SSH

Secure Shell: SSH

- Partly a tool, partly an application
- Features:
  - Encrypted login and shell connections
  - Easy, drop-in replacements for rlogin, rsh, rcp
  - Multiple means of authentication
  - Interesting case study with regards to deployability

Login sequence

- Client contacts server
- Server sends its public RSA „host" key (> 1024 bits), an RSA „server" key (> 768 bits), and list of ciphers (server key changes hourly)
- Client authenticates server
- Client generates session key and encrypts it using both host and server key
- Server decrypts it and uses it for traffic encryption
- Client authenticates to host

Two server keys: Why?

- Long key: for authentication
- Shorter key: Approx. of forward secrecy
- Why not Diffie-Hellman?
  - Speed: 768-bit RSA faster
  - Tatu Ylönen, the author, „inspired amateur“ in 1995...

Authentication?

- Server authentication by client?
  - Server is sending key not certificate
  - First time keys == ask user to accept it
    - Don‘t know if this is correct key!
  - Cache keys in „known hosts“ file
    - One does not know that the key is correct
    - Only know that key is the same as last time
  - Vulnerability on initial login only!
  - But user has to know what is happening

Sample initial login

```
$ ssh foo
The authenticity of host 'foo (192.168.77.222)' can't
RSA key fingerprint is cf:26:92:6c:01:c1:05:c7:51:de
Are you sure you want to continue connecting (yes/no)
Warning: Permanently added 'foo (RSA) to the list of
```

An attack?

```
$ ssh foo
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the
   middle attack)!
It is also possible that the RSA host key has just been
   changed.
The fingerprint for the RSA key sent by the remote host is
   f1:68:d8:0d:0a:1b:78:2c:48:3a:aa:1b:4a:8c:cb:ca.
Please contact your system administrator.
Add correct host key in /home/smb/.ssh/known_hosts to get
   rid of this message.
Offending key in /home/smb/.ssh/known_hosts:86
RSA host key for foo has changed and you have requested
   strict checking.
Host key verification failed.
```

Cipher list

- Server sends options
- Client picks
- Rollback, downgrade attack
  - Attacker substitutes list to contain only weak or cracked ciphers
  - Solution: after start of encryption send an authenticated list of original proposed algorithms

Client authentication

- Many
  - Password authentication

    + Simple

    - Password guessing possible
  - Public key authentication
    - Client sends public key to server
    - Server encrypts 256-bit random number with client public key
    - Client decrypts it and sends MD5 hash back

      + better security

      - simple key not certificate!
  - Host-based authentication
    - Client's host has public/private key pair
    - Useful only for two machines under common administration

Password guessing

```
00:01:36 foo sshd: Invalid user duane from 206.231.8
00:01:37 foo sshd: Invalid user murray from 206.231.
00:01:38 foo sshd: Invalid user kovic from 206.231.8
00:01:39 foo sshd: Invalid user mitchell from 206.23
00:01:40 foo sshd: Invalid user nance from 206.231.8
00:01:41 foo sshd: Invalid user liberty from 206.231
00:01:42 foo sshd: Invalid user alan from 206.231.8.
00:01:43 foo sshd: Invalid user wilfe from 206.231.8
00:01:45 foo sshd: Invalid user ruthy from 206.231.8
00:01:46 foo sshd: Invalid user oriana from 206.231.
00:01:47 foo sshd: Invalid user mauzone from 206.231
00:01:48 foo sshd: Invalid user leopold from 206.231
```

Storing private keys

- Compromised private key
  => all security bets are off
- Minimum
  - Read-protected
  - To circumvent, e.g., NFS problems, encrypt with some symmetric cipher...

Too many prompts!

- Problem: need to constantly enter password
- Solution: ssh agent
  - Prompts for passphrase once
  - Decrypts in memory
  - Performs public key operation on one's behalf
- SSH agent secured
  - Problem:
    - Uses Unix sockets (lives in file system)
    - File permissions on Unix-domain sockets may not be enforced
  - Solution:
    - But all systems verify permissions on containing directory
    - Put socket in protected directory (use shell to pass info)

SSH agent – usage

```
$ set|grep SSH
SSH_AGENT_PID=363
SSH_AUTH_SOCK=/tmp/ssh-00000418aa/agent.418
$ ls -la /tmp/ssh-00000418aa
total 8
drwx------ 2 smb wheel 20 Oct 11 03:15 .
drwxrwxrwt 4 root wheel 260 Oct 12 00:13 ..
srwxr-xr-x 1 smb wheel 0 Oct 10 20:57 agent.418
```

Connection forwarding

- Circumvents firewalls ☹
- Talking to an internal POP3 server:
  - **`ssh -L 110:mbox:110 firewall`**
  - followed by (in another window)
  - **`telnet 127.0.0.1 110`**
- Or, of course, configure your mailer to talk to 127.0.0.1
- Forwarding remote connections to local machine is also possible

Connection forwarding (2)

- Forwarding access to authentication agent possible
  - Never do connection-forwarding to an insecure machine!
- Circumventing policies possible
- X11 forwarding
  - Authentication?
    - Some people don‘t...
    - Kerberos „magic cookie“ mode
      - App has to read a secret value from a file and send it to X server
      - Remote sshd generates new secret
      - Local client replaces remote cookie with local one and contacts local-X-server

SSH: Deployability

- SSH success story
  - No infrastructure
    - No PKI, CAs, no central server
  - Usability
    - Drop-in replacement for rlogin
    - Same trust model ☹
    - Little user training
    - Nice features (connection/X11-forwarding)
    - Many Unix variants
  - Security
    - Defended against real attacks
    - Extra functionality
    - Add-ons: scp

SSH: Limitations

- Does not solve all problems
  - Cryptographic mistakes (i.e.: CRC instead of MD5)
  - Compromised hosts
    - Trojans possible
    - X11 and authentication agents can be captured
  - Password-guessing
  - Deliberate user misbehavior
  - SSH worms
    - Known host files indicates trust patterns
    - Transitive trust patterns
    - Encrypt your private keys

SSH conclusion

- Professional cryptographer would have designed a system around certificates issued by properly-isolated and secured CAs
  - Would have been more secure
  - Would likely have been undeployable
- Got real security from partially-secured implementation that better matched deployment patterns