Managing SSH Servers and Connections

Andrew Mallett
Linux Author and Trainer

@theurbanpenguin www.theurbanpenguin.com

Module Overview

- Determine security on a given SSH server
- Using key-based authentication
- Strengthen SSH security
- Centralizing known hosts
- Using and SSH certificate authority

SSH Security

Passwords can be guessed

The root use is a "known" account name

Using key-based authentication can replace passwords

```
$ ssh -o PreferredAuthentications=none 192.168.33.11
```

Remotely Checking Authentication Methods

Password based authentication may be enabled on some systems and on others not. This is just the build of the vagrant boxes. Vagrant, itself, uses key based authentication. But it shows how easily authentication weaknesses can be displayed.

Demo

Checking defaults on 3 systems

- Read the defaults remotely
- Create standard user on each system

Known Hosts

Each time a client connects to a server the public key presented by the server is compared to the stored version in ~/.ssh/known_hosts . If it is not present the client will be asked to accept the key. To reduce the need for each user to accept the remote server key, storage can be centralized in /etc/ssh/ssh_known_hosts.

Known Hosts and Transient VMs

If you use a lot of VMs the public-key you have stored may not match new systems using the same IP. For machines on a host-network protected from outside connections consider: -oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null

Demo

Investigating Known Hosts

- Using CLI options
- Storing options in .ssh/config

Demo

Centralizing Known Hosts

- Centralizing ssh_known_hosts
- ssh-keyscan

```
$ ssh-keygen
$ ssh-copy-id 192.168.33.11
$ ssh 192.168.33.11
```

Remove Passwords from the Equation

Using key based authentication, we will not need to add passwords. Becoming more convenient as well as more secure. Client public keys are stored in the target users authorized_keys file

Demo

Client Authentication

- ssh-keygen
- ssh-copy-id
- ssh-agent

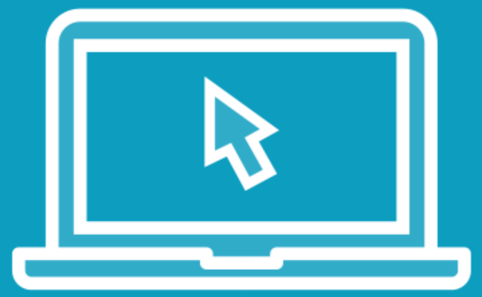
```
# cd
# ssh-keygen -f server_ca
# scp vagrant@192.168.33.11:/etc/ssh/ssh_host_rsa_key.pub alma.pub
# ssh-keygen -s server_ca -I alma -h -n 192.168.33.11 -V +52w alma.pub
# scp alma-cert.pub 192.168.33.11:/tmp/ssh_host_rsa_key-cert.pub
# cat server_ca.pub >> /etc/ssh/ssh_known_hosts
# vim /etc/ssh/ssh_known_hosts
append @cert-authority 192.168.33.* <before ssh-rsa>

# mv /tmp/ssh_host_rsa_key-cert.pub /etc/ssh
# echo "HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub" >> /etc/ssh/sshd_config
# systemctl restart sshd
```

SSH CA

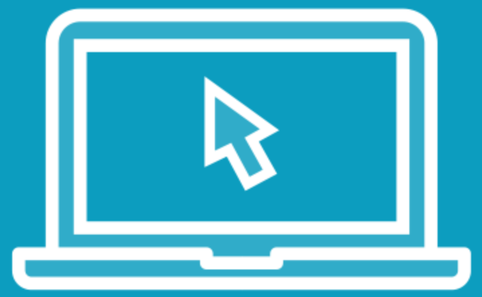
Creating a Certificate Authority on the Ubuntu system allows us to sign certificates for all host meaning clients only need to trust the CA

Demo



Configuring the CA

Demo



Signing server public keys

Summary

Congratulations, you can now secure SSH access

- ~/.ssh/known_hosts / ssh_known_hosts
- ~/.ssh/config
  - StrictHostKeyChecking
  - UserKnownHostsFile
- ssh-keygen
- ssh-copy-id
- authorized_keys
- ssh-agent / ssh-add
- ssh-keygen -s

Elevating Privileges in Linux