Using keys with SSH
Rob Judd
(rjudd@mlug.missouri.edu)
Introduction

- SSH is a drop-in replacement for telnet that allows encrypted network connections.

- There are two main versions, 1.5 “SSH-1” and 2.0 “SSH-2” [1].

- There are many implementations; the most popular are OpenSSH [2] from the OpenBSD people and the “SSH Secure Shell” distributed by SSH Communications Security Inc. [3], which I’ll call “commercial SSH”.

- SSH uses Public-Key Cryptography for key exchange and authentication, and symmetric-key encryption for the rest of the communication.
Public Key Cryptography

- Two main types of encryption: symmetric-key encryption and public key encryption (or asymmetric encryption).

- Symmetric key encryption uses the same key for encryption and decryption. Examples are DES, AES, twofish.

- Public-key encryption uses two keys. The encryption key is public, while the decryption key is private. Knowledge of the public key does not help in finding the private key.

- Public-key encryption is much slower than symmetric key encryption and is only used for small tasks such as key exchange.
An SSH session

- An SSH session has three main phases: Connection, Authentication and Transport.

- In the connection phase the client and server negotiate a secure connection.

- In the authentication phase the client authenticates the user with the server.

- Finally the main phase, transport, is where the rest of the communication takes place.

- This talk is about implementing the authentication phase using public-key cryptography.
Authentication with SSH

Three popular methods for authentication

trusted host is typically used in an internal network where a user has already logged in

password uses the system’s native password authentication; the client prompts the user for a password and then the server authenticates this

public key the user keeps the private key in a secure place, places the public key on the remote host and uses the pair together

Password authentication requires no user setup and is transparent for telnet users. Public key authentication is faster and more secure in the long run.
Public Key Authentication

Once the connection phase has established a secure (encrypted) link the client must authenticate (adapted from [4]).

1. The client requests public-key authentication with a particular key.

2. The server looks up the key in the target account and fails if the key does not match.

3. The server checks any access restrictions for the key and may deny the connection.

4. The server returns a random 256 bit string encrypted with the client’s public key.

5. The client decrypts the challenge, combines it with the session ID, hashes the result with MD5, and returns the hash.

6. The server computes the same MD5 hash; if the client’s reply matches, then the authentication succeeds.
The User’s Perspective

To use this the user must perform the following steps:

- Generate a private/public key pair;

- Save the public key on the remote machine;

- Make the private key available to the SSH client.
Generating keys

- Generate a key with `ssh-keygen -t type` where `type` is the type of key
- SSH-1 uses RSA keys (`type` is `rsa1`)
- SSH-2 specifies only DSA keys (`type` is `dsa`)
- SSH-2 may use RSA keys (`type` is `rsa`)
- Default number of bits is 1024; may be changed with `-b bits`
- You will be prompted for a filename; that is the private key. The public key is `filename.pub`

rob@localhost:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/rob/.ssh/id_dsa):
Enter passphrase (empty for no passphrase): ***************
Enter same passphrase again: ***************
Your identification has been saved in /home/rob/.ssh/id_dsa.
Your public key has been saved in /home/rob/.ssh/id_dsa.pub.
The key fingerprint is:
16:cc:5c:35:fc:5a:09:92:45:f1:0b:8a:0a:98:d1:dd rob@localhost
rob@localhost:~$
Storing Keys

You must now put the public key on the remote machine

- Copy the public key to the server: use scp, ftp, email, paste it with a mouse, anything.
- Connect to the server with ssh and password authentication.
- put the key in the “correct” place. This depends on your implementation and version.

OpenSSH (SSH-1)

   copy the key into ~/ssh/authorized_keys

OpenSSH (SSH-2)

   copy the key into ~/ssh/authorized_keys2

Sun SSH Solaris 9 (SSH-2)

   copy the key into ~/ssh/authorized_keys

Commercial SSH (SSH-1)

   copy the key into ~/ssh/authorized_keys

Commercial SSH (SSH-2)

   copy the key to the file ~/ssh2/mykey.pub
   and create a file called ~/ssh2/authorization that contains a line Key mykey.pub
Warning

The key format for commercial SSH and OpenSSH is different. To export an OpenSSH key as a commercial SSH public key you can do

```
rob@localhost$ ssh-keygen -e -f .ssh/id_dsa
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "1024-bit DSA, converted from OpenSSH by rob@localhost"
AAAAB3NzaC1kc3MAAACBAJrlkqrUbPv1LYZb71xxOp1uSeK3957kYecmgjt1QUmCfDrj76B9ZB78lnKvlaXdTuEEZ+quHPCJ5vMR4+Ej07/dq3Bs2cXv+ikM80UUXXss1Jj/M3X3wC3Xwc4z1IlXgUvwiecWmhVV2KPGCyk05RRG8G6746Jofth5VYfHccJtHAAAFQDraTbZMPEQarfJQR+vCmvbBTa3uwAAAIEAmkwB57i1/89Bzgfeg/TnxfjCdieuyRigbV9fP/qgO1WxQP0a
3xcWC43833SorR588ySY9XqRL0YMLYgR/F6ZAkdbgYT4LnLaXx1DYN8EdM4vClHVb8KZiG
PpTXIJmtvg7qB023AmEOTOvAL3X0YaNZ/sMmaFAub68LmI2+HoUAAACBAINoDNIoQ+79
xawNaeuW6v6foF6aGxNpJ0g4434wiUf4uvCbjm8XCVGzoaalJTw3ZfL/WdoiSoh3dUUXTY
mGhWF8JerkGgCh5V0xbhFIKMXXDbE+vFP1Z2Tjyw8H1eLRP3K14Bu0BB0zzajRFsDHA/km
X3+42P8hnu2zSE1Jld9c
---- END SSH2 PUBLIC KEY ----
rob@localhost$
```
Using your keys

• If you try to connect now, ssh will prompt for your key passphrase.

• To avoid this use ssh-agent to store your keys while you are logged in.

• Start ssh-agent when you first log in.

• Add your keys to it with ssh-add

• ssh will no longer ask for your passphrase

rob@localhost$ eval ‘ssh-agent’
Agent pid 17143
rob@localhost$ ssh-add
Enter passphrase for /home/rob/.ssh/id_rsa:
Identity added: /home/rob/.ssh/id_rsa (/home/rob/.ssh/id_rsa)
Identity added: /home/rob/.ssh/id_dsa (/home/rob/.ssh/id_dsa)
Identity added: /home/rob/.ssh/identity (rob@localhost)
rob@localhost$
rob@localhost$
Problems

- `ssh-agent` exports environment variables and is thus only available from the shell in which it was run and any child processes.

- To get around this run `eval 'ssh-agent'` when you first log in to a console.

- or, if you use `X` and `xdm`, then put `eval 'ssh-agent'` in either `.xsession` (if you use it) or else edit `/etc/X11/xdm/Xsession` to contain that line before it starts up the window manager.
References


2. OpenSSH: http://www.openssh.org/

3. SSH Communications Security Inc.: http://www.ssh.com/


5. These slides: http://www.cs.du.edu/~rjudd/cryptography/notes/