Secure Shell

CMSC 426/626 - Fall 2014

Outline

- Attacks and Vulnerabilities
- The Dark Ages: Telnet, Rlogin, and Rsh
- Secure Shell (SSH)

Attacks and Vulnerabilities

- *Interception* - “sniffing” unencrypted packets; session hijacking; Man-in-the-Middle (MitM).
- *Modification* - alteration of packet contents; Man-in-the-Middle.
- *Falsification* - fake hosts; Source IP Spoofing, Man-in-the-Middle, blind injection.
- *Interruption* - Denial of Service.

Telnet

- Remote terminal protocol - defined in multiple RFCs spanning 35 years.
- Data unencrypted - no additional confidentiality.

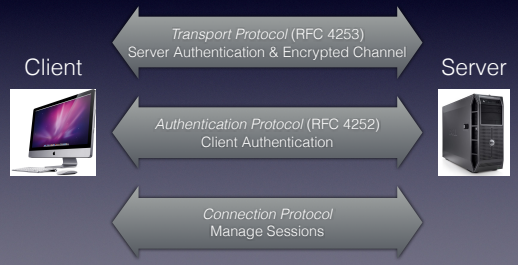
  User names and passwords sent in the clear!
- No authentication of hosts beyond what is provided by TCP/IP and DNS.

SSH History

- The first version, SSH-1, was developed in 1995 in Finland by a victim of a sniffing attack.
- Designed to replace Telnet and related utilities.
- Originally open source, but became proprietary over time (developer started a company).
- **SSH-2** developed by the IETF. Adopted as a standard in 2006.

SSH-2

SSH-2 defined and modified in multiple RFCs.



We'll only look at the *Transport* and *Authentication* protocols…

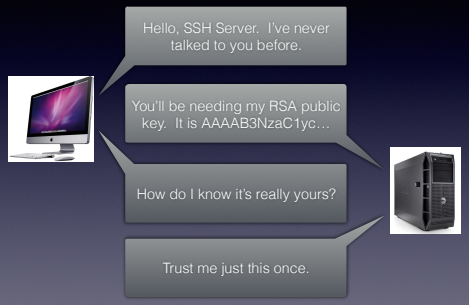
> …that's where most of the security-related processing occurs.

But first we have to understand *host keys*.

Server Host Keys

- RSA or Digital Signature Algorithm (DSA) keys (private and public parts in separate files).
- Used to authenticate the *server to the client* in the Transport Protocol; server signs a message using its private key.
- Client needs to know the server's public key.

Getting Host Keys



- In most cases, the server just sends the key, and the user must decide whether to trust it or not.
- Once a server public key is accepted, the client stores it, so in the future it doesn't have to ask the server for its key.
- This information is in `.ssh/known_hosts`:

```
linux.gl.umbc.edu,130.85.12.141 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAyrNkS6GwMRU3HBDE5gxP7unqorXtkW5B23XnTHl3
qmVVe7Ivo5BWO1+JPe6jWrIkR/g516T3J9sX0AjgND553pj++XYtrAQV7AHaq9n4ViQF
7ZP8PiV/oeMJxicglgIRo2Rd3VnCyVV+ukgZjS/Kvty/UED1ZZjWwDBRB5MwWb0=
```

In a more security conscious environment, the public host keys could be distributed more securely:

- In the form of a *certificate* signed by a *certification authority* (CA).
- Distributed out-of-band, e.g. delivered on removable media or downloaded through a separate, secure channel such as TLS.

Transport Protocol

- Identify client and server software versions
- Negotiate cryptographic algorithms
- Derive a shared secret key using DH
- Authenticate server through signed message
- Initiate encrypted connection

Details: SSH Background

Client Authentication

- Two primary methods: **password** or **public key**.
- Password method is straightforward:
  - Client sends password authentication request
  - Client sends password
- Authentication occurs *after* the connection has become encrypted, so password is protected.
- The server can reject password authentication request.

- Public key authentication is a little more involved
  - Client has public and private keys
  - Public key is known to server (how?)
  - Client requests public key authentication; sends PKC algorithm and public key
  - Client sends a signed message and the server verifies the signature
- Server can reject authentication request outright or authentication can fail due to a signature error.

Supported Encryption

OpenSSH supported symmetric algorithms:

- 3DES (triple DES)
- Blowfish
- AES
- RC4
- CAST128

Supported authentication algorithms:

- DSA
- RSA
- Elliptic Curve DSA (ECDSA)
- Edwards Curve DSA (EdDSA or Ed25519)

How does SSH Help?

- Doesn't quite defeat *Falsification*, but makes it a lot harder.
  - Both server and client are cryptographically authenticated.
  - Vulnerable when server sends host key for the first time.
- Protects against *Interception* through the use of strong encryption.
- Protects against *Modification* through use of MACs (we didn't go into this in detail).

Exercises are on the website.