquin

Host *.acme.com
    User jdoe
    Compression yes # default is no
    ServerAliveInternal 60 # keep-alives for bad firewall

Host *
    User john
    PasswordAuthentication no
```

SSH Secure Server Configuration

- Hail Mary Cloud
- Restrict Root: `PermitRootLogin no` - use sudo
- SSH Keys Only: `PasswordAuthentication no` - wait till key copied to server
- Change Port?
  - Security through Obscurity
  - Can reduce attacks/logs
- Fail2Ban or SSHGuard if you cannot restrict password login
- Verify Host/Server Key
  - DO NOT CONNECT IF HOST KEY HAS CHANGED UNTIL YOU KNOW WHY!!

SSH Server Configuration

- `UseDNS no` - prevent connection failure when DNS down
- `AllowGroups wheel` (or `sudo`) - limit who can connect
  - Restrictions processed in order list in config file
  - first-match basis
  - `{Deny,Allow}Users` - user list
  - `{Deny,Allow}Groups` - group list
  - Deny users in group then allow group
    - `DenyUsers jgballard`
    - `AllowGroup billing`
  - Can allow users per connection
    - `AllowUsers jprice@192.0.0.1/32`
- Conditional configuration - match by user, group, network etc.
  - Example, X11 Forwarding

```
Match User mwsmith
X11 Forwarding Yes
```

SSH Fun!

- SSH Local Port Forwarding
- SSH Remote Port Forwarding
- SSH X11 Forwarding
- SSH SOCKS Proxy
- Tmux

SSH Port Forwarding

- Goal: transport traffic through a secure connection
  - Work-around network filtering (firewalls)
  - Avoid sending unencrypted data on the Internet
  - But only works for TCP connections

SSH Local Port Forwarding

- 'ssh -L 12345:service:1234 user@remote-hostname'
  - `-L` - access a remote service behind a firewall
  - `telnet localhost 12345` - On client (not in SSH session)
  - Remote-hostname establishes a TCP connection to Service, port 1234
  - The traffic is tunneled inside the SSH connection to Server

SSH Local Port Forwarding

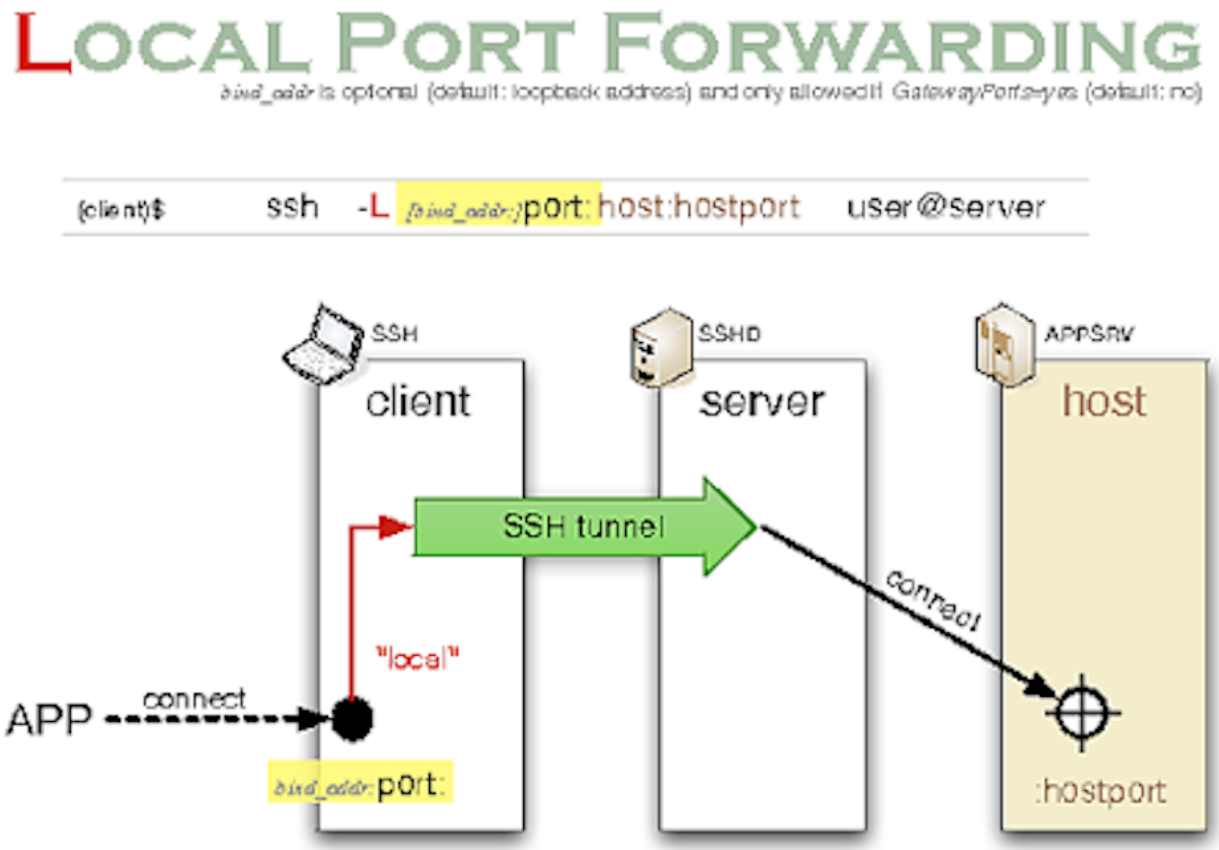


Figure 1: Local Port Forwarding

SSH Remote Port Forwarding

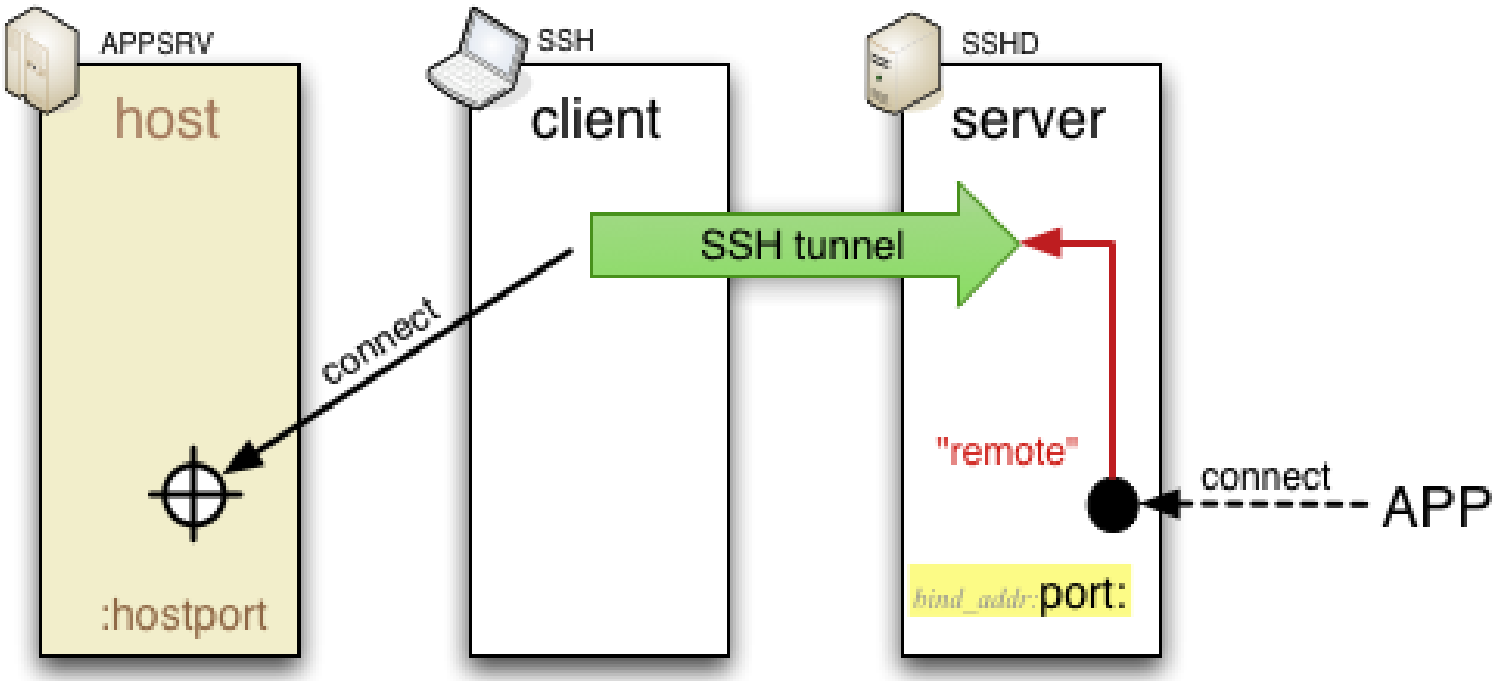
- `ssh -R 12345:service:1234 user@remote-hostname`
  - `-R` - provides remote access to local private service
  - `telnet localhost 12345` - On Server
  - Client establishes a TCP connection to Service, port 1234
  - The traffic is tunneled inside the SSH connection to Client

SSH Remote Port Forwarding

Remote Port Forwarding

bind_addr is optional and defaults to * (all interfaces)

```
(client)$  ssh  -R [bind_addr:]port:host:hostport  user@server
```



SSH X11 Forwarding

- `ssh -X user@remote-hostname`
  - Run a grapical application on a remote machine, display locally
  - Similar to VNC, but on a per-application basis
  - Then start GUI applications on server (i.e. kcalc)

SSH X11 Forwarding

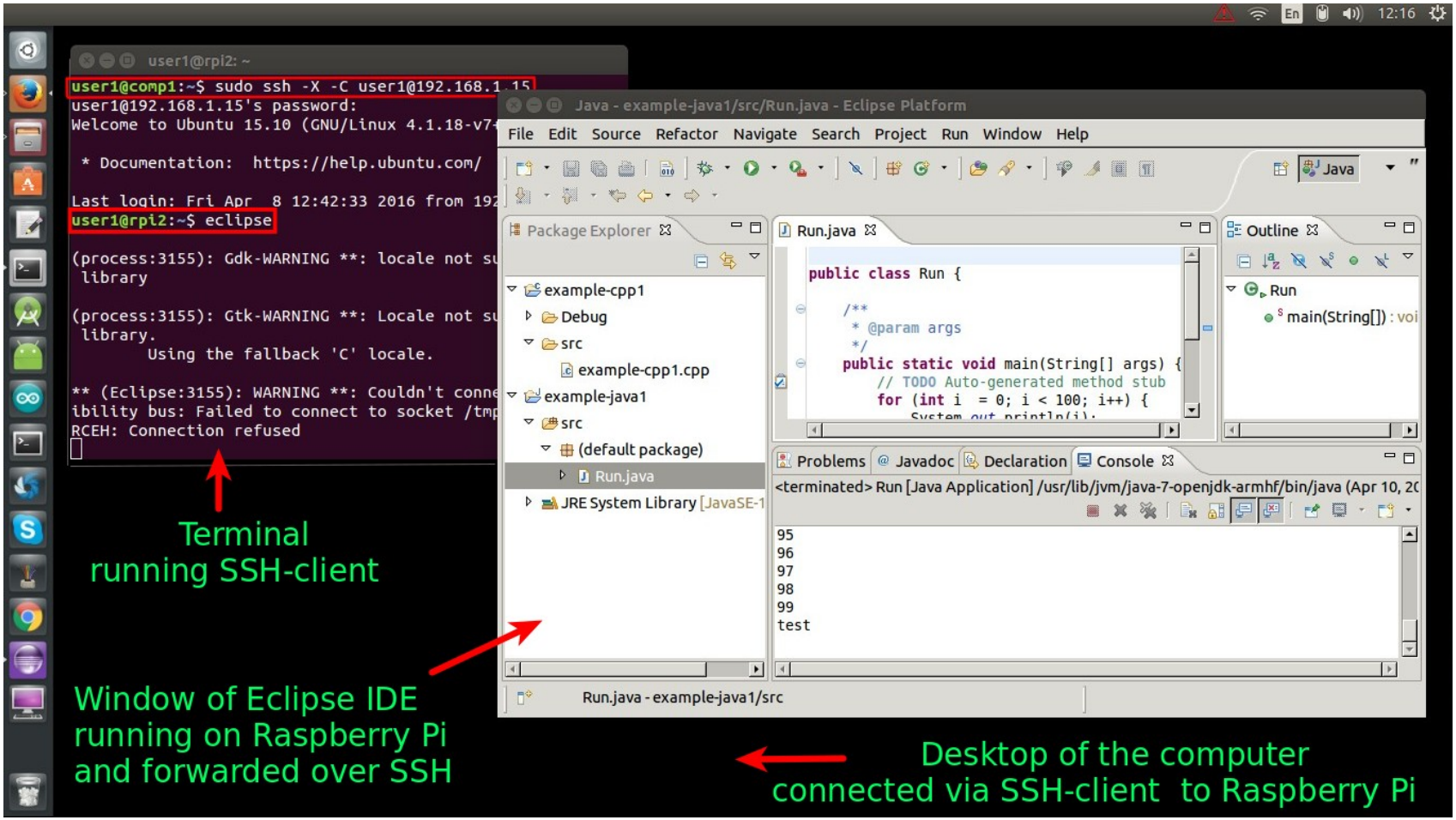


Figure 3: X11 Forwarding

SSH Socks Proxy

- `sh -D 1337 -q -C -N user@remote-hostname` does the following:
  - `-D 1337`: open a SOCKS proxy on local port :1337
  - `-C`: compress data in the tunnel, save bandwidth
  - `-q`: quiet mode, don't output anything locally
  - `-N`: do not execute remote commands, useful for just forwarding ports
  - SOCKS - protocol to proxy TCP connections via a remote machine
  - SSH can act as a SOCKS server
  - Similar to `-L` but more flexible, multiple connections
  - Configure applications to use SOCKS proxy

Firefox Socks Proxy Setup

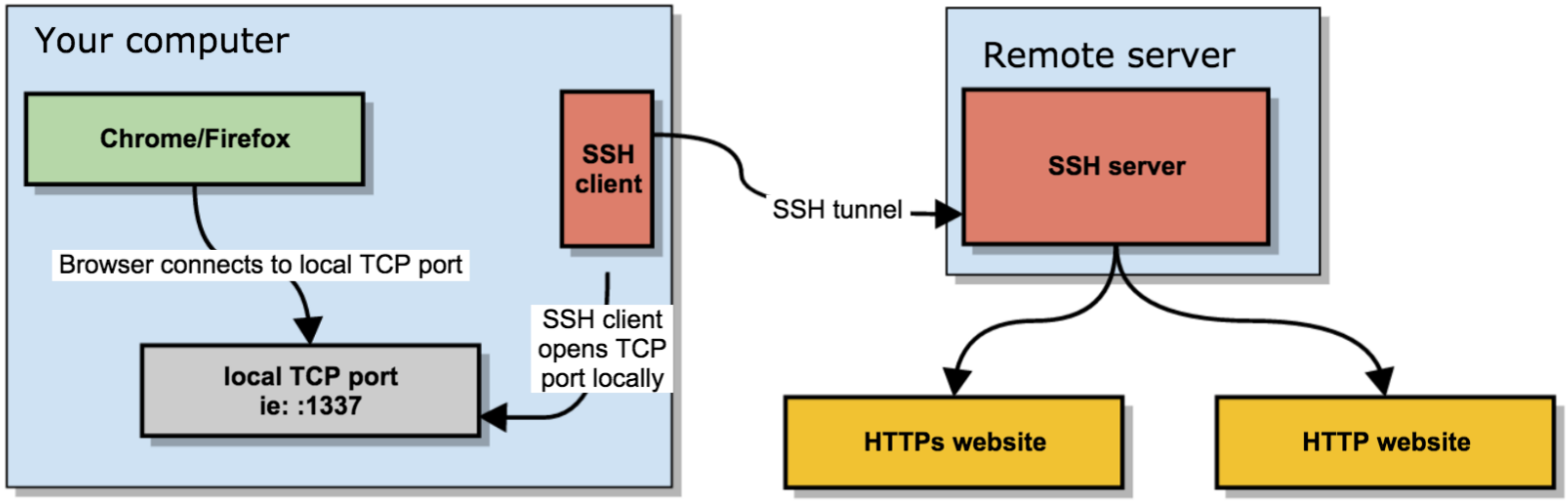


Figure 4: Firefox Socks Proxy Setup

Tmux

- Terminal Multiplexer
  - Used to keep SSH session active on server when client disconnects
  - `tmux new-session` - create new session
  - `Ctrl-b D` - to disconnect
  - `tmux attach` - to reattach to session