

Introducción a conceptos de IDS y técnicas avanzadas con Snort



**Alejandro Gramajo
BAICOM networks**

2008-2021

Temas

Introducción a IDS

- Tipos

- Técnicas de detección

- Problemas

Snort

- Open Source

- Add-ons

- Mayor performance

Creación de reglas

- Examinar trafico

- Ejemplos

Modificando paquetes

- Snort INLINE

- Ejemplos

Visualizacion

- Snort IDS en la realidad

- Para seguir leyendo

Introducción a IDS

Detección de intrusos es el proceso de monitorear computadoras o redes, para detectar entradas no autorizadas, actividad o modificación de archivos

Tipos

Host IDS

Network IDS

Hybrid IDS

Introducción a IDS

Técnicas de detección

Anomaly (anomalía)

Descubre patrones anómalos comparándolo con los considerados normales

Especial interés para la aplicación de algoritmos genéticos y redes neuronales

Signature (firma)

Comparación de firmas almacenadas contra una porción de un paquete de red

Reducción de análisis, subconjunto del total

Target (objetivo)

Ej. Busca modificaciones en un archivo específico

Introducción a IDS

Problemas

Existen 2 problemas muy importantes

Falsos positivos y Falsos negativos

Los falsos positivos se pueden dividir en categorías

Reactionary traffic alarms (ej. icmp network unreachable)

Equipment-related alarms (ej. load balancers)

Protocol violations (ej. outlook express auth=plain)

Non malicious alarms (ej. publicamos un shellcode)

True false positives

Resumiendo

Un IDS para que sea útil y confiable debe producir los mínimos falsos positivos posibles y “ningún” falso negativo

Snort

Open Source

Utiliza técnicas de detección de firmas y anomalías no estadística

Gratuito y con licencia GPL v2

detrás esta la empresa Sourcefire, ~~adquirida por~~
~~Checkpoint~~, adquirida por Cisco

Modos de ejecución

Sniffer y Packet Logger

Network IDS

IPS con FlexResp o Inline

Snort

Open Source (cont.)

Componentes

- Packet Decoder (toma los datos de libpcap o libipq)

- Preprocesadores o Input plugins

Detection Engine

Logging y alertas

Output plugins

- reportar y depositar alertas en distintos formatos

Snort

Add-ons

Facilidad para programarlos

Existen más de 14 preprocesadores.

- Port scanning (flow-portscan y sfportscan)

- Frag2 (ip packet defragmentation)

 - Frag3 (a partir de 2.4.x, mas rapido y por host)

- Stream4 (tcp stream reassembly y stateful inspection)

- Stream5 (reemplaza all 4 y al flow)

- Http-inspect

- Arp-spoof

Snort

Add-ons (cont.)

No integrados por default

- Spp-fnord (multi-architecture mutated NOP sled detector)

- Spade (statistical packet anomaly detection engine)

Output plugins

- Database (mysql, postgres, etc)

- Syslog

- XML

- Traps SNMP

- Mensajes SMB

Snort

Mayor performance

El análisis en tiempo real, requiere de muchos recursos de “cpu” y “memoria”

Optimización de reglas, necesidad de entender como funciona el engine

Tres formas de aumentar

- Mmaped pcap (utilizacion de un “shared buffer”)

- en vez de copiar los paquetes de kernel memory a userland memory, se utiliza un shared memory buffer, la libpcap encola paquetes en el buffer y Snort puede leerlos directamente

- Barnyard (utiliza el output plugin “unified”)

- PF_RING (http://www.ntop.org/PF_RING.html)

Creando nuevas reglas

Formato de una regla

Rule header (action, protocol, address, port, direction, address, port)

Rule options

```
alert tcp $HOME 22 -> $EXT any \  
  msg: "SSH version 1 support detected; \  
  flow: to_client, established; \  
  content: "SSH-1."; \  
  nocase; \  
  offset: 0; \  
  depth: 6;)
```

Creando nuevas reglas

Tips para armar reglas eficaces y veloces

Armar el patrón de regla para la vulnerabilidad, no el código exploit

Utilizar “content” siempre que sea posible

Atrapar las singularidades del protocolo

Las reglas poseen naturaleza recursiva

Probar valores numéricos (2.x)

Byte_test

Byte_jump

Creando nuevas reglas

Recursividad en las reglas

buscamos un paquete de size 1 byte con el “;” adentro, pensariamos esta regla (Fig. 1)

sin embargo si tenemos un paquete de 1000 bytes con todos “;”, macheara 1000 veces, para evitarlo. (Fig. 2)

```
content:"|29|"; dsize: 1;
```

Fig. 1

```
dsize: 1; content:"|29|";
```

Fig. 2

Creando nuevas reglas

Examinando tráfico

Cuando creamos una regla para un servicio específico, lo mejor que podemos hacer es analizar el protocolo del mismo. Buscar como funciona, donde están las fallas o donde pueden estar

Utilizar herramientas de sniffing

Creando nuevas reglas

Ejemplo

Queremos generar una alerta cuando se intenta loguear un usuario “root” al ftp

```
alert tcp any any -> any any 21 \  
  (content: “user root”; )
```

Creando nuevas reglas

Ejemplo (cont.)

Tenemos un problema ya que el protocolo acepta

```
USER R00T
user root
user    root
user<tab>root
```

Creando nuevas reglas

Ejemplo (cont.)

Flow: se utiliza para verificar que el trafico este yendo hacia el server y establecido (mantiene un tracking de conexiones)

Pcre: es para expresiones regulares

Una regla mejor

```
alert tcp any any -> any 21 \  
  (flow: to_server, established; \  
  content: "root"; \  
  pcre: "/user\s+root/i"; )
```

Modificando paquetes

Snort Intrusion Prevention System (IPS)

Necesidad de ser gateway/firewall o bridge, la topología de la red es dependiente

Tener en cuenta que cuando Snort cometa un “falso positivo” se bloqueara una conexión “válida”

Un IPS puede loguearse:

- Examinar los logs en tiempo real. (ej. SnortGuardian)

- Utilizar FlexResp o Inline.

 - Se pueden armar reglas que bloqueen o rechacen las conexiones, de acuerdo a los patrones.

Modificando paquetes

Snort Inline

También nos permite modificar el payload de un paquete, esto nos puede servir cuando tenemos un server que no podemos patchear o el patch todavía no salio y no podemos desactivar la funcionalidad que posee el bug

Limitación importante, el tamaño del paquete machado y el tamaño del paquete modificado *deben ser iguales*

Baja la performance

Modificando paquetes

Ejemplo: consultas MySQL

Existen algunos comandos “peligrosos”

UNION SELECT

LOAD_FILE ...

LOAD DATA INFILE ...

SELECT ... INTO OUTFILE ...

BENCHMARK ...

UFD

DROP DATABASE ...

Algunos se pueden eliminar desde línea desde my.cnf

ej. set-variable=local-infile=0

Modificando paquetes

Ejemplo: consultas MySQL (cont.)

Utilizando Inline, eliminamos el *drop database*

```
alert tcp any any <> any any \  
  (msg: "mysql replace"; \  
  content: "drop database"; \  
  replace: "select 'LAMER'"; )
```

```
mysql> drop database test;  
+-----+  
| test  |  
+-----+  
| LAMER |  
+-----+  
1 row in set (0.01 sec)
```

Modificando paquetes

Ejemplo: mirando el protocolo MySQL

Levantamos Snort en modo sniffer y ejecutamos un *drop database test*

HEADER

```
-----  
03/10-19:07:09.572760 200.123.146.139:32975 -> 192.168.0.2:3306  
TCP TTL:62 TOS:0x8 ID:12232 IpLen:20 DgmLen:73 DF  
***AP*** Seq: 0x35C18013 Ack: 0xE0DAC3DA Win: 0x16D0 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 156914262 165056290
```

PAYLOAD

```
-----  
11 00 00 00 03 64 72 6F 70 20 74 61 62 6C 65 20 .....drop table  
74 65 73 74 31                                test1
```

Modificando paquetes

Ejemplo: mirando el protocolo MySQL (cont.)

11 00 00	body length= 17 bytes (little endian)
00	packet= 0
03	command= QUERY
64 72 6F 70 20	args= "drop "
d r o p	
74 61 62 6C 65 20	args= "table "
t a b l e	
74 65 73 74 31	args= "test1"
t e s t 1	

Modificando paquetes

Ejemplo: mirando el protocolo MySQL (cont.)

Usando Inline le decimos que dropee el paquete (Fig. 1)

Con FlexResp no utilizamos *libipq*, *rst_snd* envía TCP-RST e *icmp_all* envía paquetes *unreachable* (Fig. 2)

```
reject tcp $EXT any -> $HOME 3306 \  
  (msg: "mysql drop reject"; \  
  content: "drop table"; \  
  offset: 5; )
```

Fig. 1

```
alert tcp $EXT any -> $HOME 3306 \  
  (msg: "mysql replace"; \  
  content: "drop table"; \  
  offset: 5; \  
  resp: rst_snd, icmp_all; )
```

Fig. 2

Modificando paquetes

Ejemplo: mirando el protocolo MySQL (cont.)

Supongamos que existe un bug en los query MySQL, y se produce cuando el tamaño del paquete es mayor a 4095 (0x0FFF) (Fig. 1)

Hacemos una regla que si el paquete query es mayor dispare una alerta (Fig. 2)

```
00 10 00  body length= 0x1000 (4096)
          (little endian)
XX  packet= 0
03  command= 0x03 == QUERY
```

Fig. 1

```
alert tcp $EXT any -> $HOME 3306 \
(msg: "mysql cmd query > 4095"; \
byte_test: 3, >, 4095, 0, little; \
content: "|03|"; \
offset: 4; )
```

Fig. 2

Modificando paquetes

Ejemplo: evitando un buffer overflow

Analizando el exploit para “Arkeia Network Backup Client”, el bo se produce cuando un paquete de tipo 77 es enviado con la seccion de datos muy grande. Mirando el protocolo. (Fig. 1)

Hacemos una regla (Fig. 2)

10 00	2-bytes type 77 request
XX XX XX XX	4-bytes ?
nn nn	2-bytes size (big)
...	seccion datos

Fig. 1

```
alert tcp $EXT any -> $HOME 617 \
(msg: "arkeia bo"; \
content:"|00 4d|"; \
depth:2; \
byte_test: 2, >, 23, 6; \
isdataat:31; \      # hay datos ?
content:!"|00|"; \  # != 0x00
offset: 8; \
depth:23; )
```

Fig. 2

Modificando paquetes

Ejemplo: detectando un SQL injection y un XSS

```
flow:to_server, established; \
uricontent:".cgi"; \

META CHARS (') (--) (#)
pcre:"/(\%27)|(\')|(\-\-\)|(\%23)|(#)/i";

STORE PROCEDURE EXEC SP XP
pcre:"/exec(\s|\+)+(s|x)p\w+/ix";

UNION
pcre:"/((\%27)|(\'))union/ix";

XSS (<) (>)
pcre:"/((\%3C)|<)[^\n]+((\%3E)|>)/i";
```

Visualización

Necesidades y problemática

- Grandes cantidades de informacion

 - Necesidad de descubrir patrones

 - Data mining

- Correlacion con otros IDS / Firewall / Netflow

- Web frontends: BASE, BASE+ (Acid obsoleto)

- NSM: Sguil

- SQL (alertas estan en DB)

Snort IDS en la realidad

Necesidades y problemática

Capacitación, conectar un IDS en la red sin customizarlo no tiene sentido alguno

Know-How en networking, para entender y armar las reglas

Recursos para ver las alertas, tampoco sirve tener el mejor IDS configurado y no mirar los logs porque genera muchos falsos positivos

Actualización constante en materia de seguridad

Por lo general cuando los recursos no alcanzan, se busca tercerizar la seguridad interna

Para seguir leyendo

El paper original se lo pueden bajar de

<http://people.baicom.com/~agramajo/notes/ids2005.pdf>

Snort docs

<http://www.snort.org/docs/>

Security Focus Archive IDS

<http://www.securityfocus.com/infocus/ids>

Listas de seguridad

Full-Disclosure, FOCUS-IDS, Vuln-Dev

Bugs – exploits

<http://www.packetstormsecurity.org/>

<http://cve.mitre.org/cve/>

<http://www.osvdb.org/>

Preguntas?

Gracias

[agramajo at baicom.com](mailto:agramajo@baicom.com)

Contacto

BAICOM
Av. Los Incas 4107 Oficina 13
(C1427DNG) Buenos Aires
Argentina

Tel/Fax: (+5411) 5032.3366
info@baicom.com
<http://www.baicom.com>