Sicurezza delle reti[1]

Mattia Monga

Dip. di Informatica e Comunicazione
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2010/11

Lezione XIV: Snort

Snort

È il piú noto e diffuso NIDS open-source creato da Martin Roesch nel 1998: molto diffuso anche in realtà aziendali.

- Viene fornito con un insieme di regole create dalla comunità
- Regole aggiornate certificate dai loro laboratori richiedono un abbonamento annuale; rilasciate gratuitamente con un ritardo di 30 giorni.
- Al contrario di quanto avviene con molti tool proprietari, le definizioni delle firme sono visibili, possono essere analizzate e modificate a piacere (c'è anche una procedura per segnalare i falsi positivi).

http://www.snort.org

Le firme di snort

Si tratta di un NIDS signature based:

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET any \
2 (msg:"SCAN SYN FIN"; \
3 flags:SF; \
4 reference:arachnids,198; \
5 classtype:attempted-recon; sid:624; rev:2;)
```

- Header della firma (riga 1):
  - azione (alert) eseguita in corrispondenza di un match positivo
  - protocollo da analizzare (es. tcp)
  - mittente e destinatario (indirizzi IP, netmask e porte)
- Body (righe 2–5):
  - messaggio di alert (es."SCAN SYN FIN")
  - condizioni per il pattern matching (es. flags:SF)
  - informazioni riguardanti la firma stessa (es. reference, classtype, sid, rev)

Le firme di snort

Si tratta di un NIDS signature based:

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET any \
2 (msg:"SCAN SYN FIN"; \
3 flags:SF; \
4 reference:arachnids,198; \
5 classtype:attempted-recon; sid:624; rev:2;)
```

Il meccanismo di pattern matching esegue un puro confronto tra stringhe.

- dall'header IP verifica se il protocollo di trasporto è tcp e se mittente e destinatario (ind.IP e porte) sono uguali o inclusi
- dall'header TCP (in questo esempio) verifica le condizioni espresse dalle opzioni, in particolare se sono settati i flag TCP SYN e FIN

Se tutti questi confronti sono soddisfatti viene generato l'alert (ad esempio, messaggio a video, log su database, email etc.)

Plug-in

I preprocessori di Snort sono plug-in che vengono attivati prima che il detection engine esegua il pattern matching delle firme.
Consultare il manuale per la lista completa dei preprocessori

Frag3 deframmenta i pacchetti riassemblandoli prima dell'analisi.

```
1  # frag3_global options:
2  # max_frags: Maximum number of frag trackers that may be active at once. Default value
3  # memcap: Maximum amount of memory that frag3 access at any given time. Default val
4  # prealloc_frags: Maximum number of individual fragments that may be processed at once
5  # instead of the memcap system, uses static allocation to increase performance. No defau
6  #
7  # frag3_engine options:
8  # timeout: Amount of time a fragmented packet may be active before expiring. Default va
9                seconds.
10 # ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
11 # Based on the initial received fragment TTL.
12 # min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this value wil
```

Plug-in

HTTP Inspect decodifica connessioni HTTP ricavandone i campi definiti dal protocollo HTTP e normalizzandone i valori. HTTP Inspect agisce sia sulle richieste dei client che sulle risposte dei server. Attualmente non gestisce lo stato applicativo, pertanto l'analisi viene fatta solo sui singoli pacchetti.

Esempio di configurazione:

```
1 preprocessor http_inspect_server: server 10.1.1.1 profile apache ports { 80 }
```

Plug-in

Stream4 riassembla stream TCP fornendo in questo modo a Snort la capacit di eseguire un'analisi stateful. Di default gestisce 8192 connessioni TCP simultanee, ma in grado di arrivare fino a oltre 100.000.

```
1  # options (options are comma delimited):
2  # detect_scans – stream4 will detect stealth portscans and generate alerts when it sees th
3  # disable_evasion_alerts – turn off the possibly noisy mitigation of overlapping sequences.
4  # min_ttl [number] – set a minium ttl that snort will accept to stream reassembly
5  # noinspect – turn off stateful inspection only
6  # timeout [number] – set the session timeout counter to [number] seconds, default is 30
7  # max_sessions [number] – limit the number of sessions stream4 keeps track of
8  # memcap [number] – limit stream4 memory usage to [number] bytes
9  # server_inspect_limit [bytes] – Byte limit on server side inspection.
10 # Available options (comma delimited):
11 # clientonly – reassemble traffic for the client side of a connection only
12 # serveronly – reassemble traffic for the server side of a connection only
13 # both – reassemble both sides of a session
```

Azioni

alert	produce un alert e logga il pacchetto.
log	logga il pacchetto.
pass	ignora il pacchetto.
activate	genera un alert e attiva una dynamic rule.
dynamic	attende che venga attivata una activate rule, poi agisce come una log rule.
drop	(snort_inline) richiede a iptables di scartare il pacchetto, dopodiché lo logga.
reject	(snort_inline) richiede a iptables di scartare il pacchetto generando un TCP Reset (o un ICMP port unreachable se il protocollo e' UDP), dopodichè lo logga.
sdrop	(snort_inline) richiede a iptables di scartare il pacchetto senza loggarlo.

Utilità delle pass action e dynamic rule

- pass: può essere utile per limitare il numero di falsi positivi senza complicare troppo la gestione delle firme: per esempio nel caso di un server con un particolare servizio che si è verificato generare un grande numero di falsi positivi
- activate/dynamic: permettono di attivare una regola successiva al match di una precedente. In questo modo si possono definire catene di firme che vengono analizzate.

Regole

Le regole (quelle fornite inizialmente sono già piú di 5000!) sono organizzate in categorie

```
1  # The following rulesets are disabled by default:
2  #
3  # web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
4  # chat, multimedia, and p2p
5  #
6  # These rules are either site policy specific or require tuning in order to not
7  # generate false positive alerts in most environments.
8  include $RULE_PATH/local.rules
9  include $RULE_PATH/bad-traffic.rules
10 include $RULE_PATH/exploit.rules
11 include $RULE_PATH/scan.rules
12 include $RULE_PATH/finger.rules
13 include $RULE_PATH/ftp.rules
14 include $RULE_PATH/telnet.rules
15 ...
```

scan.rules


```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET any
2 (msg:"SCAN XMAS"; flow:stateless; flags:SRAFPU,12;
3 reference:arachnids,144; classtype:attempted-recon; sid:625;
4 rev:7;)
```

flow:stateless significa che il pacchetto viene analizzato singolarmente.

flags:SRAFPU, 12 indica i flag TCP settati (SRAFPU), e quelli non settati (12, ovvero i due flag riservati del byte).

scan.rules


```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SCAN SSH
2 Version map attempt"; flow:to_server,established;
3 content:"Version_Mapper"; nocase; classtype:network-scan;
4 sid:1638; rev:5;)
```

flow:to_server si analizzano i pacchetti di richiesta da client a server.

nocase nell'analisi del payload non si fa distinzione tra maiuscole e minuscole.

content:"Version_Mapper" si cerca la stringa "Version_Mapper" nel payload.

dos e ddos

```
1 alert tcp $HOME_NET 12754 -> $EXTERNAL_NET any (msg:"DDOS
2 mstream handler to client"; flow:to_client,established;
3 content:">"; reference:cve,2000-0138; classtype:attempted-
4 dos; sid:248; rev:4;)
```

flow:to_client, established si analizzano i pacchetti di risposta da server a client che fanno parte di una sessione nello stato di established.

```
1 alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"DOS
2 Teardrop attack"; fragbits:M; id:242; reference:bugtraq,124;
3 reference:cve,1999-0015; reference:nessus,10279;
4 reference:url,www.cert.org/advisories/CA-1997-28.html;
5 classtype:attempted-dos; sid:270; rev:7;)
```

fragbits:M il flag More Fragment settato.

id:242 valore del campo IP Identifier

telnet


```
1 alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any
2 (msg:"TELNET root login"; flow:from_server,established;
3 content:"login|3A| root"; classtype:suspicious-login;
4 sid:719; rev:7;)
```

| serve a delimitare caratteri esadecimali: |3A| = :

ftp


```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP PASS
2 overflow attempt"; flow:to_server,established;
3 content:"PASS"; nocase; isdataat:100,relative;
4 pcre:"/^PASS\s[^\n]{100}/smi"; reference:bugtraq,10078;
5 classtype:attempted-admin; sid:1972; rev:17;)
```

Viene cercata la stringa PASS e verificato che dopo il termine della stringa (relative) ci siano almeno altri 100 bytes.
Supporta la definizione di espressioni regolari di tipo Perl Compatible Regular Expression (PCRE).

exploit


```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT ssh
2 CRC32 overflow"; flags:A+; content:"|00 01 57 00 00 00 18|";
3 offset:0; depth:7; content:"|FF FF FF FF 00 00|"; offset:8;
4 depth:14; reference:bugtraq,2347; reference:cve,CVE-2001-
5 0144; classtype:shellcode-detect; sid:1327; rev:2;)
```

offset specifica dopo quanti byte dall'inizio del payload deve iniziare la ricerca; depth indica quanti byte di payload analizzare.

Attack response


```
1 alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any
2 (msg:"ATTACK-RESPONSES Invalid URL";
3 flow:from_server,established; content:"Invalid URL";
4 nocase;
5 reference:url,www.microsoft.com/technet/security/bulletin/MS00-063.mspx;
6 classtype:attempted-recon; sid:1200; rev:10;)
```

Commenti?

web misc


```
1  alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
2  (msg:"WEB-MISC cross site scripting attempt";
3  flow:to_server,established; content:"<SCRIPT>"; nocase;
4  classtype:web-application-attack; sid:1497; rev:6;)
5  alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-
6  MISC /etc/passwd"; flags:A+; content:"/etc/passwd"; nocase;
7  classtype:attempted-recon; sid:1122; rev:2;)
8  alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
9  (msg:"WEB-MISC webadmin.dll access";
10 flow:to_server,established; uricontent:"/webadmin.dll";
11 nocase; reference:bugtraq,7438; reference:bugtraq,7439;
12 reference:bugtraq,8024; reference:cve,2003-0471;
13 reference:nessus,11771; classtype:web-application-activity;
14 sid:2246; rev:6;)
```

p2p


```
1 alert tcp $HOME_NET any -> $EXTERNAL_NET 6881:6889 (msg:"P2P
2 BitTorrent transfer"; flow:to_server,established;
3 content:"|13|BitTorrent protocol"; depth:20;
4 classtype:policy-violation; sid:2181; rev:2;)
5 alert tcp $HOME_NET any -> $EXTERNAL_NET 4242 (msg:"P2P
6 eDonkey transfer"; flow:to_server,established;
7 content:"|E3|"; depth:1;
8 reference:url,www.kom.e-technik.tu-darmstadt.de/publications/abstracts/HB02-1.html;
9 classtype:policy-violation; sid:2586; rev:2;)
```