OpenSSH

ASBL CSRRT-LU (Computer Security Research and Response Team Luxembourg)
http://www.csrrt.org/

24th February 2006

SSH - History

- 1995 Tatu Ylonen releases ssh-1.0.0 (Forms SSH Communications Security)
- 1997 IETF secsh working group formed - Work on SSH protocol v2 begun
- 1999 OpenSSH project started, based on open-source ssh-1.x code
- 2000 SSH protocol v2 support added to OpenSSH
- 2002 SSH support added to Solaris 9 (OpenSSH derived)
- SSH protocol v2 nears release as IETF RFC
- 2006 RFC released for SSH protocol v2 as secsh

SSH Operation and Protocol

- Runs on TCP port 22, initiated by client
- Client and server exchange banners at connect time:
  - SSH-1.5-SoftwareName – SSH protocol v.1
  - SSH-2.0-SoftwareName – SSH protocol v.2
  - SSH-1.99-SoftwareName – both protocols
- SoftwareName is the implementation name and version
- Usually used for backwards (bug) compatibility
- Server always has a public/private key pair
- Public key is sent during connection setup
- Server's public key is cached by client to detect MITM

SSH Protocol v1

- SSH protocol v.1 is the original version released in the free ssh-1.x code by Tatu Ylonen
- Revised between 1995 and 1997
- Final version of the protocol is officially "1.5"
- Never standardised
- Some desire to have it published as an informational RFC
- Monolithic protocol

SSH Protocol v2

- Unlike the monolithic v1 protocol SSH v2 is several protocols
- Transport protocol
  - Underlying protocol
  - Handles encryption, compression, integrity
  - Provides "services" based on text strings
- User Authentication protocol
  - Responsible for authentication of user to server
  - Supports various authentication methods
  - Password, Public key, Challenge-response, Host based
- Connection protocol
  - Interactive logins, Command execution, Port forwarding, X11 forwarding

SSH Protocol v2 - packet format

- 4 bytes Packet length
- 1 byte Padding length
- ... Payload
- paddinglen Random padding
- ... MAC[seqno, packet (sans MAC)]

SSH Protocol v2 - packet format

- Multiple MAC algorithms supported
  - hmac-md5, hmac-sha1, truncated MACs, none
- Payload may optionally be compressed prior to MAC
- Packets are optionally encrypted with a symmetric cipher
  - 3-des-cbc (MUST)
  - blowfish-cbc (RECOMMENDED)
  - twofish-cbc, aes-cbc, serpent-cbc (OPTIONAL)
  - arcfour, idea-cbc, cast128-cbc (OPTIONAL)

SSH Protocol v2 - protocol start

- Server and client exchange banners
- Client and server both send MSGKEXINIT packet
  - Random nonce
  - Supported/Allowed algorithms to use for key exchange
  - Supported/Allowed server host key formats
  - Supported/Allowed symmetric algorithms (both ways)
  - Supported/Allowed MAC algorithms
  - Supported/Allowed compression algorithms
  - Supported/Allowed languages
  - Flag indicating "KEX guess"
- The Supported/Allowed lists are comma-separated strings
  - E.g. "aes128-cbc, 3des-cbc, arcfour"

SSH Protocol v2 - protocol exchange

- Upon receipt of KEXINIT packet, both client and server
  - Calculate intersection of supported/allowed ciphers, etc
  - Run selected key exchange algorithm
  - Usually Diffie Hellman
- D-H key exchange
- Secure way for two parties to derive a shared key
  - Safe against eavesdroppers
  - Perfect Forward Secrecy
- Exchange is authenticated with hash of
  - Client / server version strings
  - Payloads of client / server KEXINIT packets
  - Server host key
  - Intermediate D-H exchange values
  - Derived shared secret

SSH Protocol v2 - protocol exchange

- Output from key exchange is two keys:
    - Server to client
    - Client to server
- The client/server may have different options. e.g.
    - client may send data encrypted with 3des-cbc, no compression
    - server may send data encrypted with arcfour, zlib compression
- MSGKEXINIT may be sent at any time during session
- "Re-keying" may change compression, ciphers as well as keys
- Once keyed, protocol moves on to authentication

SSH Protocol / OpenSSH TP

- Analyze a SSH communication stream with a simple password authentication (using Ethereal + debug mode of OpenSSH)
- Using the OpenVPN setup of the previous TP, authenticate a user (with an OpenSSH RSA public key) using OpenSSH to your server inside a VPN connection
  - Be aware that the keys scheme and authentication used in OpenVPN is different in OpenSSH...

Q and A

- Thanks for listening.
- http://www.csrrt.org.lu/
- adulau@foo.be