

Usare Snort come un leggero NIDS

Ultimamente si parla molto di *Intrusion Detection* ed in particolare della versione *Network*. Uno degli attori più discussi in questo campo, sia per motivi tecnici che ideologici, è *Snort*, un NIDS OpenSource. Partendo dall'idea che non c'è maniera migliore per capire un prodotto che quella di usarlo, in questo e nel seguente articolo proveremo ad installare, configurare ed usare *Snort* per cercare di capire che cosa fa e cosa non fa.

Non ci occuperemo in questo articolo di come fare un progetto di Intrusion Detection, né ci occuperemo di fare delle valutazioni e comparazioni fra diversi prodotti di IDS (argomento molto controverso già a livello teorico). La nostra configurazione di laboratorio è la seguente. Abbiamo una macchina con 2 schede di rete che ospita Snort, una seconda macchina anch'essa con due schede di rete, sulla quale installeremo il database e funzionerà da *console*.

Utilizziamo Snort come un sensore, ovvero poniamo Snort in ascolto su di una interfaccia (Ethernet). Snort analizza tutti i pacchetti che arrivano su questa interfaccia e se un pacchetto è selezionato da una delle regole che abbiamo impostato, Snort invia il pacchetto o parte di esso al database. Questo è di solito chiamato un *ALERT*. Il database, meglio se più di uno, ha l'unica funzione di archiviare gli *ALERT* di Snort, tocca poi alla console effettuare una ulteriore interpretazione del loro significato. Vedremo il tutto in dettaglio al momento della configurazione del sistema.

Come operazione preliminare colleghiamo una scheda di rete (Ethernet0) al Hub o Switch (se si tratta di uno switch, la porta va configurata in modalità *Mirror* o *Span* in modo che tutto il traffico che deve essere esaminato venga inviato anche alla nostra macchina, la configurazione è un po' più complicata nel caso in cui si usino VLAN). Inoltre può essere opportuno mettere l'interfaccia di ascolto in modalità *stealth*, ovvero invisibile. Per far questo bisogna non specificare un indirizzo IP e disabilitare il protocollo ARP (anche se quest'ultimo non sempre è possibile o utile). In Linux la configurazione dell'interfaccia in questa modalità è data dal semplice comando `/sbin/ifconfig eth0 up`, od, in alcune distribuzioni, dai soli parametri:

```
DEVICE=eth0
ONBOOT=yes
ARP=no
```

nel file `ifcfg-eth0`, a cui nel caso si può aggiungere `PROMISC=yes`. La seconda interfaccia Ethernet1 è collegata invece ad una LAN interna riservata alla quale è connessa anche la seconda

macchina che agisce da archivio e console. La macchina che ospita Snort è accessibile via network solo da questa seconda interfaccia. In generale su questa LAN interna riserveremo solo i sensori con Snort e le console con gli archivi, in modo che il traffico degli ALERT e dei LOG non si mischi al traffico normale della nostra rete. Le console sono collegate alla LAN aziendale attraverso la seconda (ed ultima) interfaccia Ethernet. Ovviamente sia le console che i sensori non forwardano i pacchetti da una interfaccia all'altra.

1. Installazione di Snort

Per prima cosa bisogna procurarsi Snort scaricandolo dal sito www.snort.org. Prima di fare ciò bisogna però ottenere dalla mailing-list la chiave pubblica PGP e le hash dei pacchetti. Poi dal sito si può scaricare il sorgente e compilarlo, oppure dei pacchetti pre-compilati. Per i pacchetti pre-compilati vi sono varie possibilità, le principali sono:

- se si vuole utilizzare *flexresp* (Flexible Response) che permette di inviare pacchetti IP per interrompere connessioni pericolose (codice ancora non maturo, ovvero *alpha*);
- in quale tipo di archivio si vogliono archiviare gli alert: database SQL, XML, via snmp, file locali ecc.

La nostra scelta è di utilizzare come archivio un database MySQL sulla seconda macchina, e di usare anche *flexresp*. La compilazione a mano del programma nel nostro caso richiede di avere le librerie *libpcap* e *libnet* (*libpcap* è utilizzato per mettersi in ascolto sulla interfaccia mentre *libnet* è usato da *flexresp*) ed i sorgenti (od almeno i file di header) di MySQL. Prima di compilare, controllare sempre firma digitale e hash dei pacchetti scaricati. La compilazione è poi data da un semplice: `./configure --prefix=/usr --enable-flexresp --with-mysql, make, make install` nella cartella dei sorgenti di Snort.

Utilizzando i pacchetti pre-compilati, basta, dopo aver controllato firma digitale e hash, installare nel nostro caso i due pacchetti

- `snort-1.9.0-1snort.i386.rpm`
- `snort-mysql+flexresp-1.9.0-1snort.i386.rpm`.

L'installazione produce i seguenti file:

- l'eseguibile in `/usr/sbin/snort`
- i file di configurazione nella cartella `/etc/snort/`
- una cartella per i file di log in `/var/log/snort/`
- una man page `snort` che descrive le opzioni che il programma accetta sulla linea di comando
- un script di avvio di solito in `/etc/rc.d/init.d/snortd`.

2. Configurazione di Snort

La configurazione¹ di snort avviene in due punti, le opzioni da dare alla linea di comando e di solito inserite nello script di avvio in `/etc/rc.d/init.d/snortd`, e la configurazione delle regole (il *ruleset*) presenti in `/etc/snort/`.

Le principali opzioni per la linea di comando sono riportate nella tabella 1. Se non si usa flexresp, non è necessario che Snort sia attivo come l'utente amministratore (`root`) ma si può creare un utente apposta (*snort* ad esempio), abilitare questo utente a leggere i file di configurazione in `/etc/snort/`, ed a leggere e scrivere in `/var/log/snort/`. Snort deve essere attivato dall'utente `root` e le opzioni `-u` e `-g` indicano l'utente ed il gruppo con cui il programma deve girare. (Snort può anche girare in un ambiente *chroot*, ma non lo considereremo in questo articolo). Le opzioni `-i` e `-h` indicano l'interfaccia su cui Snort deve ascoltare e l'eventuale network di indirizzi che devono essere considerati come interni. L'opzione `-z` indica che Snort può inviare ALERT o LOG solo per connessioni TCP già stabilite, quindi ALERT o LOG per pacchetti TCP di tipo SYN non saranno inviati eccetto che per gli alert inviati dai pre-processori (vedi più avanti). Questa opzione permette di ridurre l'effetto di tools come *stick* e *snot* il cui scopo è quello di confondere e subissare di false informazioni un NIDS. Infine l'opzione `-o` indica che l'ordine in cui sono verificate le regole deve essere PASS, ALERT, LOG invece di ALERT, PASS, LOG.

Infatti Snort prevede 3 tipi principali di regole, gli ALERT che inviano un avviso di infrazione, i LOG per tenere copia dei pacchetti (tipo `tcpdump`) ed i PASS che scartano un pacchetto. Come al solito, la prima regola che si applica interrompe l'esame delle regole. Le regole PASS permettono di eliminare traffico noto che non si vuole compaia nel LOG. Spesso però si vuole filtrare anche dagli ALERT del traffico noto, e perciò è possibile porre le regole PASS per prime. In questo caso bisogna però stare molto attenti a non creare delle regole PASS che eliminano i pacchetti pericolosi perché in questo caso nessun ALERT verrà mai inviato!

Nel nostro caso la linea di comando per attivare Snort è:

```
/usr/sbin/snort -u root -g snort -d -i eth0\  
-h 10.1.1.0/24 -aoIepym 027\  
-c /etc/snort/snort.conf -D
```

Nel caso avessimo più interfacce (o più VLAN sulla stessa interfaccia) sulle quali vogliamo porre Snort in ascolto, dobbiamo far girare un processo di Snort per ognuna di esse, ed opzionalmente creare un file di configurazione od una intera cartella di configurazione per ognuna di esse. Va ricordato infine che le opzioni sulla linea di comando hanno la precedenza sui corrispondenti

¹ Descriveremo qui solo le opzioni ed i parametri necessari alla nostra configurazione. Snort è altamente configurabile ed una descrizione completa di tutti i parametri, opzioni e moduli può essere trovata nel Manuale Urtente su <http://www.snort.org/>.

(quando vi siano) parametri nel file di configurazione.

Passiamo ora al cuore di Snort, la configurazione delle regole. Questa è composta da due parti, la configurazione generale e dei pre-processor, e le regole stesse. La prima parte è nel file `/etc/snort/snort.conf` che a sua volta è composto da quattro sezioni:

1. network e servizi locali
2. preprocessori
3. archivi in output
4. gruppi di regole da utilizzare

Nella prima sezione si definiscono i parametri della propria rete locale e dei servizi che vi sono, in modo che Snort possa distinguere l'attaccante dall'attaccato. Un esempio di configurazione di questi primi parametri con ovvio significato, è

```
var HOME_NET 10.1.1.0/24
var EXTERNAL_NET any
var DNS_SERVERS 10.1.1.2/32
var SMTP_SERVERS 10.1.1.4/32
var HTTP_SERVERS 10.1.1.80/32
var SQL_SERVERS $HOME_NET
var TELNET_SERVERS $HOME_NET
var HTTP_PORTS 80
# Ports you want to look for SHELLCODE on.
var SHELLCODE_PORTS !80
var ORACLE_PORTS 1521
```

I preprocessori richiedono una minima discussione. Per ottimizzare l'analisi finale delle regole, Snort utilizza dei preprocessori dei pacchetti che sono in grado di effettuare, su tutti i pacchetti, delle operazioni alle volte non possibili direttamente con le regole. In particolare alcuni dei preprocessori fanno:

- la de-frammentazione dei pacchetti e l'invio di ALERT nel caso di attacchi basati sull'uso di frammenti;
- la *stateful inspection/stream reassembly* che permette di rilevare tra l'altro portscan e simili attacchi (questo è legato all'opzione `-z`);
- la *normalizzazione* dei pacchetti prima che questi siano passati alle regole, tipico esempio la conversione dei caratteri esadecimali in ASCII o la rimozione di trucchi con UNICODE che potrebbero confondere le regole;
- il rilevamento di portscan generali tenendo statistiche sui pacchetti che vengono analizzati (tipo, sorgente, destinazione, opzioni ecc.).

I preprocessori possono essere attivati o disattivati e modificati i loro parametri. Molti dei preprocessori generano ALERT in caso di attacchi, ma come sempre è molto probabile che segnalino molti falsi positivi ed è pertanto necessario scegliere opportunamente quelli che si vogliono utilizzare e configurarli al meglio per la propria rete. Come nel caso delle regole, è necessario un lungo e delicato lavoro di regolazione fine. Come dice il loro nome, i preprocessori agiscono sui pacchetti prima delle regole normali, anche delle regole PASS.

La scelta degli archivi, ovvero dell'output, è il prossimo punto. Vi sono molte possibilità, ad esempio: *socket unix*, *syslog*, *snmp*, *smb*, *xml*, database SQL, file di testo o binari. La scelta dell'archivio deve essere basata su varie considerazioni: la quantità di dati che si vogliono archiviare, la velocità con cui i dati arrivano, i programmi che devono poi interpretare i dati. Snort offre dei metodi di archiviazione veloce dei pacchetti in formato binario (come ad esempio *tcpdump*) che permettono di archiviare anche 80Mbit al secondo, il che corrisponde ad un Ethernet a 100Mbps al massimo della capacità. Ad esempio specificando:

```
output alert_syslog: LOG_AUTH LOG_ALERT
output log_tcpdump: tcpdump.log
```

si indica che gli ALERT devono essere mandati al syslog locale e che i pacchetti di LOG devono essere scritti nel file *tcpdump.log* in formato *tcpdump* nella cartella di log. In una installazione reale, è sempre consigliato inviare ad almeno 2 archivi distinti le segnalazioni, anche nel caso in formato diverso. Per quanto riguarda il nostro esempio, abbiamo deciso di utilizzare come archivio un database MySQL sulla seconda macchina. La configurazione in questo caso è:

```
output database: alert, mysql, user=snort\
password=snortpasswd dbname=snort host=mysqlhost\
sensor_name=Snort1
```

ove l'utente e la password sono quelli dell'utente di MySQL che può scrivere nelle tavole del database *snort*. Abbiamo inoltre assegnato un nome unico a questo sensore poiché più di un sensore potrebbe inviare i propri dati alla console centrale, ed il nome può aiutare a distinguere la provenienza dei dati. Infine abbiamo deciso di inviare al database solo gli ALERT. Bisogna osservare che nel nostro caso la password del database è in chiaro nel file di configurazione e pertanto come precauzione minima bisogna porre delle protezioni restrittive al file */etc/snort/snort.conf*. Inoltre i dati tra Snort e la console passano in chiaro sul network, inclusa la password del database. Se la LAN interna è una LAN veramente privata e limitata a queste due macchine, questo potrebbe non essere un grande problema, altrimenti sarebbe meglio creare un tunnel cifrato (SSL, SSH, IPSEC od altro) che colleghi le due macchine. Oppure si possono usare altri metodi di archiviazione come XML o SNMPv3 che offrono direttamente la cifratura dei dati.

Per buon ultimo, la cosa più importante, cioè le regole. Nella cartella `/etc/snort/` sono presenti molti file organizzati per tipo contenenti regole di ALERT per attacchi conosciuti. Ad esempio attacchi su smtp, telnet, sql, netbios, oracle ecc. Includendo questi file nel file di configurazione, si caricano le corrispondenti regole in Snort. Perciò la prima cosa da fare è scegliere i file di regole che vogliamo utilizzare commentando quelle che si vogliono escludere o viceversa per quelle che si vogliono includere. Per i file di regole utilizzati, a volte è anche necessario escludere o modificare a mano le singole regole, ciò vuol dire editare direttamente i file e modificarli. Purtroppo questo fa parte del necessario, lungo e delicato lavoro di regolazione fine. L'ultimo file, vuoto, è `local.rules` ed in esso possiamo scrivere le nostre regole personali sia di ALERT che di LOG e PASS (le regole di LOG e PASS compaiono solo in questo file). Una singola regola ha la seguente semplice struttura:²

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306\  
  (msg:"MYSQL root login attempt";\  
  flow:to_server,established;\  
  content:"|0A 00 00 01 85 04 00 00 80 72 6F 6F 74 00|";\  
  classtype:protocol-command-decode; sid:1775; rev:1;)
```

La prima parola dichiara il tipo di regola, ALERT, PASS, LOG. E' anche possibile definire dei nuovi e propri tipi di regole. La seconda parola definisce il protocollo a cui applicare la regole. Segue l'indirizzo IP e la porta sorgente e l'indirizzo IP e la porta di destinazione, in questo caso la porta su cui ascolta il server MySQL. All'interno delle parentesi vi sono le azioni da effettuare sul pacchetto così selezionato: c'è il messaggio da inviare in caso di ALERT, i test da effettuare sul pacchetto con l'analisi del contenuto del pacchetto, ed infine la classificazione del tipo di ALERT. Va notato che l'analisi del contenuto del pacchetto è un'operazione molto onerosa per Snort, per cui viene effettuata per ultima solo se tutti gli altri test sono positivi. In questo caso, se il flow del pacchetto non è verso il server o non è una connessione stabilita, il test sul contenuto del pacchetto non viene fatto. Pertanto, quando si scrivono le proprie regole è meglio aggiungere almeno un test che selezioni il pacchetto sulla base di flags o altri parametri di facile verifica, prima di fare una ricerca nel contenuto del pacchetto stesso. Altri due esempi che potrebbero essere in `local.rules` sono

```
alert tcp any any -> 10.1.1.3/32 28 (flags: S+;\  
  msg: "Test FlexResp"; resp:icmp_port;)  
pass ip 10.1.1.18/32 any -> 10.1.1.22/32 any
```

La prima regola seleziona i pacchetti tcp con il bit SYN attivo indirizzati all'IP 10.1.1.3 sulla porta 28 ed invia un ALERT ed anche un pacchetto icmp port_unreachable indietro al mittente (bisogna stare molto attenti nell'uso di FlexResp a non creare loop infiniti o bloccare del traffico lecito). La

² Le regole sono di solito scritte su di un'unica riga.

seconda regola dice di non considerare i pacchetti dall'IP 10.1.1.18 all'IP 10.1.1.22 nelle seguenti regole.

3. Attivazione e manutenzione di Snort

Attivare Snort a questo punto è molto semplice, basta dare il comando che abbiamo descritto precedentemente, ed inserirlo nella procedura di avvio della macchina. All'avvio Snort comunica i dati della propria attivazione, inoltre inviandogli un segnale SIGUSR1, Snort invia a syslog le statistiche del suo funzionamento, pacchetti esaminati, persi, ALERT inviati ecc. Il segnale SIGHUP fa rileggere a Snort i file di configurazione ed è utile per aggiornare le regole senza fermare il programma.

Cosa succede quando Snort è attivo? Molto semplice, esamina tutti i pacchetti che arrivano sull'interfaccia Ethernet0, e quando un pacchetto risulta positivo rispetto ad una regola od un preprocessore, invia il corrispondente ALERT o LOG al database MySQL.

La manutenzione di Snort è delicata in quanto bisogna cercare di ottimizzare le regole per ridurre il numero di falsi positivi senza al contempo eliminare alcun vero allarme. Se inoltre si hanno molti sensori su cui gira Snort, la cosa diventa più complicata da gestire. Per questo vi sono vari strumenti, a dire il vero molti dei quali ancora in sviluppo, che gestiscono la distribuzione delle regole ai sensori. Vi sono inoltre altri *plugin* per Snort che aggiungono varie funzionalità, quali ad esempio di creare automaticamente regole da aggiungere ad un firewall in risposta ad un ALERT inviato da Snort (cosa in generale abbastanza pericolosa poiché si rischia in caso ad esempio di un attacco DDOS di peggiorare anziché migliorare la situazione). Infine non bisogna dimenticare che Snort è utilizzato come *engine* nei sensori di vari prodotti commerciali che a Snort aggiungono la gestione delle regole, degli ALERT ecc.

Il set di regole preparato dagli sviluppatori di Snort viene aggiornato costantemente con le firme di nuovi attacchi. Pertanto è necessario controllare frequentemente se nuove regole sono state pubblicate e nel caso installarle in sostituzione o aggiunta di quelle originali, facendo attenzione alle regole che erano state modificate manualmente nella fase di ottimizzazione di Snort.

4. La console

Consideriamo ora la configurazione della *console* che ospita l'archivio degli ALERT (e LOG) di Snort e l'interfaccia utente di analisi dei dati.

Come già visto nel precedentemente, abbiamo scelto di archiviare i dati in un database MySQL. Per l'interfaccia utente descriveremo ACID, un prodotto Web basato su PHP4, che si sta affermando come uno strumento potente ma al tempo stesso facile da usare.

Ricordiamo ancora la topologia della nostra piccola rete IDS. Sia la macchina con il sensore (Snort) che la console con l'archivio hanno due schede di rete. Il sensore utilizza una scheda di rete solo per ricevere i pacchetti da analizzare, la seconda scheda è connessa ad una LAN interna privata su cui transitano solo i pacchetti di ALERT e LOG ed a cui sono collegati solo il sensore e la console. La seconda scheda della console è connessa alla nostra LAN aziendale ed è su questa interfaccia che accederemo come utenti alla console. In particolare, sulla interfaccia privata della console è attivo MySQL che riceve i pacchetti da Snort, mentre sulla interfaccia sulla LAN aziendale è attivo Apache.

5. MySQL

Assumiamo che sulla macchina che utilizziamo come console sia già stato installato MySQL e che sia stata già assegnata una password all'utente *root* di MySQL. Perché Snort possa inviare i dati all'archivio di MySQL, è necessario creare il database che avevamo chiamato *snort* ed in esso alcune tavole. Inoltre ACID utilizza un database identico a quello di Snort nel quale archiviare gli ALERT vecchi, per cui creeremo anche un identico database dal nome *snort_archive*. Per creare i database è necessario procurarsi i seguenti due file contenuti nei sorgenti di Snort:

- `create_mysql`
- `snortdb-extra.gz` (opzionale)

Il secondo file è opzionale e va nel caso scompreso. Creiamo inoltre un utente *acid* che accederà ai database solo localmente dalla console Web. Per creare queste tabelle possiamo usare il file riportato in Tabella 2 (per aumentare la sicurezza possiamo sostituire a “%” , che significa qualunque indirizzo IP, gli indirizzi IP dei sensori su cui gira Snort) e, nella stessa cartella in cui abbiamo messo i file precedenti, diamo il seguente comando

```
mysql -u root -p < file_tabella_2
```

digitando la password di root di MySQL quando richiesto. Se MySQL è stato installato e configurato correttamente, Snort può ora inviare gli ALERT al database. Dobbiamo solo installare e configurare Acid.

6. Acid, Apache, Php4, Phplot , Adodb e phpMyAdmin

Come è chiaro dal titolo di questa sezione, l'installazione di ACID richiede un po' di lavoro. Acid si basa su Php4 per ottenere i dati da MySQL e creare le pagine HTML, Installeremo ACID ed i programmi di supporto nella cartella `/var/www/snort/` ed assumiamo che Apache e Php4 siano correttamente installati e configurati. E' necessario che Php4 sia compilato con il supporto per MySQL e GD, e che la libreria grafica GD sia installata, si suggerisce inoltre di dirigere i messaggi di log di Php4 su file ed è assolutamente necessario che sia settato

`magic_quotes_gpc = On`. Per quanto riguarda Apache, si suggerisce di utilizzare un Server Virtuale sicuro, con SSL, e di restringere l'accesso a `/var/www/snort/acid` e `/var/www/snort/phpMyAdmin` almeno con username e password. Non è questa comunque l'occasione per una discussione in dettaglio dei problemi di sicurezza di un server Web, Apache nel nostro caso, e di un linguaggio di scripting Web come Php. Assumiamo quindi che l'installazione e la configurazione di Apache e Php4 siano adeguate ai livelli di sicurezza richiesti. Per installare Acid ci dobbiamo procurare i seguenti programmi:³

- Acid (acid-0.9.6b21) da <http://acidlab.sourceforge.net/>
- Adodb (adodb-2.31) da <http://php.weblogs.com/adodb/>
- Phplot (phplot-4.4.6) da <http://www.phplot.com/>
- phpMyAdmin (phpMyAdmin-2.3.0) da <http://phpmyadmin.sourceforge.net/> (opzionale).

Come al solito, prima di installare i programmi è necessario verificare le hash e le firme digitali (quando sono fornite) dei pacchetti scaricati. L'installazione è molto semplice, come root:

```
mkdir -p /var/www/snort
cd /var/www/snort
tar xzvf ACID-0.9.6b21.tar.gz
tar xzvf adodb231.tgz
tar xzvf phplot-4.4.6.tar.gz
tar xzvf phpMyAdmin-2.3.0-php.tar.gz
mv phplot-4.4.6 phplot
mv phpMyAdmin-2.3.0 phpMyAdmin
ln -s phplot phplot-4.4.6
ln -s adodb adodb-2.31
ln -s acid acid-0.9.6b21
ln -s phpMyAdmin phpMyAdmin-2.3.0
```

(i quattro link creati non sono fondamentali, servono solo a ricordarci la versione installata dei vari programmi). Nessuna configurazione è necessaria per adodb e phplot che sono richiamati direttamente da ACID. La configurazione di ACID è nel file `acid/acid_conf.php`, in Tabella 3 riportiamo alcuni dei parametri che occorre o conviene modificare oltre ovviamente ad inserire la username e password di MySQL. L'installazione di phpMyAdmin non è obbligatoria, ma da la possibilità di gestire completamente MySQL via interfaccia Web, il che può essere molto comodo. La configurazione di phpMyAdmin risiede nel file `config.inc.php` ed i principali parametri da configurare per il nostro minimo setup (la configurazione di phpMyAdmin può essere alquanto complessa ed avanzata) sono indicati in Tabella 4 (sostituire a 127.0.0.1 il numero IP del

³ Le versioni indicate sono quelle che sono state utilizzate nei nostri test.

Server Virtuale di Apache). In questa configurazione di phpMyAdmin si è scelto di restringere l'accesso di phpMyAdmin solo ai 2 database creati per Snort. Sia per Acid che per phpMyAdmin vi è il tipico problema che la username e la password per l'accesso al database risiedono in forma testuale in chiaro nei file di configurazione. Entrambi i file di configurazione non dovrebbero essere accessibili tramite Apache in ogni caso (sono dei file che contengono solo delle definizioni di Php4) ma per evitare possibili problemi possiamo porre nella configurazione di Apache un vincolo esplicito che impedisce il caricamento diretto di questi file. Rimane però sempre la possibilità di un utente locale (benigno o maligno che sia) che può leggere direttamente questi due file. Per evitare ciò, assumendo che Apache giri come utente e gruppo *apache* e che quindi anche Php4 giri come questo utente, diamo i seguenti comandi sempre come root

```
cd /var/www/snort
chown -R root:apache *
chmod -R g-w,o-rwx *
```

in modo che solo l'utente root può leggere e scrivere in queste cartelle, mentre gli utenti che appartengono al gruppo apache, di solito solo l'utente apache, possono solo leggere i file contenuti in queste cartelle (si verifichi in modo particolare che i file di configurazione hanno le protezioni corrette).

L'installazione e configurazione dei programmi è finita, possiamo ora attivare Apache ed indirizzare il nostro browser alla pagina (sempre sostituendo a 127.0.0.1 il numero IP del vostro Server Virtuale) <https://127.0.0.1/Acid/> oppure <https://127.0.0.1/phpMyAdmin/> per accedere alla console di Snort od alla gestione diretta di MySQL.

7. Acid in azione

Dalla nostra scrivania possiamo ora accedere alla pagina Web iniziale di ACID sulla console (sempre che abbiamo configurato correttamente la username e password ed i filtri di accesso per Apache). Quello che ci appare è riportato in Figura1. Nella prima pagina ACID ci da la possibilità di avere un riassunto degli ALERT ricevuti da MySQL (la pagina si aggiorna automaticamente) sia in forma grafica che numerica, e permette poi di accedere direttamente agli ALERT. Sono possibili varie modalità di accesso ai dati (basta cliccare sui numeri, le percentuali, i protocolli ecc.), le principali sono

1. ricerca diretta nel database degli ALERT;
2. elenco in ordine temporale di tutti gli ALERT arrivati di un certo tipo (per protocollo [tcp, udp, icmp], per sensore, per indirizzo IP sorgente o destinazione, per porta ecc.);
3. elenco in ordine temporale inverso (ultimi per primi) degli ultimi ALERT arrivati di un certo

tipo, raggruppati per classe di ALERT (indicato dalla parola *unique*);

4. ALERT più frequenti di un certo tipo;

e così via. In particolare la modalità 3. permette di accedere in maniera rapida e concisa agli ultimi ALERT organizzati per classe di attacco. Da qui, selezionando il numero degli ALERT in una classe, si ottiene il loro elenco, e selezionando ancora il numero dell'ALERT si ottiene la descrizione completa del pacchetto, incluso il payload, che ha fatto scattare l'ALERT, vedi ad esempio Figura2. In ogni pagina che mostra degli ALERT è possibile fare delle azioni su di essi quali: inviare per email gli ALERT, gruppi di ALERT o classi di ALERT a qualcuno (l'indirizzo email è però hardcoded nella configurazione di ACID), cancellare o spostare nell'archivio l'ALERT, aggiungere l'ALERT ad un Alert Group (AG). Questa ultima funzione è molto comoda per organizzare gli ALERT a seconda di criteri personali, ad esempio per dividerli secondo dei gruppi di lavoro che rispecchiano la propria rete aziendale. Inoltre, ogni volta che ACID carica una pagina controlla se sono arrivati nuovi ALERT e segnala in rosso quanti ne sono arrivati con la frase *Added # alert(s) to the Alert cache*.

Rimangono da considerare un paio di situazioni utili e semplici da risolvere. E' spesso utile permettere a qualche altro tecnico di accedere alla console di ACID senza però dare il permesso di modificare o cancellare gli ALERT stessi. Il modo più semplice è quello di creare in Apache un nuovo Server Virtuale, con username e password diverse da quelle di Acid, con il nome di AcidViewer (<https://127.0.0.1/AcidViewer/>), diretto verso la cartella `/var/www/snort/acidviewer/` nella quale copiamo nuovamente ACID. La configurazione di questa ulteriore versione di ACID è uguale alla precedente basta però non mettere l'username e la password per il database `snort_archive`. Infine creiamo l'utente *acidviewer* in MySQL in modo simile all'utente *acid* ma ponendo *N* per il *delete_priv*. Un utente che si collega a AcidViewer è ora in grado di vedere tutti gli ALERT, aggiungerli ai gruppi di ALERT, ma non è in grado di cancellare né di spostare gli ALERT nell'archivio. Si può fare un'analoga configurazione per accedere all'archivio `snort_archive` tramite un ulteriore Server Virtuale.

Visto che di norma i sensori inviano molti dati alla console, è necessario una gestione periodica del database, cancellando o spostando nell'archivio gli ALERT vecchi ed ottimizzando le tabelle di MySQL ad esempio attraverso phpMyAdmin, per evitare possibili rallentamenti del database ed aumentarne l'efficienza.

Andrea Pasquinucci

Consulente di Sicurezza Informatica

pasquinucci@ucci.it

Riferimenti Bibliografici:

- <http://www.snort.org/>
- Judy Novak and Stephen Northcutt, *An Analyst's Handbook* (<http://vig.prenhall.com/catalog/academic/product/1,4096,0735710082,00.html>)
- Paul Innella, Oba McMillan, and David Trout *Managing Intrusion Detection Systems in Large Organizations*, <http://online.securityfocus.com/infocus/1564>.
- Kevin Timm, *Strategies to Reduce False Positives and False Negatives*, <http://online.securityfocus.com/infocus/1463>
- Anton Chuvakin and Vladislav V. Myasnyankin, *A Complete Snort-Based IDS Architecture*, <http://online.securityfocus.com/infocus/1640>
- LinuxSecurity.com, *Intrusion Detection Response*, http://www.linuxsecurity.com/feature_stories/ids-active-response.html
- Guardian Active Response for Snort, <http://www.chaotic.org/guardian/>
- per la gestione delle regole e dei sensori: IDScener (<http://www.packx.net>) e SRRAM (<http://sourceforge.net/projects/srram>)
- per il blocco automatico di attacchi tramite un firewall: SnortSAM (<http://www.snortsam.net/>)

-d	Dump the application layer data when displaying packets in verbose or packet logging mode
-i IF	interface name (also any on some platforms)
-h IP	internal network
-D	run as daemon
-u user	user to run as
-g group	group to run as
-z	log or alert <u>only</u> on established connections
-p	do not use promiscuous IF sniffing
-N	do not log, only alert
-o	first PASS rules then other
-y	add year to date
-m NNN	default umask
-e	add link-layer headers
-a	add arp packets
-I	add IF name to report
-X	dump raw data from link-layer up
-l dir	logging directory
-T	do not run, just test config

Tabella 1 Opzioni di snort sulla linea di comando

```
CONNECT mysql;
REPLACE INTO user (host,user,password)
  VALUES ('localhost','snort',PASSWORD('snortpassword'));
REPLACE INTO user (host,user,password)
  VALUES ('%', 'snort',PASSWORD('snortpassword'));
REPLACE INTO user (host,user,password)
  VALUES ('localhost','acid',PASSWORD('acidpassword'));
REPLACE INTO db (host,db,user,select_priv,insert_priv,update_priv,
  delete_priv,create_priv,drop_priv)
  VALUES ('localhost','snort','snort','Y','Y','Y','Y','Y','N');
REPLACE INTO db (host,db,user,select_priv,insert_priv,update_priv,
  delete_priv,create_priv,drop_priv)
  VALUES ('%', 'snort','snort','Y','Y','Y','Y','Y','N');
REPLACE INTO db (host,db,user,select_priv,insert_priv,
  update_priv,delete_priv,create_priv,drop_priv )
  VALUES ('localhost','snort','acid','Y','Y','Y','Y','Y','N');
REPLACE INTO db (host,db,user,select_priv,insert_priv,
  update_priv,delete_priv,create_priv,drop_priv)
  VALUES ('localhost','snort_archive','acid','Y','Y','Y',
  'Y','Y','N');

CREATE DATABASE IF NOT EXISTS snort;
CONNECT snort;
# add full path if not in current directory
source create_mysql;
# this is 12MB of data, optional, originally compressed
#source snortdb-extra;
CREATE DATABASE IF NOT EXISTS snort_archive;
CONNECT snort_archive;
# add full path if not in current directory
source create_mysql;
# this is 12MB of data, optional, originally compressed
#source snortdb-extra;

FLUSH PRIVILEGES;
```

Tabella 2. File per la creazione dei database e delle tabelle di MySQL

```
$DBLib_path = "../adodb";
$DBtype = "mysql";

$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "acid";
$alert_password = "acidpassword";

$archive_dbname = "snort_archive";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "acid";
$archive_password = "acidpassword";

$ChartLib_path = "../phplot";

$use_sig_list = 2;
$resolve_IP = 0;
```

Tabella 3. Configurazione di ACID

```
$cfg['PmaAbsoluteUri'] = 'https://127.0.0.1/phpMyAdmin/';
$cfg['Servers'][$i]['host'] = 'localhost';
$cfg['Servers'][$i]['connect_type'] = 'socket';
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'acid';
$cfg['Servers'][$i]['password'] = 'acidpassword';
$cfg['Servers'][$i]['only_db'] = array('snort', 'snort-archive');
```

Tabella 4. Configurazione di phpMyAdmin

File Edit View Go Bookmarks Tools Window Help

Analysis Console for Intrusion Databases

Added 0 alert(s) to the Alert cache

Queried on : Tue November 12, 2002 12:38:45
 Database: snort@localhost (schema version: 106)
 Time window: [2002-10-21 21:39:02] - [2002-11-11 10:15:52]

<p>Sensors: 7 Unique Alerts: 24 (5 categories) Total Number of Alerts: 698</p> <ul style="list-style-type: none"> • Source IP addresses: 48 • Dest. IP addresses: 32 • Unique IP links 81 • Source Ports: 222 <ul style="list-style-type: none"> ◦ TCP (219) UDP (3) • Dest. Ports: 218 <ul style="list-style-type: none"> ◦ TCP (210) UDP (8) 	<p>Traffic Profile by Protocol</p> <p>TCP (68%) </p> <p>UDP (2%) </p> <p>ICMP (30%) </p> <p>Portscan Traffic (0%) </p>
--	---

- Search
- Graph Alert data (**EXPERIMENTAL**)
- Snapshot
 - Most recent Alerts: any protocol, TCP, UDP, ICMP
 - Today's: alerts unique, listing; IP src / dst
 - Last 24 Hours: alerts unique, listing; IP src / dst
 - Last 72 Hours: alerts unique, listing; IP src / dst
 - Most recent 15 Unique Alerts

 - Last Source Ports: any , TCP , UDP
 - Last Destination Ports: any , TCP , UDP
 - Most frequent 5 Alerts
 - Most Frequent Source Ports: any , TCP , UDP
 - Most Frequent Destination Ports: any , TCP , UDP
 - Most frequent 15 addresses: source, destination
- Graph alert detection time
- Alert Group (AG) maintenance
- Application cache and status

[Loaded in 0 seconds]

ACID v0.9.6b21 (by Roman Danyliw as part of the AirCERT project)

Figura 1. La pagina introduttiva della console Acid

File Edit View Go Bookmarks Tools Window Help

ACID **Alert** [Home](#) [Search](#) | [AG Maintenance](#) [\[Back \]](#)

Queried DB on : Tue November 12, 2002 12:42:03

Meta Criteria	Signature "[CVE] [CVE] SNMP public access udp" ...clear...
IP Criteria	any
Layer 4 Criteria	none
Payload Criteria	any

Added 0 alert(s) to the Alert cache

Alert #1

[First] >> Next #1-(6-2424)

Meta	<table border="1"> <thead> <tr> <th>ID #</th> <th>Time</th> <th>Triggered Signature</th> </tr> </thead> <tbody> <tr> <td>6-2423</td> <td>2002-10-25 10:25:44</td> <td>[CVE] [CVE] SNMP public access udp</td> </tr> </tbody> </table>	ID #	Time	Triggered Signature	6-2423	2002-10-25 10:25:44	[CVE] [CVE] SNMP public access udp	
	ID #	Time	Triggered Signature					
	6-2423	2002-10-25 10:25:44	[CVE] [CVE] SNMP public access udp					
<table border="1"> <thead> <tr> <th>Sensor</th> <th>name</th> <th>interface</th> <th>filter</th> </tr> </thead> <tbody> <tr> <td>hal</td> <td>any</td> <td>none</td> <td></td> </tr> </tbody> </table>	Sensor	name	interface	filter	hal	any	none	
Sensor	name	interface	filter					
hal	any	none						
<table border="1"> <tr> <td>Alert Group</td> <td>none</td> </tr> </table>	Alert Group	none						
Alert Group	none							

IP	<table border="1"> <thead> <tr> <th>source addr</th> <th>dest addr</th> <th>Ver</th> <th>Hdr Len</th> <th>TOS</th> <th>length</th> <th>ID</th> <th>flags</th> <th>offset</th> <th>TTL</th> <th>chksum</th> </tr> </thead> <tbody> <tr> <td>212.78.2.248</td> <td>212.78.2.33</td> <td>4</td> <td>5</td> <td>0</td> <td>68</td> <td>25441</td> <td>0</td> <td>0</td> <td>127</td> <td>10898</td> </tr> </tbody> </table>	source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum	212.78.2.248	212.78.2.33	4	5	0	68	25441	0	0	127	10898
	source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum												
212.78.2.248	212.78.2.33	4	5	0	68	25441	0	0	127	10898													
<table border="1"> <tr> <td>Options</td> <td>none</td> </tr> </table>	Options	none																					
Options	none																						

UDP	<table border="1"> <thead> <tr> <th>source port</th> <th>dest port</th> <th>length</th> </tr> </thead> <tbody> <tr> <td>2811</td> <td>161</td> <td>48</td> </tr> </tbody> </table>	source port	dest port	length	2811	161	48	
	source port	dest port	length					
2811	161	48						
<table border="1"> <tr> <td>Payload</td> <td>length = 40</td> </tr> <tr> <td></td> <td>000 : 30 26 02 01 00 04 06 70 75 62 6C 69 63 AD 19 02 0&....public...</td> </tr> <tr> <td></td> <td>010 : 01 2F 02 01 00 02 01 00 30 0E 30 0C 06 08 2B 06 /.....0.0...+</td> </tr> <tr> <td></td> <td>020 : 01 02 01 01 02 00 05 00</td> </tr> </table>	Payload	length = 40		000 : 30 26 02 01 00 04 06 70 75 62 6C 69 63 AD 19 02 0&....public...		010 : 01 2F 02 01 00 02 01 00 30 0E 30 0C 06 08 2B 06 /.....0.0...+		020 : 01 02 01 01 02 00 05 00
Payload	length = 40							
	000 : 30 26 02 01 00 04 06 70 75 62 6C 69 63 AD 19 02 0&....public...							
	010 : 01 2F 02 01 00 02 01 00 30 0E 30 0C 06 08 2B 06 /.....0.0...+							
	020 : 01 02 01 01 02 00 05 00							

[First] >> Next #1-(6-2424)

Action
{ action } <input type="button" value="Selected"/>

[Loaded in 0 seconds]

ACID v0.9.6b21 (by Roman Danyliw as part of the AirCERT project)

Figura 2. Un pacchetto in dettaglio