ISTITUTO NAZIONALE DI FISICA NUCLEARE

INFN-14-18/CCR
10th December 2014

ASPETTI DI SICUREZZA NELLA GESTIONE DI SITI WEB IN AMBIENTE ACCADEMICO

Michele Michelotto[1], Dario Vettore[2]

[1)]INFN-Sezione di Padova, Via F. Marzolo, 8, I-35131 Padova, Italy
[2)]Garr presso INFN Sezione di Padova Via F. Marzolo, 8, I-35131 Padova, Italy

Abstract

L'obiettivo di questa guida è fornire un supporto da un punto di vista sistemistico per la configurazione di Web Server in ambito accademico Si intende seguire l'amministratore nella configurazione e installazione offrendogli dei consigli nel caso in cui ci possano essere dubbi ed evidenziando i punti più critici

CCR-51/2014/P

Pubblicato da **SIS–Pubblicazioni**
Laboratori Nazionali di Frasca

1 A CHI È RIVOLTA QUESTA GUIDA

L'obbiettivo di questa guida è fornire un supporto da un punto di vista sistemistico per la configurazione di un *Web Server* in ambito prettamente accademico. Si intende seguire l'amministratore nella configurazione e installazione offrendogli dei consigli nel caso in cui ci possano essere dei dubbi e evidenziando i punti più critici.

1.1 I server Web

Il termine *Web Server* indica un componente hardware o software che offre un contenuto di tipo testuale, multimediale o altro, il quale può essere acceduto tramite internet. La funzione principale è quella di fornire delle pagine web ai possibili client. La comunicazione avviene tramite il protocollo HTTP oppure HTTPS.

Tipicamente i web server offrono la possibilità di poter gestire uno o più virtual host all'interno dello stesso (in questa maniera è possibile utilizzare lo stesso indirizzo IP), di poter condividere file con una dimensione maggiore di 2GB, di poter suddividere la banda in modo da non saturare la rete, di supportare la generazione di pagine web dinamiche e di installare diversi siti web con un'unica installazione del server http, semplificandone la gestione sopratutto per quanto riguarda gli aspetti di sicurezza. Attualmente sono quattro i tipi di *Web Server* più diffusi[1] riassunti nella tabella 1:

Tabella 1: Diffusione dei Server Web nel mondo

Nome	Produttore	Diffusione in %
Apache	Apache	37,74
IIS	Microsoft	33,04
nginx	nginx	15,25
GWS	Google	2,19

mentre nella tabella 2 viene mostrata la diffusione in ambito INFN [2],su cui abbiamo potuto effettuare dei test più approfonditi e che assumiamo possa essere indicativo pure dal punto di vista del GARR.

Vista l'enorme diffusione di **Apache**, lo analizziamo in dettaglio.

Apache è il software che viene utilizzato nella stragrande maggioranza dei *Web Server* , quindi ci siamo occupati di studiare ed analizzare una corretta configurazione di quest'ultimo.

[1] https://en.wikipedia.org/wiki/Web_server

[2] https://agenda.infn.it/conferenceOtherViews.py?view=standard&confId=7915

Tabella 2: Diffusione dei Server Web nelle sedi INFN

Nome	Produttore	Diffusione in %
Apache	Apache	97,65
Altri	Altri	2,35

2 APACHE

2.1 Installazione

Per prima cosa è necessario installare il pacchetto *Apache*. Per esser sicuri della corretta installazione è possibile verificare recandosi con un browser alla pagina *http://127.0.0.1* oppure *http://nomeserver* e se ci appare un testo simile a questo:

```
It works!
This is the default web page for this server.
The web server software is running but no content has been added, yet.
```

Sono necessari altri pacchetti se andremo ad abilitare i contenuti dinamici nel nostro *Web Server* , ma questo argomento verrà trattato nel paragrafo 3.

2.2 Stop, Start & Restart

Per arrestare *Apache* ci sono due modi; con il primo metodo puoi utilizzare il comando *kill*, es:. red-hat:

```
kill -TERM ‘cat /var/run/httpd.pid‘
```

una differente distribuzione, potrebbe avere un diverso path. Un altro modo è usando il parametro *-k* con le opzioni: *stop, restart, graceful and graceful-stop* ovvero:

```
apachectl -k stop
apachectl -k restart
apachectl -k graceful-stop
apachectl -k graceful-restart
```

2.3 Configurazione

2.3.1 httpd.conf

Il file **httpd.conf**, tipicamente situato in **/etc/httpd/conf/** è il file di configurazione principale, dove si configura il *Web Server* in gran parte delle sue funzionalità. È suddiviso in tre sezioni:

- Global Environment
- Main server configuration
- Virtual Host

2.3.2 Global Environment

È la sezione delle impostazioni generali del server che riguardano il suo complessivo funzionamento. Una delle prime direttive che incontriamo analizzando il file è **ServerRoot**, dove impostiamo le directory dove sono situati i file di configurazione per il corretto funzionamento di *Apache*. Di seguito troviamo la direttiva **PidFile** ovvero il file dove il *Web Server* andrà a registrare il proprio numero di processo quando verrà avviato. Il parametro **Timeout** indica il tempo che intercorre tra la richiesta ed l'invio di un messaggio di errore quando questa non viene soddisfatta. La direttiva **KeepAlive** ha due parametri **Off** e **On**, se settata ad **On** permette l'invio di file multipli utilizzando la stessa connessione TCP, altrimenti aprirà una nuova connessione ad ogni richiesta.

Opzione consigliata?
On, il beneficio di tenere questo valore ad On, consiste nel fatto che un client è capace di effettuare più di una richiesta verso il tuo server, senza creare una nuova connessione TCP (che di solito consiste in una 3-way handshake), il problema che potrebbe derivare nell'avere avendo un limite delle connessioni per esempio a 300, è che nel caso ci siano 300 connessione attive tutti gli altri client andrebbero in timeout error. Disabilitare KeepAlive (Off) vorrebbe dire forzare ad avere una connessione per richiesta, fortemente penalizzante.

A KeepAlive segue l'istruzione **MaxKeepAliveRequests**: qui si può settare il numero minimo di richieste (100 per default) per singola connessione TCP. Una direttiva importante è anche **MaxRequestPerChild**: con questa direttiva è possibile impostare la quantità di richieste che possono essere prese in carico da un singolo processo figlio del processo httpd. Se indichiamo il valore con 0, avremo che i processi figli potranno avere un numero illimitato di richieste.

Opzione consigliata?
Fortemente consigliato per evitare sovraccarichi inserire un valore, 4000 è un valore accettabile, ma non infinito, cioè non **0**.

Il supporto per i **DSO** (**D**ynamic **S**hared **O**bject) è consigliato se si devono abilitare moduli o funzionalità addizionali, questa direttiva si divide in due parti: **LoadModule** e **AddModule**; quando viene aggiunto un nuovo modulo bisogna agire in entrambi le parti allo stesso modo. Per esempio aggiungendo il modulo PHP5:

```
LoadModule php5_module "PATH/php/php5apache.dll"
AddModule mod_php5.c
```

2.3.3 Main server configuration

Come prima direttiva incontriamo il settaggio della porta di ascolto. Per default *Apache* è in ascolto sulla porta **80**, ma è possibile modificare il numero di questa porta, per esempio se sono presenti più *Web Server* sullo stesso sistema. L'uso della porta di default espone il nostro server a scansioni da parte di hacker malintenzionati che effettuano dei test a tappeto sulle porte 80, per questo motivo l'amministratore di sistema dovrebbe permettere l'accesso sulla porta 80 solo ai server aggiornati e mettere su altre porte più alte (es. 5104) i server web meno sicuri, come per esempio quelli che contengono siti temporanei o con contenuti attivi.

Si può successivamente impostare l'indirizzo email di posta del **ServerAdmin**, successivamente sono presenti il **ServerName** ovvero il nome della nostra macchina server, (*localhost*

è un valore che si può inserire tranquillamente) ed infine la **DocumentRoot** ovvero la directory di partenza dove sono contenute le nostre pagine web. In linux in genere si ha come **DocumentRoot**:

```
/var/www/html
```

Successivamente troviamo ad esempio:

```
<Directory "PATH_per_la_ROOT">
Options Indexes FollowSymLinks Includes
AllowOverride All
Order allow,deny
Allow from all
</Directory>
```

Queste sono delle direttive (**container**), in altri termini delle *regole*, cioè un insieme di istruzioni, di configurazione del *Web Server* . Sono necessarie se vogliamo configurare parte del nostro *Web Server* in maniera differente. Ad esempio per riservare un'area del nostro web server a particolari tipi di utenti, accessibile solo dal nostro dominio e non dal resto del mondo:

```
1) <Directory /var/www/cgi-bin/licenze>
2)      Order Deny,Allow
3)      Deny from all
4)      Allow from .test.it
5) </Directory>
```

In questo esempio tra le righe 1-5 troviamo le directory interessate dalle nostre direttive, successivamente troviamo (riga 2) l'ordine che devono avere le nostre direttive, l'ordine in questo caso è **Deny,Allow**, in particolare come prima operazione verranno analizzate le direttive **Deny** e successivamente le direttive **Allow**, in maniera più precisa, prima sono analizzate le direttive Deny, se esiste un match la richiesta è **negata**, se non vi è nessun match si procede con l'analisi delle direttive Allow. Se nessuna delle richieste soddisfa nè l'accesso Deny nè Allow allora la richiesta viene processata per default.

Proseguendo troviamo:

```
<IfModule mod_mime.c>
TypesConfig /PATH/mime.types
</IfModule>
```

Un'istruzione come questa significa che nel caso in cui sia abilitato il modulo **mod_ mime.c** dovrà essere caricato il MIME: *mime.types*, è necessario ripetere la prima e l'ultima riga per ogni modulo importato.

Troviamo poi la sezione che riguarda i messaggi di errore, l'errore **500** che sta ad indicare *Internal server error* il **404** ovvero *File Not Found* e il **403** ovvero *Unauthorized User*, un esempio di direttive potrebbe essere:

```
ErrorDocument 500 "The server made a booommmm."
ErrorDocument 404 /missing.html
ErrorDocument 404 "/cgi-bin/missing_handler.pl"
```

2.3.4 Virtual Host

Il termine **Virtual Host** si riferisce alla pratica diventata comune di configurare più di un sito web (*test.mydomain.it test2.mydomain.it*) in una singola macchina.

Che vantaggi posso avere?
In questo modo posso disporre di un certo numero di siti web sulla stessa macchina. Questo comporta un risparmio in termini di risorse fisiche (corrente elettrica), ma anche in termini di manutenzione e sopratutto di **sicurezza** (non occorre configurare N server con conseguente e periodica attività di manutenzione, ma solamente uno.).

I **Virtual Host** sono divisi in due tipi: **IP-Based**, quando vi sono differenti indirizzi IP assegnati a differenti siti web, oppure **Name-Based** quando esistono differenti siti web con lo stesso indirizzo IP, ma con diversi Alias nel DNS.

IP-Based o Name-based?
Si utilizza il metodo IP-Based quando si vogliono applicare direttive basate sull'indirizzo IP. È quindi conveniente utilizzarlo se si hanno differenti siti web gestiti da differenti porte o interfacce. Si utilizza invece il metodo Name-Based quando vogliamo avere più siti web che condividono lo stesso numero IP. Questi ultimi sono tipicamente più facili da configurare, ma è necessario in questo caso configurare sul DNS un alias per ciascun VirtualHost che utilizza un dato IP.

Le princioali direttive di inserimento di un **Virtual Host** sono le seguenti:

```
#<VirtualHost *:80>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

Vediamo alcuni esempi di configurazioni di virtual hosts:
IP-based virtual hosting:

```
<VirtualHost 172.20.30.40>
DocumentRoot /www/example1
ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example.org
</VirtualHost>
```

In questo esempio abbiamo due IP: 172.20.30.**40** e 172.20.30.**50**, che rispondono a due distinti siti web: www.example.**com** e www.example.**org**, con due DocumentRoot differenti /www/**example1** e /www/**example2**.

Name-based virtual hosting:

```
<VirtualHost 172.20.30.50>
DocumentRoot /www/example1
ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example.org
</VirtualHost>
```

In questo caso invece abbiamo lo stesso indirizzo ip: 172.20.30.50 associato a due DocumentRoot differenti /www/**example1** e /www/**example2** e diversi siti web: www.example.**com** e www.example.**org**.

è possibile anche unire entrambe le varianti:

```
#Il server principale 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver

# This is the other address
NameVirtualHost 172.20.30.50

<VirtualHost 172.20.30.50>
DocumentRoot /www/example1
ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example.org
</VirtualHost>
```

In questo modo abbiamo un Virtual Host IP-Based sull'IP 172.20.30.**40** e due VirtualHost Name-Based sull'ip 172.20.30.**50**.

2.3.5 Alias

Il modulo **Alias** di *Apache* permette di mappare differenti parti del file system del *Web Server* nell'albero delle directory, in questa maniera è possibile utilizzare directory locali del file system che non sono presenti nella **DocumentRoot**. In particolare la direttiva Alias segue questa sintassi:

```
Alias /image /ftp/pub/image
<Directory /ftp/pub/image>
Order allow,deny
Allow from all
</Directory>
```

Con questo comando una richiesta a *http://example.com/image/foo.gif* causerà nel *Web Server* il recupero del file *foo.gif* che non è presente nella **DocumentRoot** di *Apache* ma bensì in */ftp/pub/image/foo.gif*.

Sono utili?
S, se vogliamo utilizzare dei contenuti nel nostro sito web che non sono presenti nella DocumentRoot, per esempio per ragioni si sicurezza, o che magari sono mappati in un altro server/pc all'interno della nostra rete.

2.3.6 .htaccess file

I file .htaccess(o distributed configuration files) permettono di inserire delle direttive di *Apache* specifiche riguardanti: i file; la gestione degli errori; configurazione di PHP, l'utilizzo più diffuso però risulta essere per l'autenticazione, applicate alla directory corrente. Tutte queste informazioni possono essere presenti nel file **httpd.conf**.

Vantaggio
Inserendo un file con queste direttive non è necessario riavviare il *Web Server* , dopo ogni singola modifica sul file **httpd.conf**.

Svantaggio?
Questo tipo di file viene letto ad ogni richiesta che viene fatta al *Web Server* per quella particolare directory dove risiede il file .htaccess, questo potrebbe sovraccaricare l'esecuzione del sito.

L'effetto di questi files, come detto precedentemente si applica alla sola directory (e le relative sottodirectory) in cui risiedono, gli eventuali .htaccess file presenti nelle sottodirectory avranno priorità maggiore di quello presente nella cartella padre.

Riassumendo, perchè non usarli?
I motivi per cui è sconsigliato l'utilizzo di questi file sono due:

Performance: *Apache* andrà a leggere questi file ad ogni richiesta ed andrà a cercare un file .htaccess in tutte le directory padre (anche se molto spesso si contano sulle dita di una mano).

Sicurezza: In questa maniera è possibile permettere agli utenti di modificare la configurazione del *Web Server* , in maniera tale da non averne più il controllo, è possibile però anche in questo caso permettere agli utenti l'inserimento di un numero limitato di direttive, proprio per garantire un maggior controllo sulla definizione di questi files.

2.3.7 robot.txt file

Robot.txt è un file di testo; ha lo scopo di segnalare ai motori di ricerca di non analizzare ed attraversare alcune directory allo scopo di non farle indicizzare, per esempio per ragioni di privacy. Un esempio molto semplice è il seguente:

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

con il quale si vieta a tutti i robot di cercare in tutto l'albero delle directory. Un robot.txt file più complesso potrebbe essere il seguente:

```
User-agent: *
Disallow: /
User-agent: Googlebot
Disallow: /cgi-bin/
Disallow: /privatedir/
```

Analizzando riga per riga si scopre, che tutti i robot non possono analizzare questa directory, facciamo però un'eccezione per Google, il quale ha accesso completo a tutto, *tranne che a due directory*: **/cgi-bin/** e **/privatedir/**.

Sono utili?
Si possono essere utili se si vogliono che certe sottopagine (ES. che contengono immagini) non vengano indicizzate.

2.4 I log

2.4.1 Error Log

I log degli errori di *Apache*, sono tipicamente salvati nella posizione dettata dalla direttiva **ErrorLog**:

```
ErrorLog /var/log/httpd/error_log
```

In questo log vengono memorizzate informazioni su diagnostica e ogni tipo di errore che avviene durante l'esecuzione delle richieste. Il format predefinito per gli error log è il seguente:

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied
by server configuration: /export/home/live/ap/htdocs/test
```

2.4.2 Access Log

Questo tipo di log memorizza tutte le richieste di accesso che vengono effettuate, in questo caso la direttiva si chiama: **CustomLog**, per esempio:

```
CustomLog logs/access_log
```

2.4.3 Virtual Host

Tramite le direttive del virtual host è possibile differenziare i log tra i vari virtual host. Esempio:

È possibile impostare nelle varie definizioni dei virtual host dei log personalizzati e divisi tra i vari server web, riprendendo l'esempio di prima:

```
<VirtualHost 172.20.30.50>
DocumentRoot /www/example2
ServerName www.example.org
ErrorLog  /var/log/example2_error_log
CustomLog /var/log/example2_access_log
</VirtualHost>
```

2.4.4 Server HTTPS

Quando si ha necessità che la comunicazione tra client e server sia cifrata, è necessario utilizzare il modulo **ssl**, in questo tipo di comunicazione il prefisso dell'URL cambia da *http://* a **https://**, ed è necessario abilitare il modulo **ssl**:

```
sudo a2enmod ssl
```

creiamo una cartella dove andremo a mettere i nostri certificati, come ultimo passo creiamo la richiesta di certificato utilizzando il comando:

```
mkdir /etc/apache/ssl
openssl req -new -x509 -days 365 -nodes -out /etc/apache/ssl/apache.pem
 -keyout /etc/apache/ssl/apache.key
```

il certificato deve essere inviato ad una CA per essere firmato (altrimenti potrebbe essere stato generato da chiunque). E' molto importante che la CA sia tra quelle riconosciute dal browser o come più spesso accade firmate da una delle CA riconosciute dal browser. I certificati firmati da CA non riconosciute devono essere accettate a mano dall'utente quando accede per la prima volta a quel certificato, ma in questo modo si abitua gli utenti poco esperti di sicurezza ad accettare certificati potenzialmente insicuri. Un esempio, i certificati emessi da Terena sono riconosciuti. Successivamente è necessario configurare apache che ascolti sulla porta **443**:

```
NameVirtualHost *:80
Listen 80

<VirtualHost _default_:443>
    NameVirtualHost *:443
    Listen 443
    SSLEngine on
    SSLProtocol all -SSLv2
</IfModule>
```

è necessario poi abilitare e riavviare il server virtuale:

```
sudo a2ensite default-ssl
apachectl -k restart
```

2.5 Consigli

Tenere aggiornato Apache: Sono molte le persone che collaborano al progetto **Apache**, è normale che possano essere scoperti problemi, bachi o altro; per questo è fortemente consigliato tenere aggiornato il software e installare eventuali patch. Un metodo utile per essere costantemente aggiornati potrebbe essere quello di essere iscritti ad alcune mailing list come per esempio quella degli annunci: announce@httpd.apache.org. L'elenco delle altre liste è presente qui: http://httpd.apache.org/lists.html.

Permessi nella ServerRoot: Tipicamente **Apache** viene inizialmente avviato come utente root, e di solito poi cambia in base alla scelta che abbiamo fatto (se è stata fatta) nella direttiva *User*. È opportuna cosa a questo punto modificare le protezioni delle cartelle con i file di configurazione, per non renderle accessibili da utenti non root.

Cosa potrebbe succedere se questi file fossero accessibili?
Si possono verificare alcuni spiacevoli inconvenienti, per esempio qualcuno potrebbe cambiare il binario **httpd** con uno modificato, ed al riavvio successivo potrebbe eseguire del codice diverso, se le cartelle dei log sono scrivibili da un utente non root, qualcuno potrebbe cambiare il file di log con un link simbolico.

Server Side Includes: I comandi Server Side Include (SSI) sono delle direttive inserite all'interno del codice sorgente delle pagine HTML. Non visualizzano nulla nella pagina web, ma eseguono delle istruzioni e il loro output viene inserito nella pagina che contiene il codice sorgente HTML. Anche queste direttive hanno dei potenziali rischi. Il primo riguarda il carico del *Web Server* (se abbiamo più *Web Server* questo è maggiormente visibile), tutti questi comandi infatti vengono interpretati dal *Web Server* . Queste direttive portano con se tutti i rischi riguardanti i file CGI, ovvero la possibilità di eseguire con i permessi dell'utente apache.

Come ci possiamo proteggere?
Ci sono alcune accortezze che si possono applicare per poter utilizzare gli SSI, è possibile limitare l'uso di queste direttive ai soli file es..*shtml* tramite le direttive inserite nel file *httpd.conf*. Un altra accortezza potrebbe essere quella di disabilitare l'uso di script e programmi dalle pagine SSI, cambiando *Include* con *Include NOEXEC* in questo modo disabilitiamo la possibilità di eseguire comandi come:

```
<!--#exec cmd="ls -lsa" -->
oppure
<!--#exec cgi="/cgi-bin/foo.cgi"-->
```

ovvero eseguire un comando all'interno della shell oppure uno script cgi.

Proteggere le impostazioni di sistema: E' consigliato anche impedire la creazione dei file *.htaccess* i quali come ricordo vanno a sovrascrivere le direttive di configurazione. Un modo potrebbe essere:

```
<Directory />
AllowOverride None
</Directory>
```

Dai un occhio a tuoi log: Una particolare attenzione può essere data ai file di log, ad esempio:

```
grep "client denied" error_log | tail -n 100
```

per vedere se ci sono stati dei client denied appunto, ovviamente sono molti altri i comandi che possono essere dati.

Nascondere la versione di Apache: è possibile nascondere la versione di Apache installata nel proprio sistema, è utile nel caso in cui siamo spesso soggetti a scansioni, un attaccante in base alla versione del software installato ha un ristretto numero di vulnerabilità da poter testare, ad ogni modo agendo sul file **httpd.conf** è possibile nascondere la versione, in particolare abbiamo che:

```
ServerTokens Prod displays Server: Apache
ServerTokens Major displays Server: Apache/2
ServerTokens Minor displays Server: Apache/2.2
ServerTokens Min displays Server: Apache/2.2.17
ServerTokens OS displays Server: Apache/2.2.17 (Unix)
ServerTokens Full displays Server: Apache/2.2.17 (Unix) PHP/5.3.5
```

Che opzioni è più adatta?
Sicuramente l'opzione **Prod**, in questa maniera si vede solo la versione del server Apache.

Disabilitare la vista dell'elenco dei file nelle directory: di default apache, mostra l'elenco dei file della DocumentRoot, per evitare questo è necessario aggiungere al file di configurazione le direttive:

```
<Directory /var/www/html>
    Options -Indexes
</Directory>
```

Disabilitare i moduli non in uso: è utile disabilitare i moduli che non sono utilizzati, per rendere al minimo le possibili azioni di attacco o le vulnerabilità possibili.

Non permettere ad Apache di seguire i link simbolici: è consigliato disabilitarlo e dove possibile utilizzare la direttiva SymLinksIfOwnerMatch.

```
+SymLinksIfOwnerMatch
```

Perchè
FollowSymLinks permette la creazione di link malevoli verso directory di altri server web, con l'opzione **+SymLinksIfOwnerMatch**, il nostro server seguirà i link simbolici se e solo se il file o directory di destinazione sono dello stesso utente che ha creato il link.

2.5.1 Riassunto

ServerRoot: definisce la cartella in cui sono presenti i file del server;

DocumentRoot: definisce la cartella in cui sono presenti i file che verranno richiesti al server web;

ServerAdmin: definisce l'indirizzo e-mail da includere nei messaggi di errore inviati al client;

ServerName: contiene informazioni sull'hostname al quale il server **Apache** deve rispondere. Può essere specificato un hostname diverso per ogni Virtual Host;

Listen: definisce la porta sulla quale il server è in ascolto;

User e Group: definiscono l'utente e il gruppo proprietari del processo. Il server **Apache** non dovrebbe mai girare come utente root;

ErrorLog: definisce il percorso per i file di log degli errori;

LockFile: il lock file sul file di log (da modificare solo se il log file di apache2 è montato via NFS);

PidFile: il file che contiene il PID del processo padre **Apache**;

TimeOut: il tempo dopo il quale apache chiude una connessione client in stato idle;

KeepAlive : se impostato ad *On* sono permesse connessioni persistenti, ovvero in grado di servire più di una richiesta di uno stesso client;

MaxKeepAliveRequests: il massimo numero di richieste che in una singola connessione un client può fare al server. Dopo tale numero le richieste verranno scartate;

KeepAliveTimeout: tempo massimo, in secondi, da attendere per consentire di effettuare un'altra richiesta da parte di uno stesso client su una stessa connessione preesistente;

MaxRequestsPerChild: numero massimo di richieste che il server è in grado di processare per ciascun thread.

3 PHP

PHP acronimo di (Hypertext Preprocessor) è un linguaggio di programmazione interpretato, originariamente ideato per la realizzazione di pagine web dinamiche, attualmente è utilizzato per la realizzazione di applicazioni web lato server, ma può essere usato anche per script da riga di comando.

3.1 Installazione

I pacchetti tipici per una installazione di **PHP** sono **php, php-cli, php-common, php-pdo** in una red hat linux, recuperabili con qualsiasi sistema di gestione dei pacchetti yum oppure apt. L'installazione può essere quindi fatta in maniera totalmente automatica.

3.2 Configurazione

Il file di configurazione principale di **PHP** è il file **php.ini**, situato in */etc*. Analizziamo ora le impostazioni più comuni di **PHP**:

memory_limit(integer): questo parametro serve per impostare la dimensione massima in di byte occupabile dallo script, in questa maniera è possibile impedire a script fatti male (**o maligni**) di occupare inutilmente la memoria del server;

post_max_size(integer): Questo parametro imposta la dimensione masssima dei post;

include_path(string): qui è possibile trovare un elenco di directory in cui le funzioni require(), include() e fopen_with_path() cercheranno i files;

upload_max_filesize(integer): la dimensione massima di un file inviato: Es.: 20MB 100MB1GB;

max_execution_time: il tempo massimo di esecuzione di uno script;

display_errors: che può assumere i valori *On* oppure *Off*, per indicare se verranno mostrati a video gli errori;

log_error: è possibile memorizzare i log di php in un file;

disable_functions = [function1, function2... :] è possibile disabilitare alcune funzioni di **PHP** per ovvi motivi di sicurezza

```
log_errors = On
error_reporting = E_ALL & ~E_NOTICE
log_errors_max_len = 4096
error_log = /var/log/php.err
```

Attenzione!
Solitamente dopo le modifiche apportate a questo file è buona cosa effettuare un riavvio dell'interprete php, tipicamente il nostro *Web Server* .

4 MySQL

MySQL è il più diffuso DBMS (DataBase Managemente System), è open source e basato sul linguaggio **SQL**. In DataBase indica un archivio dati, o un insieme di archivi, in cui le informazioni sono strutturate e collegate tra loro secondo un particolare modello logico (Relazionale, Gerarchico, ad oggetti ecc). Tipicamente questi DBMS sono basati su un'architettura di tipo Client/Server.

4.1 Installazione

È possibile scaricare **MySQL** dal sito ufficiale, ma è possibile recuperare i pacchetti anche con qualsiasi sistema di gestione dei pacchetti.Tipicamente i pacchetti necessari sono (mysql-server, mysql-admin).

4.2 Avvio

MySQL è composto dal programma principale **mysqld** e da tutta una serie di tool associati: **mysqladmin, mysqlcheck, mysqldump, mysqlhotcopy, mysqlimport, mysqlshow, mysqlaccess, mysql_zap**; non è compito di questa guida analizzarli tutti, ma verranno discussi ed analizzati i più importanti.

mysqld_safe e **mysql.server**
sono degli script disponibili in ambiente unix, che permettono l'avvio di **MySQL** che aggiungono alcune funzionalità di sicurezza come per esempio riavviare il server quando si presenta un errore e la contemporanea registrazione in un file di log delle informazioni relative all'errore riscontrato.

4.3 Configurazione

Tipicamente il file di configurazione di **MySQL** è */etc/my.cnf* dove ci sono le opzioni generali, *$MYSQL_HOMEmy.cnf* è il file di configurazione relativa ad un server, mentre invece ˜my.cnf contiene le opzioni relative a un utente.

4.4 I permessi

4.4.1 Accesso a MySQL

Il sistema di permessi di accesso ai database è piuttosto complicato, l'utente viene identificato non solo sulla base dell'utente stesso, ma anche dalla macchina da cui si collega. Una conseguenza di questo è che se lo stesso utente si collega da due macchine diverse per **MySQL** non è lo stesso utente. All'interno di **MySQL** , è presente il db **mysql**, che contiene la tabella *user*, consultata per prima quando un utente prova a connettersi a mysql. Quindi in base alla coppia di campi *Host* e *User* ed in base alla *password* inserita verrà autenticato o no. Nella colonna *Host* della tabella *user* è possibile sia inserire nomi di host come: *test.example.it*, oppure indirizzi IP come *1.2.3.4* oppure *151.42.62.0255.255.255.0*. Inoltre è possibile utilizzare wildcard come % ed inserire valori come *%.example.it* dove tutti gli host appartenenti a quel dominio possono accederci. Un altro campo della tabella *user* è il campo *password*, il quale se vuoto, non significa che l'accesso dell'utente verrà sempre autorizzato: in realtà gli verrà sempre negato. La password viene memorizzata con una stringa di 41 caratteri, di cui il primo è un asterisco e i successivi 40 sono la password cifrata con un algoritmo di hashing.

4.4.2 Accesso ai database

Dopo aver correttamente avuto accesso, l'utente deve a sua volta avere i privilegi per poter lavorare con il database. I campi successivi della tabella *user*, come ad esempio (*Select_priv, Insert_priv, Update_priv, ecc*). In ciascuno di questi campi è possibile inserire o il valore *Y* oppure *N*. È intuibile che se è presente il valore *Y* è possibile compiere l'azione.

Attenzione
È fortemente consigliato impostare, per gli utenti che creiamo e che hanno accesso al database tutti valori ad **N**. Se fossero impostati tutti a *Y*, avremmo che ogni utente che creiamo avrebbe accesso a tutti i database presenti e questo è fortemente sconsigliato. Come procedere?

Porre soluzione a questo quesito è molto facile, infatti ci viene in aiuto la tabella **DB**, dove sono raggruppati i diritti di ciascun utente per ciascun database. In questa tabella la chiave è composta dai tre campi: *User*, *Host* e *Db*. Quindi ogni riga stabilisce i diritti che ha un utente, da dove si collega e a quale particolare database. Se la casella *Host* è vuota, entra in gioco la tabella **Host**, che specifica i diritti che ha un utente su un particolare db in base all'host da cui si collega.

4.4.3 Assegnazione dei permessi

GRANT e REVOKE
I diritti di amministrazione ad un database vengono assegnati ad un utente tramite l'utilizzo di questi due comandi (**Grant** e **Revoke**), ma anche con delle normali query SQL di *insert* oppure di *update*.

Alcuni esempi possono essere:

```
GRANT ALL ON Accounts.* TO root@localhost IDENTIFIED BY "password";
GRANT SELECT ON acquisti.* TO test@localhost IDENTIFIED BY "password";
~
REVOKE SELECT on acquisti.* FROM test@localhost
REVOKE ALL PRIVILEGES, GRANT OPTION FROM test@localhost
~
flush privileges;
```

4.5 Programmi di supporto

PhpMyAdmin: tool di supporto completamente visuale, scritto in PHP che semplifica notevolmente all'amminnistratore la gestione di un database **MySQL** ;

Attenzione
È consigliabile scegliere o l'autenticazione http oppure quella basata sui cookie, in quanto l'autenticazione normale comporta la memorizzazione in chiaro della password dell'utente.

MySQL Administrator: programma multipiattaforma che fornisce un'interfaccia grafica per accedere al database. Forse è molto più complicato rispetto al primo tool descritto, ma risulta essere più sicuro, perchè non vulnerabile alle problematiche che può introdurre PhpMyAdmin.

4.6 BackUp & Dump dei database

mysqldump è il comando che ci permette di effettuare il dump del database (tutti o quelli che scegliamo). Un comando semplice e veloce e per comprire anche il file di backup può essere:

```
mysqldump db_name --password="pass" | bzip2 > /mybackups/db_name.bz2
```

5 CMS

I CMS (Content Management System) sono un strumento software installato su un web server che servono per facilitare la pubblicazione di contenuti di siti web. In questo modo non è necessario che il webmaster abbia conoscenze di programmazione web. I più diffusi CMS sono: **Joomla**, Wordpress, Contao, Dokuwiki, Drupal, Mambo, MediaWiki, WordPress, la maggior parte sono basati su PHP e Java, a livello di rete INFN [3]:

Tabella 3: Diffusione dei CMS nelle reti INFN

CMS	Versione	Diffusione in %	Supportato fino a:
Wordpress	3.3.2	5	12 dic 2011
Wordpress	3.5.2	5	11 dic 2012
Wordpress	3.8.1	5	12 dic 2013
Joomla	1.5-1.6	55	11 dic 2012
Drupal	–	25	

Vista la diffusione di Joomla, analizziamo in dettaglio gli aspetti di sicurezza di questo CMS.

5.1 Installazione

Joomla, è reperibile direttamente dal sito ufficiale[4].

5.2 Configurazione

Attenzione
Come ogni applicazione anche Joomla è sensibile ad attacchi di ogni tipo, ma si possono utilizzare alcuni accorgimenti:

Aggiornare Joomla e le sue estensioni costantemente: l'azione più importante che si possa compiere è tenere tutto il software aggiornato. Tipicamente ci sono sempre delle correzioni di bug o eventuali altri problemi di sicurezza, non solo per Joomla, ma anche per le estensioni che vi sono installate. Molti aggiornamenti possono essere compiuti direttamente via interfaccia web di amministratore.

Impostare una password sicura: avere una password sicura è un buon criterio si sicurezza, una password sicura dovrebbe avere dagli 8 ai 10 caratteri, contenere lettere maiuscole, lettere minuscole, caratteri non alfabetici e numeri.

Impostare i diritti di accesso alle directory: MAI usare come permessi **777**, in particolare:

- le directory di Joomla devono essere impostate a **755**

[3] https://agenda.infn.it/conferenceOtherViews.py?view=standard&confId=7915
[4] http://www.joomla.org/

- i file, devono avere i permessi **644**

Usare alcune estensioni ad hoc per la sicurezza: utili sono estensioni come

- jHackGuard
- Akeeba Admin Tools
- jomDefender
- jSecure

Backup: effettuare spesso un backup del vostro sito Joomla

Proteggere la pagina di accesso: rendere non accessibile a chi non serve la pagina di login di Joomla, utilizzando alcune tecniche viste nel paragrafo 2.3.3.