IDS

Carles Mateu

Requirements

To do this labs you'll need:

- To have the CentOS 8 virtual machine for the lab. Available at: http://codeandbeer.org/virtual/SegAC/ova/CentOS-IDS-updated.ova

  Password for root user: *test*

> **IMPORTANT**
> We should have VirtualBox running with a CentOS virtual machine, with the network on NAT network, the provided one is already at step 3 of this guide (to save time).

Snort installation

1. We start by installing and upgrading our CentOS system:

```
dnf install epel-release -y

dnf config-manager --add-repo /etc/yum.repos.d/CentOS-
    PowerTools.repo

dnf config-manager --set-enabled PowerTools

dnf install -y https://forensics.cert.org/cert-forensics-
    tools-release-el8.rpm
```

2. This will add additional repositories and enable PowerTools repo. So we can upgrade:

```
dnf upgrade -y
```

3. **After upgrade reboot**
4. Now we will install some required libraries and tools:

```
dnf install daq
dnf localinstall snort-2.9.17-1.centos8.x86_64.rpm
```

   o

```
dnf install https://www.snort.org/downloads/snort/snort
    -2.9.17-1.centos8.x86_64.rpm
```

Old version was:

```
dnf install https://www.snort.org/downloads/snort/snort
    -2.9.15.1-1.centos7.x86_64.rpm
```

5. We should be able (we won't) to run snort, for example in sniffer mode:

```
snort -v
```

6. Now we have to solve some issues with paths (the RPM is incorrectly built):

   One missing depending library:

```
dnf install libdnet
```

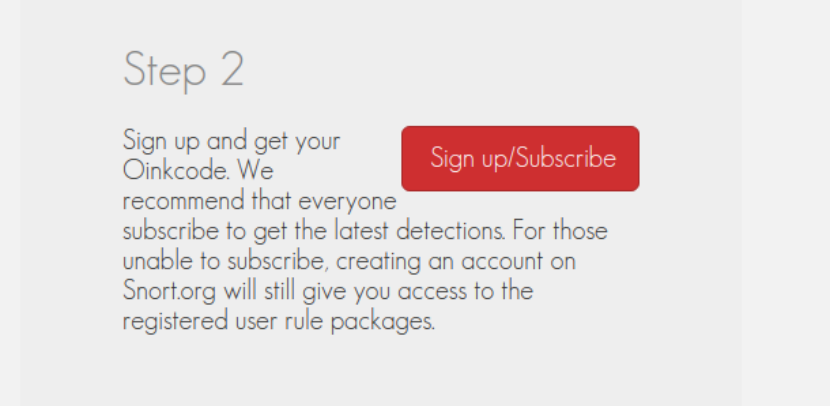
   One incorrectly referenced DL:

```
ln -s /usr/lib64/libdnet.so.1 /usr/lib64/libdnet.1
```

7. We should be able now to run snort, for example in sniffer mode:

```
snort -v
```

8. Now we will need to subscribe to get our oinkcode, so we will be able to download more rulesets. On the snort website http://snort.org, on get started, search for this:



Subscribe to get the oinkcode (a numerical ID), so we can download rulesets:

9. Download rulesets for our version:

```
cd ~ (go home)
dnf install -y wget
wget https://www.snort.org/rules/snortrules-snapshot
    -29151.tar.gz?oinkcode=<our oinkcode> -O snortrules-
    snapshot-29151.tar.gz
wget https://www.snort.org/downloads/community/community-
    rules.tar.gz -O community-rules.tar.gz
dnf install snort-sample-rules
```

This gives us 3 rulesets. The provided one (snort-sample-rules), some community created rules, and an update of snort rulesets (snortrules-snapshot). And we have one such set installed (sample), although is very scarce.

10. Test if we can run snort in IDS mode (we won't).

```
    snort -c /etc/snort/snort.conf
```

We will have to fix a couple variables on the config file.

1. This variable:

```
dynamicdetection directory /usr/local/lib/
    snort_dynamicrules
```

points to an inexisting path. We can either change it or create the path.

```
mkdir /usr/local/lib/snort_dynamicrules
```

2. Snort now will complain about this path:

```
/etc/snort/../rules/white_list.rules
```

Search on snort.conf the line where WHITE_LIST_PATH is set, and change it (and BLACK_LIST_PATH) to:

```
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

11. Now we should be able to run snort in IDS mode, so we have snort correctly configured:

```
snort -dev -l ./log -c /etc/snort/snort.conf
```

Using Snort

We will use two virtual machines, the Kali linux one (for instance), and the snort one. If we don't have another virtual machine (like the Kali one), we can stop the snort virtual machine. Clone it, and use two copies of snort.

12. Note down the IP adresses of the snort machine ($SNORT_IP) and of the other machine ($HOST_IP), to make it easier to follow. If you want, you can define shell variables with those adresses, so writing commands is easier and less error prone.

```
SNORT_IP=10.100.6.138
HOST_IP=10.100.6.80
export SNORT_IP HOST_IP
```

Snort as sniffer

We can use snort as a sniffer. With:

```
snort -v host $SNORT_IP
```

We'll see all traffic originating or destinated to our snort machine. Then, from our host, we can ping snort, and we'll see our ICMP messages.

13. If we access our snort machine by ssh, we'll see our own access traffic, creating a feedback effect. In this case we can filter out our own ssh access:

```
snort -v host $SNORT_IP and not tcp port 22
```

Or by filtering our workstation IP if it's not the same as the HOST_IP:

```
snort -v host $SNORT_IP and not host $WORKSTATION_IP
```

14. We can increase the amount of information shown, by using -e and -d:

- -v, verbose sniffer mode, show packets on console
- -e, display (log) link layer headers.
- -d, display (log) application layer data (with -v)

So, we can do, first:

```
snort -ev host $SNORT_IP
```

and repeat the ping.

15. Then with application data:

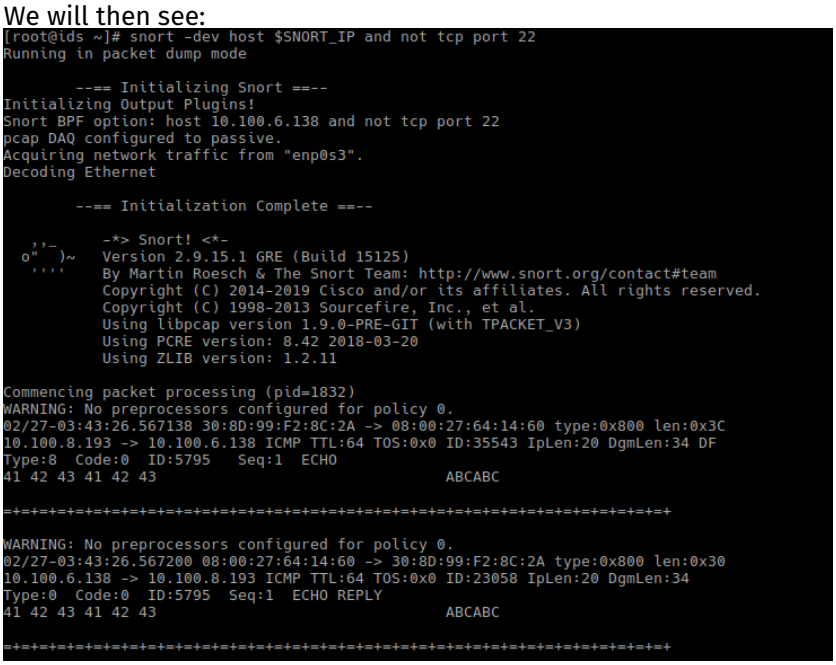
```
snort -dev host $SNORT_IP
```

We can, also, using ping, filter the amount of packets, the size of those packets, and the payload (data) in the ICMP messages. This will make easier to track our packets on the log files.

```
ping -c 1 -s 6 -p "414243414243" $SNORT_IP
```

Will send 1 packet only, with 6 bytes of payload, consisting of ABCABC (41 is hex for A, etc.)

We will then see:



Snort as logger

Besides sniffing traffic, snort can log and save all packets it sees, to replay them. The format used is PCAP format, so, tcpdump, wireshark, that also use PCAP can read snort files.

Snort stores its logs in /var/log/snort, for this lab, we'll use a local directory, to make things easier. So, create a directory, called log, and we'll use it:

```
mkdir log
```

16. Now do another sniff run, logging packets to that dir:

```
snort -dev -l ./log host $SNORT_IP
```

We'll have some files there, one called:

```
snort.log.TTTTTTTTTT
```

TTTTTT is the time the capture started (unix time). We'll see its a pcap file:

```
file -ki ./log/snort.log.TTTTTTTT
```

17. We can read (replay) it, with tcpdump if we have it:

```
dnf install -y tcpdump
tcpdump -nn -r  ./log/snort.log.TTTTTTTT
```

We'll get a read of the file:

```
[root@ids ~]# tcpdump -nn -r ./log/snort.log.1582793714
reading from file ./log/snort.log.1582793714, link-type EN10MB (Ethernet)
03:55:19.196493 IP 10.100.8.193 > 10.100.6.138: ICMP echo request, id 5935, seq 1, length 14
03:55:19.196544 IP 10.100.6.138 > 10.100.8.193: ICMP echo reply, id 5935, seq 1, length 14
03:55:22.025660 IP 10.100.6.138.59626 > 10.69.1.38.123: NTPv4, Client, length 48
03:55:22.026113 IP 10.69.1.38.123 > 10.100.6.138.59626: NTPv4, Server, length 48
03:55:24.202495 ARP, Request who-has 10.100.8.193 tell 10.100.6.138, length 28
```

18. Or with snort:

```
snort -r ./log/snort.log.TTTTTTTT | less
```

Snort as IDS

Snort can operate as an IDS (in fact, that's its intended purpose). For that we'll need rules, those rules are in: /etc/snort/rules

We probably have some rules, or placeholders (empty files) for that rules, if we have not added rulesets.

19. We will start by adding a simple rule, to test how this works.

```
alert icmp any any -> any any (msg:"UDL: ICMP Packet
    found"; sid:1000001; rev:1;)
```

This rule will generate an alert for **any** ICMP (ping for instance) packet found. The message includes a string, UDL, easy to search on the logs.

We start snort, adding -L to change the name of the log file, and -c pointing to the configuration file.

```
snort -dev -l ./log -L udl -c /etc/snort/snort.conf host
    $SNORT_IP
```

We do a couple pings, to generate traffic that we can inspect from the other computer (or terminal, or whatever). Afterwards, on the log dir, we'll have a udl.TTTTTT file, we can read with tcpdump or snort.

We will also have an alert file, with text descriptions of all alerts we have encountered. We can use any log watcher to inspect that file.

20. We'll add now a real one, from a real issue. Add to /etc/snort/rules/local.rules:

```
alert icmp any any -> $HOME_NET any (msg:"UDL: ICMP Large
     ICMP Packet"; dsize:>800; reference:arachnids,246;
    classtype:bad-unknown; sid:499; rev:4;)
```

and remove or comment previous ICMP rule (so we don't have doubts on which rule applies).

Now we can do a ping, of large size, to raise the alert:

```
ping -c 1 -s 4000 -p "414243414243" $SNORT_IP
```

Watch logs

21. Watching log files all day is not practical, we want to receive alerts, as soon as possible, if something happens. To that avail, there are programs complementing Snort, like Barnyard, Snorby, etc. One simple program, we can use today in the lab, is swatch.

Swatch watches logfiles and takes actions following some predefined rules. Install it:

```
dnf install swatch
```

and create a small text file, swatchconfig, for example, with this content:

```
watchfor /UDL/
    echo=red
    exec /bin/echo "UDL message" | /bin/mail -s "ABCD
        again!" one@gemail.address
```

This file tells swatch to wait for the text UDL to appear on the watched file, and when it appears to do two things, first, echo on screen the line, on red, and second, send “UDL message” to the mail command (will fail, we don’t have it installed, and would fail anyway due to security policies).

We can use this (without the echo, the idea is to avoid watching the screen) to send emails, SMS, Telegram, Whatsapp, play music, ...

22. Start now the watching on another terminal:

```
swatch -c swatchconfig -t log/alert
```

and repeat the ping:

```
ping -c 1 -s 4000 -p "414243414243" $SNORT_IP
```

Content rules

23. We can create rules that inspect content of the packets, for example:

```
alert icmp any any -> any any (msg:"UDL: root"; content:"
    root";)
```

Will check all ICMP traffic that has root in its payload.