Creare un server web con un Raspberry Pi

Giovan Battista Rolandi

2013

Sommario

Indice

1 Introduzione

Dopo aver analizzato le ragioni per le quali si può avere necessità di creare un proprio server web casalingo, questa guida analizzerà dettagliatamente l'installazione e la configurazione del server su un Raspberry Pi. La guida suppone che si conosca già come funziona un sito internet (il webserver, PHP, ...) e che, magari, se ne abbia già uno personale; inoltre è richiesta una certa dimestichezza con la shell Linux, in quanto la guida si rivolge ad un pubblico che sa cos'è e come funziona Linux.

1.1 Licenza

1.2 Perché creare un server casalingo?

Le ragioni che possono spingere alla creazione di un webserver casalingo sono molteplici. Evidentemente, tenere nella propria abitazione il proprio sito web, crea numerosi vantaggi:

- Controllo completo sull'hardware: possibilità di ampliarlo, migliorarlo, gestirlo come più ci aggrada, allo stesso modo di un normalissimo computer
- Controllo completo sul software: possibilità di installare qualsiasi programma e strumento utile, libertà di configurare senza alcuna limitazione il proprio sistema
- Imparare qualcosa di nuovo

Ma ci sono altrettanti svantaggi:

- Costo dell'energia elettrica: tenere un computer acceso 24h può incidere una centinaia di euro all'anno sulla bolletta.
- Manutenzione: l'hardware deve essere raffreddato; spesso, è necessario anche tenerlo in una stanza separata a causa della rumorosità; in più, in caso di sbalzi di tensione o blackout temporanei, router e server devono essere riavviati. Inoltre, i software devono essere costantemente aggiornati, almeno mensilmente.
- Sicurezza: ogni apertura della nostra rete verso l'esterno, è la porta d'accesso per un potenziale attacco verso i nostri computer di casa; più probabilmente, potremmo essere soggetti a spam e a un sovraccarico della rete.
- Velocità: la velocità di upload delle connessioni ADSL domestiche in genere è molto limitata e non supera normalmente i 100K/s

- Costo della connessione a Internet: questo non dovrebbe essere un problema, perché chi ha interesse a crearsi un server casalingo, di norma ha già almeno un abbonamento a Internet senza tempo.

Se tuttavia si è davvero determinati a fronteggiare qualsiasi controversia, o non si ha la necessità di un sistema stabile al 100%, non resta che continuare a leggere.

1.2.1 Velocità della linea

Un problema che limita le capacità del server è indubbiamente la velocità con cui questo può comunicare verso l'esterno; sebbene le interfacce di rete Ethernet consentano lo scambio di dati da 10M/s a 100M/s, la normale linea telefonica, in grado di far fronte ad un massimo di 20M/s, è ulteriormente limitata da fattori quali la lunghezza dei cavi e il loro percorso, la distanza dalla centralina, e, non ultimo, l'avidità degli *Internet Service Provider* (ISP). Una comune ADSL Telecom 7Mega consente di inviare al massimo 40K/s; Fastweb su rame ha un'upload di 100K/s, mentre su fibra ottica fino a 10M/s.

Per un piccolo sito/blog personale con poche immagini e senza molti contenuti multimediali, 100K/s sono sufficienti per servire quattro/cinque utenti contemporaneamente.

Altro elemento da non trascurare è il tempo di *ping*, che deve essere il più basso possibile. Buoni valori si aggirano al di sotto di 100ms, anche se fino a 200ms è comunque trascurabile. Il tempo di ping varia in base al luogo dal quale viene effettuata la connessione al vostro server: più lontani ci si trova, più tempo è necessario per stabilire la connessione. Per stabilire il tempo di ping, si scelgono alcuni server di cui conosciamo la collocazione geografica e si pingano. Ovviamente bisogna scegliere dei server vicini ai luoghi dai quali immaginiamo di raggiungere più visite: sarebbe completamente inutile provare un server in Alaska se il nostro sito parla di abbigliamento estivo. Per avere un'idea generale del tempo che impiega un visitatore a stabilire una connessione col nostro server da una determinata località, si può usare questo strumento: https://www.site24x7.com/ping-test.html

1.3 Perché utilizzare un Raspberry Pi?

La scelta della macchina che funga da server deve essere ben ponderata. Utilizzare un Raspberry Pi come server invece che un normale computer, ha innumerevoli vantaggi:

- Un computer consuma in media 100W, e ha un costo annuo di $\sim$ 200 €; un Raspberry Pi, che consuma 5W in media, ha un costo annuo di soli $\sim$ 10 €
- Un computer è spesso rumoroso, richiede una certa ventilazione e un locale adatto ad essere ospitato; un Raspberry Pi non ha parti meccaniche in movimento, quindi non genera rumore, produce poco calore (un piccolo dissipatore è sufficiente) e, non essendo più grande di un pacchetto di sigarette, può essere collocato dove preferiamo.

- Mentre un computer si spegne al minimo calo di tensione, essendo il Raspberry Pi alimentato attraverso un trasformatore, può resistere a blackout di alcuni secondi; tuttavia, un gruppo di continuità (UPS) è indispensabile per evitare eventuali downtime.

- Un Raspberry Pi ha un processore a 700MHz (eventualmente lo si può overclockare, anche se lo sconsiglio in quanto potrebbe recare danni all'hardware, in particolare alla scheda SD): questa capacità di calcolo è più che sufficiente per gestire un sito statico di sole pagine HTML, o dinamico in PHP. Si può utilizzare anche un database MySQL, a patto di non non sovraccaricarlo troppo con CMS piuttosto complessi (come Wordpress o Mediawiki – in alternativa si può usare Drupal o Dokuwiki, oppure spostare il server MySQL su un'altra macchina).

2 Scelta della distribuzione

L'universo Linux fornisce moltissime distribuzioni, diverse tra loro per filosofia e modalità di utilizzo. Tra tutte le distribuzioni, è indispensabile sceglierne una che abbia il supporto per l'architettura ARM (la nuova architettura dall'alto rapporto di velocità/consumo presente sui nuovi smartphone e sul Raspberry Pi). Tra quelle più gettonate, presenti sul sito uffuciale del Raspberry Pi, troviamo ArchlinuxARM e Debian Armel. Tra le due, Debian è sicuramente la più adatta ad ospitare un webserver, per via della sua stabilità. Pertanto, scarichiamo l'immagine di Debian Armel già pronta dal sito ufficiale del Raspberry Pi.

Al momento in cui sto scrivendo, l'immagine più recente si trova a questo indirizzo: `http://downloads.raspberrypi.org/images/raspbian/2013-02-09-wheezy-raspbian/2013-02-09-wheezy-raspbian.zip`

3 QEMU

Se non avete ancora un Raspberry Pi, oppure volete configurare il sistema più comodamente, seduti dinnanzi al vostro computer, potete virtualizzare il sistema utilizzando l'emulatore QEMU.

Attenzione: virtualizzare un'architettura diversa da quella del proprio processore richiede una notevole potenza di calcolo: si consiglia di utilizzare QEMU solamente se si ha almeno un processore dual core @ 1.5 GHz. In caso contrario, copiare l'immagine sulla scheda SD e configurare il sistema direttamente dal Raspberry Pi.

Se non si è interessati alla virtualizzazione e si vuole installare subito il sistema sul Raspberry Pi per configurarlo direttamente dal minicomputer, si può copiare l'immagine sulla scheda SD (Sezione 6) e procedere con la configurazione (Sezione 4).

3.1 Installazione e preparazione dell'ambiente di lavoro

Innanzitutto, installiamo `qemu` e creiamo una cartella di lavoro:

```
# sudo apt-get install qemu-system    # Debian e derivate
# pacman -Syu qemu                    # Archlinux e derivate
$ mkdir ~/qemu
```

dopodiché procuriamoci il kernel ARM per QEMU:

```
$ cd ~/qemu/ && wget -c http://xecdesign.com/downloads/linux-qemu/kernel-qemu
```

Una volta ottenuti tutti gli strumenti e tutto il materiale necessario, bisogna modificare l'immagine per renderla adatta a funzionare anche su macchina virtuale; pertanto, individuiamo la partizione di / (root) all'interno dell'immagine e montiamola per modificarla:

```
$ file 2013-02-09-wheezy-raspbian.img
```

Come vediamo, ci sono due partizioni: a noi interessa la seconda (formattata in ext4). Considerando il settore di inizio di tale partizione, stabiliamo il byte di inizio della partizione all'interno dell'immagine moltiplicando tale valore per 512, e usiamo il nuovo valore ottenuto come offset per montare la partizione:

```
# mount 2013-02-09-wheezy-raspbian.img -o offset=62914560 /mnt
```

Una volta montata la partizione, modifichiamo il file `/etc/ld.so.preload` commentando l'unica linea presente.

```
  # nano /mnt/etc/ld.so.preload
```

Smontiamo l'immagine:

```
# umount /mnt
```

3.2 Avvio dell'emulatore

Con la seguente riga di comando, invochiamo QEMU per avviare Debian:

```
qemu-system-arm -kernel kernel-qemu -cpu arm1176 -m 256 -M versatilepb \
-no-reboot -serial stdio -append "root=/dev/sda2 panic=1" \
-hda 2013-02-09-wheezy-raspbian.img \
-redir tcp:5022::22 -redir tcp:5080::80
```

Si noti l'opzione `-redir tcp:5022::22`, la quale redirige la porta 5022 del sistema host (il nostro computer fisico) alla porta 22 del sistema guest (il Raspberry Pi virtualizzato). In questo modo, una volta avviato l'emulatore, col seguente comando saremmo in grado di connetterci al Raspberry Pi fittizio tramite ssh, come se fosse un computer sulla rete:

```
$ ssh pi@127.0.0.1 -p 5022
```

Molto probabilmente, l'immagine sarà stata danneggiata da queste operazioni poco ortodosse; pertanto, Debian ci notificherà un errore sul disco, che sarà velocemente corretto impartendo questo comando al terminale di recovery che ci viene presentato:

```
# fsck /dev/sda2
```

Riavviamo la macchina virtuale (eventualmente chiudendo QEMU nel caso non collabori) e il sistema sarà pronto per essere modificato:

```
# reboot
```

Trucco: una volta constatato il corretto funzionamento del sistema virtualizzato, si può inibire il terminale grafico forzando l'utilizzo del solo terminale testuale, semplicemente aggiungendo l'opzione `-cursors` alla riga di comando di QEMU.

4 Configurazione di base del sistema

Il primo accesso al sistema si può effettuare con le credenziali fornite:
Utente: `pi`
Password: `raspberry`

L'utente `pi` è anche `sudoers` e può diventare amministratore digitando semplicemente `$ sudo su` dalla riga di comando.

4.1 Impostare la password di root

Impostare la password di amministrazione sui sistemi Unix è *fondamentale* per la sicurezza del sistema. Su Linux, infatti, l'amministratore è letteralmente autorizzato a fare di tutto (il *tutto* implica anche *distruggere* il sistema). Effettuiamo l'accesso come utente `pi`, dopodichè digitiamo:

```
$ sudo su
# passwd
```

ed impostiamo una nuova password per l'amministratore.

4.2 Creare un utente

Mantenere l'utente `pi` con la password di default è pura follia se si decide di rendere disponibile l'accesso al proprio Raspberry Pi dall'esterno; pertanto, creare un nuovo utente, impostare una password sicura e eliminare l'utente di default:

```
# adduser utente
# userdel pi
# rm -r /home/pi
```

Trucco: se l'utente deve modificare i file contenuti nella DocumentRoot di Apache, impostare come gruppo base `www-data`; altrimenti, se l'utente è già stato creato, seguire la procedura alla Sezione 5.1.2.

4.3 Ottimizzazione del software

A meno che non si voglia utilizzare il Raspberry Pi anche per altri scopi, tutto quanto riguarda la gestione dell'audio e del video, compresi i programmi con interfaccia grafica, può essere disinstallato.

```
# apt-get remove xorg-server lxterminal leafpad alsa-utils ...
# apt-get autoremove
```

A questo punto, aggiornare l'intero sistema (richiede che la connessione a Internet sia attiva):

```
# apt-get update && apt-get upgrade
```

4.4 Impostare un hostname

È buona norma dare un nome ad ogni macchina della rete, e il Raspberry Pi non deve fare eccezione. Per impostare un nome al minicomputer, scriverlo nel file `/etc/hostname`, per esempio:

```
glgprograms
```

Analogamente, aggiungiamo la seguente riga anche a `/etc/hosts`:

```
127.0.0.1 glgprograms:localdomain glgprograms
```

dove *glgprograms* è ovviamente l'hostname che abbiamo scelto.

5 Installazione del webserver

Una volta completata la configurazione del sistema, riavviamo il Raspberry Pi (o comunque la macchina virtuale di QEMU, se stiamo operando sull'immagine dal nostro computer) e, dopo aver constatato che tutto funziona alla perfezione, procediamo con l'installazione dei programmi per il webserver.

Nota: durante l'installazione di questi programmi, `dpkg` richiederà interattivamente come configurarli (per esempio, con quale webserver si deve interfacciare phpMyAdmin) e le password da utilizzare (per esempio, la password di amministrazione del database MySQL). Onde evitare di perdere inavvertitamente queste preziose informazioni, consiglio di annotarle in un posto sicuro.

5.1 Apache

Sebbene esista Lighttpd che è molto più leggero, Apache, essendo più diffuso, è più facilmente personalizzabile ed espandibile, mettendo a disposizione una vasta gamma di moduli.

5.1.1 Installazione di Apache

Per installare Apache (attualmente alla versione 2):

```
# apt-get install apache2
```

ATTENZIONE: per rendere effettiva ogni modifica ai file di configurazione di Apache e PHP, è indispensabile riavviare il webserver (o, al limite, ricaricare la sua configurazione), attraverso uno dei seguenti comandi

```
# service apache2 restart   # riavvia il webserver
# service apache2 reload    # ricarica la configurazione
# service apache2 stop      # spegne il webserver
# service apache2 start     # avvia il webserver
```

oppure riavviando completamente il Raspberry Pi.

5.1.2 Abilitazione dei permessi utente

Anche se può essere configurato diversamente (come vedremo nella Sezione 5.1.3), Apache serve i files presenti sotto la cartella `/var/www`; i file creati in questa cartella da Apache appartengono all'utente `www-data:www-data` e hanno permessi `664` (ossia `rw-rw-r--`), perciò non possono essere modificati da un utente normale. Tuttavia, aggiungendo l'utente al gruppo `www-data`, gli verranno garantiti i permessi per poter modificare i file in `/var/www`. Rimane tuttavia un problema: se l'utente crea un nuovo file nella suddetta cartella, questo verrà creato con appartenenza a `utente:utente` e permessi `644` (cioè `rw-r--r--`), e ciò implica che Apache non può più modificarlo. Perciò, affinché anche i file creati dall'utente possano essere scritti sia da Apache, sia dall'utente, si dovrà fare in modo che i file che crea abbiano appartenenza al gruppo `www-data` e permessi `664` (ossia `rw-rw-r--`).

Per fare questo, ottenere il GID (*Group IDentifier*) del gruppo `www-data`, con il seguente comando:

```
$ cat /etc/group | grep www-data | cut -d: -f3
```

supponendo che il GID sia *33*, modificare il file `/etc/passwd` alla riga dell'utente interessato, sostituendo il campo relativo al gruppo base (ossia il quarto) con il GID *33*, affinché assomigli a:

```
glgprograms:x:1007:33:GLG Programs,,,:/home/glgprograms:/bin/zsh
```

Dopodiché, abilitare la creazione di file con permessi `664` aggiungendo al file `/etc/profile` la linea

```
umask 002
```

NOTA: per rendere effettive queste modifiche, eseguire nuovamente il login.

ATTENZIONE: se si considera di modificare i file anche montando la cartella da remoto tramite il protocollo `sftp` (con `sshfs` oppure `FileZilla`, Sezione 7.1), modificare anche il file `/etc/ssh/sshd_config` alla seguente linea, aggiungendo *-u 002*:

```
Subsystem sftp /usr/lib/openssh/sftp-server -u 002
```

5.1.3 Configurazione di un Virtual Host

Può sempre risultare utile poter ospitare più siti web sulla stessa macchina, oppure semplicemente separare i contenuti pubblici del sito dalle pagine di amministrazione. Inoltre, si può presentare la necessità di consentire un accesso sicuro al sito, tramite certificato SSL e protocollo *HTTPS* o di limitare accessi e larghezza di banda: in tutti questi casi, è necessario creare degli host virtuali.

NOTA: la creazione di un host virtuale è a discrezione dell'amministratore del server; se si è sicuri di non avere nessuna necessità di abilitare, anche in futuro, una delle suddette caratteristiche, allora si può saltare direttamente alla Sezione 5.1.4. In ogni caso, alcune modifiche minori possono essere effettuate anche sul file di configurazione iniziale, `/etc/apache2/sites-available/default`.

Prima di configurare un host virtuale, assicuriamoci che la configurazione base di Apache non possa andare in conflitto con la nostra; in particolare, commentare tutte le linee `Listen` all'interno del file `/etc/apache2/ports.conf`, se queste non sono già commentate (la presenza di queste linee in un file diverso da quello degli host virtuali veri e propri, è causa di notevoli mal di testa per molti, compreso il webmaster sottoscritto; una veloce ricerca con Google su qualche forum chiarirà immediatamente questo concetto).

Per configurare un host virtuale, creiamo due cartelle all'interno di `/var/www`, che ospiteranno i due siti, ad esempio:

```
$ cd /var/www
$ mkdir glgprograms.it
$ mkdir glgprograms.glg
```

dopodiché, configuriamo Apache per poter usare i due host virtuali, creando due file (che, per semplicità, chiameremo come le due cartelle) all'interno di `/etc/apache2/sites-available`:

```
# cd /etc/apache2/sites-available
# touch glgprograms.it glgprograms.glg
```

All'interno di `glgprograms.it` scriviamo:

```
Listen 80
NameVirtualHost *:80
<VirtualHost *:80>
  ServerAdmin myemail@provider.com
  ServerName www.glgprograms.it
  ServerAlias glgprograms.it
  DocumentRoot "/var/www/glgprograms.it"
  DirectoryIndex index.html index.htm index.php
</VirtualHost>
```

dove

- *80* è la porta sulla quale il webserver Apache deve rimanere in attesa delle connessioni dall'esterno;
- *myemail@provider.com* è l'indirizzo email dell'amministratore del webserver, che Apache utilizza nei messaggi di errore e/o per notificare eventuali malfunzionamenti all'amministratore stesso;
- *ServerName* e *ServerAlias* indicano i dominî ai quali risponde il webserver;
- *"/var/www/glgprograms.it"* è la cartella all'interno della quale si trovano i file del sito

Analogamente, creiamo il file di configurazione per l'altro sito, modificando la porta di ascolto e la DocumentRoot, ad esempio, *81* e */var/www/glgprograms.glg*. Per ulteriori dettagli, si rimanda alla risorsa ufficiale di Apache.
Infine, disabilitiamo l'host di default, e abilitiamo i nostri:

```
# a2dissite default
# a2ensite glgprograms.it
# a2ensite glgprograms.glg
```

Riavviando Apache, noteremo che, interrogandolo con un browser all'indirizzo *http://dominio.it/*, risponderà col contenuto del primo virtual host, mentre interrogandolo all'indirizzo *http://dominio.it:81* risponderà col contenuto del secondo virtual host (dove per *dominio.it* si intende il dominio vero e proprio associato al Raspberry Pi, oppure il suo indirizzo IP).

Nella Sezione 5.1.4, vedremo una primitiva applicazione pratica per gli host virtuali. Per applicazioni più avanzate, consultare la Sezione 8.4.

5.1.4 Ottimizzazione di Apache

Dal momento che le ADSL casalinghe non forniscono molta larghezza di banda, è molto probabile che anche un solo visitatore possa saturarvi la linea scaricando immagini o altri contenuti particolarmente grandi. Per evitare di trovarsi impossibilitati a navigare

quando questo accade, si può limitare la banda utilizzata da Apache, così da lasciare sempre almeno qualche K/s libero per la navigazione personale. Per limitare la banda utilizzata da Apache, si può utilizzare il modulo `mod_bw`.
Per installare `mod_bw`:

```
# apt-get install libapache2-mod-bw
```

dopodiché, all'interno della direttiva `<VirtualHost>` dell'host virtuale interessato, aggiungere, facendo particolare attenzione alla distinzione tra lettere maiuscole e minuscole:

```
BandwidthModule On
ForceBandWidthModule On
Bandwidth all 35840
MinBandwidth all -1
```

dove *35840* è la massima larghezza di banda, espressa in bytes al secondo, che Apache può utilizzare complessivamente (ossia indipendentemente da quanti utenti sono collegati) quando serve quel sito; nel caso specifico, Apache invierà dati con una frequenza massima di 35 K/s ($35 \cdot 1024$), velocità ragionevole con una comune ADSL Telecom da circa 40 K/s in upload.

5.2 PHP

PHP è un potente linguaggio di programmazione interpretato a tipizzazione debole, caratteristiche che lo rendono perfetto per creare piccole e semplici applicazioni in maniera veloce ed elastica. Disponendo inoltre di numerose funzioni dedicate all'elaborazione e al controllo dei dati provenienti da maschere, all'integrazione con i database relazionali SQL e ad altre procedure di uso comune nella programmazione orientata ai siti web, si integra perfettamente con il webserver Apache.

5.2.1 Installazione di PHP

Per installare PHP:

```
# apt-get install php5 libapache2-mod-php5 php5-intl php5-mcrypt \
  php5-curl php5-gd php5-sqlite
```

5.2.2 Ottimizzazione di PHP con APC (cache)

Poiché il Raspberry Pi non ha una grande capacità di calcolo, eseguire ripetutamente lo stesso programma, anche per un piccolo sito, quando questo produce sempre lo stesso risultato, può essere inutile causa di generali rallentamenti nella navigazione, specialmente se si collegano diversi utenti. Pertanto, per rendere il sito più veloce e reattivo, può essere utile installare APC, un performante strumento di *cache*.
Per compilare e installare APC:

```
# apt-get install php-pear php5-dev apache2-prefork-dev build-essential \
  make && pecl install apc
```

Per abilitare APC, modificare il file `/etc/php5/apache2/php.ini` aggiungendo alla sezione *Dynamic Extension* la seguente direttiva:

```
extension=apc.so
```

5.2.3 Invio di email con PHP e sendmail

Per ricevere commenti o messaggi dai visitatori del sito senza dover necessariamente lasciare il proprio indirizzo email in chiaro alla mercé dello spam, è possibile utilizzare la funzione `mail()` di PHP, la quale, appunto, permette di inviare email in maniera automatica. Per poter inviare email, è necessario `sendmail`. Per installare `sendmail`:

```
# apt-get install sendmail
```

dopodiché configurare PHP modificando il file `/etc/php5/apache2/php.ini`, alla seguente linea, come suggerito, del resto, dal commento:

```
sendmail_path = /usr/sbin/sendmail -t -i
```

5.3 MySQL

MySQL è uno tra i migliori database relazionali SQL esistenti oggigiorno, e, sebbene con un semplice Raspberry Pi sia pressocché impossibile sfruttarne appieno tutte le potenzialità, rimane comunque un ottimo strumento per la gestione di un sito web, considerando anche il fatto che è facilmente scalabile e portabile, e rimane comunque facile da spostare, in caso di necessità, su una macchina più potente affiancata al Raspberry Pi.

5.3.1 Installazione di MySQL

Per installare MySQL:

```
# apt-get install mysql-server mysql-client php5-mysql
```

e per accedere alla riga di comando:

```
$ mysql -u mysqluser -p
```

dove *mysqluser* è il nome dell'utente MySQL col quale si desidera accedere al database.

NOTA: non ritengo necessario aggiungere un ulteriore supporto per la cache delle query SQL, in quanto quello integrato con MySQL è sufficiente.

5.3.2 phpMyAdmin

Sebbene sia possibile interrogare il database semplicemente utilizzando la riga di comando, come descritto sopra, è sicuramente molto più comodo e veloce l'utilizzo di un software esterno per la gestione del database con un'interfaccia grafica. A questo scopo, esistono numerosi software, tra cui `Emma` oppure il più conosciuto e intuitivo `phpMyAdmin`, che è quello che andremo ad installare.
Per installare `phpmyadmin`:

```
# apt-get install phpmyadmin
```

dopodiché, sarà possibile interagire graficamente col database aprendo la pagina di `phpmyadmin`, interrogandolo con un browser all'indirizzo *http://dominio/phpmyadmin*. Poiché questo è l'indirizzo standard, consiglio caldamente di modificarlo, come spiegato nella Sezione 8.3.

TRUCCO: `phpMyAdmin` è un software estremamente potente, ma per questa ragione pesante e un po' lento; per consultare il database o eseguire velocemente delle query, si può utilizzare anche `phpMiniAdmin` (Sezione 5.3.3).

5.3.3 phpMiniAdmin

Installare `phpMiniAdmin` è molto semplice: infatti è sufficiente scaricare un solo script PHP dal sito del progetto ufficiale phpminiadmin.sourceforge.net e copiarlo in una cartella del webserver, dopodiché sarà accessibile come una qualsiasi normalissima altra pagina.

Per configurare `phpMiniAdmin`, inserire le proprie credenziali di accesso al database nell'array `$DBDEF` all'interno dello stesso script `phpminiadmin.php`.

6 Installazione sulla scheda SD

Per copiare l'immagine del sistema sulla scheda SD occorre innanzitutto individuare il device node che corrisponde alla scheda SD:

```
# fdisk -l
```

Supponiamo che la scheda SD sia `/dev/sdb`, allora

```
# dd if=2013-02-09-wheezy-raspbian.img of=/dev/sdb conv=sync
```

Per comodità, possiamo vedere come procede la copia inviando un segnale di tipo USR1 a `dd`, in questo modo:

```
kill -USR1 $(ps -A | grep -w dd | awk '{print $1;}')
```

In ogni caso, armiamoci di pazienza perché può essere necessario aspettare anche una mezz'ora.

Una volta copiata l'immagine sulla scheda SD, è necessario espandere il filesystem su tutta la scheda, in modo da sfruttarne tutta la capacità (altrimenti lo spazio a disposizione è di ∼2G). Per far ciò, si può utilizzare il tool presente sull'immagine, oppure più comodamente espanderlo con GParted direttamente dal nostro computer. Se non abbiamo GParted, installiamolo e avviamolo:

```
# apt-get install gparted   # Debian e derivate
# pacman -S gparted         # Archlinux e derivate
# gparted &
```

Individuiamo la nostra scheda SD (per esempio, `/dev/sdb`) e la partizione da espandere (nel caso dell'esempio, `sdb2`); facciamoci click col tasto destro del mouse e scegliamo *Ridimensiona/Sposta*: nella finestra che compare, allarghiamo la partizione fino a riempire tutto lo spazio a disposizione e confermiamo le modifiche. A questo punto, armiamoci di pazienza e attendiamo il ridimensionamento del filesystem – anche se in genere questa operazione richiede al massimo qualche minuto.

6.1 Backup della scheda SD e ripristino

`dd` può essere analogamente utilizzato anche per fare una copia di sicurezza (backup) della scheda SD e quindi di tutte le modifiche che vi abbiamo effettuato sopra (per risparmiare spazio, l'immagine copia di sicurezza viene compressa con `gzip`). Prima di eseguire questi comandi, è necessario smontare la scheda SD.

```
# dd if=/dev/sdb conv=sync | gzip -c > backup.gz
```

Per ripristinare l'immagine di backup precedentemente creata tramite l'ausilio di `dd` e `gzip` sul disco `/dev/sdb`:

```
# cat backup.gz | gzip -d > /dev/sdb
```

7 Accesso al Raspberry Pi

Una volta disinstallata l'interfaccia grafica e tutti gli inutili (per il nostro scopo) strumenti che dipendono da essa, per accedere al Raspberry Pi sarà necessario utilizzare strumenti diversi da un monitor e una tastiera. In questa sezione è descritto il modo in cui si può operare da remoto sul Raspberry Pi.

7.1 SSH

SSH, acronimo ricorsivo di Secure Shell, è uno strumento a riga di comando, utile, versatile e soprattutto sicuro, per accedere da remoto ad una macchina. Per collegarsi al Raspberry Pi tramite SSH è necessario installare il client corrispondente sul proprio computer:

```
# apt-get install openssh-client    # Debian e derivate
# pacman -S openssh                 # Arch Linux e derivate
```

e, per collegarsi,

```
$ ssh utente@hostname
```

dove, `utente` è il nome dell'utente che avete creato sul Raspberry Pi e `hostname` è il suo indirizzo IP (o eventuale nome di dominio). Ad esempio, *$ ssh glgprograms@127.0.0.1*

7.2 Copia di file sul server

Il protocollo di SSH può essere sfruttato per trasferire in maniera sicura, oltre ai comandi, anche file, facendo uso dello strumento che si preferisce (vedere l'elenco che segue).

NOTA: per il controllo dei permessi, vedere anche la Sezione 5.1.2.

7.2.1 scp

`scp` permette di copiare file (o gruppi di file appositamente selezionati tramite caratteri jolly) da/su un computer remoto, nel nostro caso il Raspberry Pi. La sintassi di `scp` è:

```
$ scp utente@hostname:/percorso/file/remoto /percorso/cartella/locale
  oppure
$ scp /percorso/file/locale utente@hostname:/percorso/cartella/remota
```

dove il primo argomento indica il file da copiare, mentre il secondo la destinazione, ad esempio *$ scp glgprograms@glgprograms.it:/srv/http/index.php ~/Documenti/*

7.2.2 sshfs

`sshfs` è forse lo strumento più comodo per trasferire files in maniera sicura da/a un computer remoto. Attraverso `sshfs` è possibile, infatti, montare una cartella di un computer remoto direttamente nel filesystem locale, rendendo così trasparente e automatico il suo accesso da tutti i programmi. Per montare una cartella di un computer remoto sul computer locale, installare `sshfs`:

```
# apt-get install sshfs   # Debian e derivate
# pacman -S sshfs         # Arch Linux e derivate
```

e, per collegarsi,

```
$ mkdir -p ~/discoremoto
$ sshfs utente@hostname:/percorso/cartella/remota ~/discoremoto/
```

Così, aprendo la cartella locale ~/discoremoto/ si potrà operare direttamente sul contenuto della cartella remota specificata.

Molti file manager permettono di eseguire questa operazione anche tramite una comoda interfaccia grafica (ad esempio, *Connessione remota cartella* di `Nautilus` (gestore di file di Gnome) oppure, con `pcmanfm` e con `Thunar` (gestori di file di LXDE e XFCE rispettivamente), semplicemente digitando nella barra degli indirizzi:

```
sftp://utente@hostname:/percorso/cartella/remota
```

7.2.3 FileZilla

Infine, FileZilla è un programma ad interfaccia grafica (disponibile anche per Microsoft Windows e Apple Mac OS) che permette lo scambio di files tramite SSH. Per installare FileZilla:

```
# apt-get install filezilla     # Debian e derivate
# pacman -S filezilla           # Arch Linux e derivate
```

e, per collegarsi, inserire nel campo *Host* in alto a sinistra

```
sftp://utente:password@hostname
```

e premere *Invio*.

8 Sicurezza e spam

Una volta reso il sito pubblicamente accessibile all'interno della vasta rete Internet, sarà indispensabile prendere delle precauzioni aggiuntive per evitare che qualche malintenzionato possa accedervi abusivamente. Le precauzioni di seguito elencate sono solo alcune delle più comuni: del resto, non possiamo certamente descrivervi dettagliatamente come è protetto il nostro server e come si *buca*!

Avvertenza: questa sezione dà solamente alcune indicazioni generali su come proteggere il proprio server. GLG Programs non è da ritenersi responsabile per nessun motivo nel caso in cui il proprio server venga violato, anche dopo aver seguito accuratamente tutti i consigli qui elencati.

8.1 Sicurezza di SSH

Come abbiamo visto nella Sezione 7.1, SSH è uno strumento molto potente per la gestione da remoto del nostro server; tuttavia, è anche alquanto pericoloso lasciare che qualcuno possa trovare la vostra password e accedere al server.
Per limitare i tentativi di accesso al server, è utile cambiare la porta sulla quale SSH accetta le connessioni, semplicemente modificando il file `/etc/ssh/sshd_config` alla linea che riporta:

```
Port 22
```

8.2 Sicurezza di Apache con .htaccess

Per negare o permettere l'accesso a determinati utenti oppure a determinati cartelle o files del sito, è possibile inserire delle direttive all'interno dei file `.htaccess` (per maggiori informazioni, consultare un motore di ricerca).

8.2.1 Abilitare gli .htaccess

Per permettere l'utilizzo di questi file, è indispensabile abilitarli nella configurazione dell'host virtuale per il quale si desidera renderli attivi, aggiungendo le seguenti righe:

```
<Directory "/var/www/glgprograms.it/">
  Options FollowSymLinks MultiViews
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>
```

dove */var/www/glgprograms.it* è la cartella entro la quale si desidera abilitare l'utilizzo dei files `.htaccess`.

8.2.2 Esempio di .htaccess

Ad esempio, per negare a chiunque l'accesso ad una determinata cartella del sito, si può creare un file `.htaccess` all'interno di tale cartella con contenuto:

```
deny from all
```

8.3 Sicurezza di phpMyAdmin

8.3.1 Prevenire attacchi dizionario e forza bruta

Per prevenire attacchi al proprio database, è consigliabile cambiare la pagina di accesso a phpMyAdmin, modificando il file `/etc/phpmyadmin/apache.conf` alla riga:

```
Alias /phpSecureAdmin /usr/share/phpmyadmin
```

dove */phpSecureAdmin* è l'indirizzo al quale deve rispondere `phpMyAdmin`.

8.3.2 Prevenire l'accesso come amministratore

Nella sfortunata ipotesi in cui qualche malintenzionato riuscisse a trovare l'indirizzo a cui risponde `phpMyAdmin`, e riuscisse a trovare una delle nostre password, possiamo comunque negargli l'accesso come amministratore, semplicemente aggiungendo al file di configurazione `/etc/phpmyadmin/config.inc.php` la linea:

```
$cfg['Servers'][$i]['AllowRoot'] = FALSE;
```

8.3.3 Prevenire attacchi man-in-the-middle

Abilitando SSL (come descritto nella Sezione 8.4), è possibile configurare phpMyAdmin affinché richieda sempre una connessione HTTPS per avviarsi, semplicemente aggiungendo al file `/etc/phpmyadmin/config.inc.php` la linea:

```
$cfg['ForceSSL'] = TRUE;
```

8.4 Connessione SSL con protocollo HTTPS

Per garantire sicurezza ai visitatori del sito, oppure per rendere più sicuro l'accesso alle pagine di amministrazione, si può crittografare la connessione al webserver utilizzando SSL con protocollo HTTPS. Per garantire questo tipo di accesso, è necessario avere un certificato, che può essere acquistato (o concesso gratuitamente in prova per un periodo limitato, circa 2-4 settimane) da un'azienda autorizzata, oppure, più semplicemente, può essere creato al volo e autofirmato. In questo secondo caso, è necessario tenere presente che, normalmente, i comuni browser visualizzano un messaggio di avviso quando si naviga su un sito la cui connessione crittografata è consentita da un certificato autofirmato: pertanto, consiglio di utilizzarlo solamente per le pagine di amministrazione del sito – non accessibili ai comuni visitatori e ai motori di ricerca (Sezione 8.5).

Per poter utilizzare un certificato SSL, occorre installare `openssl`:

```
# apt-get install openssl
```

Se si è acquistato un certificato da un'autorità esterna, è possibile abilitarlo saltando direttamente alla Sezione 8.4.2, altrimenti procedere con la prossima Sezione 8.4.1.

8.4.1 Generazione di un certificato SSL

Secondo quanto spiegato dettagliatamente anche in questa guida del wiki ufficiale di Debian, creiamo una cartella in cui tenere i certificati, generiamoli, e rendiamoli accessibili solamente a `openssl`:

```
# mkdir -p /etc/ssl/localcerts
# openssl req -new -x509 -days 365 -nodes -out \
  /etc/ssl/localcerts/apache.pem -keyout /etc/ssl/localcerts/apache.key
# chmod 600 /etc/ssl/localcerts/apache*
```

Durante la creazione del certificato, generiamo quanta più *entropia* possibile, impegnando il Raspberry Pi in varie attività (potrà sembrare *strano*, ma ciò rende il certificato più robusto); saranno richieste interattivamente alcune informazioni circa il sito: in particolare, alla richiesta del *Common Name (e.g. server FQDN or YOUR name)* inseriamo il dominio al quale risponde Apache (ad esempio, *glgprograms.it*).

8.4.2 Abilitazione del certificato SSL

Una volta creato il certificato, aggiungiamo il modulo SSL ad Apache:

```
# a2enmod ssl
```

e infine configuriamo un host virtuale che utilizzi il certificato, abilitandolo all'accettazione delle connessioni sulla porta `443` e inserendo nelle direttive `<VirtualHost>` quanto segue:

```
SSLEngine On
SSLCertificateFile /etc/ssl/localcerts/apache.pem
SSLCertificateKeyFile /etc/ssl/localcerts/apache.key
```

Una volta riavviato Apache, interrogandolo con un browser all'indirizzo *https://dominio/*, potremmo navigare sito al riparo da occhi indiscreti e attacchi *man in the middle* (sempre ammesso di ricordarsi di controllare che il certificato sia il nostro).

8.5 robots.txt

Nella gestione di un sito può essere utile rendere raggiungibili alcune pagine per effettuare dei test, ma senza che queste vengano indicizzate dai motori di ricerca. Per evitare che Google, Bing o altri rendano visibili contenuti che debbono restare nascosti, è possibile comunicare loro quali pagine (o cartelle) non devono indicizzare. Per fare questo, bisogna creare un file, denominato `robots.txt`, che deve essere collocato nella cartella radice del nostro webserver, in modo che sia raggiungibile, ad esempio, attraverso un indirizzo del tipo *http://www.glgprograms.it/robots.txt*.
Volendo evitare che la cartella `amministrazione` venga indicizzata da un qualsiasi motore di ricerca, il file `robots.txt` deve contenere:

```
User-agent: *
Disallow: /amministrazione/
```

dove *User-agent: ** indica i bot dei motori di ricerca che devono seguire le regole, identificati per *user agent* (nel caso specifico, tutti), mentre i *Disallow:*, che possono essere ripetuti, indicano i file/cartelle che devono essere esclusi dall'indicizzazione.

Attenzione: in questo modo le pagine vengono nascoste agli occhi dei motori di ricerca, ma sono comunque accessibili se si conosce il loro indirizzo – per renderle inaccessibili, usare un file `.htaccess`.

Attenzione: non è garantito che i motori di ricerca che visitano il nostro sito rispettino queste regole.

9 Sitografia

Siti che sono stati utili:

- http://www.raspberrypi.org/phpBB3/viewtopic.php?t=37386&p=314321
- http://www.cyber-space.it/blog/webmaster/configurare-php-per-inviare-mail-con-se 938/
- http://forum.directadmin.com/showthread.php?t=14782
- http://www.debian-administration.org/articles/412
- http://wiki.debian.org/Self-Signed_Certificate
- http://linuxaria.com/article/linux-shell-understanding-umask-with-examples? lang=it
- https://bbs.archlinux.org/viewtopic.php?id=140734
- http://wiki.debian.org/LaMp

— End of Document —