DoS Apache - IDS e Firewall HTTP

DoS Apache – Prevenire attacchi Denial of Service e Distribuited Denial of Service con mod_evasive e mod_security

MOD EVASIVE

Proteggere il nostro webserver senza ricorrere a sistemi IDS particolarmente complessi o costosi è possibile, mod_evasive e mod_security sono i due moduli da installare e configurare per prevenire attacchi per Denial of Service (Dos) e Distribuited Denial of Service (DDoS), il primo lavora come un IDS, mentre il secondo usa delle regole similari ad un firewall.

Iniziamo impostando i valori di TimeOut e KeepAlive:

- La direttiva **RequestReadTimeout** consente di limitare il tempo di un client per effettuare una richiesta .
- Il valore della direttiva **TimeOut** dovrebbe essere abbassato su siti che sono oggetto di attacchi DoS , è opportuno impostare questo a partire da un paio di secondi . Un valore troppo basso porterà problemi con l'esecuzione di script CGI che richiedo molto tempo per il loro completamento.
- Il parametro per la direttiva **KeepAliveTimeout** può essere abbassato anche su siti che sono oggetto di attacchi DoS . Disattivare il **KeepAlive** con impostazione Off, così come accade per alcuni siti, produce inconvenienti prestazionali, se impostata su On, *permette di usare, come da specifiche HTTP/1.1, la stessa connessione TCP per inviare più file, è pertanto consigliata questa configurazione, che evita l'apertura di una connessione TCP per ogni richiesta HTTP*.

Il mod_evasive intercetta e blocca un determinato indirizzo IP che svolge un determinato numero di richieste in un breve lasso di tempo.

Prima di procedere installiamo alcuni pacchetti fondamentali

```
# yum install make autoconf
# yum install gcc httpd-devel pcre-devel
# yum install libxml2 libxml2-devel curl curl-devel
```

Passiamo all'installazione,può essere fatta tramite yum:

```
# yum install -y mod_evasive
```

oppure scaricando il pacchetto e compilandolo:

```
# cd /usr/src
# wget
http://www.zdziarski.com/blog/wpcontent/uploads/2010/02/mod_evasive_1.10.1.ta
r.gz
# tar xzf mod_evasive_1.10.1.tar.gz
# cd mod_evasive
# apxs -cia mod_evasive20.c
```

Passiamo ora alla configurazione:

```
# vi /etc/httpd/conf/httpd.conf
```

Abilitiamo il modulo e inseriamo le direttive:

```
LoadModule evasive20_module   /usr/lib64/httpd/modules/mod_evasive20.so
```

Editiamo il file

```
# vim /etc/httpd/conf.d/mod_evasive.conf
```

Inseriamo le entry di base:

```
# mod_evasive configuration
LoadModule evasive20_module modules/mod_evasive20.so
<IfModule mod_evasive20.c>
    DOSHashTableSize    3097
    DOSPageCount        2
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   10
    DOSEmailNotify      dos@lbit-solution.it
    #DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
    DOSLogDir           "/var/log/httpd/mod_evasive"
    DOSWhitelist  95.110.245.202
    #DOSWhitelist   192.168.0.*
</IfModule>
```

Ora vediamo nel dettaglio le direttive:

- **DOSHashTableSize**: dimensione della tabella di hash per la collezione dei dati di campionamento.
- **DOSPageCount**: identifica la soglia di richiesta di una stessa pagina da parte di un host in un certo intervallo di tempo.
- **DOSSiteCount**: identifica la soglia di richiesta di un qualsiasi oggetto da parte di un host in un certo intervallo di tempo.
- **DOSPageInterval**: intevallo di tempo per la soglia del parametro **DOSPageCount** in secondi.
- **DOSSiteInterval**: intevallo di tempo per la soglia del parametro

DOSSiteCount in secondi.

- **DOSBlockingPeriod**: parametro che specifica l'intervallo di tempo utilizzato per mostare l'http error 403 ai client che stanno eseguendo un probabile attacco DoS.
- **DOSEmailNotify**: parametro che specifica l'indirizzo mail al quale inviare una mail di notifica, se un certo indirizzo IP sta eseguendo un probabile attacco Dos.
- **DOSWhitelist**: con questo parametro è possibile aggiungere una lista di IP che non devono essere bloccati dal modulo, nella configurazione di esempio abbiamo applicato la regola per l'indirizzo IP 95.110.245.202
- **DOSLogDir**: specifica un path alternativo alla temp directory per la collezione dei dati.
- **DOSSystemCommand**: lancia uno specifico comando quando viene superata la soglia da parte di un client. Per ricavare l'indirizzo IP che ha sfortato la soglia si deve usare la variabile "%s".

Per testare che tutto sia funzionante, e che le nostre richieste vengano bloccate possiamo usare uno script PERL:

```
#!/usr/bin/perl
# test.pl: small script to test mod_dosevasive's effectiveness
  use IO::Socket;
  use strict;
  for(0..100) {
      my($response);
      my($SOCKET) = new IO::Socket::INET( Proto   => "tcp",
                                          PeerAddr=> "127.0.0.1:80");
      if (! defined $SOCKET) { die $!; }
      print $SOCKET "GET /?$_ HTTP/1.0\n\n";
      $response = <$SOCKET>;
      print $response;
      close($SOCKET);
  }
```

Il risultato del test sarà il seguente:

```
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
```

MOD SECURITY

Anche per il mod_security vale la stessa regola del mod_evasive per

l’installazione, possiamo scegliere se installarlo tramite repository oppure compilarlo.

Installazione tramite yum:

```
# yum install mod_security
```

Oppure scaricare il pacchetto ed installarlo:

```
# cd /usr/src
# wget http://www.modsecurity.org/download/modsecurity-apache_2.6.6.tar.gz
# tar xzf modsecurity-apache_2.6.6.tar.gz
# cd modsecurity-apache_2.6.6
# ./configure
# make install
# cp modsecurity.conf-recommended /etc/httpd/conf.d/modsecurity.conf
```

File di configurazione di mod_security

1. **/etc/httpd/conf.d/mod_security.conf** – file di configurazione principale del modulo mod_security di Apache
2. **/etc/httpd/modsecurity.d/** – tutti gli altri file di configurazione modulo Apache mod_security.
3. **/etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf** – La configurazione presente in questo file deve essere personalizzata in base alle vostre esigenze prima di essere messa in esercizio.
4. **/var/log/httpd/modsec_debug.log** – Usa i messaggi di debug per il debugging e altri problemi
5. **/var/log/httpd/modsec_audit.log** – Tutte le richieste che attivano ModSecuirty (come rilevato) o gli errori server (“RelevantOnly”) vengono scritti nel file di log.

Editiamo il file /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf

```
# vi  /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf
```

E attiviamo la protezione del webserver

```
# SecRuleEngine On
```

Riavviamo il servizio httpd

```
# service httpd restart
```

Vediamo dal file di log se non si sono problemi:

```
# tail -f /var/log/httpd/error_log
```

Abbiamo terminato l'installazine dei due moduli che ridurranno gli attacchi, ora in base all'hardware e alle proprie esigenze andranno configurati tutti i servizi.

Scarica il PDF Proteggere Apache da attacchi DoS e DDoS.

MySQL UDF Perl Regular Expression

Nel realizzare nuovi scraper per g4play.it Emanuele si è reso conto che la nostra istanza MySQL non supporta le espressioni regolari, a lui non servono solo query di ricerca ma manipolazioni di dati complesse. Con estrema semplicità mi chiede di installare la libreria lib_mysqludf_preg, non è complicato, ma neanche così banale. Iniziamo subito con l'installazione dei pacchetti che ci serviranno:

```
[root@mysqllbit lib_mysqludf_preg]# yum install pcre pcre-devel
[root@mysqllbit lib_mysqludf_preg]# yum install make gcc gcc-c++
[root@mysqllbit lib_mysqludf_preg]# yum install mysql-devel
```

Questo per evitare tutti gli errori relativi al compilatore, a pcre e mysql. Scarichiamo il paccheto da GitHub:

```
[root@mysqllbit lib_mysqludf_preg]# wget
https://github.com/mysqludf/lib_mysqludf_preg/archive/testing.zip
[root@mysqllbit lib_mysqludf_preg]# unzip testing.zip
```

ora lanciamo il configuratore

```
[root@mysqllbit lib_mysqludf_preg]# ./configure
```

ci siamo risparmiati gli errori avendo installato preventivamente i pacchetti, l'unico messaggio a video con la parola ERROR è

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using
password: NO)
```

Possiamo rilassarci, avendo settato la password di root è normale che non riesca ad accedere. Ora installiamo:

```
[root@mysqllbit lib_mysqludf_preg]# make
[root@mysqllbit lib_mysqludf_preg]# make install
[root@mysqllbit lib_mysqludf_preg]# make installdb

ERROR 1548 (HY000) at line 5: Cannot load from mysql.proc. The table is
probably corrupted
make: *** [uninstalldb] Error 1
```

Sull'ultimo passaggio ho ricevuto errore di tabella corrotta, per questo ho dovuto prima "sistemare" le tabelle MySQL e poi rilanciare il make installdb

```
[root@mysqllbit lib_mysqludf_preg]# make installdb
/usr/bin/mysql -p <./uninstalldb.sql
Enter password:
cat installdb.sql | sed 's/\.so/.dll/g' >installdb_win.sql
if test -f .libs/lib_mysqludf_preg.dll; then  \
                /usr/bin/mysql -p <./installdb_win.sql; \
        else \
                /usr/bin/mysql -p <./installdb.sql;\
        fi
Enter password:
[root@mysqllbit lib_mysqludf_preg]# make test
cd test; make test
make[1]: Entering directory `/usr/local/lib/lib_mysqludf_preg/test'
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-fi...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
make[1]: Leaving directory `/usr/local/lib/lib_mysqludf_preg/test'
```

Finito, ora Emanuele potrà usare le espressioni regolari per manipolare i dati di g4play.it.

^M carattere di fine riga

Il carattere ^M è il simbolo di fine riga utilizzato dagli editor windows e mal interpretati da unix/linux.

Prendiamo questo file, scritto con il notepad:

```
$ cat appo.sh
#!/bin/bash
echo "ciao"
```

Questo dovrebbe stampare “ciao”, ma se eseguito va in errore:

```
$ ./appo.sh
./appo.sh: line 2: $'\r': command not found
```

Se aggiungiamo l’opzione -v al cat, vediamo che…

```
$ cat -v appo.sh
#!/bin/bash^M
^M
echo "ciao"^M
```

Ops.. i ^M.. questi vengono interpretati come caratteri e non come “a capo”..

ripuliamo il file..

abbiamo un modo velocissimo per rimediare, dalla shell, sia usando un comando apposito sia utilizzando il perl, il risultato è lo stesso.

Dalla shell, digitate:

```
dos2unix appo.sh
```

Questo sovrascrive il file stesso convertendo i ^M (nelle versioni linux, nelle unix dovete redirigere su un nuovo file).

```
$ dos2unix appo.sh
dos2unix: converting file appo.sh to Unix format ...
$ cat -v appo.sh
#!/bin/bash
echo "ciao"
```

oppure, per utilizzare il perl, digitate:

```
$ perl -pi -e 's/\r//g' appo.sh
$ cat -v appo2.sh
#!/bin/bash
echo "ciao"
```

Adesso funziona..

```
$ ./appo.sh
ciao
```
