Large Scale SSH

Netconf 2020

Michael W Lucas

https://www.mwl.io
Twitter - @mwlauthor
Mastodon - @mwlucas@bsd.network

About Me

- Author
- Unix since 198(mumble), network admin since 1995
- Founding member of semibug.org
- ~~Blatant BSD bigot~~
- ~~Notorious BSD demagogue~~
- ~~BSD Geezer~~
- Long-time BSD advocate
- Author of many books, including SSH Mastery, SNMP Mastery
- As Michael Warren Lucas, writes novels like git commit murder

Prerequisites

- You use and configure OpenSSH
- You use key-based authentication
- scp(1) and sftp(1) don’t scare you
- Automation is cool

- Mostly talking OpenSSH, some PuTTY

One Problem, Two Faces

- Keys
  - Distributing User Keys
  - Validating Host Keys

User Keys

- Distributing authorized_keys to individual hosts
- Querying the network for authorized_keys entries

Distributing User Keys

- Copy $HOME/.ssh/authorized_keys
- Problem: you can’t trust users
  - Deliberate mucking
  - Intruders

Automated Solution

- Users don’t get to update their own authorized keys, submit to automation system
- In sshd_config, do

  AuthorizedKeysFile /etc/ssh/keys/%u
- Keys go in /etc/ssh/keys/mwl, /etc/ssh/keys/bagel, etc

Querying the Network for Keys

- Do you have LDAP?
- Load the proper schema into your LDAP directory
- Configure sshd_config

  AuthorizedKeysCommand /usr/scripts/getAuthorizedKeys.pl

  AuthorizedKeysCommandUser keymaster
- The script you need varies with LDAP implementation

Simple Network Script

```
#!/usr/bin/perl
die unless $ARGV[0];
open (LDAP, "/usr/bin/ldapsearch -L -xZb\"dc=michaelwlucas,dc=com\"  \
        '(&(objectClass=posixAccount)(uid=$ARGV[0]))' sshPublicKey |") \
        || die "ldapsearch failed $!\n";
while (<LDAP>) {
        next if /^#|^version|^dn\:|^\s*$/;
        s/\n//;
        s/\://g;
        s/sshPublicKey/\n/;
        s/^ //;
        print;
}
print "\n";
```

No LDAP? No Problem!

- LDAP is the most common network-available database
- Your network is special
- AuthorizedKeysCommand is a script
- Do you have SQL? Any kind of directory?

Host Keys

- How many of you know you should meticulously verify a server's SSH host key before logging in?
- How many of you actually verify a server's SSH host key?
- SSH host key verification is like flossing.
- Solution: don't have a human do it
  - Distribute known_hosts
  - Look up known_hosts

Creating known_hosts

- Before exposing to the Internet, get a known good host key with ssh-keyscan(1)

  $ ssh-keyscan www > www.known_hosts

  - I keep these per-hostname files
- Can simplify known_hosts by reducing supported algorithms in sshd_config

  HostKeyAlgorithms ssh-ed25519, ssh-rsa
- Concatenate into known_hosts

Revoking known_hosts

- If a key is compromised, don’t give users the chance to manually accept it. Revoke it & redistribute.

@revoked www2 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTIt…

Distributing known_hosts

- Leave $HOME/.ssh/known_hosts alone
- Place in /etc/ssh/ssh_known_hosts
- User’s known_hosts can contain obsolete keys, move it away upon first deployment
- Once you have automation: /etc/ssh/ssh_config contains global SSH settings

Distributing PuTTY host key cache

- PuTTY stores host keys in registry
- The kh2reg.py script in PuTTY source code converts known_hosts to PuTTY registry keys

```
$ hk2reg.py pristine-known_hosts > putty.reg
```

- Have AD install this at user logon

Host Keys in DNS

- SSHFP (SSH Fingerprint) records
- Requires DNSSEC

$ mail~; **ssh-keygen -r mail**
mail IN SSHFP 1 1 f0a2cc23ed07b4bf1201eaae4aba712bae739945
mail IN SSHFP 1 2 4e7eb91fedc66b0ca59c8e74244826302a4e0ee4568d4c6a0149543244c3339b
mail IN SSHFP 2 1 68a0e9db5e3ee92db1e2d8399b05a3e7ab244a1a
mail IN SSHFP 2 2 7d04969d6e75f95a84f5cb3430e49172ebfddc03e987bb4f176f34b6e8753b94
…

- Key files from a different host? Use –f flag

SSH Certificate Authorities

- Totally different from X.509 (TLS) certificate authorities
- A CA is a method of delegating trust. Requires two features:
  - Encryption
  - Signing
- An SSH key can provide both of these
- SSH CA: an SSH key you choose to use as a CA
- Much like a self-signed TLS certificate
- Don’t need external trust
- 100% requires automation: copy files & restart sshd

Certificate Expiration

- Certificates Expire
- 25-year certs will be cryptographically insecure long before they expire
- Expire every year or so
- Add bonus time for appendicitis
- Or, auto-renew in half the expiration time

Organizing your CA

- One CA for users, one for hosts
- Don’t use /etc/ssh
  - /usr/local/sshca/users
  - /usr/local/sshca/hosts
  - Give each host or user their own subdirectory, like /usr/local/sshca/users/mwl or /usr/local/sshca/hosts/www
  - Every host has host key files, keep them separate from other hosts

Generating Host & User CA keys

- Our old friend ssh-keygen(1)

$ ssh-keygen –t rsa –f host-myca-key –c 'host CA key 2018-06-09'

$ ssh-keygen –t rsa –f user-myca-key –c ‘user CA key 2018-06-09’

- Secure the keys to the kingdom!
- Now teach sshd(8) and ssh(1) to trust these CAs

CA and sshd(8)

- Sshd authenticates users, it must trust the user CA key
- Create a file containing all trusted user CA public keys, one key per line
- In sshd_config, use

    TrustedUserCAKeys /etc/ssh/user-ca-keys.pub

CA and ssh(1)

- ssh(1) authenticates hosts, it needs to trust host CA
- ssh_known_hosts is for exactly this
- Mark the key with @cert-authority and domains it’s valid for

```
@cert-authority *.mwl.io,michaelwlucas.com ssh-rsa AAAA…
```

Certificate Identity

- Says what the cert is for
- Set at cert creation
- Logged when key is used
- I use host_ for host certs and user_ for user certs

Certificate Archives

- Easier to revoke a key when you have the key
- Users can’t be trusted
- Keep a copy of cert and public key in CA archive

Creating Host Certificates

- Use ssh-keygen(8) - **not** ssh-keysign(8)!

```
# ssh-keygen -s host-mwlca-key -I host_sloth -h \
    -n sloth.mwl.io -V +56w5d ssh_host_*pub
```

- -s = host CA key
- -I = identity
- -h = host cert
- -n = host or hosts this cert is good for
- -V = expiration date
- Last, key file

Host Certificate Files

- You get a -cert file for each public key file
  - ssh_host_rsa_key.pub, ssh_host_rsa_key-cert.pub
  - ssh_host_dsa_key.pub, ssh_host_dsa_key-cert.pub
  - ssh_host_ecdsa_key.pub, ssh_host_ecdsa_key-cert.pub
  - ssh_host_ed25519_key.pub, ssh_host_ed25519_key-cert.pub

- Copy the certs to server’s /etc/ssh

```
HostKey /etc/ssh/ssh_host_rsa_key
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
…
HostKey /etc/ssh/ssh_host_ed25519_key
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
```

Viewing Certificates

 # ssh-keygen -Lf ssh_host_ed25519_key-cert.pub

ssh_host_ed25519_key-cert.pub:
Type: ssh-ed25519-cert-v01@openssh.com host certificate
Public key: ED25519-CERT SHA256:nNtyIQidY3MXAEfpWZ0wzkXK...
Signing CA: RSA SHA256:ZQHNMc2TmWlnygGy9+UoOYFK92RdbguzN...
Key ID: “sloth”
Serial: 0
Valid: from 2017-12-04T11:52:00 to 2017-12-25T11:53:17
Principals:
sloth.mwl.io
Critical Options: (none)
Extensions: (none)

Testing Host Certificates

- Move known_hosts out of the way
- Should connect without being prompted for key verification
- Doesn’t work? Add 1-3 –v to get debugging output
  - Don’t have CA installed on client?

Creating User Certificates

- Use ssh-keygen(8) - **not** ssh-keysign(8)!

```
# ssh-keygen -s user-mwlca-key -I user_mwl \
    -n mwl -V +56w5d id_rsa.pub
```

- -s = host CA key
- -I = identity
- -n = user this cert is good for
- -V = expiration date
- Last, user’s public key file

Using User Certificates

- Copy certificate file into $HOME/.ssh/
- Corresponding key file should be there, i.e.:
  - id_rsa.pub, id_rsa-cert.pub

Disabling authorized_keys

- Users cannot be trusted
- Someone will upload an authorized_keys just for convenience
- In sshd_config
  - AuthorizedKeysFile none

Massive Scale SSH

- Millions of servers?
- Tens of thousands of sysadmins?
- UID range of 1-65535 too small?
- LDAP servers releasing smoke from too much load?

Principals

- A named entity, not tied to a hostname or UID
- Can be structured any way you want
- Principals can be authorized to log in via sshd

AuthorizedPrincipalsFile

- Contains a list of principals allowed to use the service
  - AuthorizedPrincipalsFile /etc/ssh/principals
- Might contain:

  everywhere-root
  europe-root
  europe-database

- Can also do AuthorizedPrincipalsFile /etc/ssh/principals/%u

Create Certs with Principals

- Use ssh-keygen

```
# ssh-keygen -s user-mwlca-key \
    -I user_87181_Michael_Lucas -n peasants,vermin \
    -V +52w id_rsa.pub
```

- Identity contains employee ID number and real name
- -n gives assigned principals, peasants and vermin
- Must re-issue cert to assign new principals

Look Up Principals

- Text files across millions of servers? No no no

  AuthorizedPrincipalsCommand /usr/scripts/principals.pl

  AuthorizedPrincipalsCommandUser apc

Questions and Answers?