Università degli studi di Trieste

Laboratorio di reti di calcolatori

DIPATIMENTO DI
ELETTROTECNICA
ELETTRONICA
INFORMATICA

Installazione e configurazione di

Allievi: Argenton Marco
Codognotto Alberto
D'Orlando Marco
Meneghel Francesco
Relatore: prof. Bartoli Alberto
Anno accademico: 2001/02

Introduzione

La seguente documentazione è stata prodotta al fine di chiarire le modalità d'installazione e configurazione del programma OpenSSH: secure shell, in altre parole un'applicazione (composto da server e client) capace di emulare un terminale Windows, su di una macchina Windows remota. L'esperienza è stata assegnata come esercitazione pratica, al termine del corso di reti di calcolatori. Lo scopo è di installare l'applicazione (lato server e lato client) e di configurare il sistema con le modalità di autenticazione viste a lezione.

Installazione del server

- Eseguire il programma d'auto installazione "`setup.exe`". Il programma richiede che gli siano forniti alcuni dati da parte dell'utente (in questo caso administrator):
  - directory di installazione,
  - componenti da installare, server, client o server e client.

  L' auto installazione procede col decomprimere i file, nella cartella di installazione.
  Genera le coppie di chiavi pubbliche e private dell'host (`ssh_host_key, ssh_host_dsa_key, ssh_host_key.pub, ssh_host_dsa_key.pub` i file con estensione `.pub` contengono le chiavi pubbliche).
  Configura automaticamente i file di sistema per l'esecuzione del programma server.
  L'installazione del server è stata eseguita con privilegio di administrator.

- Quindi si deve compilare correttamente il file di configurazione degli utenti. Il file si chiama `[Install directory]\etc\passwd`. Si dovrebbe aprire automaticamente al termine dell'installazione. Altrimenti si deve utilizzare un editor per visualizzarlo e modificarlo.

```
Administrator::::Administrator:/:
alberto::500::Alberto Codognotto:/usr_ssh/alberto:/[installdirpath]/ssh/switch.exe
marco::500::Marco D'Orlando:/usr_ssh/marco:/[installdirpath]/ssh/switch.exe
marco2::500::Marco Argenton:/usr_ssh/marco2:/[installdirpath]/ssh/switch.exe
francesco::500::Francesco Meneghel:/usr_ssh/francesco:/[installdirpath]/ssh/switch.exe
```

Sopra si può vedere il file di registrazione utenti compilato. Ci sono vari campi, separati dal carattere " : ".
Il primo campo rappresenta il nome utente, il secondo deve rimanere vuoto, infatti è il campo password e non deve essere memorizzato sul disco in "chiaro". Il terzo campo è lo userid che abbiamo posto 500 per ogni utente .Questo è necessario per sistemi windows per rendere possibile l' autenticazione a chiave pubblica come indicato nella documentazione . Il quarto campo deve rimenere vuoto. Il quinto campo rappresenta la descrizione utente. Il sesto campo la home directory dell'utente sul server. Il settimo campo identifica quale shell aprire all'utente dopo il logon.
Dato che OpenBSD (da cui è estratto OpenSSH) è simile ai sistemi UNIX, non si possono considerare le altre unità DOS servendosi della notazione lettera-due punti (es: D:, E:, etc) per specificare un'unità diversa anzichè C:, si deve scrivere: `/cygdrive/DRIVER_LETTER,` es: `/cygdrive/d/usr_ssh.`
E' importante che la linea administrator non venga eliminata.
Si sottolinea, inoltre, che gli utenti devono essere "utenti di Windows NT" della macchina server. La password a cui si fa riferimento, usata per l'autenticazione "debole", è quella di accesso alla macchina server.

- Creazione delle varie directory utente `[Home]`, in questo caso si dovranno creare le directory:

  Es: `c:\usr_ssh\alberto`
  `c:\usr_ssh\marco`
  `c:\usr_ssh\marco2`
  `c:\usr_ssh\Francesco`
  Per ogni utente poi si dovrà creare una sotto cartella `.ssh.`
  Es:
  `c:\usr_ssh\alberto\.ssh`

- Si compila correttamente il file `start-sshd.bat:`
  E necessario scrivere correttamente la posizione del file `cmd.exe.` Nel file si troverà una voce:
  `Set ALTCMD=` indica di default la directory `c:\winnt\system32\cmd.exe`
  Per cui se il sistema operativo NT è installato in una directory diversa dalla consueta si dovrà aggiungere il path effettivo, specificando dove si trova il file:
  Es: `Set ALTCMD=d:\winnt\system32\cmd.exe`

- Creazione dei file: `.rhosts, .shosts, authorized_keys, authorized_keys2.` E' necessaria una copia per ogni utente. Di default questi file non sono creati automaticamente dal programma di installazione, perchè sono utilizzati solo dalla modalità di autenticazione più "debole", quindi si dovranno creare manualmente. La loro presenza tuttavia non creerà problemi agli altri tipi d'autenticazione ne falle di sicurezza.
  - `[HOME]/.rhosts` è costituito da coppie nome host, nome utente, separate da uno spazio, una per riga.
  - `[HOME]/.shosts` è costituito da coppie nome host, nome utente, separate da uno spazio, una per riga come `.rhosts.`
  - `[HOME]/.ssh/authorized_key(2)` elencano le chiavi pubbliche (RSA\DSA) che possono essere utilizzate dall'utente per l'autenticazione.
    Dato che abbiamo utilizzato il protocollo 1 che si avvale di chiavi RSA, è stato sufficiente migrare sul server il file `identity.pub` (tramite floppy) e rinominarlo (qualora venga utilizzato il protocollo 2, le chiavi in `id_dsa.pub` e `id_rsa.pub` devono essere inserite nel file `authorized_key2`).

- Creazione dei file `rhost.equiv, shost.equiv`.
  - `[Install directory]/etc/rhost.equiv` il file è costituito da il nome degli host abilitati al logon nel host-server in questione. Si possono inoltre specificare i nomi degli utenti, per ogni host, abilitati alla connesione al server. Il formato del file è nome: `host(space)utente1(space)utente2(space)utente3` e cosi via.
  - `[Install directory]/etc/shost.equiv` analogo al file precedente.

Configurazione del server

Dopo aver creato tutti i file necessari per i vari metodi di autenticazione supportata, si deve procedere con la configurazione del sistema. Il file preposto a tale scopo è denominato sshd_config. Nella installazione di default è contenuto nella directory di installazione.
Riportiamo qui il contenuto del file

```
HostKey             /ssh/ssh_host_key
HostDSAKey               /ssh/ssh_host_dsa_key
PidFile                  /ssh/sshd.pid
Protocol            1

PermitRootLogin                     yes
PasswordAuthentication              yes
IgnoreRhosts                        yes
IgnoreUserKnownHosts                yes
RhostsAuthentication                no
RhostsRSAAuthentication              no
RSAAuthentication                   yes

SyslogFacility AUTH
LogLevel INFO

Subsystem sftp /ssh/sftp-server
```

Rispetto alla versione di default, abbiamo permesso l'utilizzo del solo protocollo 1 .

Installazione Client

- Analogamente a quanto fatto in precedenza sul lato server, basta mandare in esecuzione su un'altra macchina il file "`setup.exe`" selezionando l'installazione **del solo client.** Abbiamo fatto questo perché non ci sembra ragionevole installare il server su più nodi della rete: si noti che questo particolare sarà rilevante per le considerazioni che riporteremo in seguito.
- Installando il solo client non avviene alcuna generazione di chiavi di nodo come avveniva in automatico per quanto riguarda il server.
- Abbiamo generato il file `[install directory]/ssh/.ssh/known_hosts` contentenente la chiave pubblica del server. Il formato è il seguente:
  `[nome DNS del server],[ip address](space)[chiave].`
  Il file qualora non esista viene creato automaticamente al primo tentativo di connessione al server chedendo all'utente il permesso di aggiungere la chiave pubblica del server.
  Per questioni di sicurezza ci è sembrato corretto definire a priori il server fidato.
- A questo punto non resta altro che creare la/le coppia/e di chiavi utente: questo viene fatto mediante l'applicazione ssh-keygen.exe. E' possibile generare i seguenti file: identity e identity.pub (contenenti chiavi RSA usate dal protocollo1), `id_dsa` e `id_dsa.pub` (contenenti chiavi dsa usate dal protocollo 2), id_rsa e id_rsa.pub (contenenti chiavi RSA usate dal protocollo2). Questi vanno in `[install directory]/ssh/.ssh`
- In fase di generazione, abbiamo sfruttato l' opzione di protezione delle chiavi mediante una "passphrase" che viene poi richiesta in fase di autenticazione (cfr. log1)
- Per la connessione remota l'utente si serve del programma ssh.exe. E' possibile passare parametri opzionali o tramite linea di comando (es: opzione –v "verbose" per visualizza re i log) o tramite il file `config` (stesso formato di ssh_config sul lato server) che si può opzionalmente creare in `[install directory]/ssh/.ssh` In questo modo è possibile forzare il client ad usare una particolare autenticazione tra quelle permesse dal server.

Considerazioni

- Una volta terminata la fase di installazione e configurazione del tool, abbiamo provato ad utilizzare le modalità di autenticazione viste a lezione. Come prima cosa, abbiamo notato che l'autenticazione fra host non è mutua[1]: il client infatti non ha nessuna chiave di nodo! Questo non risulta subito chiaro dalla documentazione, ma trova conferma dalla esecuzione di ssh in modalità "verbose" (cfr. log1). Si vede chiaramente che il client aspetta la chiave pubblica di nodo del server, la confronta con la propria copia contenuta nel file Known_host e se coincidono genera un numero casuale, lo critta con la chiave appena ricevuta e lo invia al server; a questo punto il random-number viene usato da entrambi i lati come session-key per crittare la comunicazione (descrizione semplificata).
  Questi passaggi sono sì spiegati nel manuale sshd, ma in quest'ultimo ci sono anche frasi fuorvianti che vanno interpretate nel giusto contesto: per esempio si dice che ogni host è caratterizzato da una chiave di nodo, ma dal contesto si capisce si sta parlando di un host in cui sia installato il server.

- Abbiamo notato, inoltre, che l' utente che desidera accedere al server deve essere un utente NT della macchina su cui questo è installato. Lo username ssh deve quindi coincidere con quello NT.
  Non c'è correlazione invece con lo username utilizzato per accedere alla macchina client.
  Quindi nel caso di autenticazione con password l' utente pippo abilitato sul client ma non sul server,
  può logarsi a quest ultimo dicendo di essere pluto (utente abilitato) a patto di conoscerne la password.

- Le due modalità di autenticazione di utente che siamo quindi riusciti a realizzare sono quella a chiave pubblica e quella tramite password. L' autenticazione utilizzata è la più forte tra quelle ammesse da entrambi i lati.Se per esempio sono previste entrambe le modalità su ambo i lati e se l' utente è sprovvisto delle chiavi ,automaticamente il sistema si "abbassa" richiedendo la password;qualora questa venga digitata in modo errato per 3 volte viene restituito il seguente messaggio di errore: "too many authentications faild".
  Si noti che in condizioni operative è giusto che sia il server a fissare la modalità di autenticazione .

- Una prova interessante è stata la reinstallazione del server,con con conseguente generazione di nuove chiavi di nodo, senza l'aggiornamento del file `known_hosts` sul lato client.
  Quando l'utente tenta di accedere al server il client non riconosce la nuova chiave ,
  avvisa l'utente con un messaggio di warning e disabilita l'autenticazione con password per evitare che l' utente ignaro digiti la password pregiudicandone la segretezza .
  Il sistema ammette comunque l' autenticazione a chiave pubblica (cfr log2).
  In questo caso l' utente non può essere sicuro di connettersi al server corretto.

- Tra gli allegati inseriamo due capture dello sniffer: una in cui compaiono i pacchetti relativi alla risposta crittata ad un comando "dir", ed una in cui i pacchetti sono sempre relativi ad un comando "dir" ma questa volta eseguito su sessione telnet e quindi in chiaro.

[1] Per dissolvere i nostri dubbi circa le modalità di autenticazione fra host, abbiamo contattato via mail uno dei realizzatori del tool presso la NetworkSimplicity: alleghiamo la mail di risposta che conferma le nostre ipotesi.

LOG1

```
C:\PROGRA~2\NETWOR~1\ssh>ssh -1 -v alberto@egle
OpenSSH_2.9p2, SSH protocols 1.5/2.0, OpenSSL 0x0090600f
debug1: Reading configuration data /usr/lib/.ssh/config
debug1: Seeding random number generator
debug1: restore_uid
debug1: ssh_connect: getuid 500 geteuid 500 anon 1
debug1: Connecting to egle [169.0.0.1] port 22.
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: Connection established.
debug1: identity file /usr/lib/.ssh/identity type 0
debug1: Remote protocol version 1.99, remote software version OpenSSH_2.9p2
debug1: match: OpenSSH_2.9p2 pat ^OpenSSH
debug1: Local version string SSH-1.5-OpenSSH_2.9p2
debug1: Waiting for server public key.
debug1: Received server public key (768 bits) and host key (1024 bits).
debug1: Host 'egle' is known and matches the RSA1 host key.
debug1: Found key in /usr/lib/.ssh/known_hosts:1
debug1: Encryption type: 3des
debug1: Sent encrypted session key.
debug1: Installing crc compensation attack detector.
debug1: Received encrypted confirmation.
debug1: Trying RSA authentication with key 'alberto@HELENA'
debug1: Remote: /usr_ssh/alberto/.ssh/authorized_keys, line 1: bad key syntax
debug1: Remote: /usr_ssh/alberto/.ssh/authorized_keys, line 2: bad key syntax
debug1: Received RSA challenge from server.
Enter passphrase for RSA key 'alberto@HELENA':
debug1: Sending response to host key RSA challenge.
debug1: Remote: RSA authentication accepted.
debug1: RSA authentication accepted by server.
debug1: Requesting pty.
debug1: Requesting shell.
debug1: Entering interactive session.
Last login: Tue Jan 22 12:30:20 2002 from helena
Microsoft Windows 2000 [Versione 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

c:\usr_ssh\alberto>dir
 Il volume nell'unit. C S MAIN
 Numero di serie del volume: 206E-0FEB

 Directory di c:\usr_ssh\alberto

22/01/2002  10.53       <DIR>          .
22/01/2002  10.53       <DIR>          ..
22/01/2002  10.53       <DIR>          .ssh
22/01/2002  11.30                   99 .rhosts
22/01/2002  11.31                   99 .shosts
               2 File            198 byte
               3 Directory     385.945.600 byte disponibili

c:\usr_ssh\alberto>exit
Connection to egle closed.
debug1: Transferred: stdin 9, stdout 646, stderr 28 bytes in 12.0 seconds
debug1: Bytes per second: stdin 0.8, stdout 54.0, stderr 2.3
debug1: Exit status 0

C:\PROGRA~2\NETWOR~1\ssh>
```

LOG2

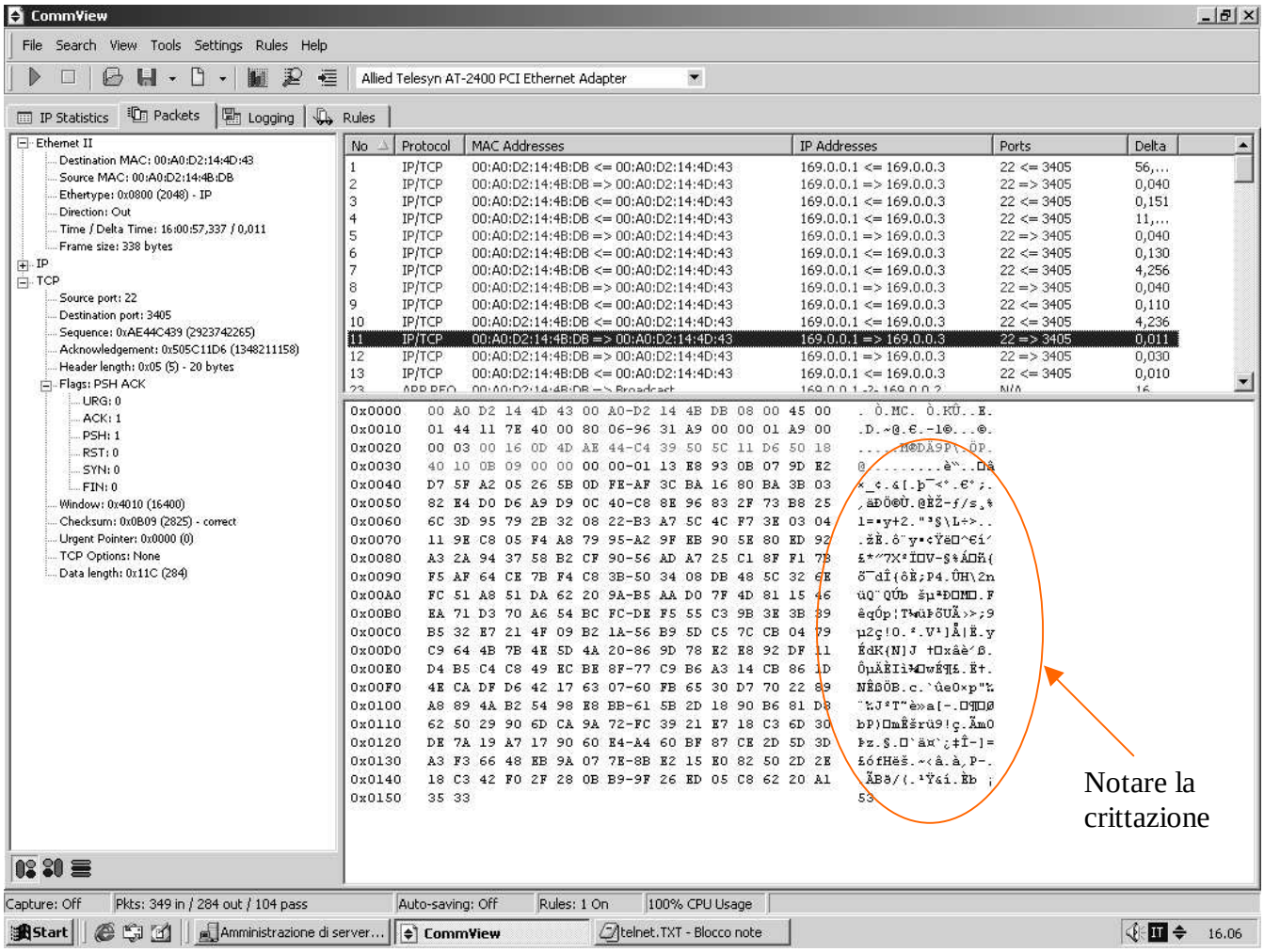
```
C:\Program Files\NetworkSimplicity\ssh>ssh alberto@egle
debug1: Seeding random number generator
debug1: restore_uid
debug1: ssh_connect: getuid 500 geteuid 500 anon 1
debug1: Connecting to egle [169.0.0.1] port 22.
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: Connection established.
debug1: identity file /usr/lib/.ssh/identity type 0
debug1: Remote protocol version 1.5, remote software version OpenSSH_2.9p2
debug1: match: OpenSSH_2.9p2 pat ^OpenSSH
debug1: Local version string SSH-1.5-OpenSSH_2.9p2
debug1: Waiting for server public key.
debug1: Received server public key (768 bits) and host key (1024 bits).
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA1 host key has just been changed.
The fingerprint for the RSA1 key sent by the remote host is
0f:ef:9d:e8:aa:bf:73:60:97:77:e2:f7:ea:ce:d3:21.
Please contact your system administrator.
Add correct host key in /usr/lib/.ssh/known_hosts to get rid of this message.
Offending key in /usr/lib/.ssh/known_hosts:1
Password authentication is disabled to avoid trojan horses.
debug1: Encryption type: 3des
debug1: Sent encrypted session key.
debug1: Installing crc compensation attack detector.
debug1: Received encrypted confirmation.
debug1: Trying RSA authentication with key 'alberto@HELENA'
debug1: Remote: /usr_ssh/alberto/.ssh/authorized_keys, line 1: bad key syntax
debug1: Remote: /usr_ssh/alberto/.ssh/authorized_keys, line 2: bad key syntax
debug1: Received RSA challenge from server.
Enter passphrase for RSA key 'alberto@HELENA':
debug1: Sending response to host key RSA challenge.
debug1: Remote: RSA authentication accepted.
debug1: RSA authentication accepted by server.
debug1: Requesting pty.
debug1: Requesting shell.
debug1: Entering interactive session.
Last login: Tue Jan 22 12:50:50 2002 from max
Microsoft Windows 2000 [Versione 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

c:\usr_ssh\alberto>
```

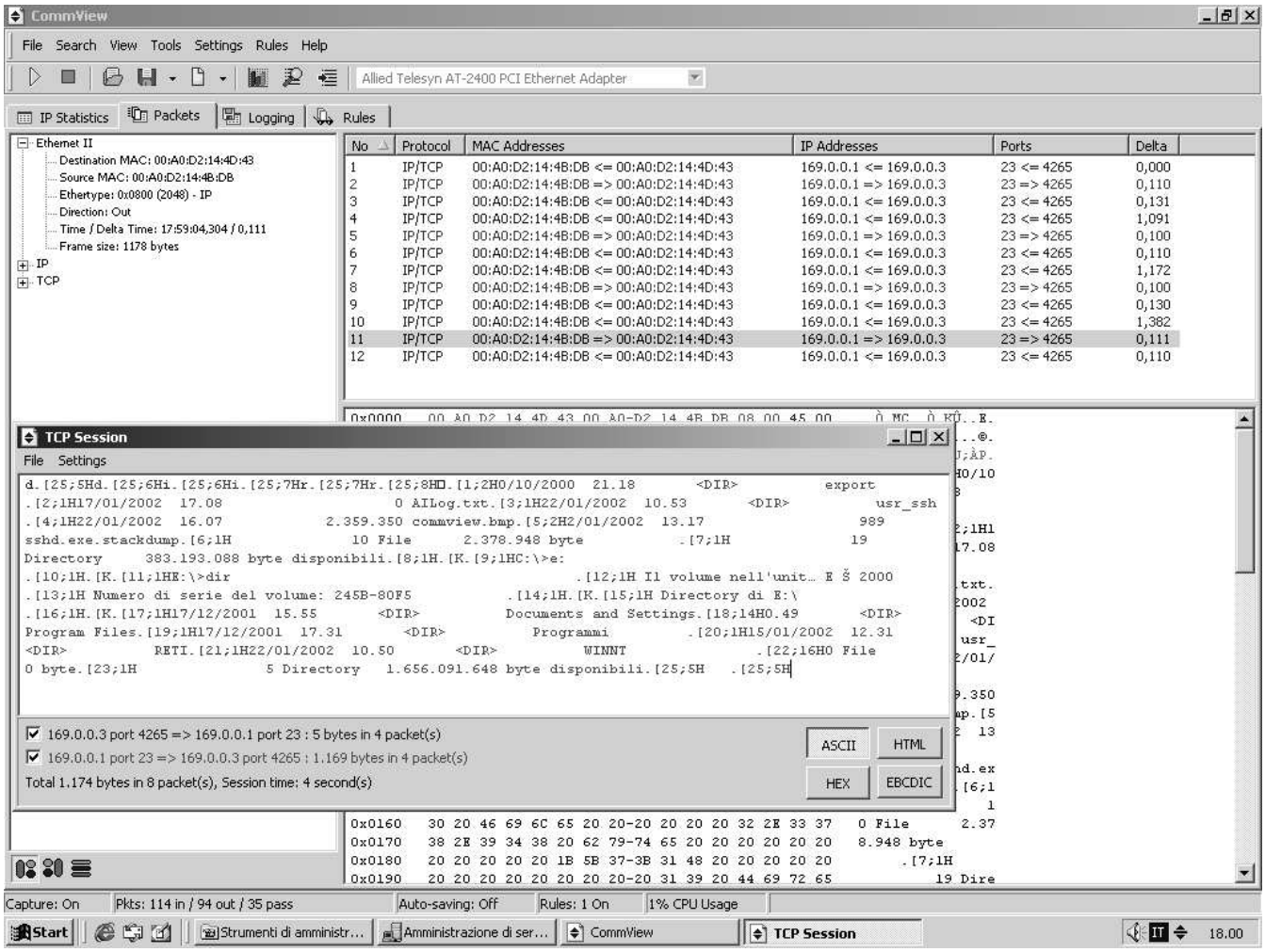
Log3

```
C:\>ssh alberto@egle
debug1: Seeding random number generator
debug1: restore_uid
debug1: ssh_connect: getuid 500 geteuid 500 anon 1
debug1: Connecting to egle [169.0.0.1] port 22.
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: temporarily_use_uid: 500/544 (e=500)
debug1: restore_uid
debug1: Connection established.
debug1: identity file /usr/lib/.ssh/identity type 0
debug1: Remote protocol version 1.99, remote software version OpenSSH_2.9p2
debug1: match: OpenSSH_2.9p2 pat ^OpenSSH
debug1: Local version string SSH-1.5-OpenSSH_2.9p2
debug1: Waiting for server public key.
debug1: Received server public key (768 bits) and host key (1024 bits).
debug1: Host 'egle' is known and matches the RSA1 host key.
debug1: Found key in /usr/lib/.ssh/known_hosts:1
debug1: Encryption type: 3des
debug1: Sent encrypted session key.
debug1: Installing crc compensation attack detector.
debug1: Received encrypted confirmation.
debug1: Doing challenge reponse authentication.
debug1: No challenge.
debug1: Doing password authentication.
alberto@egle's password:
debug1: Requesting pty.
debug1: Requesting shell.
debug1: Entering interactive session.
Last login: Wed Jan 16 17:57:00 2002 from max
Microsoft Windows 2000 [Versione 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

c:\usr_ssh\alberto>
```



Capture 1: Esecuzione di "dir" su ssh.



Capture 2: Esecuzione di "dir" su telnet.