Secure Shell

CMSC 426 - Computer Security

Outline

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

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.

Transport Protocol (RFC 4253)
Server Authentication & Encrypted Channel

Client

Server

Authentication Protocol (RFC 4252)
Client Authentication

Connection Protocol
Manage Sessions

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

Hello, SSH Server. I've never talked to you before.

You'll be needing my RSA public key. It is AAAAB3NzaC1yc…

How do I know it's really yours?

Trust me just this once.

- 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

Required symmetric ciphers:

- 3DES (CBC)
- AES (CBC)

Optional symmetric ciphers

- Blowfish / Twofish (CBC)
- Serpent (CBC)
- RC4
- IDEA (CBC)
- CAST (CBC)

Required authentication algorithms:

- DSA (SHA1 or SHA2)
- RSA (SHA1 or SHA2)

Optional authentication algorithms:

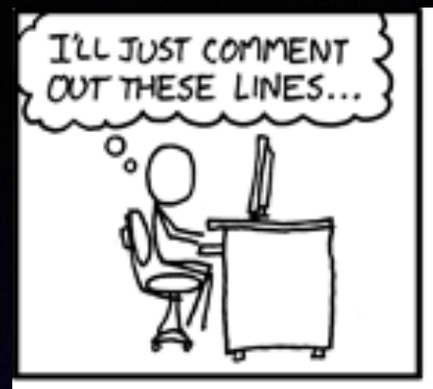
- Elliptic Curve DSA (ECDSA)
- Edwards Curve DSA (EdDSA or Ed25519)
- and others…

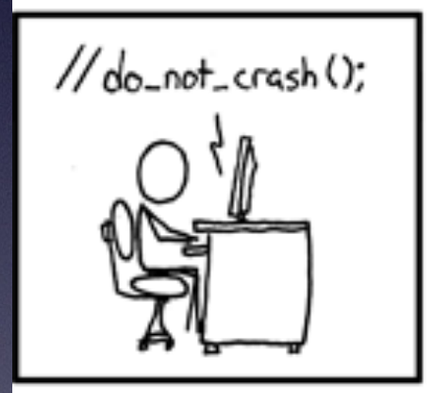
How does SSH Help?

- 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).

The Debian Fiasco

- Error in Debian OpenSSL discovered May 2008; present since 2006.
- Code change eliminated entropy for PRNG resulting in “weak” keys.
- Weak keys affected SSH, OpenVPN, DNSSEC, and key material in X.509 certificates.





IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:

AFFECTED SYSTEM	SECURITY PROBLEM
FEDORA CORE	VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)	GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO	VULNERABLE TO FLATTERY
OLPC OS	VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE	GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU	TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES

- Valgrind flagged use of uninitialized memory by a line of code in the OpenSSL PRNG.

- Debian maintainers commented-out the line, but the purpose of the code was to introduce random data into PRNG buffer.

- Maintainers *did* discuss their intentions on openssl-dev mailing list.

***Bottom Line**:* only 2^{15} possible SSH keys

“Fiasco” References

- Debian Security Advisory, 13 May 2008, http://www.debian.org/security/2008/dsa-1571
- Bruce Schneier, Random Number Bug in Debian Linux, https://www.schneier.com/blog/archives/2008/05/random_number_b.html
- Steinar Gunderson, Some maths , http://plog.sesse.net/blog/tech/2008-05-14-17-21_some_maths.html
- Ben Laurie, Debian and OpenSSL: The Aftermath, http://www.links.org/?p=328#comment-177420
- Openssl-dev discussion, http://marc.info/?t=114651088900003&r=1&w=2
- XKCD, security_holes.png, http://imgs.xkcd.com/comics/security_holes.png

Next time: Network Authentication