Setup dell'ambiente virtuale

1.1 L'immagine ISO

Il file `sicureti.iso` è un'immagine ISO *live*: significa che può essere utilizzata in diversi modi:

1. può essere masterizzata su di un CD o un DVD e usata in modalità *live,* cioè avviando una macchina fisica con il CD o DVD[1];
2. può essere copiata su di una memoria USB e usata in modalità *live,* cioè avviando una macchina fisica con la chiavetta USB[2];
3. può essere usata simulando le precedenti modalità con una **macchina virtuale** come Qemu (vedi § 1.2), VMWare, VirtualBox,...

L'immagine contiene un programma d'installazione, qualora la si voglia trasferire in maniera permanente sull'hard-disk della macchina utilizzata. Gli esercizi del corso sono pensati per l'uso in modalità 3.

Il sistema contenuto nell'immagine è una versione personalizzata per il corso di Sicurezza delle reti di Debian GNU/Linux. L'utente `user` può autenticarsi con la password `live` e accedere ai privilegi di amministrazione con il programma `sudo`. Per ulteriori informazioni si veda `http://live.debian.net`.

1.2 Qemu

Qemu (`http://qemu.org`) è un emulatore che permette di simulare sistemi hardware completi. È software libero e disponibile per ambienti GNU/Linux, MS Windows, Mac OS X. Nel seguito si farà riferimento alla versione Linux, ma le differenze in altri contesti sono minime e riguardano principalmente il modo di lanciare il programma. I parametri a linea di comando indicati sono disponibili in tutte le versioni.

Per lanciare il programma simulando la modalità 1

```
1 qemu -cdrom sicureti.iso
```

Utilizzando uno dei comandi descritti sopra, si ottiene un sistema GNU/Linux i386, con 128MB di RAM e in grado di sfruttare la rete disponibile al sistema *host*, cioè alla macchina in cui viene fatto girare (cioè *ospita*) Qemu[3]

Se la macchina *host* dispone di sufficiente RAM, si consiglia di aumentare la memoria a disposizione della macchina *guest*. L'opzione `-m 1024` fornisce 1GB di memoria alla

[1] La macchina deve essere capace di *boot* da CD/DVD e avere un processore compatibile i386
[2] La macchina deve essere capace di *boot* da USB e avere un processore compatibile i386
[3] La macchina virtuale *ospitata* all'interno di Qemu viene detta invece "sistema *guest*".

macchina *guest,* rendendo molto piú efficiente l'esecuzione dei programmi, in special modo quelli in modalità grafica[4].

Se la macchina *host* è un sistema GNU/Linux, esiste una versione di Qemu che sfrutta le capacità di virtualizzazione dei kernel (e processori) moderni:

```
1 qemu -enable-kvm -cdrom sicureti.iso
```

Attenzione! Per mantenere le dimensioni ridotte, molti dei programmi contenuti nell'immagine iso sono in realtà ottenuti tramite `busybox`: p.es. `ls` è un *link* simbolico a `busybox`, perciò l'esecuzione di `ls` equivale a chiamare `busybox ls`. Il manuale di queste versioni ridotte è descritto nella pagina di manuale di `busybox` (`man busybox`).

1.3 Esercizi

1.3.1 Capire che cosa fa Qemu

Leggere il manuale di Qemu alla pagina `http://wiki.qemu.org/download/qemu-doc.html` (non aggiornata all'ultima versione, ma sufficientemente sintetica per farsi un'idea d'insieme. La versione corrente è `http://en.wikibooks.org/wiki/QEMU`), soffermandosi in particolare sulle opzioni riguardanti dischi e rete.

1.3.2 Gestire la macchina virtuale

Agendo con i tasti `Ctrl-Alt-2` mentre è in esecuzione Qemu, si accede al monitor di sistema della macchina *guest;* con `Ctrl-Alt-1` si torna alla macchina *guest.*

Accedere al monitor e provare il comando `help,` ottenendo cosí un elenco delle possibilità. Quando non ci stanno tutte nella finestra corrente, le schermate del monitor possono essere esaminate con `Ctrl-PgUp` e `Ctrl-PgDown`. In modalità monitor con il comando `sendkey` è possibile mandare una combinazioni di tasti alla macchina *guest:* provare `sendkey ctrl-alt-f2`.

[4]Per mantenerne le dimensioni ridotte, l'immagine iso contiene solo programmi di tipo 'console'

Tool di base per analizzare lo stack TCP/IP

Per esaminare (e, in alcuni casi, variare) le caratteristiche dello *stack* TCP/IP installato sulla macchina *guest*, si possono utilizzare alcuni tool di base, generalmente presenti in ogni distribuzione di Linux. Prima di iniziare gli esperimenti meglio controllare di riuscire a connettersi all'esterno tramite le applicazioni della macchina guest, per esempio il browser. Le operazioni che seguono richiedono per lo piú i privilegi di amministrazione: il modo piú semplice per iniziare una sessione comandi, da eseguire come `root` user, si apra un terminale e si dia il comando

```
1 sudo -s
```

Dopo di ciò il prompt indica che si sta agendo come `root` (e l'ultimo carattere del prompt è #)

2.1 ifconfig

ifconfig è il comando per configurare le *interfacce* di rete.

```
1 ifconfig -a
```

Serve a mostrare tutte (all) le interfacce configurate nel sistema. Un'interfaccia è attiva quando è `UP`. Affinché sia utilizzabile in modalità TCP/IP deve avere (almeno[1]) un numero IP.

*L*e interfacce attive nella macchina virtuale sono `eth0` e il *local loopback* `lo`.

Per cambiare il numero IP di `eth0`

```
1 ifconfig eth0 10.0.2.16
```

Gli esperimenti possono alterare lo stack rendendolo non funzionante. Potete sempre riavviare la macchina virtuale per ricominciare. In molti casi basta però riavviare semplicemente la configurazione delle interfacce. Nei sistemi Debian ciò avviene tramite gli script `ifup` e `ifdown`. Per ripristinare

```
1 ifdown eth0
2 ifup eth0
```

2.1.1 Esercizi

- Leggere il manuale di `ifconfig` (man ifconfig)
- Rilevare la netmask di `eth0`
- Cosa succede se si imposta un numero IP su una sottorete diversa? La connessione con l'esterno continua a funzionare?

[1] È possibile associare piú di un numero IP alla stessa interfaccia fisica

- Cambiare il numero MAC dell'interfaccia `eth0` in `00:01:02:03:04:05`

2.2 netstat

Il dispatching dei pacchetti è regolato dalla *routing table*, che può essere manipolata tramite il comando `route`.

Con `netstat` si può controllare lo stato della tabella di *routing*.

```
1 netstat -nr
```

Il `-n` evita di coinvolgere il risolutore DNS nell'operazione, rendendola piú veloce e affidabile (la risoluzione richiede infatti probabilmente la generazione di pacchetti).

`netstat` permette anche di esaminare lo stato delle connessioni in corso o in ascolto

```
1 netstat -taun
```

- Leggere il manuale di `netstat` e dar conto dei parametri usati negli esempi
- Quale processo è in ascolto sulla porta TCP 22?
- In un altro terminale aprire una sessione SSH (ssh user@localhost, la password è `live`): controllare lo stato delle connessioni dopo questa operazione.

Una versione piú raffinata e moderna del programma `netstat` è `ss` (*socket stat*): le opzioni sono pressoché identiche.

La virtualizzazione di nodi di rete

All'interno della macchina generata con il Live CD e magari virtualizzata con Qemu è possibile attivare un ulteriore livello di virtualizzazione con *Virtual Square*. Virtual Square è un progetto di software libero per virtualizzazione dell'Università di Bologna (Renzo Davoli, `http://wiki.virtualsquare.org`) e mette a disposizione diversi componenti per la virtualizzazione della rete (VDE2, LWIPV6) e dell'interfaccia del sistema operativo (UMview), attivabili in user-mode, generalmente senza bisogno di particolari privilegi di amministrazione (necessari solo quando si vuole utilizzare la speciale interfaccia *tun/tap*).

3.1 VDE2

VDE2 (Virtual Distributed Ethernet) permette di realizzare *switch* virtuale con il comando vde_switch. La terminologia è la seguente:

wire qualsiasi cosa sia capace di fornire uno *stream* di dati può essere un *wire*

plug un terminale cui è attaccato un *wire* e finisce in uno *switch*

cable è un *wire* con due *plug* e connette i nodi della rete virtuale

```
1 vde_switch -s /tmp/switch
```

In questa maniera si crea uno *switch* controllato da una *unix socket* `ctl` contenuta nella directory `/tmp/switch`. Il processo mette a disposizione un'interfaccia di *monitoring*, come è possibile sperimentare scrivendo un comando (p.es. `help`).

L'interfaccia di monitoring può essere gestita piú comodamente attivando un'apposita *unix socket*.

```
1 vde_switch -s /tmp/switch -M /tmp/mgmt
```

A questo punto ci si può collegare con

```
1 vdeterm /tmp/mgmt
```

molto piú comodo perché il terminale cosí attivato dispone di *history* e completamento dei comandi.

3.2 UMview

UMView è lo strumento principale di *View OS*: un approccio alla virtualizzazione in cui ogni processo "vede" una versione personalizzate delle chiamate di sistema.

```
1 umview bash
```

In questo modo si esegue `bash` in una *view*, rendendo possibile la ridefinizione delle chiamate di sistema: per `bash` il sistema risulterà quindi "virtualizzato".

```
1 um_add_service umnet
```

In questa maniera si carica la ridefinizione delle chiamate di rete.

```
1 umview -V test -p umnet bash
```

Cosí si precarica il modulo *umnet* e si dà alla *view* il nome `test` (utile a riconoscerla quando ce n'è piú di una...)

3.3 Virtualizzazione dello stack

Grazie a umnet si possono costruire *stack* di rete con le proprietà volute. Il meccanismo fondamentale consiste nel "montare" una specifica implementazione della pila protocollare, p.es. `umnetlwipv6` una implementazione (in user mode!) dello stack TCP/IP (versioni 4 e 6).

```
1 mount -t umnetlwipv6 -o vd0=/tmp/switch none /dev/net/prova
```

A questo punto è possibile far "vedere" lo stack `/dev/net/prova` a qualsiasi processo con il comando `mstack`.

```
1 ip link
2 mstack /dev/net/prova ip link
```

Attenzione! Bisogna avere l'accortezza di usare programmi che usano direttamente le *system call*: p.es. `ifconfig`, `netstat` e `route` non va bene perché agiscono manipolando lo pseudo-file-system `/proc`: in pratica usano le system call di manipolazione file invece che quelle di manipolazione di rete (le uniche che vengono ridefinite da `umnet`). Inoltre, l'implementazione di LWIPV6 non è completa e alcuni programmi che utilizzano opzioni avanzate (con `setsockopt`) potrebbero funzionare solo parzialmente (è il caso di `nmap`).

3.3.1 Esercizi

Dopo aver letto il manuale di `ip`, creare una *view* `prima` in cui impostare le seguenti proprietà di rete:

```
1 ip addr add 192.168.1.100/24 dev vd0
2 ip link vd0
```

Creare poi un'altra *view* `seconda` con numero IP `192.168.1.200/24`.

3.4 Netcat

Netcat (`http://nc110.sourceforge.net/`) è un programma che permette di stabilire comunicazioni TCP e UDP senza alcun livello applicativo: fornisce sostanzialmente il servizio di creazione di un socket a linea di comando.

L'uso di base prevede che il server si metta in ascolto (-l listen) su di una porta (-p): per quelle < 1024 servono i privilegi di root. La porta non deve naturalmente essere già in uso da un altro programma.

```
nc -l -p 12345
```

Il client si connetterà con un parallelo

```
nc 192.168.1.100 12345
```

Cosí però non è molto utile: bisogna far viaggiare dei dati sulla connessione.

```
echo ciao | nc 192.168.1.100 12345
```

In questo caso il server vede il messaggio “ciao”. Vale anche nell’altro senso.

```
echo ciao | nc -l -p 12345
```

Il client riceverà il messaggio al momento del collegamento.

3.4.1 Esercizi

- Leggere il manuale di netcat
- Scambiare il messaggio ‘ciao’ con netcat f