TD 1 : Système de détection d'intrusions :

Partie 1 :

1) Lorsqu'on lance snort avec les options -vde, on obtient la sortie suivante :

```
root@efd84df1e289:/# snort -vde
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=757)
```

Dès qu'on commence à utiliser l'un des 2 sites web, snort commence à capturer les paquets et à les afficher :

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-15:53:38.785059 02:42:61:2D:F0:B1 -> 02:42:AC:11:00:02 type:0x800 len:0x42
172.17.0.1:51228 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:8902 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x74A2E880  Ack: 0x9FC5B16D  Win: 0x200  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2745854384 1931732363

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-15:53:38.785256 02:42:61:2D:F0:B1 -> 02:42:AC:11:00:02 type:0x800 len:0x326
172.17.0.1:51228 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:8903 IpLen:20 DgmLen:792 DF
***AP*** Seq: 0x74A2E880  Ack: 0x9FC5B16D  Win: 0x200  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2745854384 1931732363
47 45 54 20 2F 72 65 73 74 2F 75 73 65 72 2F 77  GET /rest/user/w
68 6F 61 6D 69 20 48 54 54 50 2F 31 2E 31 0D 0A  hoami HTTP/1.1..
48 6F 73 74 3A 20 6C 6F 63 61 6C 68 6F 73 74 3A  Host: localhost:
33 30 30 30 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E  3000..Connection
3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 73 65  : keep-alive..se
63 2D 63 68 2D 75 61 3A 20 22 47 6F 6F 67 6C 65  c-ch-ua: "Google
20 43 68 72 6F 6D 65 22 3B 76 3D 22 31 30 35 22   Chrome";v="105"
2C 20 22 4E 6F 74 29 41 3B 42 72 61 6E 64 22 3B  , "Not)A;Brand";
76 3D 22 38 22 2C 20 22 43 68 72 6F 6D 69 75 6D  v="8", "Chromium
22 3B 76 3D 22 31 30 35 22 0D 0A 41 63 63 65 70  ";v="105"..Accep
74 3A 20 61 70 70 6C 69 63 61 74 69 6F 6E 2F 6A  t: application/j
73 6F 6E 2C 20 74 65 78 74 2F 70 6C 61 69 6E 2C  son, text/plain,
20 2A 2F 2A 0D 0A 44 4E 54 3A 20 31 0D 0A 73 65   */*..DNT: 1..se
63 2D 63 68 2D 75 61 2D 6D 6F 62 69 6C 65 3A 20  c-ch-ua-mobile:
3F 30 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20  ?0..User-Agent:
```

On observe que les paquets sont lisibles directement en clair.

2) Lorsqu’on arrête snort, il nous fournit des statistiques sur les paquets analysés. Ici il n’en a pas filtré un seul.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

===============================================================================
Run time for packet processing was 61.429859 seconds
Snort processed 93 packets.
Snort ran for 0 days 0 hours 1 minutes 1 seconds
   Pkts/min:           93
   Pkts/sec:            1
===============================================================================
Memory usage summary:
  Total non-mmapped bytes (arena):       786432
  Bytes in mapped regions (hblkhd):      13180928
  Total allocated space (uordblks):      680560
  Total free space (fordblks):           105872
  Topmost releasable block (keepcost):   103456
===============================================================================
Packet I/O Totals:
   Received:           95
   Analyzed:           93 ( 97.895%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            2 (  2.105%)
   Injected:            0
===============================================================================
```

```
Breakdown by protocol (includes rebuilt packets):
        Eth:           93 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:           93 (100.000%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:            4 (  4.301%)
        TCP:           86 ( 92.473%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
   IP6 Opts:            0 (  0.000%)
      Frag6:            0 (  0.000%)
      ICMP6:            0 (  0.000%)
       UDP6:            0 (  0.000%)
       TCP6:            0 (  0.000%)
     Teredo:            0 (  0.000%)
    ICMP-IP:            0 (  0.000%)
    IP4/IP4:            0 (  0.000%)
    IP4/IP6:            0 (  0.000%)
    IP6/IP4:            0 (  0.000%)
    IP6/IP6:            0 (  0.000%)
        GRE:            0 (  0.000%)
    GRE Eth:            0 (  0.000%)
   GRE VLAN:            0 (  0.000%)
    GRE IP4:            0 (  0.000%)
    GRE IP6:            0 (  0.000%)
GRE IP6 Ext:            0 (  0.000%)
   GRE PPTP:            0 (  0.000%)
    GRE ARP:            0 (  0.000%)
    GRE IPX:            0 (  0.000%)
```

3) Allons lire les options de la ligne de commande sur le manuel en ligne : http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-3-SECT-3.html.

-v : Option “verbose”, affiche tous les paquets sur la console.

Sans cette option : Le résultat est le même parce qu’on utilise snort en mode packet logging

-d : Affiche les charges utiles des paquets.

Sans cette option :

```
10/04-16:08:31.364590 02:42:AC:11:00:02 -> 02:42:61:2D:F0:B1 type:0x800 len:0x42
172.17.0.2:3000 -> 172.17.0.1:51290 TCP TTL:64 TOS:0x0 ID:13630 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xBE11E52B  Ack: 0xC3F48AA  Win: 0x1F5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1932624943 2746746964
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-16:08:31.365540 02:42:AC:11:00:02 -> 02:42:61:2D:F0:B1 type:0x800 len:0x1B3
172.17.0.2:3000 -> 172.17.0.1:51290 TCP TTL:64 TOS:0x0 ID:13631 IpLen:20 DgmLen:421 DF
***AP*** Seq: 0xBE11E52B  Ack: 0xC3F48AA  Win: 0x1F5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1932624944 2746746964
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-16:08:31.365597 02:42:61:2D:F0:B1 -> 02:42:AC:11:00:02 type:0x800 len:0x42
172.17.0.1:51290 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:43660 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xC3F48AA  Ack: 0xBE11E69C  Win: 0x1FE  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2746746965 1932624944
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-16:08:31.365738 02:42:61:2D:F0:B1 -> 02:42:AC:11:00:02 type:0x800 len:0x3BD
172.17.0.1:51310 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:54317 IpLen:20 DgmLen:943 DF
***AP*** Seq: 0x8DC1BDF1  Ack: 0xA3B0917E  Win: 0x200  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2746746965 1932624942
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

-e : Affiche les information correspondants à la couche de liaison (adresse MAC).

Sans cette option :

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/04-16:12:38.583895 172.17.0.2:34048 -> 172.217.18.46:80
TCP TTL:64 TOS:0x0 ID:17182 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x33BE06A7  Ack: 0x8EE17357  Win: 0x1F5  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
10/04-16:12:38.583928 172.217.18.46:80 -> 172.17.0.2:34048
TCP TTL:37 TOS:0x0 ID:0 IpLen:20 DgmLen:324 DF
***AP*** Seq: 0x8EE17357  Ack: 0x33BE06A7  Win: 0xFFFF  TcpLen: 20
65 69 67 68 74 3A 35 34 70 78 3B 77 69 64 74 68  eight:54px;width
3A 31 35 30 70 78 7D 0A 20 20 3C 2F 73 74 79 6C  :150px}.  </styl
65 3E 0A 20 20 3C 61 20 68 72 65 66 3D 2F 2F 77  e>.  <a href=//w
```

4) Lorsqu’on lance une image docker en mode port-forwarding comme indiqué dans le TD, la pile réseau n’est pas entièrement initialisée. Il est donc nécessaire de lancer l’image normalement. On obtient donc une image qui répond aux pings :

```
thomas@DESKTOP-4DF2HCG:~$ ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=58 time=2.46 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=58 time=3.44 ms
```

(On note au passage que le délai de réponse est 50 fois plus élevé que lors d'un ping vers l'hôte local).

Lorsque j'ai lancé snort en mode sniffer sur le docker configuré sur WSL, il n'est pas parvenu à décoder correctement les paquets ICMP, et ne détectait rien du tout.

J'ai donc recommencé cette question sur un docker installé sur une machine virtuelle Ubuntu.

Depuis docker, j'ai lancé : **snort -l log -dve icmp**

On ping ensuite l'image docker:

```
thomas@thomas-VirtualBox:~/Bureau$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.199 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.084 ms
```

(ça fonctionne cette fois-ci)

Evidemment, snort n'affiche rien à l'écran puisqu'il sauvegarde tout dans le fichier de log.

Et enfin on lit le fichier de sortie de snort, lui aussi en binaire :

On constate qu'il est illisible tel quel

Pour afficher le contenu du fichier : **snort -dvr log/snort.log.1664902520**

Il s'agit bien de paquets ICMP :

```
Commencing packet processing (pid=119)
WARNING: No preprocessors configured for policy 0.
10/04-18:55:24.534827 172.17.0.1 -> 172.17.0.2
ICMP TTL:64 TOS:0x0 ID:36012 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:4   Seq:1  ECHO
7C 65 3C 63 00 00 00 00 76 28 08 00 00 00 00 00  |e<c....v(......
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
10/04-18:55:24.534843 172.17.0.2 -> 172.17.0.1
ICMP TTL:64 TOS:0x0 ID:20652 IpLen:20 DgmLen:84
Type:0  Code:0  ID:4  Seq:1  ECHO REPLY
7C 65 3C 63 00 00 00 00 76 28 08 00 00 00 00 00  |e<c....v(......
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

5) Pour logger les paquets au format binaire (au format tcpdump) : **snort -b icmp**

Si on devait capturer beaucoup de paquets, la sauvegarde sous ce format permettrait de gagner en performances et d'avoir des fichiers plus petits.

6) Pour afficher le fichier binaire, on peut utiliser snort : **snort -r /var/log/snort/snort.log.1664903854**

On obtient le même résultat que précédemment :

```
WARNING: No preprocessors configured for policy 0.
10/04-19:17:38.194511 172.17.0.2 -> 172.17.0.1
ICMP TTL:64 TOS:0x0 ID:1434 IpLen:20 DgmLen:84
Type:0  Code:0  ID:7  Seq:1  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
10/04-19:17:39.194056 172.17.0.1 -> 172.17.0.2
ICMP TTL:64 TOS:0x0 ID:39911 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:7   Seq:2  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Mais on peut aussi procéder avec la commande *tcpdump,* qu'il faut au préalable installer :

tcpdump -r /var/log/snort/snort.log.1664903854 (ajouter l'option *-X* pour avoir le contenu des paquets au format hexadécimal)

```
root@7b3f580e1469:~# tcpdump -r /var/log/snort/snort.log.1664903854
reading from file /var/log/snort/snort.log.1664903854, link-type EN10MB (Ethernet)
19:17:38.191640 IP 172.17.0.1 > 7b3f580e1469: ICMP echo request, id 7, seq 1, length 64
19:17:38.194511 IP 7b3f580e1469 > 172.17.0.1: ICMP echo reply, id 7, seq 1, length 64
19:17:39.194056 IP 172.17.0.1 > 7b3f580e1469: ICMP echo request, id 7, seq 2, length 64
19:17:39.194091 IP 7b3f580e1469 > 172.17.0.1: ICMP echo reply, id 7, seq 2, length 64
19:17:40.213444 IP 172.17.0.1 > 7b3f580e1469: ICMP echo request, id 7, seq 3, length 64
19:17:40.213460 IP 7b3f580e1469 > 172.17.0.1: ICMP echo reply, id 7, seq 3, length 64
root@7b3f580e1469:~#
```

Partie 2 :

1) Pour commencer, je fais en sorte qu'il soit possible de se connecter en SSH à mon docker. Pour cela, j'installe le paquet **openssh-server**. Je change le mot de passe **root** avec la commande **passwd**. J'autorise la connexion de l'utilisateur root dans le fichier **/etc/ssh/sshd_config** (**PermitRootLogin yes**). Lorsque j'essaie de me connecter, je constate que docker ne veut pas qu'on écoute sur le port 22. Je change donc le port du serveur SSH à **1022** dans le même fichier de configuration. Je redémarre le serveur, et il est désormais possible de se connecter en SSH à mon docker :

```
thomas@thomas-VirtualBox:~/Bureau$ ssh root@172.17.0.2 -p 1022
The authenticity of host '[172.17.0.2]:1022 ([172.17.0.2]:1022)' can't be established.
ED25519 key fingerprint is SHA256:kAw4p6KuofyVzlM1GzbkYKdDdy+Z7jqBavy4T2/myug.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.17.0.2]:1022' (ED25519) to the list of known hosts.
root@172.17.0.2's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-48-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Oct  4 20:18:01 2022 from 127.0.0.1
root@ba0f777b4e8d:~# exit
logout
Connection to 172.17.0.2 closed.
```

Commentons donc toutes les règles de détection snort sauf celles dans **/etc/snort/rules/local.rules**.

Ajoutons ensuite la règle **alert tcp any any -> any 1022 (msg: "Tentative de connexion SSH, bikkuri shita :O"; sid:460)**

On lance maintenant snort en mode IDS :

snort -dev -l log -c /etc/snort/snort.conf -i eth0

Puis on essaie de se connecter en SSH au docker :

```
thomas@thomas-VirtualBox:~/Bureau$ ssh root@172.17.0.2 -p 1022
root@172.17.0.2's password:
```

(on interrompt la connexion pour ne pas surcharger de paquets).

Aucune alerte levée.

Je consulte une FAQ officielle : https://www.snort.org/faq/i-m-not-receiving-alerts-in-snort

Il s'agit peut-être d'un problème de « TCP offload », lorsque la carte réseau calcule elle-même le checksum des paquets pour décharger le CPU de cette tâche.

Lançons la commande correcte : **snort -dev -l log -c /etc/snort/snort.conf -i eth0**

Cela semble fonctionner

```
Action Stats:
     Alerts:            14 ( 48.276%)
     Logged:            14 ( 48.276%)
     Passed:             0 (  0.000%)
```

On reçoit une alerte sur environ la moitié des paquets, ce qui est logique étant donné que l'autre moitié correspond aux réponses du serveur.

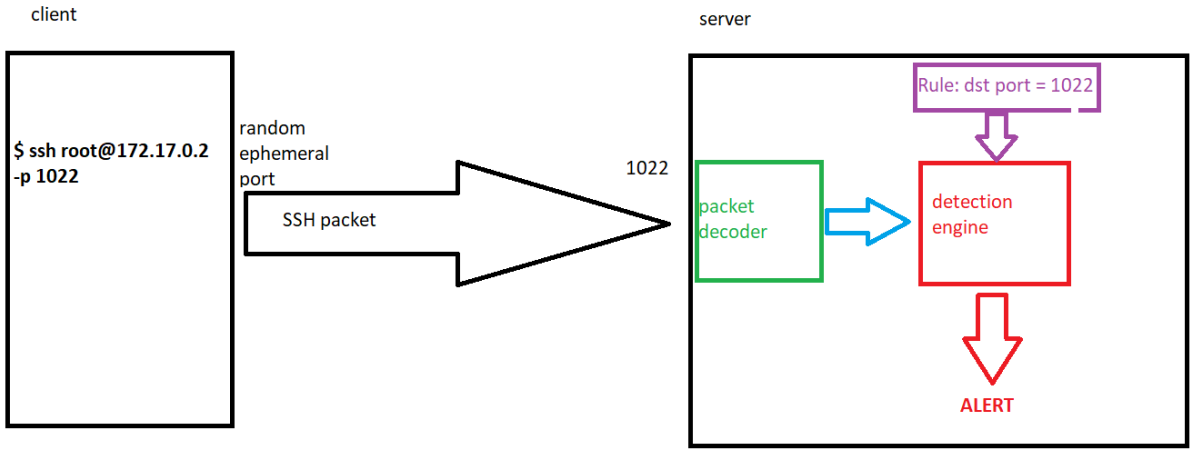
Le fichier de log s'est rempli :

```
root@ba0f777b4e8d:~# cat log/snort.log && echo
h<c<o●●●●●●●V●fc<●●c<●●●●JB●B●●●E<fO@@|7●●●V●   ●●i●●●XT●
n●●h<c<o●●●●●●●V●^c<●●c<●●●●BB●B●●●E4fP@@|>●●●V●        ●●j●$T"●●XL
n●●)●D>h<c<o●●●●●●●V●~c<●●c<●●●●bB●B●●●ETfQ@@|●●●V●     ●●j●$T"●●Xl
n●●)●D>SSH-2.0-OpenSSH_8.9p1 Ubuntu-3
h<c<●●□●●●V●^c<●●c<●●□B●B●●●E4fR@@|<●●●V●     ●●●$TK●●XL
n●●)●DKh<c<●●●"●●●●V●^c<●●c<●●●"BB●B●●●E4fU@@|9●●●V●    ●●j●$Xk●●XL
n●●)●DMh<c<●●●E●●●●V●●c<●●c<●●●ErB●B●●●EdfV@@●●●V●      ●●j●$Xk●●X|
n●●)●DM, j●●●●●●●●7●rX●WT●mh#U^}●H●D~h<c<●●V●●●●V●^c<●●c<●●VBB●B●●●E4fW@@|7●●●V●        ●●●$Z●●XL
n●●)●DWhc<●●●●●●●V●c<●●c<●●●RB●B●●●EDfX@@|&●●●V●       ●●●$Z●●X\
n●)●DW

h<      c<●●●●●●V●●     c<●●c<●●nB●B●●●E`fY@@|  ●●●V●   ●●●$Z●●Xx
n●)●D\●J●●●8r●●LSV●●●●●5●●●n%●●t●●B●kso●●f●●h<
●●●●V●^
BB●B●●●E4fZ@@|4●●●V●    ●●●●$ZK●●XL
n●)●Djh<
        c<●●P`●●●●V●●
                  c<●●c<●●P`~B●B●●●Epf[@@{●●●●V●     ●●●●$ZK●●X●
n●)●Dj●Y_2
p●9[●●i●lvR●●gJ●P●●7ğ70b●2●●g●e●●●C{e●●●●h<
                                        c<●●i●●●●●V●^
                                                    c<●●c<●●i●BB●B●●●E4f\@@|2●●●V●  ●●●$Z●●XL
c<●●c<●●U+BB●B●●●E4f]@@|1●●●V●  ●●●$Z●●XL
n●)●Dph<c<●●Y?●●●●V●^c<●●c<●●Y?BB●B●●●E4f^@@|0●●●V●     ●●●$Z●●●XL
n●)●G

root@ba0f777b4e8d:~#
```

Mais impossible de le lire

```
root@ba0f777b4e8d:~# snort -r log/snort.log
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to read-file.
ERROR: Can't initialize DAQ pcap (-1) - unknown file format
Fatal Error, Quitting..
```

Je n'ai hélas pas trouvé d'explications.



2) Pour mener à bien le scan XMAS et sa détection, je me suis basé sur le tutoriel suivant :

Commençons par retirer la règle précédemment créée et ajoutons la règle suivante :

alert tcp any any -> any 1022 (msg: "Tentative de scan XMAS, bikkuri shita :O"; flags: FPU; sid: 460; rev: 1)

Cette règle va détecter tous les scans XMAS sur le port SSH 1022. Essayons d'abord de lancer une connexion SSH conventionnelle :

ssh root@172.17.0.2 -p 1022

Aucune alerte levée :

```
Action Stats:
     Alerts:               0 (   0.000%)
     Logged:               0 (   0.000%)
     Passed:               0 (   0.000%)
```

Essayons maintenant un scan XMAS sur ce même port :

sudo nmap -sX -p1022 172.17.0.2

Cette fois-ci, snort a bien détecté un scan XMAS sur le port 1022.

```
Action Stats:
     Alerts:            2 ( 40.000%)
     Logged:            2 ( 40.000%)
```

Essayons maintenant de lancer un scan avec Python scappy :

J'ai utilisé le script à l'adresse suivante : https://github.com/PacktPublishing/Python-Penetration-Testing-Cookbook/blob/master/Chapter06/xmas-scanner.py

Le port est bien détecté :

```
Possible Open or Filtered Ports:
[1022]
```

De plus, une alerte est bien levée par snort :

```
Action Stats:
     Alerts:            1 (  1.351%)
     Logged:            1 (  1.351%)
     Passed:            0 (  0.000%)
```

3) Le fichier acquisitions.md se récupère à l'adresse http://localhost:3000/ftp/acquisitions.md

On l'obtient avec un listing directory sur le dossier ftp.

Ajoutons maintenant la règle suivant à snort : **alert tcp any any -> any 3000 (msg: "Someone is trying to steal our acquisitions!"; content: "/ftp/acquisitions.md";sid: 460; rev:1)** après avoir au préalable retiré les autres.

Il faut aussi penser à aujouter l'option **-k none** à snort puisqu'on accède au site via un port-forwarding :

snort -dev -l log -c /etc/snort/snort.conf -i eth0 -k none

Essayons de récupérer un document quelconque sur notre site. Aucune alerte levée :

```
Action Stats:
     Alerts:            0 (  0.000%)
     Logged:            0 (  0.000%)
     Passed:            0 (  0.000%)
```

Essayons de récupérer notre fichier d'acquisitions :

```
Action Stats:
     Alerts:            1 (  3.846%)
     Logged:            1 (  3.846%)
     Passed:            0 (  0.000%)
```

En effet notre paquet contient bien la chaîne recherchée :

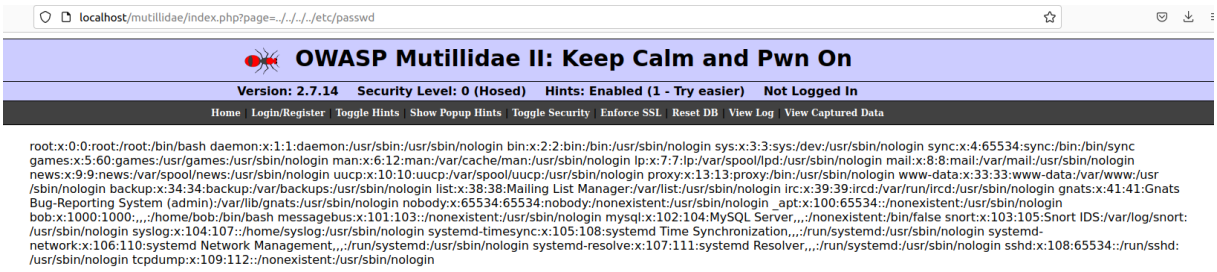
```
WARNING: No preprocessors configured for policy 0.
10/05-17:47:04.800891 02:42:52:40:3C:4B -> 02:42:AC:11:00:02 type:0x800 len:0x2EE
172.17.0.1:56536 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:62967 IpLen:20 DgmLen:736 DF
***AP*** Seq: 0xF2CEE1C1  Ack: 0xED8D7E8F  Win: 0x1F6  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3588569688 1031958709
47 45 54 20 2F 66 74 70 2F 61 63 71 75 69 73 69  GET /ftp/acquisi
74 69 6F 6E 73 2E 6D 64 20 48 54 54 50 2F 31 2E  tions.md HTTP/1.
31 0D 0A 48 6F 73 74 3A 20 6C 6F 63 61 6C 68 6F  1..Host: localho
73 74 3A 33 30 30 30 0D 0A 55 73 65 72 2D 41 67  st:3000..User-Ag
65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30  ent: Mozilla/5.0
20 28 58 31 31 3B 20 55 62 75 6E 74 75 3B 20 4C   (X11; Ubuntu; L
69 6E 75 78 20 78 38 36 5F 36 34 3B 20 72 76 3A  inux x86_64; rv:
31 30 35 2E 30 29 20 47 65 63 6B 6F 2F 32 30 31  105.0) Gecko/201
30 30 31 30 31 20 46 69 72 65 66 6F 78 2F 31 30  00101 Firefox/10
35 2E 30 0D 0A 41 63 63 65 70 74 3A 20 74 65 78  5.0..Accept: tex
74 2F 68 74 6D 6C 2C 61 70 70 6C 69 63 61 74 69  t/html,applicati
6F 6E 2F 78 68 74 6D 6C 2B 78 6D 6C 2C 61 70 70  on/xhtml+xml,app
```

4) Recherchons maintenant des vulnérabilités sur le site http://localhost/mutillidae/

Testons au hasard une page du site : http://localhost/mutillidae/index.php?page=text-file-viewer.php

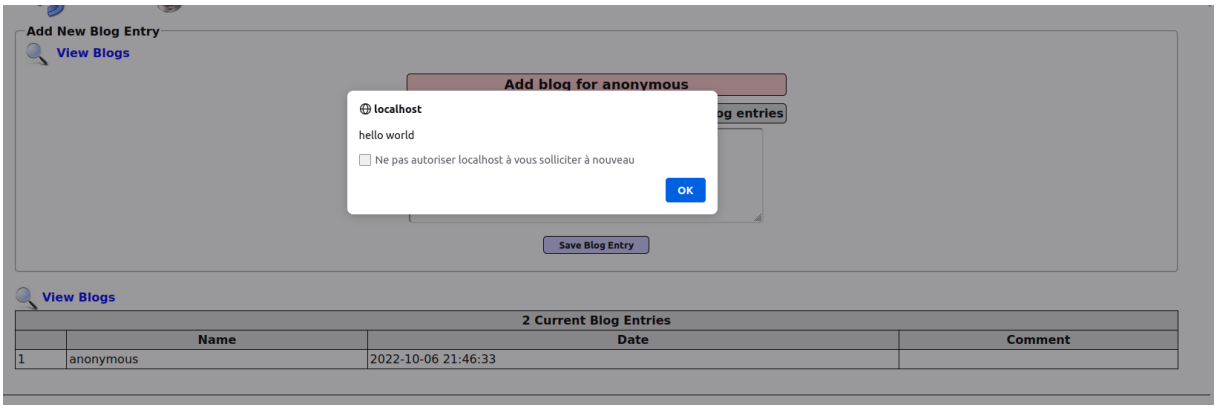
On remarque que le fichier PHP à afficher est lu comme un paramètre, nous pourrions donc essayer de lire un fichier système en modifiant un peu l'URL :

http://localhost/mutillidae/index.php?page=../../../../etc/passwd



Je n'ai pas eu à chercher bien loin pour trouver une deuxième faille de sécurité : la page http://localhost/mutillidae/index.php?page=add-to-your-blog.php nous propose en effet de remplir un formulaire. Nous pourrions essayer de vérifier s'il est bien protégé contre les injections XSS :

Il semblerait que cette page ne soit pas protégée correctement contre les failles XSS. Ces dernières permettent par exemple de voler les cookies de l'utilisateur ou bien de charger du contenu à son insu.



Note : j'ai aussi essayé de procéder à une injection SQL avec l'outil sqlmap, sans succès malheureusement.

Essayons maintenant de détecter l'exploitation de ces failles via snort.

Ajoutons la règle locale snort suivante : **alert tcp any any -> any 80 (msg: "Someone is trying to access forbidden path"; content: "/..";sid: 460; rev:1)**

Lançons snort: **alert tcp any any -> any 80 (msg: "Someone is trying to access forbidden path"; content: "/..";sid: 460; rev:1)**

Si nous essayons de consulter le site « normalement », aucune alerte n'est levée.

```
Action Stats:
     Alerts:            0 (   0.000%)
     Logged:            0 (   0.000%)
     Passed:            0 (   0.000%)
```

Par contre, si nous essayons de consulter http://localhost/mutillidae/index.php?page=../../../../etc/passwd

, notre attaque est bien détectée par snort :

```
Action Stats:
     Alerts:            1 (  4.545%)
     Logged:            1 (  4.545%)
     Passed:            0 (  0.000%)
```

Si nous cherchons à détecter les injections XSS, cela pourrait s'avérer beaucoup plus ardu. Contentons-nous donc simplement des injections de script. Il suffit pour cela de détecter l'envoi du pattern suivant dans une requête : **%3Cscript%3E(.*)%3C%2Fscript%3E** (on note que les caractère spéciaux sont encodés au format URL).

On peut essayer avec les règles disponibles sur cette page : https://github.com/OpenIxia/sample-cloud-ids/blob/master/sensor/snort/local.rules

Il est cependant plus facile d'écrire la règle suivante, qui n'a pas généré de faux positif lors de mes tests : **alert tcp any any -> any 80 (msg:"XSS detected :'("; flow:established,to_server; content: "%3Cscript"; classtype: Web-application-attack; sid:640; rev:1;)**

On se contente ici d'essayer de détecter le pattern **%3Cscript**, ce qui fonctionne également parfaitement.

Sur une navigation normale :

```
Action Stats:
     Alerts:            0 (  0.000%)
     Logged:            0 (  0.000%)
     Passed:            0 (  0.000%)
```

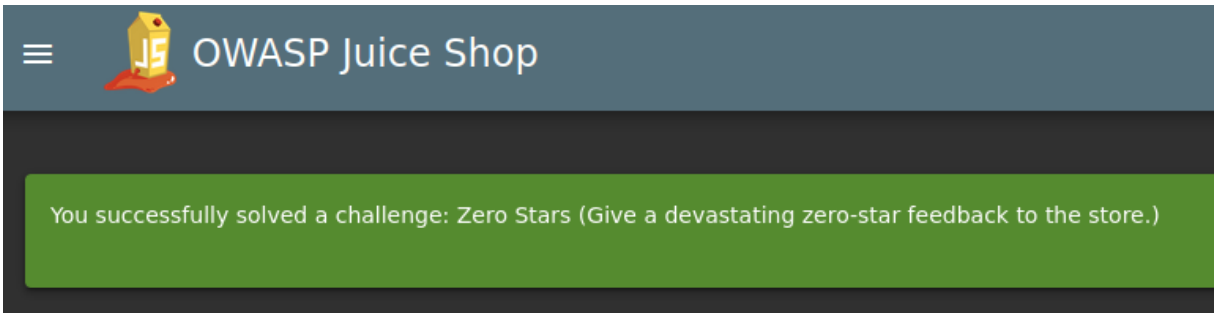
Sur l'injection testée plus haut :

```
Action Stats:
     Alerts:            1 (  3.571%)
     Logged:            1 (  3.571%)
     Passed:            0 (  0.000%)
```

5) Essayons de trouver l'adresse email de l'administrateur du site.

La page de login se trouve sur http://localhost:3000/#/login. Dans un premier temps, essayons d'obtenir cette adresse email avec une simple méthode d'OSINT (en cherchant par exemple sur la page de contacts), puis dans le code source.

Cela m'a permis de réussir un autre challenge par hasard, en modifiant le bouton d'envoi du feedback :



Cela dit, je n'ai toujours pas trouvé l'email souhaité.

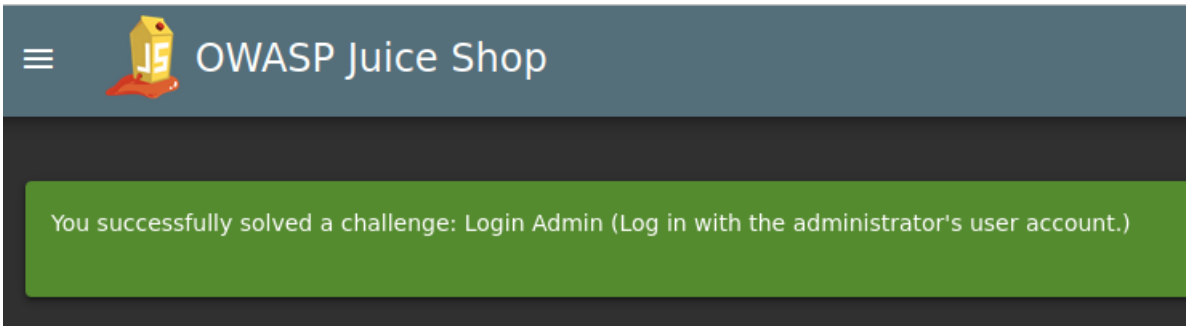
Après un certain temps de recherche, je me décide à tenter l'injection SQL. Je cherche d'abord une URL où un paramètre est injectable (http://site.come/dir/page?par=inject_here), sans succès. J'essaie donc sur le formulaire de connexion en entrant le login suivant : **test' OR id = 1;#** , puis un mot de passe aléatoire.

Si le formulaire est injectable, la requête effectivement exécutée devrait être quelque chose comme :

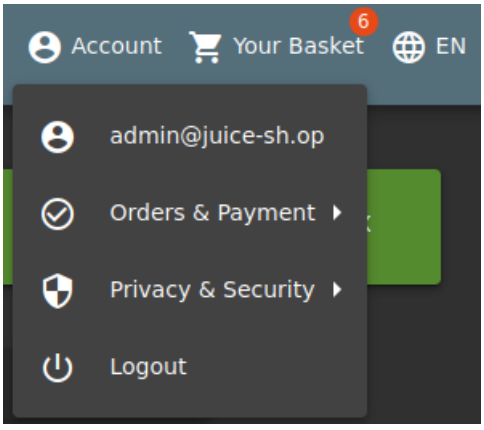
SELECT * FROM users WHERE login = 'test' OR id = 1;#' AND password = pass;

Ce qui en SQL signifie "sélectionner le premier utilisateur avec l'identifiant 1 », donc probablement l'administrateur.

Cela fonctionne :



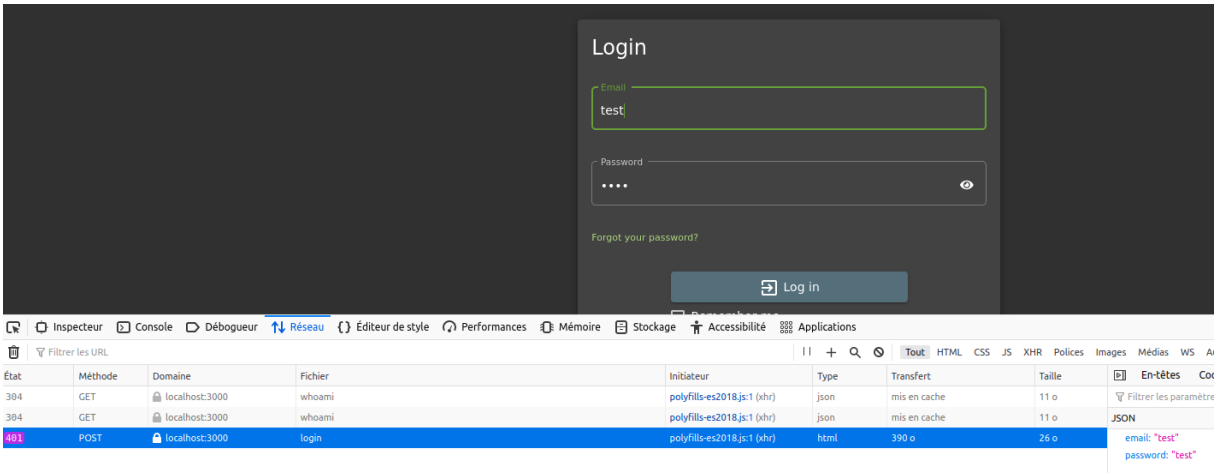
Je me suis bien emparé du compte administrateur, et je connais donc son adresse email :



J’ai techniquement pris le contrôle du compte administrateur, je pourrais donc m’arrêter ici. Ce n’est cependant pas ce qui est demandé dans le TD.

Je télécharge la liste des 1050 identifiants les plus communs sur https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/best1050.txt (nb : je suggère d’utiliser le dictionnaire **howsecureismypassword** ou **rockyou** pour une liste plus exhaustive).

Commençons par regarder le format de la requête d’authentification :

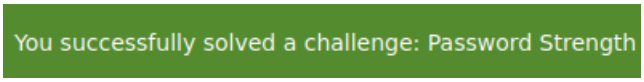


On observe que les paramètres d’authentification sont transmis sous la forme **email=email&password=password** à l’adresse **http://localhost:3000/rest/user/login**. De plus, en cas d’echec, on reçoit un code de retour **HTTP 401**.

Ecrivons rapidement un petit script python pour tester tous les mots de passe de notre dictionnaire :

```
import requests

url = 'http://localhost:3000/rest/user/login'
best_1050_file = open("best1050.txt")

for password in best_1050_file:
        password = password.strip() # remove new line character
        credentials = {'email': 'admin@juice-sh.op', 'password': password}

        http_response = requests.post(url, data = credentials)
        if http_response.status_code != 401:
                print("password found:", password)
                break

best_1050_file.close()
```

```
thomas@thomas-VirtualBox:~/Documents$ python3 hack_pass.py
password found: admin123
thomas@thomas-VirtualBox:~/Documents$
```



Essayons maintenant de détecter ce type d'attaque avec snort. La difficulté est de distinguer une attaque de bruteforce d'un utilisateur de bonne foi qui se trompe de mot de passe. Il faudrait donc déclencher l'alerte qu'en cas d'un certain nombre d'erreurs dans une courte fenêtre de temps.

Pour cela, snort nous propose le paramètre de **threshold** : https://www.snort.org/faq/readme-thresholding

Nous obtenons donc la règle suivante :

alert tcp any 3000 -> any any (msg:"Bruteforce attack detected ! :'("; content: "HTTP/1.1 401 Unauthorized"; threshold: type threshold, track by_src, count 10, seconds 60; sid:640; rev:1;)

Une alerte sera déclenchée si le serveur renvoie un code 401 plus de 10 fois au même client dans une fenêtre de 60 secondes.

Un utilisateur de bonne foi se trompe de mot de passe 3 fois, aucune alerte :

```
Action Stats:
     Alerts:               0 (   0.000%)
     Logged:               0 (   0.000%)
     Passed:               0 (   0.000%)
```

Je lance le script de bruteforce, alerte :

```
Action Stats:
     Alerts:              11 (   0.927%)
     Logged:              11 (   0.927%)
     Passed:               0 (   0.000%)
```

6) Après avoir installé hping3 à l'aide de la commande apt, je lance la commande suivante :

sudo hping3 -S --flood -V -p 3000 172.17.0.2

On remarque que les droits root sont nécessaires sous Linux pour envoyer beaucoup de paquets réseau en même temps. Au passage j'inscris directement l'adresse IP de l'image docker et non pas l'hôte local, même si cela fonctionnerait puisque nous avons précédemment configuré docker pour faire du port forwarding. Cela afin d'éviter une perte de performances et du calcul CPU inutiles.

Je lance donc la commande et **l'interrompt après environ 10 secondes** :

```
thomas@thomas-VirtualBox:~/Documents$ sudo hping3 -S --flood -V -p 3000 172.17.0.2
using docker0, addr: 172.17.0.1, MTU: 1500
HPING 172.17.0.2 (docker0 172.17.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.17.0.2 hping statistic ---
1470120 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
thomas@thomas-VirtualBox:~/Documents$
```

Nous envoyons donc en moyenne environ **150 000 paquets / seconde**.

Sans surprise, snort relève la détection de nombreux paquets sur le port 3000 :

```
WARNING: No preprocessors configured for policy 0.
10/06-23:29:49.310461 02:42:52:A0:BD:28 -> 02:42:AC:11:00:02 type:0x800 len:0x36
172.17.0.1:22186 -> 172.17.0.2:3000 TCP TTL:64 TOS:0x0 ID:22430 IpLen:20 DgmLen:40
******S* Seq: 0x5114E31  Ack: 0x659450B9  Win: 0x200  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
10/06-23:29:49.310463 02:42:AC:11:00:02 -> 02:42:52:A0:BD:28 type:0x800 len:0x3A
172.17.0.2:3000 -> 172.17.0.1:22186 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x9085B4FC  Ack: 0x5114E32  Win: 0xFAF0  TcpLen: 24
TCP Options (1) => MSS: 1460

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Notons au passage que le site reste toujours accessible via le navigateur, l'attaque DOS a donc une portée assez limitée.

Maintenant, notre objectif ne va plus seulement consister en la détection d'une attaque, mais aussi dans le blocage de celle-ci. Il faut donc configurer snort pour bloquer des paquets : http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node29.html.

Commençons par écrire une règle en nous inspirant du **threshold** de la question précédente pour détecter une attaque DOS et lever une alerte :

alert tcp any 3000 -> any any (msg:"DOS attack detected ! :'("; threshold: type threshold, track by_src, count 1000, seconds 10; sid:640; rev:1;)

Cette fois-ci, on cherche à détecter 1000 requêtes sans pattern particulier dans une fenêtre de 10 secondes, provenant de la même adresse IP. Le nombre de requêtes autorisées est en effet un peu plus élevé pour ne pas déclencher de faux positif si l'utilisateur télécharge un gros fichier par exemple. Testons notre règle :

Navigation « honnête » :

```
Action Stats:
     Alerts:            0 (  0.000%)
     Logged:            0 (  0.000%)
     Passed:            0 (  0.000%)
```

Attaque DOS :

```
Action Stats:
     Alerts:           36 (  0.033%)
     Logged:           36 (  0.033%)
     Passed:            0 (  0.000%)
```

Essayons de remplacer la règle d'**alert** par un **drop** :

drop tcp any 3000 -> any any (msg:"DOS attack detected ! :("; threshold: type threshold, track by_src, count 1000, seconds 10; sid:640; rev:1;)

snort indique qu'il a réussi à éliminer environ **50% des paquets de l'attaque DOS**

```
Packet I/O Totals:
   Received:       2584157
   Analyzed:        201853 (  7.811%)
    Dropped:       2382302 ( 47.968%)
   Filtered:             0 (  0.000%)
Outstanding:       2382304 ( 92.189%)
   Injected:             0
```

Sur une période de temps de 10s, hping n'a transmis que 860 000 paquets, soit 86 000 paquets / seconde.

```
thomas@thomas-VirtualBox:~/Documents$ sudo hping3 -S --flood -V -p 3000 172.17.0.2
using docker0, addr: 172.17.0.1, MTU: 1500
HPING 172.17.0.2 (docker0 172.17.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.17.0.2 hping statistic ---
861397 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Nous avons réussi à réduire l'attaque de moitié, mais nous pourrions essayer de faire mieux.

J'ai essayé d'améliorer mon taux de mitigation en suivant les règles proposées sur ou https://www.researchgate.net/publication/335803817_Mitigation_of_DoS_and_Port_Scan_Attacks_Using_Snort ou bien https://www.researchgate.net/publication/338660054_DETECTING_DDoS_ATTACK_USING_Snort, mais sans grand succès hélas.

Afin d'expérimenter l'attaque **DDOS** (Distributed DOS, attaque DOS depuis plusieurs machines), j'ai lancé hping3 depuis plusieurs terminaux, puisque je n'ai pas pour le moment un botnet avec plusieurs milliers de machines à ma disposition. Le résultat sur la console snort est très similaire.

Une façon envisageable d'échapper à la détection DOS avec une attaque DDOS serait donc de limiter l'envoi de requête par machine individuelle. Nous aurions donc beaucoup de machines envoyant chacune des requêtes à une fréquence acceptable pour l'analyseur de paquets. L'ensemble des requêtes envoyées serait par contre suffisant pour perturber le fonctionnement de notre serveur.

Le gros inconvénient de cette méthode réside cependant dans le fait de disposer d'un nombre conséquent de machines à disposition.