NETLinK S.a.S.
Progettazione e Realizzazione
Sistemi Informativi

http://www.netlink.it e-mail: info@netlink.it

Apache e la Sicurezza

Introduzione

Purtroppo il web server rappresenta sempre un punto di debolezza in una rete, attraverso di esso spesso si possono effettuare delle vere e proprie intrusioni sul sistema.
Chi gestisce webserver esposti su internet conosce questo problema, risolvibile solo ponendo grande attenzione all'amministrazione del server web stesso, aggiornandolo costantemente specie se vengono segnalati bug.
Sicuramente usare https, quando possibile, garantisce uno standard di sicurezza più elevato, specie nel caso in cui si pubblichino informazioni sensibili, o che comunque necessitano di un accesso selettivo.
In effetti utilizzare la Basic Authentication e creare i file di password con htpasswd non impedisce che le connessioni vengano intercettate perché le password vengono trasmesse in chiaro.
Con un pò più di sforzo si può invece utilizzare la **Digest Authentication** che consente di utilizzare MD5 per cifrare user e password nel momento dell'accesso alle directory protette.

Come è noto, Apache offre diversi modi per autenticare gli utenti e proteggere l'accesso al webserver o a parti in esso contenute, sfruttando le direttive allow e deny.
Ma se volessimo autenticare in modo massiccio utenti sarebbe bene criptare l'invio di user e password. Per fare questo ci sono sostanzialmente due metodi ovvero usare https e autenticazione Basic, oppure nel caso in cui sia necessario mantenere l'accesso http standard su porta 80, allora occorre sostituire l'autenticazione Basic con quella Digest che consente di utilizzare l'algoritmo md5 per cifrare le sessioni di autenticazione.
I principali browser oramai supportano questo tipo di autenticazione e per questo, se possibile, è bene implementarla.
Un altro metodo che vi propongo, legato non solo a questioni di sicurezza, riguarda la possibilità di gestire l'**autenticazione degli utenti** attraverso un **server mysql**.
Questo metodo risulta particolarmente utile quando abbiamo a che fare con un elevato numero di utenti, soprattutto perché la crezione di utenti può essere gestita con interfacce web o applicativi che semplificano tali operazioni.
Come fare? Io incomincerei con l'implementazione della digest Authentication:

Requirements

Conviene ripartire da zero, ricompilando Apache, con il supporto per il mod_ssl e per il mod_digest.
I sorgenti sono disponibili presso i link indicati:

- http://httpd.apache.org/dist/httpd/apache_1.3.28.tar.gz
- ftp://ftp.modssl.org/source/2.8.15-1.3.28.tar.gz
- ftp://ftp.openssl.org/source/openssl-0.9.7c.tar.gz

Compilare openssl

Per prima cosa compiliamo openssl o eventualmente aggiorniamolo:

```
$ cd openssl-0.9.7c
$ ./config
$ make
$ make install
```

Compilare mod_ssl

Stessa cosa va fatta con il mod_ssl per cui va eseguito il configure indicando la posizione dei sorgenti apache.

```
$ cd mod_ssl-2.8.8-1.3.24
$ ./configure --with-apache=../apache_1.3.28 --with-ssl=../openssl-0.9.7c \
--prefix=/usr/local/apache
```

Compilare Apache

A questo punto, per ultimo, effettuiamo il configure di Apache abilitando sial il mod_digest che il mod_ssl.

```
$./configure –-prefix=/usr/local/apache \
--enable-module=ssl \  # abilita il supporto per SSL
--enable-module=auth_digest \ # abilita MD5 hashes per l' HTTP auth
--enable-module=rewrite
```

verificare con attenzione la presenza della stringa alla file del configure

```
+ using -ldb1 for DBM support
    enabling DBM support for mod_rewrite
  o digest_auth_module uses ConfigStart/End
    using /dev/random for the random seed
  o ssl_module uses ConfigStart/End
```

```
$ make
$ make certificate
$ make install
```

L'apache così ottenuto supporta anche la crittografia grazie al mod_ssl è può essere lanciato con *apachectl startssl*, consentendoci di sperimentare entrambi i tipi di autenticazione (Basic o Digest) proteggendo comunque l'invio dei dati durante una sessione grazie all'utilizzo dell'https.

Configurare Apache

Prima di eseguire Apache poniamo l'attenzione su alcuni elementi riguardanti la configurazione di Apache riguardanti l'autenticazione degli utenti su aree del web che si vogliono proteggere o di cui si vuole restringere l'accesso solo a determinati utenti o a determinati host:

Definiamo le directory che devono essere protette dall'autenticazione Digest:

<Location /riservata/>
AuthType Digest
AuthName "Utenti Autorizzati"
AuthDigestFile /usr/local/apache/passwd.md5
Require valid-user
</Location>

oppure creare un file .htaccess con questa sintassi

#.htaccess
AuthName "Utenti Autorizzati"
AuthType Digest
AuthDigestFile /usr/local/apache/passwd.md5
require valid-user
#fine

Gli utenti si aggiungono con **htdigest** al posto di htpasswd usata per l'autenticazione basic.

htdigest [-c] passwdfile realm username

Con l'opzione -c viene creato il file di password, e se esistente viene sovrascritto. In genere questa opzione si usa la prima volta che il fiel viene creato

Una piccola parentesi va aperta per specificare come Apache si comporta nel caso in cui si voglia abilitare le restrizioni e le autenticazioni:

<Directory "/var/lib/apache/htdocs">
Options -Indexes FollowSymLinks MultiViews
AllowOverride Limit AuthConfig
Order allow,deny
Allow from all
</Directory>

La direttiva *AllowOverride* impostata a Limit e ad AuthConfig specifica infatti che possono essere impostati dei limiti per l'accesso al web server o che viene richiesta un'autenticazione. Questo sia che si specifichino i vincoli per directory, oppure con i file .htaccess depositatati nelle singole directory che si vogliono proteggere.
Ad esempio impostando entrambe le opzioni noi possiamo sia richiedere un'autenticazione protetta da MD5 hash che **limitando l'accesso solo ad alcuni IP o domini** oppure escludendo alcuni IP e domini. Vediamo alcuni esempi:

→ Accesso selettivo: nego tutti tranne
<Directory /ristretta>
AllowOverride Limit AuthConfig
Order deny,allow → prima nega poi consente
deny from all → nega tutto in partenza e poi consente solo per alcuni
allow from 127.0.0.1 192.168.17.150 → accedono solo il localhost è l'host con l'IP indicato
</Directory>

→ Accesso selettivo: consento a tutti tranne
<Directory /ristretta>

AllowOverride Limit AuthConfig
Order allow,deny → prima consente poi nego
allow from all → consente tutto in partenza mentre nega per alcuni
deny from 192.168.20.0/24 → accedono solo il localhost è l'host con l'IP indicato
</Directory>

La stessa cosa è fattibile specificando per "allow from" o "deny from" il nome di un host oppure un dominio.
Nei file di log (error_log) di Apache dovreste avere delle righe del tipo:

[Thu Nov 6 11:43:39 2003] [error] [client 192.168.20.5] **client denied** by server configuration: /usr/local/apache/htdocs/ristretta

In questo modo possiamo restringere o consentire l'accesso a determinati IP o classi o domini che in più devono autenticarsi utilizzando la Digest Authtentication che evita il trasferimento in chiaro delle password su web.

Nel caso in cui si utilizzino entrambi i metodi di restrizione (filtro IP e password MD5) per la stessa directory, come nel caso sopra esposto, è possibile utilizzare la direttiva "Satisfy" per indicare ad Apache di applicare tutti e due i metodi (Satisfy all) oppure solo uno dei due (Satisfy any). Nel secondo caso l'accesso sarà consentito sia ai client con l'IP indicato oppure a tutti i client dotati dell'account.

Ora possiamo invece vedere come configurare Apache per autenticare gli utenti utilizzando mysql. Per fare questo è necessario utilizzare un modulo di terze parti il **mod_auth_mysql:**

Requirements

Per prima cosa è necessario procurarsi il modulo e installarlo sul sistema, questa volta senza ricompilare apache, sfruttiamo la possibilità di utilizzare il modulo in modalità dinamica (DSO):

http://modauthmysql.sourceforge.net/

Scompattiamo il modulo:
root@fly:/usr/src/mod_auth_mysql# ls
README mod_auth_mysql.c

Compilazione del modulo

Per compilare in modalità DSO ci avvaliamo di apxs già presente sul nostro sistema sistema:

root@fly:/usr/src/mod_auth_mysql# apxs -c -D APACHE1 -lmysqlclient -lm -lz mod_auth_mysql.c
gcc -DLINUX=22 -I/usr/include/db1 -DUSE_HSREGEX -DEAPI -fpic -DSHARED_MODULE -I/usr/include/apache -DAPACHE1 -c mod_auth_mysql.c
mod_auth_mysql.c: In function `mysql_authenticate_basic_user':
mod_auth_mysql.c:667: warning: passing arg 2 of `strcmp' makes pointer from integer without a cast
gcc -shared -o mod_auth_mysql.so mod_auth_mysql.o -lmysqlclient -lm -lz
root@fly:/usr/src/mod_auth_mysql# ls
README mod_auth_mysql.c mod_auth_mysql.o **mod_auth_mysql.so***

Installazione del modulo

Dopo la sua compilazione può essere installato con i moduli di default di Apache, in genere nella directory /usr/libexec, ma può essere anche copiato manualmente, in un'altra posizione.

apxs -i mod_auth_mysql.so

Configurazione del modulo in Apache

Aggiungere in httpd.conf la direttiva to httpd.conf:

LoadModule mysql_auth_module libexec/mod_auth_mysql.so
AddModule mod_auth_mysql.c

Creazione tabella utenti in mysql

La tabella che dovrà contenere le informazioni su gli account degli utenti (user e password) dovrà avere una struttura simile a quella indicata nel README che accompagna il modulo, ma volendo potrebbe anche essere diversa, diciamo che è personalizzabile:

```
#Tab user_info
CREATE TABLE user_info (
  user_name CHAR(30) NOT NULL,
  user_passwd CHAR(20) NOT NULL,
  user_group CHAR(10),
  PRIMARY KEY (user_name)
)
```

E' importante che il campo user_info sia la chiave primaria e per questo motivo non potrà mai essere nullo.

A questo punto creiamo un DB sul sistema:
mysqladmin create apache –p

e ci carichiamo la struttura della tabella:
mysql apache -p <tab.sql

Otterremo un DB di questo tipo:

```
root@fly:/usr/src/mod_auth_mysql# mysqlshow -p apache user_info
Enter password:
Database: apache  Wildcard: user_info
+-----------+
|  Tables   |
+-----------+
| user_info |
+-----------+
```

```
mysql> select * from user_info;
+-----------+--------------------+------------+
| user_name | user_passwd        | user_group |
+-----------+--------------------+------------+
| paolo     | password           | amministra |
| paolop    | 21214db37cf66c40   | amministra |
+-----------+--------------------+------------+
2 rows in set (0.00 sec)
```

Inseriamo due utenti, uno con la password in chiaro e uno con la password criptata, entrambi del gruppo amministratori:

```
INSERT INTO user_info VALUES ( 'paolo',’passwd,'amministratori');
INSERT INTO user_info VALUES ( 'paolop',password('passwd '),'amministratori');
```

Configurazione di Apache

Adesso abbiamo la tabella con due utenti (paolo e paolop) che possiamo utilizzare per i nostri test, passiamo quindi alla configurazione di Apache.
La prima mostrata è la configurazione standard che prevede le password in chiaro (vedi oltre):

```
<Location /mysql>
AuthName "Autenticazione MySQL"
AuthType Basic
AuthGroupFile /dev/null
AuthMySQLHost localhost
AuthMySQLDB apache
AuthMySQLCryptedPasswords Off
AuthMySQLUserTable user_info
require valid-user
</Location>
```

E’ senza dubbio meglio utilizzare la criptazione ed inserire gli utenti nella tabella user_info con la password criptata:

```
<Location /mysql>
AuthName "Autenticazione MySQL"
AuthType Basic
AuthGroupFile /dev/null
AuthMySQLHost localhost
AuthMySQLDB apache
AuthMySQLScrambledPasswords On
AuthMySQLUserTable user_info
require valid-user
</Location>
```

La direttiva AuthMySQLScrambledPasswords On ci consente di utilizzare gli utenti con le password criptate secondo il metodo password (), all’atto dell’inserimento della password nella tabella.
Ma vediamo nel dettaglio i diversi metodi di gestione delle password degli utenti:

Metodi di autenticazione

Per le password in chiaro impostare:
AuthMySQLCryptedPasswords Off

Nella tabella user_info del DB apache
```
INSERT INTO user_info VALUES ( 'test',’test’,'amministratori');
```

La password è in chiaro, può essere facilmente intercettata. Meglio utilizzare password criptate.

Per le password criptate

AuthMySQLScrambledPasswords On

Nella tabella user_info del DB apache
INSERT INTO user_info VALUES ('test',**password**('passwd'),'amministratori');

In questo caso la password viene criptata.

Per le password criptate con md5

AuthMySQLMD5Passwords On

Nella tabella user_info del DB apache
INSERT INTO user_info VALUES ('test',**md5**('passwd'),'amministratori');

Autenticazione per gruppi

E' molto utile nel caso di molti utenti autenticare per gruppi, facendo in modo che solo gli utenti associati ad in particolare gruppo, nel nostro caso il gruppo *amministratori*, possano loggarsi al sistema o alla risorsa condivisa:

```
CREATE TABLE user_group (
   user_name char(50) DEFAULT '' NOT NULL,
   user_group char(20) DEFAULT '' NOT NULL,
   create_date int,
   expire_date int,
   PRIMARY KEY (user_name,user_group)
 );
```

Configurazione Apache

```
<Location /mysql>
AuthName "MySQL Testing"
AuthType Basic
AuthGroupFile /dev/null
AuthMySQLHost localhost
AuthMySQLDB apache
AuthMySQLUser apache
AuthMySQLPassword apache
AuthMySQLPasswordField user_passwd
AuthMySQLScrambledPasswords On
AuthMySQLUserTable user_info
AuthMySQLGroupTable user_group
AuthMySQLGroupField user_group
require group amministratori
</Location>
```

Inserimento Utenti

```
mysql> show tables;
+------------------+
| Tables_in_apache |
+------------------+
| user_group       |
```

```
| user_info       |
+-----------------+
2 rows in set (0.00 sec)

mysql> INSERT INTO user_group VALUES ( 'prova','amministratori','','');
Query OK, 1 row affected (0.00 sec)
```

L'utente prova del gruppo amministratori si potrà loggare

Invece se aggiungiamo un utente in gruppo diverso, ad esempio un nuovo gruppo chiamato utenti:

```
mysql> INSERT INTO user_info VALUES ( 'pippo',password('pippo'),'utenti');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO user_group VALUES ( 'pippo','utenti','','');
Query OK, 1 row affected (0.00 sec)
```

Avremo che questo utente non potrà loggarsi, perché non fa parte del gruppo amministratori.

Note

Non ho avuto modo di fare un test esaustivo di questo sistema di autenticazione, il numero di utenti era limitato ed Apache non ne ha risentito. Sarei curioso di provarlo sotto un carico di lavoro molto maggiore per vedere di quanto possa risentire in termini di prestazioni effettuare l'autenticazione sul server mysql.
Se qualcuno ha avuto esperienze nell'implementazione di questi sistemi, me le segnali in modo da poter completare il documento anche sotto questo aspetto.

***Doc: apache_auth_2*.pdf**

Dott. Paolo PAVAN [Netlink Sas]– pavan@netlink.it
Data: *Settembre 2003*

Note finali

- Il presente documento è a semplice scopo divulgativo
- L'autore non si assume la responsabilità di eventuali danni diretti o indiretti derivanti dall'uso dei programmi, o dall'applicazione delle configurazioni menzionate nel seguente articolo
- L'uso o il riutilizzo del presente articolo è liberamente consentito per scopi didattici o informativi previa citazione della fonte
- Sono possibili errori o imprecisioni, segnalatemele a pavan@netlink.it
- Chi volesse integrare il presente documento, può scrivere a pavan@netlink.it.