

CORSO DI SICUREZZA DEI SISTEMI WEB E MOBILI

Modulo 4: Sicurezza dei Sistemi Web

Introduzione ad Apache

1 Breve storia

Lo sviluppo di Apache è curato dalla Apache Software Foundation, un'organizzazione no-profit.

Il progetto Apache nacque nel 1995; è un progetto nato per creare un Web server stabile, affidabile e veloce per piattaforme Unix (anche se ora supporta anche Windows). A quel tempo, il server Web più diffuso era il *daemon* HTTP pubblico sviluppato da Rob McCool al NCSA (National Center for Supercomputing Application), Università dell'Illinois. A partire dal 1994 lo sviluppo di questo server si era fermato in quanto il suo autore aveva lasciato l'NCSA, per cui un gruppo di webmaster aveva iniziato a sviluppare dei *patch* in maniera autonoma. Apache nasce nell'aprile 1995 come evoluzione di `httpd 1.3`, ne ingloba le caratteristiche, risolvendo i problemi ed implementando nuove *feature*; da qui il nome: A PAthChy sErver! (in realtà il nome viene dato in onore della tribù di nativi americani).

Nel dicembre 1995 viene rilasciata la versione 1.0 del server Apache completamente indipendente dal codice della NCSA. Il Web server è free e open source. Ora siamo alla versione 2.4.

2 Apache: caratteristiche generali

È un Web server che utilizza il protocollo HTTP per ricevere e servire le richieste del client. Gira come un processo *standalone* e sulla base delle impostazioni contenute nel file di configurazione (`apache2.conf` nella versione di Ubuntu della macchina virtuale che vi è stato suggerito di installare) ed eventualmente in altri file di configurazione permette l'accesso alle risorse che gestisce (pagine HTML).

Ha una struttura modulare per una più semplice ed efficace gestione che permette a chiunque di aggiungere nuove funzionalità al server. I moduli possono venire rimossi o rimpiazzati, oppure se ne possono aggiungere di nuovi.

Ci sono alcuni moduli standard inclusi nella distribuzione, altri sono scritti da sviluppatori esterni e non sono garantiti. Ad esempio i seguenti moduli sono quelli relativi al controllo degli accessi e all'autenticazione:

- `mod_access` (dalla versione 2.1 `mod_authz_host`);
- `mod_auth` (dalla versione 2.1 `mod_auth_basic` e `mod_auth_digest`).

3 Riferimenti

- Pagina web di Apache: <http://httpd.apache.org>
- Pagina di documentazione: <http://httpd.apache.org/docs/>
- Articoli interessanti su Apache: <http://www.apacheweek.com/>
- Breve tutoria scritto bene: <http://wiki.ubuntu-it.org/Server/Web>

4 Configurazione del server

Il WS Apache è configurato inserendo delle direttive (o regole) nei file di configurazione che sono dei file di testo. Il principale file di configurazione di Apache HTTP Server è `apache2.conf` (oppure `httpd.conf` dalla versione 2.4 in poi). La locazione del file è stabilita a tempo di compilazione e dipende dalla distribuzione. Per esempio, nella versione di Ubuntu della Virtual Machine che vi è stata proposta per i laboratori il file è installato in `/etc/apache2/apache2.conf`. Si tratta di un tipico file di testo di configurazione di UNIX con linee di commento che cominciano col carattere `#`.

Alcuni dei parametri di configurazione fondamentali sono:

- `ServerRoot`

Specifica la gerarchia di directory di default per l'installazione di Apache.

- `DocumentRoot`

Stabilisce in che posizione il server Apache deve cercare i file che compongono il sito. Di default tutte le richieste sono girate a questa directory, ma link simbolici ed alias possono essere usati per puntare ad altre locazioni. Nel caso della vostra versione di Ubuntu, il valore predefinito è `/var/www/`.

- `ErrorLog`

Specifica dove si trovano i file dei log di errore. Nel caso della vostra versione di Ubuntu, il valore predefinito è `/var/log/apache2/error.log`.

- `ServerAdmin`

Specifica a quale indirizzo email il sistema deve indirizzare la posta destinata agli amministratori (il valore predefinito è `webmaster@localhost`).

- `Listen`

Specifica la porta (opzionalmente l'indirizzo IP) su cui il server Apache dovrebbe essere in ascolto, il valore predefinito è 80.

- `DirectoryIndex`

Specifica la pagina predefinita proposta dal server alle richieste dell'indice di una directory, specificate attraverso l'uso di una barra (/) come postfisso al nome della directory.

Trovate una descrizione di tutte le possibili opzioni di configurazione nella documentazione ufficiale di Apache.

Si possono aggiungere altri file di configurazione usando la direttiva `Include`. Un file di configurazione incluso per default è il file `/etc/apache2/sites-enabled/000-default` (dove si trovano alcuni dei parametri di configurazione appena descritti).

4.1 Il file `.htaccess`

Apache permette una gestione distribuita e decentralizzata della configurazione mediante l'uso di file "speciali" inseriti nel Web tree. In genere i file speciali sono chiamati `.htaccess` (il nome può venire modificato all'interno del file di configurazione usando la direttiva `AccessFileName`). Il file segue la stessa sintassi degli altri file di configurazione; le direttive in esso contenute si applicano alla directory in cui viene messo il file, e alle sotto-directory.

Se si intende usare anche il file bisogna comunicare al server che deve cercare gli eventuali file `.htaccess` mediante la direttiva `AllowOverride All`.

Va ricordato che, mentre se si fanno delle modifiche al file di configurazione principale va fatto ripartire il server in modo che le legga, il file `.htaccess` viene letto ad ogni richiesta quindi le modifiche hanno effetto immediato.

I file `.htaccess` vengono combinati secondo il principio *most specific takes precedence*: data una richiesta di accesso ad un file, la ricerca di eventuali file `.htaccess` parte dalla directory dove si trova il file e sale fino alla radice del server. Se

non viene trovato alcun file `.htaccess`, per decidere se dare accesso alla risorsa o meno, si utilizza il file di configurazione principale.

5 Direttive per il controllo degli accessi

Le direttive per il controllo degli accessi possono essere:

- *Host-based*: le regole vengono definite in base all'indirizzo IP, al nome di dominio o al valore di una variabile d'ambiente del richiedente;
- *User-based*: le regole definite in base all'identità dell'utente (in questo caso richiedono autenticazione).

5.1 Direttive host-based

Le direttive principali sono:

- `Allow` (= permetti)

Specifica chi può accedere ad una risorsa:

- `Deny` (= nega)

Specifica a chi è negato l'accesso ad una risorsa;

- `Order`

Stabilisce l'ordine di valutazione delle direttive `allow` e `deny`.

Tali direttive sono state sostituite dalle direttive `Require host address` e `Require ip ip.address` dalla versione 2.4 in poi, che comunque rimane retro-compatibile. Per negare l'accesso si inserisce la clausola `not`, come in `Require not ip 192.168.205.13`. Si rimanda alla documentazione relativa alla versione 2.4 per l'intera sintassi dei comandi.

5.1.1 La direttiva `Allow`

La sintassi è la seguente:

```
Allow from host
```

Dove `host` può essere:

- la parola chiave `all` che indica che l'accesso è dato a tutti;
- un nome di dominio completo (come `mioserver.miodominio.it`);
- un nome di dominio parziale (come `miodominio.it` o `.org`);
- un indirizzo IP completo (come `217.199.180.21`);
- un pattern IP (come `217.199.180`);
- una rete/maschera (come `192.20.250.0/255.255.255.0`);
- `env = variabile`, dove `variabile` è una variabile di ambiente basata sull'header HTTP.

Si possono indicare anche una lista di indirizzi o di nomi di dominio.

Alcuni esempi di utilizzo della direttiva:

```
Allow from 10.252.46.165
Allow from host.example.com
Allow from 192.168.205
Allow from phishers.example.com moreidiots.example
Allow from all
Allow from miodominio.it
Allow from .org altrodominio.biz
Allow from 217.171.50 62.110
```

Esempio di uso della direttiva con variabili:

```
BrowserMatch MSIE ms_browser
Allow from env=ms_browser

SetEnvIf Referer www.miosito.it internal_pg
Allow from env=internal_pg
```

5.1.2 La direttiva Deny

La sintassi è la seguente:

```
Deny from host
```

Dove *host* può essere:

- la parola chiave `all` che indica che l'accesso è negato a tutti;
- un nome di dominio completo (come `mioserver.miodominio.it`);
- un nome di dominio parziale (come `miodominio.it` o `.org`);
- un indirizzo IP completo (come `217.199.180.21`);
- un pattern IP (come `217.199.180`);
- una rete/maschera (come `192.20.250.0/255.255.255.0`);
- `env = variabile`, dove *variabile* è una variabile di ambiente basata sull'header HTTP.

Si possono indicare anche una lista di indirizzi o di nomi di dominio.

Alcuni esempi di utilizzo della direttiva:

```
Deny from 10.252.46.165
Deny from host.example.com
Deny from 192.168.205
Deny from phishers.example.com moreidiots.example
Deny from env=GoAway
```

5.1.3 La direttiva Order

La sintassi è la seguente:

```
Order tipo_ordine
```

Dove *tipo_ordine* può essere:

- `Deny,Allow:deny` viene valutata prima di `allow`, l'accesso è permesso di default (**politica aperta**);
- `Allow,Deny:allow` viene valutata prima di `deny`, l'accesso è negato di default (**politica chiusa**);
- `Mutual-failure`: solo gli host della lista di `allow` che non compaiono in `deny` hanno il permesso di accedere.

Un esempio di utilizzo della direttiva `Order` è il seguente:

```
Order Deny,Allow
Deny from all
Allow from dominiofidato.it
```

5.2 Granularità delle regole d'accesso

Nella precedente sezione abbiamo visto la sintassi per specificare come permettere o negare l'accesso ad una risorsa. Quello di cui ora abbiamo bisogno è di essere in grado di specificare a quali risorse (in genere file) tali direttive vadano applicate.

Se le direttive si trovano nel file `.htaccess` c'è già un'indicazione dello scope (ovvero tutti i file contenuti nella directory dove si trova il file `.htaccess`), altrimenti lo si deve specificare mediante l'uso delle seguenti direttive:

- `Directory`: le direttive contenute si applicano alla directory indicata e alle sue sotto-directory;
- `Files`: le direttive si applicano a tutti i file corrispondenti al nome indicato, indipendentemente dalla directory in cui si trovano;
- `FilesMatch`: le direttive si applicano a tutti i file che hanno un nome che corrisponde all'espressione regolare indicata;
- `Location`: si riferisce allo "spazio Web";
- `Limit`: restringe l'uso dei metodi HTTP.

La sintassi delle direttive appena elencate è la seguente (per maggiori dettagli, consultare la documentazione ufficiale; si veda Figura 1 come esempio di pagina della documentazione ufficiale relativa alla direttiva `Files`):

- `<Directory nomeDir> direttive </Directory>`
- `<Files nomefile> direttive </Files>`
- `<FilesMatch regexpr> direttive </FilesMatch>`
- `<Location URL-path> direttive </Location>`
- `<Limit method> direttive principali </Limit>`

Nell'esempio:

```
<Directory /dir_controllata>
Order Allow,Deny
...
</Directory>
```

di partenza si è deciso di mettere una politica chiusa (`Order Allow,Deny`), per cui l'accesso di default è negato.

Ad esempio, in:

```
<Directory /miosito/dir_controllata>
Order Allow,Deny
Allow from dominiofidato.it
Deny from escluso.dominiofidato.it
</Directory>
```

si è specificata una sotto rete da cui si accettano le richieste, a parte che dall' host specifico `escluso.dominiofidato.it`.

Nell'esempio:

```
<Limit POST PUT>
Order Deny,Allow
Deny from all
Allow from dominiofidato.it
```

<Files> Directive

Description: Contains directives that apply to matched filenames

Syntax: <Files filename> ... </Files>

Context: server config, virtual host, directory, .htaccess

Override: All

Status: Core

Module: core

The <Files> directive limits the scope of the enclosed directives by filename. It is comparable to the <Directory> and <Location> directives. It should be matched with a </Files> directive. The directives given within this section will be applied to any object with a basename (last component of filename) matching the specified filename. <Files> sections are processed in the order they appear in the configuration file, after the <Directory> sections and .htaccess files are read, but before <Location> sections. Note that <Files> can be nested inside <Directory> sections to restrict the portion of the filesystem they apply to.

The *filename* argument should include a filename, or a wild-card string, where ? matches any single character, and * matches any sequences of characters:

```
<Files "cat.html">
  # Insert stuff that applies to cat.html here
</Files>

<Files "?at.*">
  # This would apply to cat.html, bat.html, hat.php and so on.
</Files>
```

Regular expressions can also be used, with the addition of the ~ character. For example:

```
<Files ~ "\.(gif|jpe?g|png)$">
```

would match most common Internet graphics formats. <FilesMatch> is preferred, however.

Note that unlike <Directory> and <Location> sections, <Files> sections can be used inside .htaccess files. This allows users to control access to their own files, at a file-by-file level.

See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received

Figura 1: Esempio di pagina della documentazione ufficiale di Apache

```
</Limit>
```

di partenza si è deciso di mettere una politica chiusa: infatti anche se è presente la direttiva `Order Deny,Allow`, di seguito è indicato `Deny from all`, per cui l'accesso di default è negato a tutti coloro che fanno richieste usando i metodi `HTTP POST` e `PUT`. L'accesso è poi dato alle richieste che provengono dal dominio `dominiofidato.it`.

Nell'esempio:

```
<Files nonsitocca.html>
  Order allow,deny
  Deny from all
</Files>
```

l'accesso al file `nonsitocca.html` è negato a tutti.

Importante: Data una richiesta di accesso ad un file, la valutazione per decidere se dare accesso o meno al file parte dalla directory dove si trova il file e sale fino alla radice del server. Come già detto, se presenti, file `.htaccess` vengono combinati secondo il principio *most specific takes precedence*. L'ordine di valutazione caso di utilizzo contemporaneo di diversi "contenitori" è <Directory>, <Files> e <Location>.

6 Usare Apache

Il Web Server Apache2 è disponibile anche per Linux. A seconda della distribuzione, si utilizza un comando diverso per installare Apache. Nel caso di Ubuntu, a linea di comando va inserito il seguente comando (a meno che non si sia già root):

```
sudo apt-get install apache2
```

Ora come fare a sapere se il server è già in esecuzione? Basta eseguire uno dei due comandi seguenti:

- `ps aux | grep apache2`
- `service apache2 status`

Nel caso si debba far partire (o ripartire il server) basta eseguire uno dei due comandi (da `root`!):

- `service apache2 start`
- `service apache2 restart`

Nel caso si apra una finestra del browser nella stessa macchina in cui si è installato il server, per visualizzare le pagine del Web tree del server basta inserire l'URL `http://localhost/` oppure `http://127.0.1.1/`.

Per vedere quale versione di Apache è installata: `apache2 -v`.

Per vedere quali librerie sono già installate: `apache2 -l`.