Protocoale de nivel aplicaţie:

SSH, SFTP

SSH

- Protocol de acces securizat între 2 maşini despărţite de o reţea necunoscută.
- Înlocuieşte telnet, rsh, rlogin; poate înlocui ftp.
- Foloseşte criptare.
- SSH nu este un shell precum sh, bash (Unix Bourne shell), csh (C shell)

Proprietăţi

- Transmisia e sigură.
- Transmisia poate fi comprimată.
- Parola de login nu e strict necesară
- remote login = remote command shell - de aici numele

Probleme Telnet

- Autentificarea şi datele sînt transmise în clar.
- Orice host de pe traseu poate intercepta comunicaţia.

Istoria SSH

- Creat de Tatu Ylönen în iulie 1995, student la Helsinki University of Technology (HUT)
- SSH1 - complet free
- Fondează SSH Communications Security, Ltd
- unele restricţii
- versiunea SSH 2 corectează probleme de securitate
- Open SSH - pentru a elimina restricţiile; bazat iniţial pe SSH 1 apoi pe SSH 2

Funcţii

- Command Shell securizat
- Port Forwarding, inclusiv X11 forwarding
- Transfer de fişiere securizat.

Command Shell securizat

- Acces la un terminal la distanţă.
- Orice se poate face într-un terminal local se poate face prin ssh.

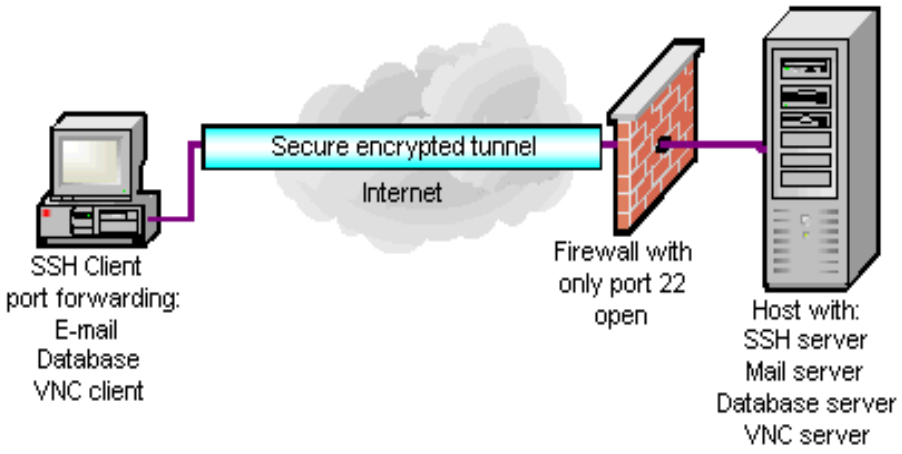
Port Forwarding

- Foarte puternic şi versatil.
- poate securiza aplicaţii TCP/IP diverse cum ar fi e-mail, contactarea unor baze de date la distanţă, etc.
- poate securiza transferul de date pentru aplicaţii TCP/IP oarecare

- `ssh -L localport:tohost:toport remotehost`
- Exemplu: ascultăm portul 5110 local; pachetele ajunse aici sînt trimise către mailserver, port 110:

```
ssh -L 5110:mailserver:110 mailserver
```

Port Forwarding - ex. 2



Transfer de fişiere securizat

- Secure File Transfer Protocol (SFTP) este un subsistem al Secure Shell protocol.
- Nivel separat de protocol, peste protocolul SSH, care se ocupă de transferul de fişiere.

SFTP

- SFTP criptează atît sesiunea de autentificare, cît şi fluxul de date.
- Utilizează acelaşi port ca şi SSH - nu trebuie deschis un port suplimentar în router/firewall.
- elimină necesitatea NAT şi problemele aferente

- SFTP: interactiv; scp: neinteractiv, *command-line*
- Sintaxa scp:

```
scp -P port [[user@]host:]fis_sursa
[[user@]host:]fis_destinatie
```

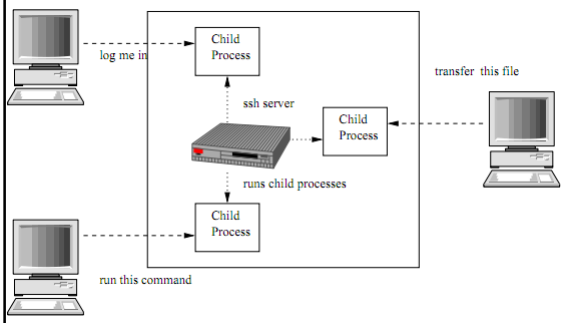
exemplu:

```
scp -P 2222 archive.tgz ms@remote.host.com:~ms
```

Componente Secure Shell

- SSHD Server: un server (*daemon*) care acceptă conexiunile clienţilor.
- Clienţi: un program care se conectează la serverele SSH şi face cereri pentru servicii
- Sesiune: o conexiune în curs, între client şi server, care începe după autentificarea clientului la server

Arhitectura SSH



- Utilizatorul, prin clientul SSH, se conectează la portul TCP/22 al serverului
- Dacă autentiticarea are succes, SSHD de pe srver crează (“forks”) un proces (“child process”) SSHD care se va ocupa de transferul SSH între cele 2 părţi.
- procesul-copil SSHD crează (“forks”) comenzile lansate de către clientul SSH şi care sînt primite la server; tipic se crează un proces bash pentru o sesiune interactivă.
- procesul-copil SSHD criptează schimbul de mesaje
- clientul SSH decriptează informaţia şi o trimite către aplicaţia utilizator

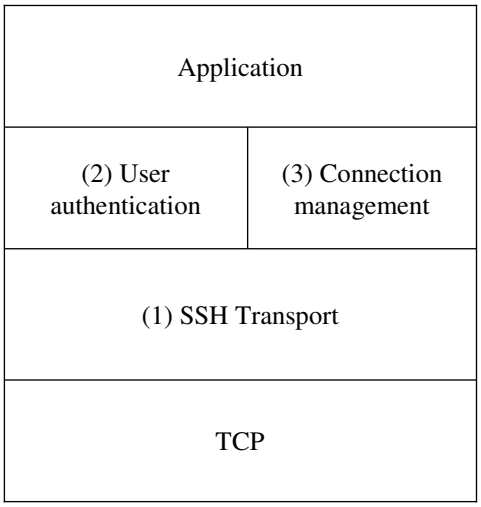
Aspecte de securitate furnizate de SSH

- Autentificarea utilizatorilor
- Autentificarea hostului
- Criptarea datelor
- Integritatea datelor

Versiuni SSH - comparaţie

SSH version 1	SSH version 2
monolithic (integrated) design	separation of the authentication, connection and transport functions into layers
integrity via CRC32 (not secure)	integrity via HMAC (hash encryption)
one and only one channel per session	unrestricted number of channels per session
negotiation using only a symmetric cipher in the channel, session identification with unique key on both sides	more detailed negotiation (symmetric cipher, public keys, compression, ...), and a separate session key, compression and integrity on both sides
only RSA for the public key algorithm	RSA and DSA for the public key algorithm
session key transmitted by the client side	session key negotiated thru the Diffie-Hellman protocol
session key valid for the entire session	renewable session key

Stiva SSH: 3 protocoale (de la SSH2)



(1) Protocolul de transport

Sursa: rfc4253

Asigură:

- Criptarea conexiunii
- Autentificarea la nivel de host (nu de user)
- Integritatea datelor

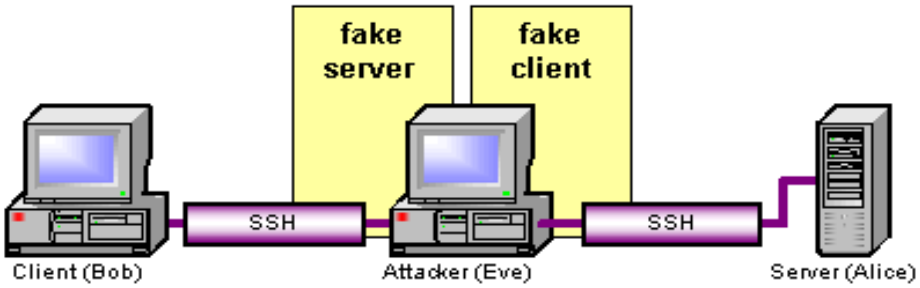
Se negociază între client şi server:

- Metoda schimbului de chei
- Algoritmul de chei publice
- Algoritmul de criptare simetrică
- Algoritmul MAC
- Algoritmul de hash

Autentificarea hostului

- Un server foloseşte un “host key” generat la prima conexiune cu clientul
- cheile sînt persistente (nu se schimbă des) şi sînt asimetrice
- foloseşte un sistem similar cu chei publice - cheia publică a clientului este trimisă pe server la prima conexiune
- serviciu suplimentare faţă de telnet, ftp etc
- schimbarea cheii este imediat detectată
- previne atacurile de tip *man-in-the-middle*

Man-in-the-middle



- autentificarea hostului (Alice) previne înlocuirea sa cu un alt host (Eve)

Integritatea datelor

- Confirmarea faptului că datele care ajung la server sînt chiar cele transmise de client
- Criptarea datelor nu e sinonimă cu integritatea; un atac de tip *insertion and replay* constă în inserarea de către o terţă persoană, în fluxul de date, a unor date suplimentare, criptate
- SSH1 foloseşte un CRC-32
- CRC-32 este sigur împotriva alterării accidentale a datelor şi nu împotriva atacurilor
- SSH2 foloseşte algoritmi MAC (*Message Authentication Code*) pentru a proteja integritatea

Criptarea datelor

- protejează la atacurile de tip “sniffing” sau “*eavesdropping*”
- funcţie: privacy
- SSH2 permite negocierea tipului de cifrare care va fi utilizat:
  - la conectare, serverul trimite clientului o listă cu tipurile de cifru suportate
  - clientul alege primul tip de pe lista serverului pe care îl suportă

(1) continuare

Negocierea alg. de criptare:

```
3des-cbc        REQUIRED                three-key 3DES in CBC mode
blowfish-cbc    RECOMMENDED      Blowfish in CBC mode
     ...
aes256-cbc      OPTIONAL                AES (Rijndael), CBC, 256-bit key
```

Negocierea alg. de autentificare MAC:

```
hmac-sha1       REQUIRED        HMAC-SHA1 (digest length = key
                                      length = 20)
hmac-sha1-96    RECOMMENDED     first 96 bits of HMAC-SHA1 (digest
                                      length = 12, key length = 20)
hmac-md5        OPTIONAL        HMAC-MD5 (digest length = key
                                      length = 16)
hmac-md5-96     OPTIONAL        first 96 bits of HMAC-MD5 (digest
                                      length = 12, key length = 16)
none            OPTIONAL        no MAC; NOT RECOMMENDED
```

(2) Autentificarea userului

- rhosts (doar ssh v 1, obsolete)
  - Bazat pe fişiere .rhosts din /home/user
- Parole
  - Protocolul trimite username/password
  - parola **nu** se transmite în clar
  - dezavantaj: utilizatorul trebuie să introducă parola de fiecare dată, potenţial pentru interceptarea ei prin alte mijloace
- Chei publice
  - Vezi demo
  - Metoda cea mai sigură - nu se cere parola
  - Se foloseşte o pereche de chei, generate automat de către program, fiecare între 1024 şi 2048 biţi lungime
  - după generare, doar cheia publică a userului se trimite pe serverul pe care se doreşte conectarea prin ssh
- Kerberos
- Host keys for server authentication

(3) Managementul conexiunii

Administrează un set de fluxuri TCP

- Fiecare are:
  - Local and remote ID
  - Window size
  - Max packet size (bulk vs interactive)
- Se ocupă de:
  - Pseudo-terminals
  - Remote commands
  - X11
  - Flow control

Demo - ssh fără parolă

client$ **mkdir ~/.ssh**

client$ **chmod 700 ~/.ssh**

client$ **ssh-keygen -q -f ~/.ssh/id_rsa -t rsa**

Enter passphrase (empty for no passphrase): …

Enter same passphrase again: …

OBS: s-au generat cele 2 chei (publică/privată)

client$ **scp** ~/.ssh/id_rsa.**pub** server.example.org:

OBS: cheia publică (id_rsa.**pub**) se copiază pe server, cheia privată (id_rsa) rămîne pe client !

server$ mkdir ~/.ssh

server$ chmod 700 ~/.ssh

server$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys

server$ chmod 600 ~/.ssh/authorized_keys