

IES Valle Inclán



Práctica SNORT

Ethan Erwin Sánchez
Víctor Rodríguez Pérez

Índice

Índice	2
Preparación inicial	3
Configuración de reglas.....	4
comprobación de las reglas.....	5
Conectar con telegram	6
comprobación telegram	8
Conclusión.....	8

Preparación inicial

El primer paso es activar el servicio de iptables en el equipo, en nuestro caso no hizo falta ya que venían activadas por defecto.

El siguiente paso es instalar el servicio de snort usando

```
Sudo apt update | sudo apt install snort
```

Durante la instalación nos pedirá que red queremos proteger, en nuestro caso pusimos la 192.168.0.0/24

Configuración de snort
<p>Tiene que utilizar el formato CIDR, esto es, 192.168.1.0/24 para un bloque de 256 IPs o 192.168.1.42/32 para sólo una dirección. Debe separar múltiples direcciones por «,» (comas) y sin espacios.</p> <p>Puede dejar este valor en blanco y configurar HOME_NET en /etc/snort/snort.conf en su lugar. Esto es útil si utiliza Snort en un sistema que cambia frecuentemente de red y no tiene una dirección IP estática asignada.</p> <p>Tenga en cuenta que si Snort está configurado para utilizar múltiples interfaces se utilizará esta definición como valor de «HOME_NET» para todos ellos.</p> <p>Intervalo de direcciones para la red local:</p> <p>192.168.0.0/24</p> <p><Ok></p>

Aquí terminaría la configuración inicial de snort, si hacemos un comando para ver el estado veremos que esta activo

```
ethan at ethan-kawai in 家
UwU sudo service snort status 0 (49.183s) < 12:14:23
● snort.service - LSB: Lightweight network intrusion detection system
   Loaded: loaded (/etc/init.d/snort; generated)
   Active: active (running) since Tue 2024-12-10 12:14:16 CET; 26s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 4458 ExecStart=/etc/init.d/snort start (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 9456)
   Memory: 78.8M
      CPU: 200ms
    CGroup: /system.slice/snort.service
            └─4478 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g snort --pid-path /run/snort

dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_FTPTELNET Version 1.2 <Build>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_GTP Version 1.1 <Build 1>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_SDF Version 1.1 <Build 1>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_DCERPC2 Version 1.0 <Build>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_REPUTATION Version 1.1 <Bui>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_DNS Version 1.1 <Build 4>
dic 10 12:14:16 ethan-kawai snort[4478]: Preprocessor Object: SF_SIP Version 1.1 <Build 1>
dic 10 12:14:16 ethan-kawai snort[4478]: Commencing packet processing (pid=4478)
lines 1-21/21 (END)
```

Configuración de reglas

El propio snort nos facilita un fichero por defecto sin ninguna regla para poder agregar nuestras propias reglas, pero nosotros por facilidad de organizar y saber que es cada regla crearemos nuestros propio ficheros en el directorio

/etc/snort/rules/XXX.rules

La primera regla que crearemos será para detectar los ping que se hagan a nuestra red, crearemos nuestro archivo en /etc/snort/rules/regla1SAD.rules

```
ethan at ethan-kawai in 家
UwU cat /etc/snort/rules/regla1SAD.rules 0 (02:02.651) < 12:17:26
alert icmp any any -> $HOME_NET any (msg:"Se ha recibido un ping";sid<
<PING";sid: 1000001;rev:1;classtype:icmp-event;)
ethan at ethan-kawai in 家
UwU 0 (0.001s) < 12:17:37
```

Nuestra segunda regla será para detectar ssh, por lo cual crearemos una regla nueva en /etc/snort/rules/regla2SAD.rules

```
ethan at ethan-kawai in 家
UwU cat /etc/snort/rules/regla2SAD.rules 1 (0.003s) < 12:54:40
alert tcp any any -> $HOME_NET any (msg: "Conexion hacia el server ssh UwU"; sid: 1000002; rev:1; classtyp
e: tcp-connection;)
ethan at ethan-kawai in 家
UwU 0 (0.000s) < 12:54:46
```

Después hay que añadir estas 2 reglas creadas al archivo de configuración y ya que te encuentras en el archivo de configuración hay que cambiar el HOME_NET de any a la ip especificado en la configuración inicial, que se encuentra en /etc/snort/snort.conf

```
# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overridden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.0.0/24
```

```
GNU nano 6.2 /etc/snort/snort.conf *
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/community-sql-injection.rules
include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
include $RULE_PATH/regla1SAD.rules
include $RULE_PATH/regla2SAD.rules
#####
# Step #8: Customize your preprocessor and decoder alerts
```

comprobación de las reglas

Para poner a funcionar el snort hay que usar el siguiente comando:

```
Sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
```

Una vez ejecutado esto si hacemos un ping o una conexión ssh nos saldrá una alerta

```
(ethan@alexakis)-[~]
$ ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=0.560 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=64 time=0.375 ms
64 bytes from 192.168.0.5: icmp_seq=3 ttl=64 time=0.424 ms
64 bytes from 192.168.0.5: icmp_seq=4 ttl=64 time=0.463 ms
64 bytes from 192.168.0.5: icmp_seq=5 ttl=64 time=0.253 ms
64 bytes from 192.168.0.5: icmp_seq=6 ttl=64 time=0.371 ms
^C
— 192.168.0.5 ping statistics —
6 packets transmitted, 6 received, 0% packet loss, time 5112ms
rtt min/avg/max/mdev = 0.253/0.407/0.560/0.093 ms

UwU sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3 0 (0.001s) < 12:47:12
12/10-12:47:28.870347 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:28.870365 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
12/10-12:47:29.888893 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:29.888898 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
12/10-12:47:30.923337 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:30.923365 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
12/10-12:47:31.944793 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:31.944813 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
12/10-12:47:32.968409 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:32.968425 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
12/10-12:47:33.984954 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.10 -> 192.168.0.5
12/10-12:47:33.984969 [**] [1:1000001:1] se esta realizando un ping [**] [Classification: Generic ICMP event] [Priority: 3] (ICMP) 192.168.0.5 -> 192.168.0.10
```

Ahora comprobaremos el ssh

```
(ethan@alexakis)-[~]
$ ssh ethan@192.168.0.5
The authenticity of host '192.168.0.5 (192.168.0.5)' can't be established.
ED25519 key fingerprint is SHA256:JBI+YaxMFLV4iLos8gFRLh7oqw78so9VGsrbYPGn1I.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.5' (ED25519) to the list of known hosts.
ethan@192.168.0.5's password:
Welcome to Ubuntu 22.10 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

592 updates can be applied immediately.
422 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ethan@ethan-kawai
OS: Uwuntu 22.10 LTS x86_64
Uptime: 8 mins
Shell: fish 3.3.1
Terminal: /dev/pts/2
CPU: Intel i7-10700K (6) @ 3.792GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 901MiB / 7949MiB
CPU Usage: 2%
Local IP: 10.224.18.250

ethan at ethan-kawai in 家
UwU
```

```

ethan at ethan-kawai in ~
UwU sudo snort -A console -n -c /etc/snort/snort.conf -i enp0s3 0 (01:11.290) < 12:57:19
12/10-12:57:43.612891 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.613504 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.613673 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.637000 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.648908 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.689843 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.700658 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22
12/10-12:57:43.710944 [**] [1:1000002:1] Conexión hacia el server ssh UwU [**] [Classification: A TCP connection was detected] [Priority: 4] (TCP) 192.168.0.10:44836 -> 192.168.0.5:22

```

Conectar con telegram

Ahora para que las alertas se puedan ver sin tener que estar delante de la pantalla configuraremos un bot de telegram para que nos envíen las alertas, para ellos clonaremos un repositorio de git que nos viene todo lo necesario

```

ethan at ethan-kawai in 家/Documentos/telegram
UwU git clone https://github.com/gagaltotal/Snort-Bot-Telegram-Shell 0 (0.000s) < 08:54:37
Cloning into 'Snort-Bot-Telegram-Shell'...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 101 (delta 4), reused 0 (delta 0), pack-reused 88 (from 1)
Receiving objects: 100% (101/101), 556.37 KiB | 1.39 MiB/s, done.
Resolving deltas: 100% (46/46), done.

```

También tendremos que crear un bot nuevo con BotFather en telegram, asique en el buscador de telegram pondremos @botfather y hablaremos con la cuenta que esta verificada

```

/deletegame - delete an existing game 09:08
/newbot 09:09 ✓
Alright, a new bot. How are we going to call it? Please choose a
name for your bot. 09:09
Avisos SNORT 09:09 ✓
Good. Now let's choose a username for your bot. It must end in
'bot'. Like this, for example: TetrisBot or tetris_bot. 09:09
Snort_bot 09:09 ✓
Sorry, this username is already taken. Please try something
different. 09:09
SnortUwU_bot 09:09 ✓
Done! Congratulations on your new bot. You will find it at
t.me/SnortUwU_bot. You can now add a description, about
section and profile picture for your bot, see /help for a list of
commands. By the way, when you've finished creating your cool
bot, ping our Bot Support if you want a better username for it.
Just make sure the bot is fully operational before you do this.
Use this token to access the HTTP API:
7277831578:AAH4xsboHhkCm_ga1shKu6HfYFpCJYA-PYQ
Keep your token secure and store it safely, it can be used by
anyone to control your bot.
For a description of the Bot API, see this page:
https://core.telegram.org/bots/api 09:09

```

Una vez hayamos clonado el repositorio git y creado el nuevo bot hay que configurar el archivo que se encuentra dentro de donde hayamos clonado el repositorio de git llamado Bot-tele.sh

Dentro de ese archivo hay que poner el token y chat-id que sacaremos de telegram, pero para ello hay que hacer una configuración previa.

Para saber el token nos lo dice botfather al crear el bot y con ese token podremos sacar el chat_id, para ello iremos al buscador del navegador y pondremos

Api.telegram.org/"token de nuestro bot" y si nunca hemos interactuado con nuestro bot nos saldrá en blanco, así que iremos al chat del bot con el enlace que nos da botfather y pondremos /start, después refrescamos la página y nos saldrá mas información relacionado con el bot y también el chat_id

```
{ "ok": true, "result": { { "update_id": 379437636,
"message": { "message_id": 5, "from":
{ "id": 852663754, "is_bot": false, "first_name": "Ethan", "last_name": "Erwin", "username": "ethane66", "language_code": "en", "chat":
{ "id": 852663754, "first_name": "Ethan", "last_name": "Erwin", "username": "ethane66", "type": "private", "date": 1733995661, "text": "/start", "entities": { { "offset": 0, "length": 6, "type": "bot_command" } } } } } }
```

Con esto y el token del bot podremos configurar el archivo bot-tele.sh

```
#Chat ID dan bot token Telegram
chat_id="4608874111"
token="7277831578:AAH4xsboHhkCm_ga1shKu6HfYFPcJYA-PYQ"
```

Por ultimo hay que crear e indicar donde pondremos los logs de telegram, por comodidad he creado el archivo logs en el mismo directorio donde esta el bot-tele.sh y indicaremos el archivo en de configuración la ruta de los logs

```
GNU nano 6.2
#!/bin/bash

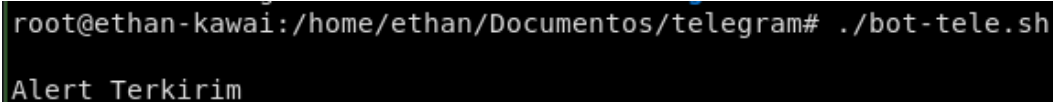
#init
initCount=0
logs=/home/ethane/telegram/log-tele.txt
```

comprobación telegram

Para poder comprobar que funciona el bot de telegram iniciaremos snort con unos parámetros distintos

```
Sudo snort -i enp0s3 -c /etc/snort/snort.conf -l /var/logs/snort/ -d -A  
console > /home/Ethan/documentos/telegram/log-tele.sh
```

En otra terminal cambiaremos los permisos del archivo bot-tele.sh para que pueda tener permisos de ejecución y después lo ejecutaremos con sudo, una vez tengamos el snort y el archivo sh ejecutados haremos un ssh a la maquina y nos saldrá esta alerta



```
root@ethan-kawai:/home/ethan/Documentos/telegram# ./bot-tele.sh  
Alert Terkirim
```

(Debido a que la red de clase bloquea telegram no puedo adjuntar una imagen del mensaje recibido en telegram, pero esta captura indica que se envió el mensaje)

Conclusión

En esta practica hemos visto que un programa simple como snort nos puede ayudar a detectar intentos de ataques a nuestra red, ya que viene con un montón de opciones preconfigurados mas las reglas especificas que queremos tener y que se puede hacer de manera que se pueda monitorear de manera desatendida con telegram