

SSH — Secure Shell

Open Source Security Tools for Information Technology Professionals

School of Professional Studies (SPS)

The City University of New York (CUNY)

Aron Trauring
Adjunct Professor
CEO, Zoteca

Class 4 October 17th, 2005

Clear Text Problem

- Data in packets available to anyone to read — clear text, plain text
- How to share while keeping data confidential?
- Remote Administration — How to allow access without making it easy to break in?

Types of Encryption

- Symmetric/Shared Secret — same key used to encrypt and decrypt
- Problem: if key is intercepted you are wide-open
- Asymmetric/Public Key — different key used to encrypt and decrypt

Public Key Encryption

- Whitfield Diffie, Martin Hellman, Ralph Merkle — 1976
- Split key into two parts — private and public
- Sender encrypts message with recipients public key
- Recipient decrypts message with private key
- Advantage: two entities can communicate secretly without sharing a secret key
- Created by using one way functions — easy to compute in one direction but not the reverse
- Prime number factors — easy to multiply two prime numbers but no algorithm to factor

Practical Implementation of PKE

- Allows for both key generation and digital signature
- RSA — Ron Rivest, Adi Shamir and Len Adleman at MIT, 1977 (U.S. patent expired in 2000)
- RSA <http://en.wikipedia.org/wiki/RSA>
- DSA — Digital Signature Algorithm. US Federal standard, 1991.
- DSA http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

SSH

- Based on PKE — uses either RSA or DSA (RSA default)
- Establishes a secure connection
- All traffic over connection is encrypted
- Client-server model
- OpenSSH — FOSS version available on all Unices and Windows (PuTTY)

Stages of Establishing the SSH connection

1. Client needs to establish that it is talking to the machine you asked it to, and not another machine that's spoofing it (or sitting in the middle accepting data and passing it on transparently)
2. Server on the remote machine may want to establish that you are connecting from the machine you appear to be, and not another machine that's spoofing it
3. Client and server exchange keys for encrypting all future traffic between them
4. Client needs to convince the server that you are who you claim to be, and are authorized to do things on the server

Host Authentication

- Hosts that support SSH arrange to have a host identification key consisting of a public and private part.
- Server sends its public key to the client
- Client encrypts a random session key with it and sends the result back to the server.
- Client clearly knows the session key as it generated it, and only the server can use its private key to decrypt the message the client sent to it.
- This both secures the session, and assures the client that it must be talking to the correct server machine

Server Check

- Client has a list of public keys for server machines (taken from /etc/ssh_host_key.pub on server)
- Per-machine in /etc/ssh/ssh_known_hosts
- Per-user ~/.ssh/known_hosts
- If a public key is not locally held, get warning: “Host key not found from the list of known hosts. Are you sure you want to continue connecting (yes/no)?”
- If answer yes, the host public key will be added to that user’s personal list of host public keys
- Minor danger of DNS spoofing

Client Challenge

- Administrator has included the public key for the client machine in the per-machine list on the server machine.S
- Server creates a "challenge" encrypted with the client's host public key
- Only if the client is the machine it claims to be will it have access to the correct private key which enables it to decrypt the challenge and send it back to the server

Encryption — Session Key

- Goal: to prevent any data (both authentication information, and subsequent session data) transferred (password, keystrokes and program output) from being snoopable
- Ssh client and server use exchange keys for one of a number of different encryption methods used to encrypt all subsequent data transfers in the session
- Attacker would need to know the key in order to decrypt any part of the session, and the keys are negotiated in an encrypted form that requires knowledge of one or other hosts' secret host keys.
- Various encryption methods are available (TBD)

User Authentication

- rhosts — checks if in .rhosts file and only does host authentication: not secure and not often used
- rhosts with client challenge — slightly better
- password — user enters password. Password sent encrypted. Usually don't allow root login.
- Public key encryption

PKE User Authentication

- User creates key pair using ssh-keygen
- Puts copy of public key (`~/.ssh/id_ds.pub` or `id_rsa.pub`) in server machine
- Per-machine in `/etc/ssh/authorized_keys`
- Per-user in `~/.ssh/authorized_keys`
- Server creates a challenge encrypted with user's public key.
- If the client knows the corresponding private key, it can decrypt the challenge and send it back to the server, which
- Server now knows that the client must have the secret and should therefore be allowed access.

SSH Uses

- Remote login
- Port forwarding