PKIX SSH Setup Instructions

Installing the PKIX SSH server/client and provisioning a single user 

The tutorial at <http://tech.ciges.net/blog/openssh-with-x509-certificates-how-to/> doesn't detail all the steps needed for configuring the PKIX server. In particular, differences in the install location for the package (and probably the version of PKIX being installed) requires modifications to the steps provided. The steps below include installation and configuration of PKIX SSH itself – so there should be less room for error.

**Setup VMs**

I setup 2 "linked clones" of a U18.04 Server Virtual Box image I already had on-hand - ensuring the new VM images had randomized MAC addresses. The hostnames were set to "pkix-ssh-server" and "pkix-ssh-client" respectively (editing /etc/hosts and /etc/hostname), setting the server and client IP addresses to 10.20.0.90 and 10.20.0.91, respectively (editing /etc/netplan/50-cloud-init.yaml), and rebooting.

Note: System configuration can obviously vary widely. And you can use whatever hosts/VMs you like. But I wanted to make it clear what I was using here and what IP addresses/hostnames I’m using in various examples below.

**Build and install the PKIX SSH package**

On the test client and server, download and extract the PKIX server source:

mkdir ~/packages

cd ~/packages

wget <https://roumenpetrov.info/secsh/src/pkixssh-12.5.1.tar.gz>

cd ~/src

tar xf ~/packages/pkixssh-12.5.1.tar.gz

cd pkixssh-12.5.1

Install PKIX SSH prerequisites:

 sudo apt-get install gcc make libssl-dev zlib1g-dev

Note: This list can vary based on your distro and what packages you’re previously installed. These represent the packages required for my base system.

Configure the the PKIX SSH build to install/run into a custom prefix:

./configure --prefix=/opt/pkix-ssh

Note that this could error out if dependencies are missing. For instance, if you’re on a different distro than Ubuntu 18.04 Server. Install any necessary packages and re-run configure until it completes.

Build PKIX SSH:

make

This should take a few minutes and (hopefully) complete without error. Any errors here usually have to do with package dependencies. The usual procedure here is to look at configure.log, example necessary dependencies, and either disable unnecessary deps by passing options to configure or installing packages. (and google searches if you really get deperate)

Install PKIX SSH

This will install the PKIX SSH executables, libraries, config files, and other resources into the /opt/pkix-ssh tree.

sudo make install

**Create the CA cert and provision the SSH server**

Creating the CA cert

On the Host OS (I used this as the "Control" machine, as it's referred to in the tutorial) generate the CA cert and private key:

openssl genrsa -des3 -out ca.key 2048

openssl req -new -key ca.key -out ca.csr

openssl x509 -req -days 3650 -in ca.csr -signkey ca.key -out ca.crt

Copy the CA cert to the server and client

scp ca.crt [craig@10.20.0.90](mailto:craig@10.20.0.90):

scp ca.crt [craig@10.20.0.91](mailto:craig@10.20.0.91):

Note: this will put the CA certs in the home directories on the client and server - change the username appropriately of course.

Install the CA cert on the server

cd /opt/pkix-ssh/etc/ca/crt/

sudo mv -v ~/ca.crt ./

sudo ln -sv ca.crt `openssl x509 -in ca.crt -noout -hash`.0

Note: The tutorial has you install the CA cert on the client as well - but not sure why. I was able to login without setting it up. All it dod was avoid a "cannot build certificate chain, code=20, msg='unable to get local issuer certificate'" on the client. Perhaps if some mutual host-based auth was setup it would be necessary. But that doesn't seem to be in scope for this tutorial.

**Creating a user cert**

From the client machine, generate a pub/private key and a CSR for cert generation on the host/"Control" machine:

cd ~

ssh-keygen -t rsa -b 2048 -f ~/.ssh/id\_rsa -N ""

openssl req -new -key .ssh/id\_rsa -out id\_rsa.csr

Note that you want to enter some user-specific attributes in the CSR, since you'll want to use attributes in the cert to bind to a particular user on the server (more on this below). For example, I used:

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:Oregon

Locality Name (eg, city) []:Portland

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Craigus Inc

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:Craig Pratt

Email Address []:

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

From the host/"Control" machine copy the CSR from the client, generate a cert, and return it to the client:

scp craig@10.20.0.91:id\_rsa.csr ./

openssl x509 -req -days 3650 -in id\_rsa.csr -out id\_rsa.crt -CA ca.crt -CAkey ca.key -CAcreateserial

scp id\_rsa.crt [craig@10.20.0.91](mailto:craig@10.20.0.91):

On the client device, install the uploaded cert:

 cat id\_rsa.crt >> ~/.ssh/id\_rsa

**Associating a certificate with a server-side user account**

Now that the user has a cert binding a public key to an identity - and signed by a mutually trusted CA - we setup the server to bind the user cert to a particular account on the server.

From the client or host/Control machine, extract the Subject info from the user cert:

openssl x509 -noout -subject -in ./id\_rsa.crt

e.g. This returned for me:

subject=C = US, ST = Oregon, L = Portland, O = Craigus Inc, CN = Craig Pratt

From the user account on the server ([craig@10.20.0.90](mailto:craig@10.20.0.90) in my case), add an entry which authorizes the user based on cert-provided credentials. e.g. I added this entry (based on the subject field from the cert extracted above) to the file ~/.ssh/authorized\_keys:

x509v3-sign-rsa subject= /C=US/ST=Oregon/L=Portland/O=Craigus Inc/CN=Craig Pratt

Note: Presumably, all of the fields must match the cert presented by the client to be associated with the ccount on the server (the account who’s home directory contains the ~/.ssh/authorized\_keys file). There are probably subset matching criteria that can be used that would allow multiple user certs to be granted access to the same server-side account.

**Running the PKIX SSH server**

Change the default bind port

Modify the SSH server config to run on a different port by editing /opt/pkix-ssh/etc/sshd\_config and changing:

#Port 22

#AddressFamily any

#ListenAddress 0.0.0.0

#ListenAddress ::

 to read:

Port 2222

#AddressFamily any

#ListenAddress 0.0.0.0

#ListenAddress ::

and save the file.

Note: I’m doing this presuming you may already have an SSH OpenSSH server installed and running on port 22.

Start the SSH server:

On the server host,  start the PKIX SSH server manually (in debug mode) bu running:

sudo /opt/pkix-ssh/sbin/sshd -d

It should output:

debug1: ssh\_set\_validator: ignore responder url

announced algorithms: x509v3-ecdsa-sha2-nistp256,x509v3-ecdsa-sha2-nistp384,x509v3-ecdsa-sha2-nistp521,x509v3-sign-rsa,x509v3-ssh-rsa,x509v3-rsa2048-sha256,x509v3-sign-dss,x509v3-ssh-dss,ssh-ed25519,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521

debug1: sshd version PKIX-SSH 12.5.1, OpenSSH\_8.3p1, OpenSSL 1.1.1  11 Sep 2018

debug1: private host key #0: ssh-rsa SHA256:8+fbc/PcKRgWFSmgob6U6HySqnHcQISCdVEN48UBLpo

debug1: private host key #1: ecdsa-sha2-nistp256 SHA256:ymYinIKKz8B3dbHrZznnw/ukuZu9M+ahkq8/q/vTtV4

debug1: private host key #2: ssh-ed25519 SHA256:vSQV045kIY+NgJoBwjImLb7mWk7vqJ6ti/woc39mRA4

debug1: rexec\_argv[0]='/opt/pkix-ssh/sbin/sshd'

debug1: rexec\_argv[1]='-d'

debug1: Set /proc/self/oom\_score\_adj from 0 to -1000

debug1: Bind to port 2222 on 0.0.0.0.

Server listening on 0.0.0.0 port 2222.

debug1: Bind to port 2222 on ::.

Server listening on :: port 2222.

 If it outputs anything about “cannot build certificate chain, code=20, msg='unable to get local issuer certificate'” check to ensure the CA cert is installed correctly (described above).

**Logging into the server from the client**

Attempt to login to the server from the client using:

/opt/pkix-ssh/bin/ssh -v -p 2222 [craig@10.20.0.90](mailto:craig@10.20.0.90)

This should login to the server without any prompts or errors. E.g.:

craig@pkix-ssh-client:~$ /opt/pkix-ssh/bin/ssh -v -p 2222 [craig@10.20.0.90](mailto:craig@10.20.0.90)

PKIX-SSH 12.5.1, OpenSSH\_8.3p1, OpenSSL 1.1.1  11 Sep 2018

cannot build certificate chain, code=20, msg='unable to get local issuer certificate'

Offering public key: 'x509v3-ssh-rsa' /home/craig/.ssh/id\_rsa RSA+cert SHA256:MTXpDXXnWX4oZDoVXTpUJE5Kh0YDHXayUe3T/lhWLag

cannot build certificate chain, code=20, msg='unable to get local issuer certificate'

Authenticated to 10.20.0.90 ([10.20.0.90]:2222).

Last login: Fri Aug 21 20:27:29 2020 from 10.20.0.91

Environment:

.  .  .

craig@pkix-ssh-server:~$

And the corresponding server output:

debug1: Server will not fork when running in debugging mode.

debug1: rexec start in 5 out 5 newsock 5 pipe -1 sock 8

debug1: ssh\_set\_validator: ignore responder url

announced algorithms: x509v3-ecdsa-sha2-nistp256,x509v3-ecdsa-sha2-nistp384,x509v3-ecdsa-sha2-nistp521,x509v3-sign-rsa,x509v3-ssh-rsa,x509v3-rsa2048-sha256,x509v3-sign-dss,x509v3-ssh-dss,ssh-ed25519,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521

debug1: sshd version PKIX-SSH 12.5.1, OpenSSH\_8.3p1, OpenSSL 1.1.1  11 Sep 2018

debug1: private host key #0: ssh-rsa SHA256:8+fbc/PcKRgWFSmgob6U6HySqnHcQISCdVEN48UBLpo

debug1: private host key #1: ecdsa-sha2-nistp256 SHA256:ymYinIKKz8B3dbHrZznnw/ukuZu9M+ahkq8/q/vTtV4

debug1: private host key #2: ssh-ed25519 SHA256:vSQV045kIY+NgJoBwjImLb7mWk7vqJ6ti/woc39mRA4

debug1: inetd sockets after dupping: 5, 5

Connection from 10.20.0.91 port 54468 on 10.20.0.90 port 2222 rdomain ""

debug1: Local version string SSH-2.0-OpenSSH\_8.3 PKIX[12.5.1]

debug1: Remote protocol version 2.0, remote software version OpenSSH\_8.3 PKIX[12.5.1]

debug1: match: OpenSSH\_8.3 PKIX[12.5.1] pat OpenSSH\* compat 0x04000000

debug1: x.509 compatibility rfc6187\_missing\_key\_identifier=no: pattern 'OpenSSH\*PKIX[??.\*' match 'OpenSSH\_8.3 PKIX[12.5.1]'

debug1: x.509 compatibility rfc6187\_asn1\_opaque\_ecdsa\_signature=no: pattern 'OpenSSH\*PKIX[??.\*' match 'OpenSSH\_8.3 PKIX[12.5.1]'

debug1: x.509 compatibility broken list with accepted publickey algorithms=no: pattern 'OpenSSH\*PKIX\*' match 'OpenSSH\_8.3 PKIX[12.5.1]'

debug1: permanently\_set\_uid: 110/65534 [preauth]

debug1: list\_hostkey\_types: ssh-rsa,rsa-sha2-256,rsa-sha2-512,ecdsa-sha2-nistp256,ssh-ed25519 [preauth]

debug1: SSH2\_MSG\_KEXINIT sent [preauth]

debug1: SSH2\_MSG\_KEXINIT received [preauth]

debug1: kex: algorithm: curve25519-sha256 [preauth]

debug1: kex: host key algorithm: ecdsa-sha2-nistp256 [preauth]

debug1: kex: client->server cipher: [chacha20-poly1305@openssh.com](mailto:chacha20-poly1305@openssh.com) MAC: <implicit> compression: none [preauth]

debug1: kex: server->client cipher: [chacha20-poly1305@openssh.com](mailto:chacha20-poly1305@openssh.com) MAC: <implicit> compression: none [preauth]

debug1: expecting SSH2\_MSG\_KEX\_ECDH\_INIT [preauth]

debug1: rekey out after 134217728 blocks [preauth]

debug1: SSH2\_MSG\_NEWKEYS sent [preauth]

debug1: Sending SSH2\_MSG\_EXT\_INFO [preauth]

debug1: expecting SSH2\_MSG\_NEWKEYS [preauth]

debug1: SSH2\_MSG\_NEWKEYS received [preauth]

debug1: rekey in after 134217728 blocks [preauth]

debug1: KEX done [preauth]

debug1: userauth-request for user craig service ssh-connection method none [preauth]

debug1: attempt 0 failures 0 [preauth]

debug1: userauth-request for user craig service ssh-connection method publickey [preauth]

debug1: attempt 1 failures 0 [preauth]

debug1: userauth\_pubkey: test pkalg x509v3-ssh-rsa pkblob RSA+cert SHA256:MTXpDXXnWX4oZDoVXTpUJE5Kh0YDHXayUe3T/lhWLag [preauth]

debug1: temporarily\_use\_uid: 1000/1000 (e=0/0)

debug1: trying public key file /home/craig/.ssh/authorized\_keys

debug1: fd 6 clearing O\_NONBLOCK

debug1: /home/craig/.ssh/authorized\_keys:1: matching key found: X509(rsa) C=US,ST=Oregon,L=Portland,O=Craigus Inc,CN=Craig Pratt

debug1: /home/craig/.ssh/authorized\_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding

Authorized by X509(rsa) : C=US,ST=Oregon,L=Portland,O=Craigus Inc,CN=Craig Pratt

debug1: restore\_uid: 0/0

Postponed publickey for craig from 10.20.0.91 port 54468 ssh2 [preauth]

debug1: userauth-request for user craig service ssh-connection method publickey [preauth]

debug1: attempt 2 failures 0 [preauth]

debug1: temporarily\_use\_uid: 1000/1000 (e=0/0)

debug1: trying public key file /home/craig/.ssh/authorized\_keys

debug1: fd 6 clearing O\_NONBLOCK

debug1: /home/craig/.ssh/authorized\_keys:1: matching key found: X509(rsa) C=US,ST=Oregon,L=Portland,O=Craigus Inc,CN=Craig Pratt

debug1: /home/craig/.ssh/authorized\_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding

Authorized by X509(rsa) : C=US,ST=Oregon,L=Portland,O=Craigus Inc,CN=Craig Pratt

debug1: restore\_uid: 0/0

debug1: auth\_activate\_options: setting new authentication options

Accepted publickey for craig from 10.20.0.91 port 54468 ssh2: RSA+cert SHA256:MTXpDXXnWX4oZDoVXTpUJE5Kh0YDHXayUe3T/lhWLag

debug1: monitor\_child\_preauth: craig has been authenticated by privileged process

debug1: auth\_activate\_options: setting new authentication options [preauth]

debug1: monitor\_read\_log: child log fd closed

User child is on pid 32183

debug1: permanently\_set\_uid: 1000/1000

debug1: rekey in after 134217728 blocks

debug1: rekey out after 134217728 blocks

debug1: ssh\_packet\_set\_postauth: called

debug1: active: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding

debug1: Entering interactive session for SSH2.

debug1: server\_init\_dispatch

debug1: server\_input\_channel\_open: ctype session rchan 0 win 1048576 max 16384

debug1: input\_session\_request

debug1: channel 0: new [server-session]

debug1: session\_new: session 0

debug1: session\_open: channel 0

debug1: session\_open: session 0: link with channel 0

debug1: server\_input\_channel\_open: confirm session

debug1: server\_input\_global\_request: rtype [no-more-sessions@openssh.com](mailto:no-more-sessions@openssh.com) want\_reply 0

debug1: server\_input\_channel\_req: channel 0 request pty-req reply 1

debug1: session\_by\_channel: session 0 channel 0

debug1: session\_input\_channel\_req: session 0 req pty-req

debug1: Allocating pty.

debug1: session\_new: session 0

debug1: session\_pty\_req: session 0 alloc /dev/pts/1

debug1: server\_input\_channel\_req: channel 0 request shell reply 1

debug1: session\_by\_channel: session 0 channel 0

debug1: session\_input\_channel\_req: session 0 req shell

Starting session: shell on pts/1 for craig from 10.20.0.91 port 54468 id 0

debug1: Setting controlling tty using TIOCSCTTY.

Logging out/restarting

If you logout of the client, the server will terminate (in debug mode). Just restart it to attempt another login.

**Provisioning other clients**

….