

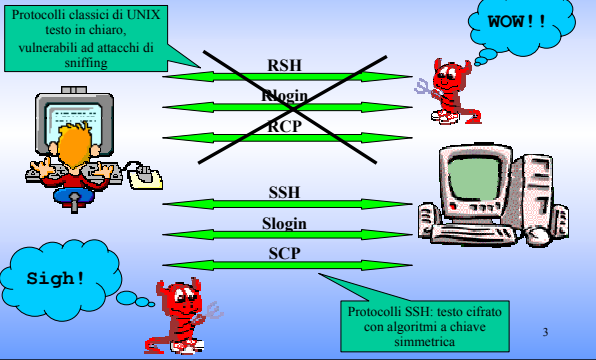


Fabio Petagna matr. 56/01051 fpetagna@tiscalinet.it  
 Vittorio Fucella matr. 56/00981 vfucel@libero.it

## Motivazioni

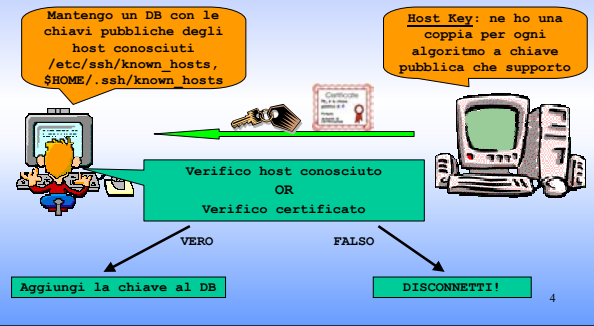
- Protocollo per il login remoto sicuro e altri servizi di rete sicuri su un canale insicuro
- Nato per rimpiazzare i comandi Berkeley r\* (**rsh**, **rlogin**, **rcp**) con le rispettive versioni sicure (**ssh**, **slogin**, **scp**)
- Fornisce:
  - Una infrastruttura per connessioni crittografate
  - Autenticazione forte tra host e host e tra utente e host
  - Possibilità di creare un canale di comunicazione sicuro attraverso il quale veicolare qualsiasi connessione TCP/IP
- Risolve alcuni noti problemi di sicurezza dei protocolli TCP/IP come:
  - L'IP spoofing (falsificazione dell'indirizzo IP del mittente)
  - Il DNS spoofing (falsificazione delle informazioni contenute nel DNS)
  - Routing spoofing (falsificazione delle rotte intraprese dai pacchetti e instradamento su percorsi diversi)

## Motivazioni Connessioni crittografate



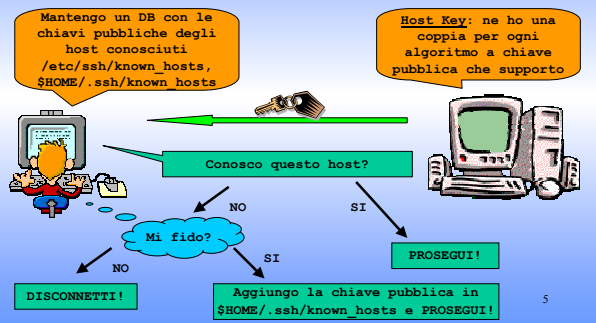
## Motivazioni Autenticazione forte

Esempio di verifica dell'identità del server: chiave+certificati



## Motivazioni Autenticazione forte

Esempio di verifica dell'identità del server: chiave



## Motivazioni Autenticazione client -> server

- **PROBLEMA:** Metodo *classico* di autenticazione *rhosts* di UNIX vulnerabile ad attacchi di *spoofing*

Autenticazione RHOSTS:  
 I files /etc/hosts.equiv e \$HOME/.ssh/known\_hosts Contengono IP o nomi host dei client ammessi al login senza password

- Un noto attacco: **IP Spoofing**

- Azioni per un tipico attacco IP SPOOFING da parte di un hacker (H):
1. Scegliere l'host target (B)
  2. Scoprire l'indirizzo IP di un host fidato (A)
  3. Disabilitare A (con TCP SYN flooding)
  4. Utilizzare l'indirizzo IP di B per ottenere una shell su B cercando di indovinare i Segue Number
    - N.B.: H potrebbe catturare i pacchetti in uscita da B falsificando la rotta di ritorno (ROUTING SPOOFING)
  5. Aggiungere il proprio indirizzo in /etc/hosts.equiv su B

## Motivazioni

### Autenticazione client -> server

- Un'altro attacco: **DNS Spoofing**

Azioni per un tipico attacco DNS SPOOFING da parte di un hacker (H):

1. Scegliere l'host target (B)
2. Scoprire il nome host di un host fidato (A)
3. Falsificare il DNS di B facendo in modo il nome host di A venga risolto con l'indirizzo IP di H
4. Stabilire una connessione con B, fingendosi A

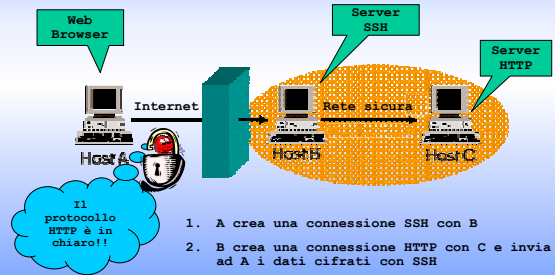
- **SOLUZIONE:** SSH aggiunge un controllo sull'host più rigoroso:  
Autenticazione *rhosts* + *chiave pubblica*:
  - il client invia un pacchetto firmato con la chiave privata del proprio host
  - Il server verifica la firma con la chiave pubblica dell'host del client

7

## Motivazioni

### TCP port forwarding

- Una qualsiasi connessione TCP può essere resa sicura:



8

## Credits e Versioni

- 2 Versioni non compatibili tra loro
  - SSH1
  - SSH2
- Gestito da 2 società finlandesi:
  - **SSH Communications Security** (<http://www.ssh.org>) ha sviluppato i protocolli SSH1 e SSH2 e mantiene le distribuzioni principali di SSH
  - **Data Fellows** (<http://www.datafellows.com>) ha acquistato dalla prima la licenza di vendere e supportare SSH

9

## Licenze d'uso

- Data Fellows ha imposto i limiti di utilizzo per il software:
  - SSH1 può essere utilizzato liberamente a patto che non se ne ricavi guadagno
  - SSH2 può essere utilizzato liberamente solo privatamente o in ambito educational
- Protocolli SSH1 e SSH2 pubblicati come Internet Drafts
- OpenSSH (<http://www.openssh.com>)
  - Open Source
  - nato per far parte del sistema operativo OpenBSD
  - utilizza solo algoritmi di crittografia senza restrizioni di utilizzo

10

## Caratteristiche

### Protocollo estendibile



- Controllo dell'evoluzione del protocollo
  - Internet Draft con scadenza semestrale
- Insieme minimale di algoritmi supportato da tutte le implementazioni
- Algoritmi identificati con nomi riassuntivi
- Possibilità di definire algoritmi personalizzati
  - Nomi tipo DNS. Esempio: ourcipher-[cbc@ssh.com](mailto:cbc@ssh.com)

11

## Caratteristiche

### Politiche Locali

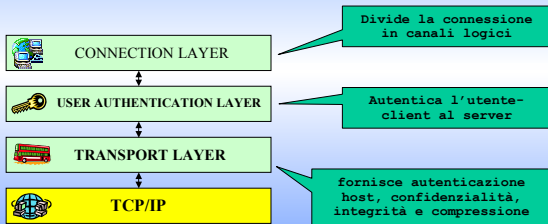


- Piena negoziazione degli algoritmi di crittografia, integrità, scambio delle chiavi e compressione
- Per ogni categoria le politiche locali specificano un algoritmo preferito
- A seconda dell'utente e della sua locazione possono essere richiesti differenti modalità di autenticazione o addirittura autenticazioni multiple
- E' possibile limitare od estendere l'insieme delle operazioni consentite a determinati utenti

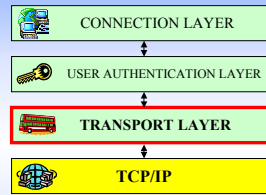
12

# Architettura

Tre componenti maggiori:



# Transport Layer Protocol

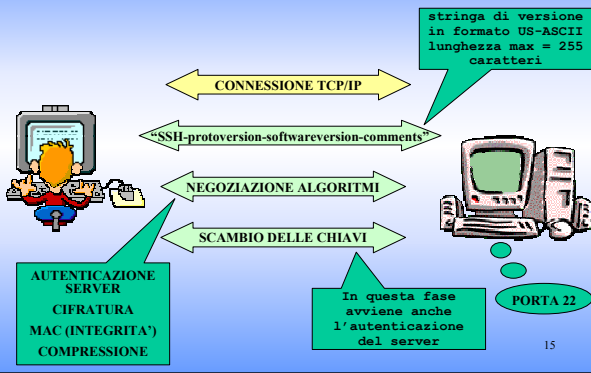


Esegue

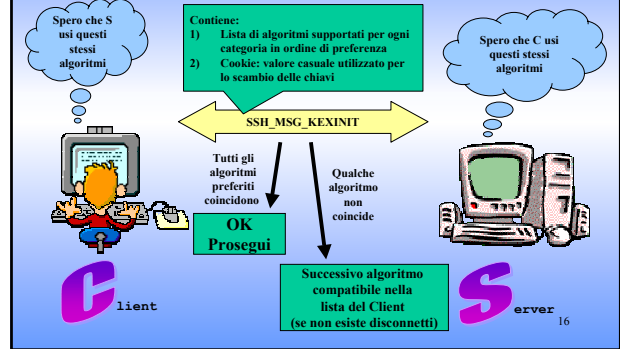
- Autenticazione del server
- Scambio di chiavi
- Cifratura
- Protezione dell'integrità
- Calcolo di un unico *session\_id*, utilizzato dai protocolli di livello superiore

- Progettato per essere utilizzato su un livello di trasporto affidabile
- Autenticazione dell'utente non implementata a questo livello, bensì al protocollo al livello più alto
- Progettato per essere semplice e flessibile: sono necessari in media solo 2 (max 3) round trip per una operazione

# Transport Layer Protocol



# Transport Layer Protocol Negoziazione algoritmi



# Transport Layer Protocol Negoziazione Algoritmi

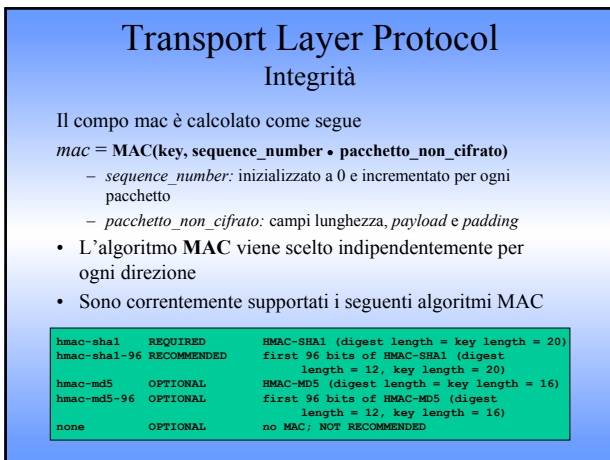
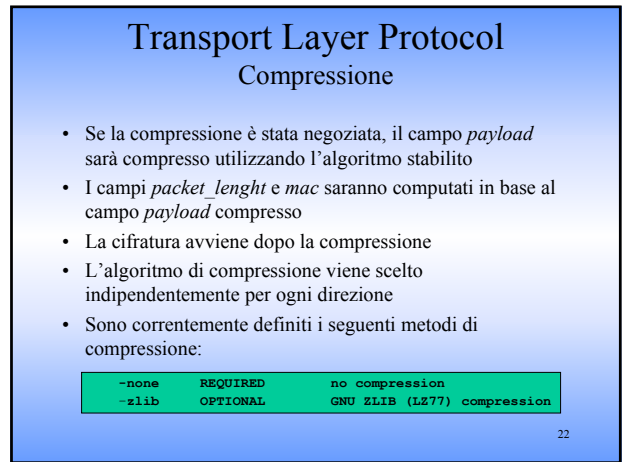
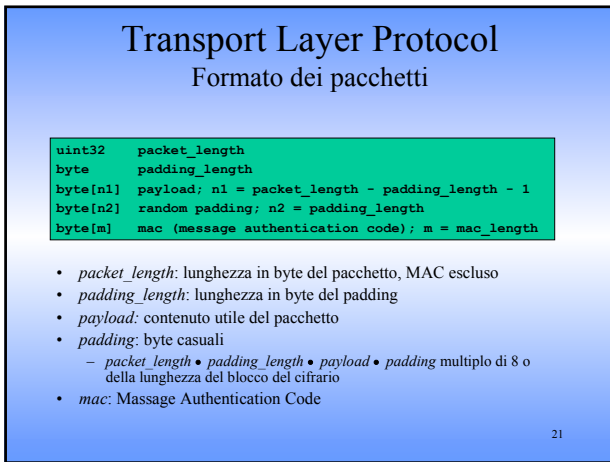
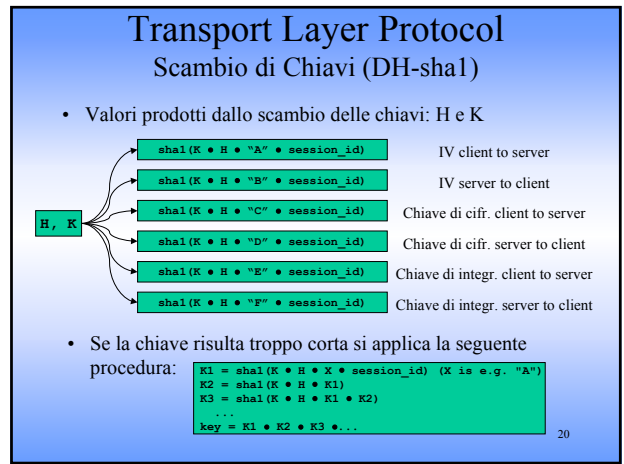
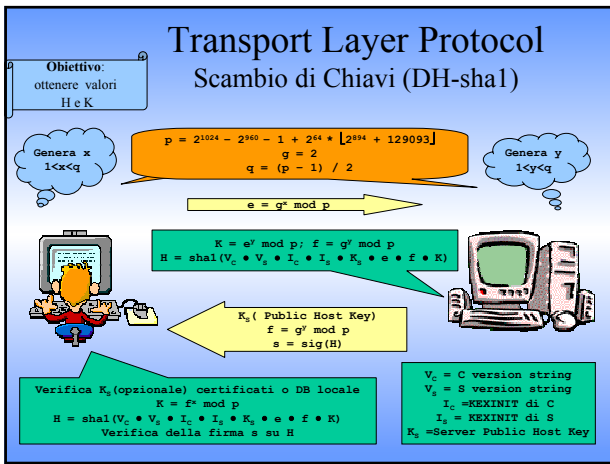
Pacchetto `SSH_MSG_KEXINIT`

```

byte      SSH_MSG_KEXINIT
byte[16]  cookie (random bytes)
string    kex_algorithms
string    server_host_key_algorithms
string    encryption_algorithms_client_to_server
string    encryption_algorithms_server_to_client
string    mac_algorithms_client_to_server
string    mac_algorithms_server_to_client
string    compression_algorithms_client_to_server
string    compression_algorithms_server_to_client
string    languages_client_to_server
string    languages_server_to_client
boolean   first_kex_packet_follows
uint32    0 (reserved for future extension)
    
```

# Transport Layer Protocol Scambio di Chiavi

- Obiettivo: ottenere dei valori H e K, che verranno utilizzati per il calcolo delle chiavi
- Il valore H è usato anche come identificatore di sessione
- L'identificatore di sessione (*session\_id*)
  - È utilizzato nel calcolo delle chiavi
  - Una volta computato non cambia, anche se le chiavi sono rinegoziate in seguito
- DH-sha1 è l'unico algoritmo richiesto
- La verifica dell'identità del server avviene in questa fase



# Transport Layer Protocol Cifratura

- Sono correntemente supportati i seguenti cifrari

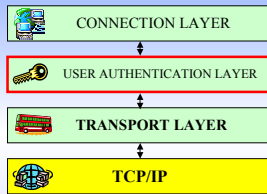
3des-cbc	REQUIRED	three-key 3DES in CBC mode
blowfish-cbc	RECOMMENDED	Blowfish in CBC mode
twofish256-cbc	OPTIONAL	Twofish in CBC mode, with 256-bit key
twofish-cbc	OPTIONAL	alias for "twofish256-cbc" (this historical reasons)
twofish192-cbc	OPTIONAL	Twofish with 192-bit key
twofish128-cbc	RECOMMENDED	Twofish with 128-bit key
aes256-cbc	OPTIONAL	AES (Rijndael) in CBC mode, with 256-bit key
aes192-cbc	OPTIONAL	AES with 192-bit key
aes128-cbc	RECOMMENDED	AES with 128-bit key
serpent256-cbc	OPTIONAL	Serpent in CBC mode, with 256-bit key
serpent192-cbc	OPTIONAL	Serpent with 192-bit key
serpent128-cbc	OPTIONAL	Serpent with 128-bit key
arcfour	OPTIONAL	the ARCFour stream cipher
idea-cbc	OPTIONAL	IDEA in CBC mode
cast128-cbc	OPTIONAL	CAST-128 in CBC mode
none	OPTIONAL	no encryption; NOT RECOMMENDED

# Transport Layer Protocol Algoritmi a chiave pubblica

- Il protocollo è stato progettato per operare con quasi tutti i formati di chiave, codifiche ed algoritmi
- Sono supportati i seguenti formati di chiavi pubbliche e di certificati

ssh-dss	REQUIRED	sign	Simple DSS
ssh-rsa	RECOMMENDED	sign	Simple RSA
x509v3-sign-rsa	RECOMMENDED	sign	X.509 certificates (RSA key)
x509v3-sign-dss	RECOMMENDED	sign	X.509 certificates (DSS key)
spki-sign-rsa	OPTIONAL	sign	SPKI certificates (RSA key)
spki-sign-dss	OPTIONAL	sign	SPKI certificates (DSS key)
pgp-sign-rsa	OPTIONAL	sign	OpenPGP certificates (RSA key)
pgp-sign-dss	OPTIONAL	sign	OpenPGP certificates (DSS key)

# Authentication Protocol



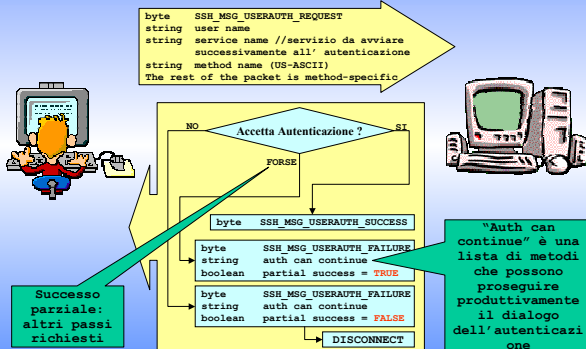
- Autentica l'utente-client al server
- All'avvio del protocollo, lo strato di trasporto gli fornisce l'identificatore di sessione (Il valore H del primo pacchetto di scambio di chiavi)

# Authentication Protocol

- Si assume che il protocollo sottostante fornisca protezione dell'integrità e confidenzialità
- 3 metodi di autenticazione:
  - Chiave Pubblica
  - Password
  - Host Based
- Il server guida l'autenticazione:



# Authentication Protocol Schema generale

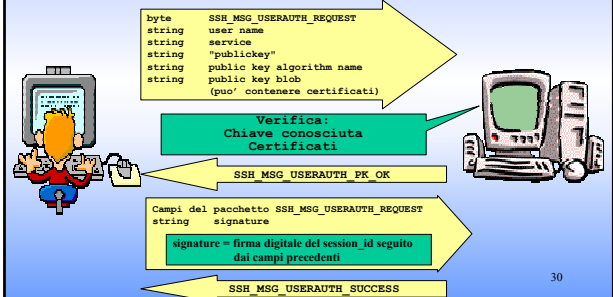


"Auth can continue" è una lista di metodi che possono proseguire produttivamente il dialogo dell'autenticazione

Successo parziale: altri passi richiesti

# Authentication Protocol Autenticazione a Chiave Pubblica

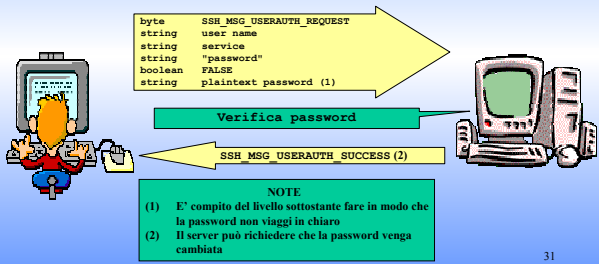
- Questo metodo deve essere supportato da ogni implementazione
- Autenticazione basata su algoritmi a chiave pubblica



# Authentication Protocol

## Autenticazione con Password

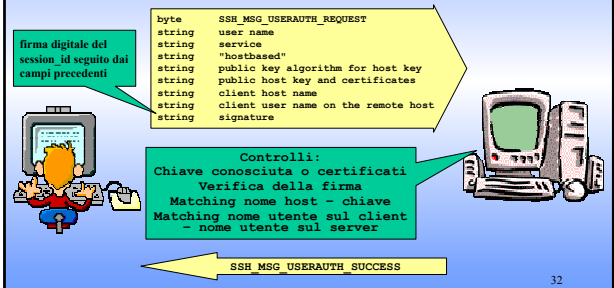
- Tutte le implementazioni dovrebbero supportare questo metodo dato che non è obbligatorio il possesso di una chiave pubblica



# Authentication Protocol

## Autenticazione Host Based

- Questo tipo di autenticazione è opzionale (è simile a Rhosts di UNIX)
- Si basa su nome host, sulla host key del client e sul nome utente



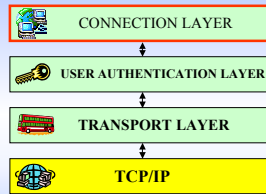
# Authentication Protocol

## Considerazioni sulla sicurezza

- Si assume il protocollo di trasporto sicuro abbia già
  - Autenticata la macchina server
  - Stabilito un canale di comunicazione cifrato
  - Computato un identificatore di sessione
- Il server può andare in uno stato di *sleep* dopo ripetute autenticazioni fallite
- Se il livello di trasporto non utilizza il MAC, non deve essere possibile cambiare la password per evitare attacchi del tipo *denial of service*
- La politica locale decide quali metodi accettare per ogni utente
- I messaggi di debug possono essere disabilitati per evitare che forniscano troppe informazioni sull'host



# Connection Protocol



Fornisce:

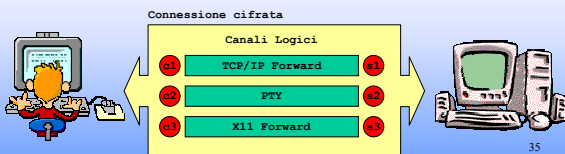
- Sessioni interattive di login;
- Esecuzione remota di comandi;
- Inoltro di connessioni TCP/IP;
- Inoltro di connessioni X11

Divide la connessione in canali logici; tutti questi canali sono *multiplexati* in un singolo tunnel cifrato

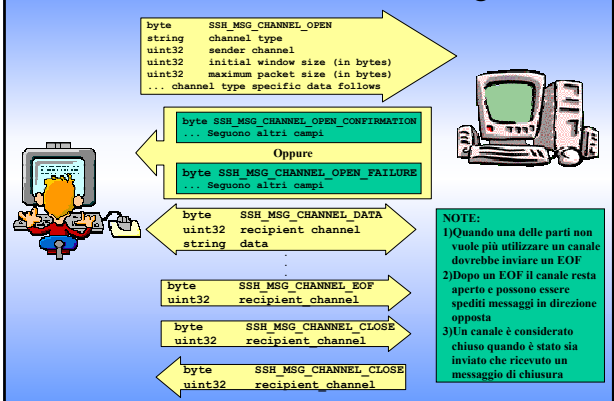
# Connection Protocol

## Meccanismo dei canali

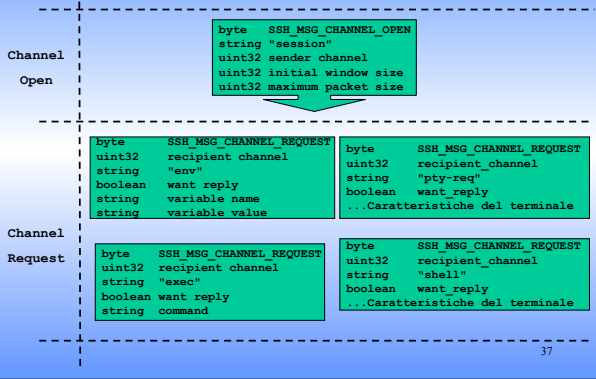
- Tutte gli pseudo-terminali, connessioni inoltrate, ecc. sono canali
- I canali sono identificati da numeri(ad entrambe le parti), che possono essere differenti tra client e server
- I canali hanno un meccanismo di controllo del flusso



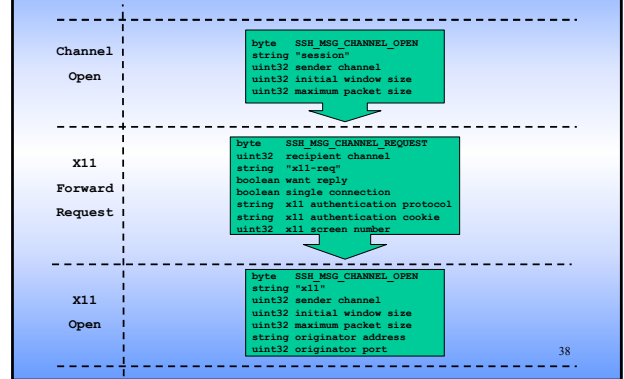
# Meccanismo dei canali: Schema generale



## Connection Protocol Sessioni Interattive

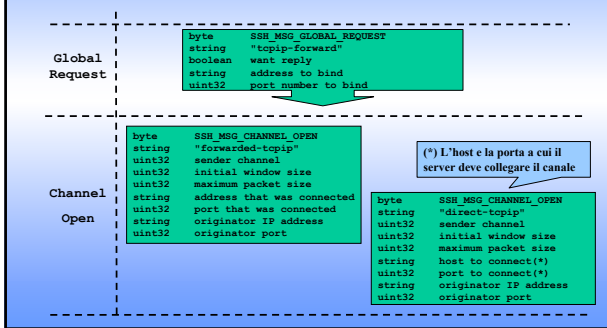


## Connection Protocol Sessioni Interattive: X11



## Connection Protocol TCP/IP Port Forwarding

- permette di creare un canale di comunicazione sicuro attraverso il quale veicolare qualsiasi connessione TCP



## Connection Protocol Considerazioni sulla sicurezza

- Si assume che questo protocollo giri al di sopra di un protocollo di trasporto sicuro che ha già
  - Autenticato la macchina server
  - Stabilito un canale di comunicazione cifrato
  - Computato un identificatore di sessione
  - Autenticato l'utente
- Migliorata la sicurezza per connessioni al server X11
- Il port forwarding può potenzialmente permettere ad un intruso di scavalcare sistemi di sicurezza come i firewall; tuttavia questo poteva già essere fatto in altri modi
- In quanto cifrato, il traffico non può essere esaminato da un firewall



40

## Considerazioni sulla sicurezza

- Algoritmi di crittografia e autenticazione ben conosciuti e testati
- Chiavi di taglia sufficiente a garantire protezione a lungo termine
- Algoritmi negoziati: in caso di rottura di uno di essi, è possibile utilizzarne altri senza modifiche al protocollo

41



- Suite di protocolli gratuita
- Caratteristiche principali
  - Progetto Open Source
  - Licenza Gratuita
  - Crittografia forte (3DES, Blowfish)
  - X11 Forwarding (cifratura del traffico X11)
  - Inoltro di porte (canali criptati per altri protocolli)
  - Autenticazione Forte (Chiave Pubblica, One-Time Password e Kerberos)
  - Interoperabilità (Conformità con SSH 1.3, 1.5, e gli Standard del protocollo 2.0)
  - Supporto per client e server SFTP nei protocolli SSH1 e SSH2
  - Compressione Dati

42

## Installazione: preliminari

- È possibile effettuare il download dal sito [www.openssh.com](http://www.openssh.com)
- Sono disponibili due formati: RPM e TGZ
- Per il formato RPM si ha bisogno dei seguenti file:
  - openssl-versione.rpm
  - openssl-clients- versione.rpm
  - openssl-server- versione.rpm
- Per il formato TGZ si ha bisogno del seguente file:
  - openssl-versione.tar.gz
- Sono inoltre necessari per l'installazione i seguenti pacchetti
  - Zlib
  - OpenSSL

43

## Installazione di Zlib

- Libreria di compressione dati Open Source e Patent free
- Sito <http://www.freeware.com/pub/infozip/zlib/>
- Passi per l'installazione:
  - Formato RPM

```
rpm -ivh zlib-versione.rpm
```

- Formato TGZ

```
tar xzvf zlib-versione.tar.gz
cd zlib-versione
./configure
make
su -c "make install"
```

44

## Installazione di OpenSSL

- Libreria Open Source che implementa:
  - Secure Socket Layer (SSL)
  - Transport Layer Security (TLS)
  - Libreria contenente algoritmi a crittografia forte
- Sito [www.openssl.org](http://www.openssl.org)
- Passi per l'installazione:

- Formato RPM

```
rpm -ivh openssl-versione.rpm
```

- Formato TGZ

```
tar xzvf openssl-versione.tar.gz
cd openssl-versione
./configure
make
su -c "make install"
```

45

## Installazione OpenSSH

- Con la versione TGZ:

```
tar xzvf openssl-versione.tar.gz
cd openssl-versione
./configure --prefix=/usr
--sysconfdir=/etc/ssh
make
su -c "make install"
ssh-keygen -b 1024 -f /etc/ssh/ssh_host_key -N
```

/usr/(bin,sbin,man)  
Per i file binari

/etc/ssh  
File di Configurazione

- Con la versione RPM

```
rpm -i openssl-1.2.2-1.i386.rpm
rpm -i openssl-server-1.2.2-1.i386.rpm
rpm -i openssl-clients-1.2.2-1.i386.rpm
```

46

## Note all'installazione via TGZ

- `su -c "make host-key"`  
consente di creare sia le chiavi host RSA sia quelle DSA
- Cartella **contrib**: *script init*  
Si deve copiare lo *script di startup* all'interno della cartella **init.d** la cui posizione varia a seconda della distribuzione di Linux
- Avviare il *demone SSH*: `/usr/sbin/sshd`
- `telnet 127.0.0.1 porta 22`. Se il demone è in esecuzione si dovrebbe ottenere quanto segue:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'
SSH-1.99-OpenSSH 2.3.pl
```

47

## File Interessati

- `$HOME/.ssh/known_hosts`  
Registrano le chiavi pubbliche degli host in cui l'utente ha effettuato il login
- `$HOME/.ssh/identity`, `$HOME/.ssh/id_dsa`  
Contengono le chiavi private RSA SSH1 e DSA dell'utente rispettivamente
- `$HOME/.ssh/identity.pub`, `$HOME/.ssh/id_dsa.pub`  
Contengono le chiavi pubbliche RSA SSH1 e DSA dell'utente rispettivamente
- `$HOME/.ssh/config`  
File di configurazione per l'utente usato dal client SSH
- `$HOME/.ssh/authorized_keys`  
Lista le chiavi pubbliche RSA SSH1 degli utenti autorizzati ad effettuare il login

48

## File Interessati

- `$HOME/.ssh/authorized_keys2`  
Lista le chiavi pubbliche RSA/DSA SSH2 degli utenti autorizzati ad effettuare il login
- `/etc/ssh/ssh_known_hosts`,  
`/etc/ssh/ssh_known_hosts2`  
Lista delle host key per gli host conosciuti dal server. Il primo contiene chiavi pubbliche RSA SSH1 mentre il secondo anche chiavi pubbliche DSA SSH2
- `/etc/ssh/ssh_config`  
File di configurazione di default per i client

49

## File Interessati

- `/etc/ssh/sshd_config`  
File di configurazione per l'utente usato dal server SSH
- `/etc/ssh/ssh_host_key`,  
`/etc/ssh/ssh_host_dsa_key`,  
`/etc/ssh/ssh_host_rsa_key`  
Contengono la parte privata delle chiavi SSH1 RSA, SSH2 DSA e SSH2 RSA del server rispettivamente
- `/etc/ssh/ssh_host_key.pub`,  
`/etc/ssh/ssh_host_dsa_key.pub`,  
`/etc/ssh/ssh_host_rsa_key.pub`  
Contengono la parte pubblica delle chiavi SSH1, SSH2 DSA e SSH2 RSA del server rispettivamente

50

## `/etc/ssh/sshd_config`

- Principali direttive:

- `AllowHosts <modello>...`
- `DenyHosts <modello>...`
- `AllowUsers <modello>...`
- `DenyUsers <modello>...`

Permettono di definire uno o più modelli (attraverso l'uso dei caratteri jolly '\*' e '?') riferiti a nomi di host e di utenti a cui si intende permettere o impedire l'accesso

51

## `/etc/ssh/sshd_config`

- Principali direttive(continua):

- `HostKey <file>`  
Permette di indicare il file contenente la chiave privata del nodo, in alternativa a quello standard ('/etc/ssh/ssh\_host\_key')
- `LoginGraceTime <durata>`  
Permette di stabilire il tempo massimo concesso per completare la procedura di accesso. Il valore predefinito è di 600 secondi, pari a 10 minuti
- `PasswordAuthentication {yes|no}`  
Stabilisce se l'autenticazione attraverso la password è consentita oppure no. Il valore predefinito è 'yes', cosa che permette questo tipo di autenticazione

52

## `/etc/ssh/sshd_config`

- Principali direttive(continua):

- `PermitRootLogin {yes|no|nopwd}`  
Permette di abilitare o meno l'accesso da parte dell'utente 'root'. Il valore predefinito è 'yes'. 'nopwd' esclude solo la forma di autenticazione attraverso password
- `RSAAuthentication {yes|no}`  
Permette di abilitare o meno l'autenticazione RSA
- `StrictModes {yes|no}`  
Se attivato, fa in modo che 'sshd' verifichi la proprietà dei file di configurazione nelle directory personali degli utenti, rifiutando di considerare i file appartenenti a utenti «sbagliati». Il valore predefinito è 'yes'

53

## `/etc/ssh/sshd_config`: Esempio

```
# This is ssh server systemwide configuration file.
Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh/ssh_host_key
RandomSeed /etc/ssh/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
KillForwarding yes
KillDisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility AUTH
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# PidFile /var/run/sshd.pid
# AllowHosts * our.com friend.other.com
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny on
```

54

## /etc/ssh/ssh\_config

- Principali direttive:

- Host <modelli>**  
Permette di definire uno o più modelli (attraverso l'uso dei caratteri jolly '\*' e '?') riferiti a nomi di host, a cui sono riferite le direttive successive, fino alla prossima direttiva 'Host'
- IdentityFile <file>**  
Permette di indicare il file contenente la chiave privata dell'utente, in alternativa a quello standard
- User <utente>**  
Permette di indicare l'utente da utilizzare nella connessione remota. Ciò è particolarmente utile nella configurazione personalizzata, in cui si potrebbe specificare l'utente giusto per ogni nodo presso cui si ha accesso

55

## /etc/ssh/ssh\_config

- Principali direttive:

- Cipher {idea|des|3des|blowfish|arcfour|tss|none}**  
Permette di indicare il tipo di cifratura preferita, se ammissibile anche per il server. La cifratura predefinita è 'idea'; in alternativa viene usata '3des'. Con 'none' si intende di non volere alcun tipo di cifratura
- Compression {yes|no}**  
Se attivato, permette di utilizzare una comunicazione di dati compressa. Il valore predefinito è 'no'
- StrictHostKeyChecking {yes|no}**  
Se attivato, fa in modo le chiavi pubbliche dei server contattati non possano essere aggiunte automaticamente nell'elenco personale e che la connessione a nodi sconosciuti o irriconoscibili venga impedita. Il valore predefinito è 'no'

56

## /etc/ssh/ssh\_config: Esempio

```
# Site-wide defaults for various options
# Host *
# ForwardAgent yes
# ForwardX11 yes
# RhostsAuthentication yes
# RhostsRSAAuthentication yes
# RSAAuthentication yes
# TISAuthentication no
# PasswordAuthentication yes
# FallBackToRsh yes
# UseRsh no
# BatchMode no
# StrictHostKeyChecking no
# IdentityFile ~/.ssh/identity
# Port 22
# Cipher idea
# EscapeChar ~
```

57

## Comandi Principali

**ssh** [<opzioni>] <host>[<comando>] 'ssh' è in grado di instaurare una connessione per l'accesso presso un server in cui sia in funzione il demone 'sshd'

Alcune opzioni

- l <utente>** specifica il nominativo-utente remoto
- i <file-di-identificazione>** specifica il file con la chiave privata diverso da quello di default

**scp** [<opzioni>][<utente>@]<host>[:<origine file>]...[[<utente>@]<host>:]<destinazione file>

'scp' usa 'ssh' per la copia di file tra elaboratori

Alcune opzioni:

- p** Fa in modo che gli attributi originali dei file vengano rispettati il più possibile nella copia
- r** Copia ricorsiva delle directory

58

## Esempi

Esempi:

```
•$ ssh -l tizio linux.brot.dg ls -l /tmp
```

User Name    Host name    Comando

```
•$ ssh -l tizio linux.brot.dg tar czf~/home/tizio>backup.tar.gz
```

Comando: Copia della directory personale dell'utente 'tizio' nell'elaboratore remoto. L'operazione genera il file 'backup.tar.gz' nella directory corrente dell'elaboratore locale

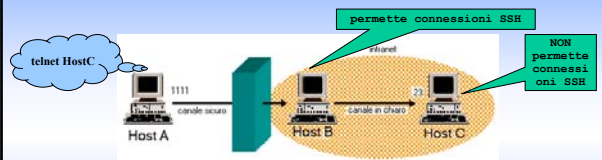
```
•$ scp tizio@linux.brot.dg:/etc/profile .
```

User Name, Host e file remoto da copiare    Directory destinazione locale

59

## Esempio: Port Forwarding

- Immaginiamo il seguente scenario:



hostA> ssh -L 1111:HostC:23 HostB

Porta locale libera    Host e porta da raggiungere    Ricevente del comando

hostA> telnet localhost 1111

60