# Taller | Armando un laboratorio para detectar amenazas en la red


Dictado por: Sheyla Leacock

# Objetivos del taller:

El objetivo de este taller es armar un laboratorio de detección de amenazas con herramientas OpenSource y simular ataques a nuestro entorno de red que nos permitan conocer cómo podemos mejorar nuestras defensas.

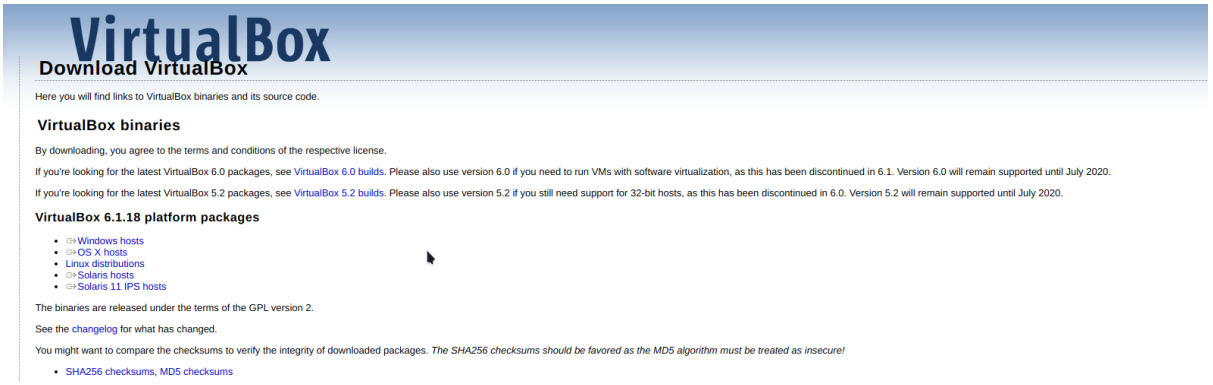
Se propone desplegar de forma automatizada el sistema de detección de intrusos Snort en un entorno Linux, así como realizar las configuraciones e implementar reglas que nos permitan detectar amenazas a nivel de red.

Se implementará también una arquitectura cliente servidor con Pytbull para generar tráfico malicioso en nuestra red a fin de que pueda ser detectado por el IDS.

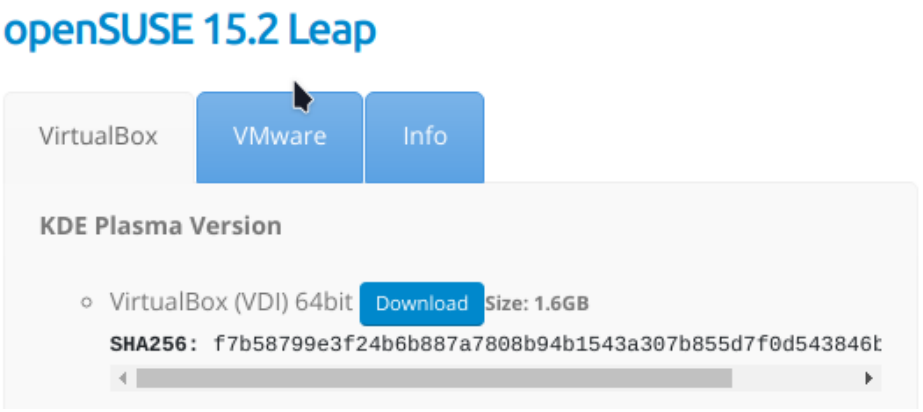
Además desplegaremos herramientas que nos permitan monitorear las alertas generadas por el IDS y realizar correlación de eventos

# Prerrequisitos:

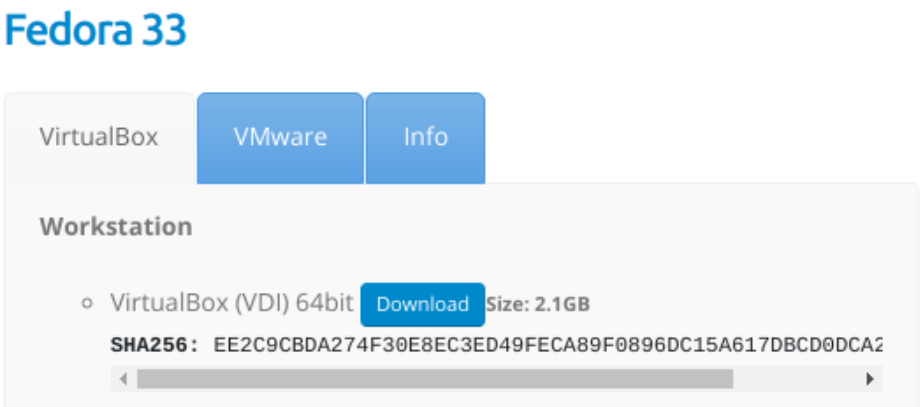
Descargar e instalar VirtualBox: https://www.virtualbox.org/wiki/Downloads




Descargar una máquina virtual con OpenSUSE 15.2 desde OSboxes: https://www.osboxes.org/opensuse/




Descargar una máquina virtual con Fedora 33 desde OSboxes: https://www.osboxes.org/fedora/

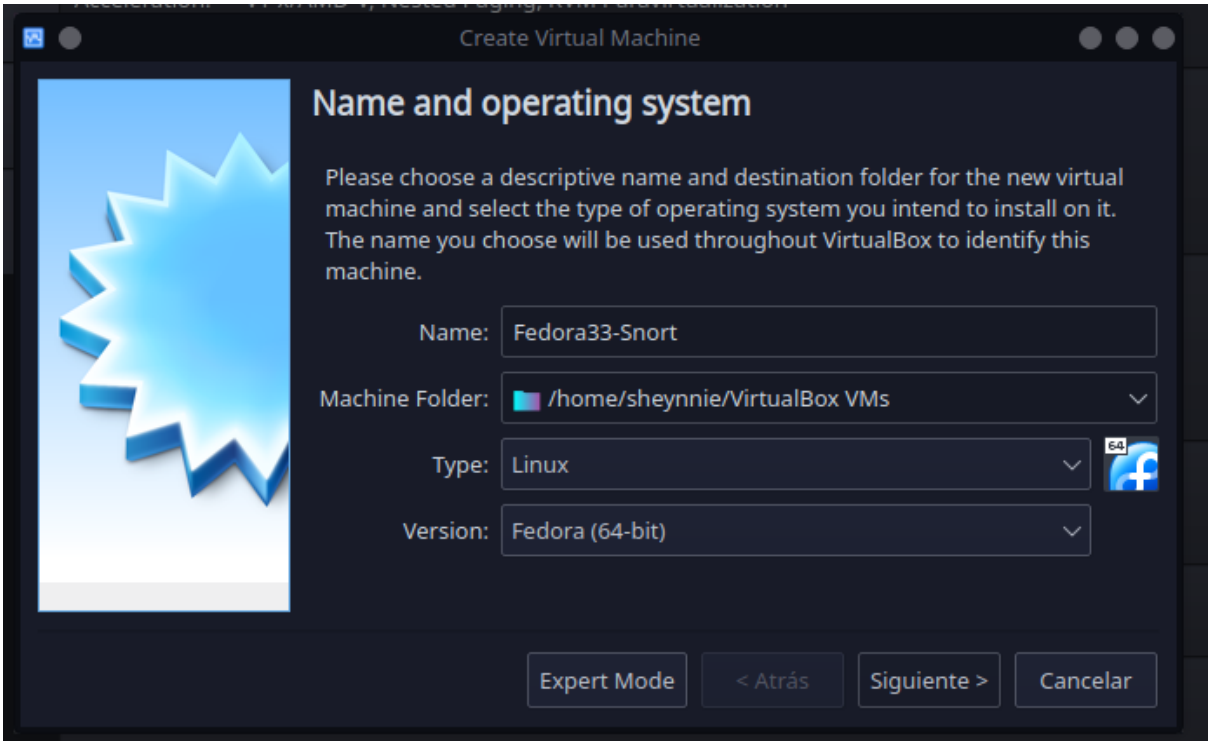


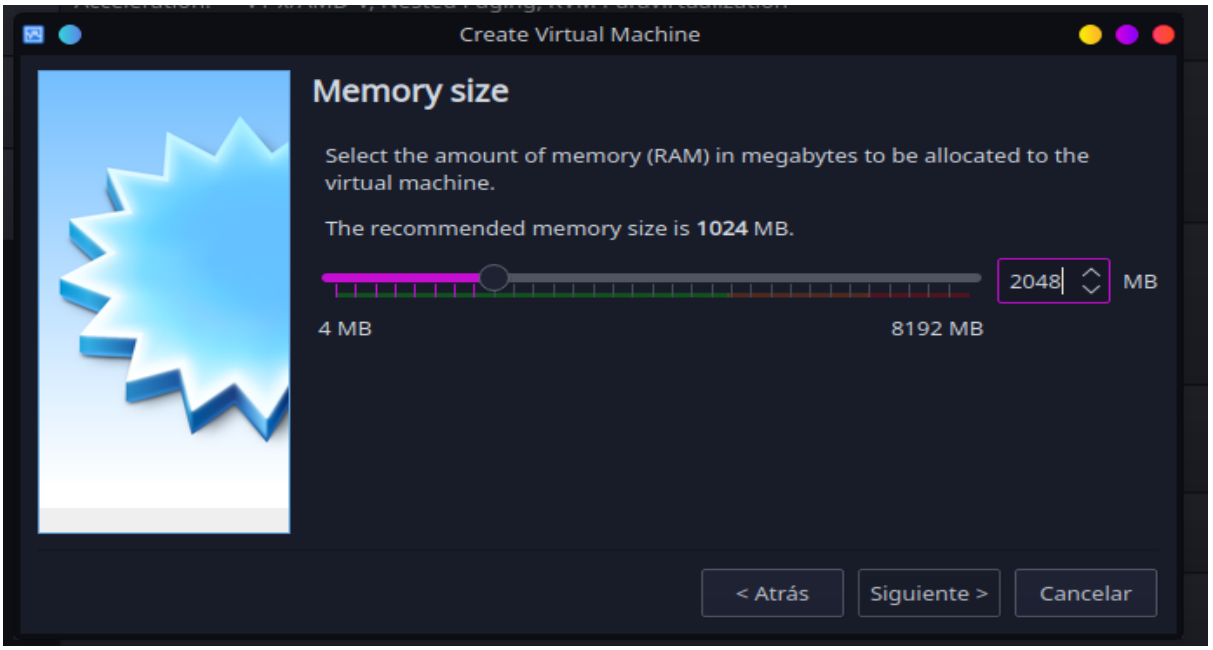
Nota: La ventaja de las máquinas de OSBoxes es que ya se encuentran “Listas para usar”. Utilizan el siguiente usuario y contraseña por defecto: Usuario: **osboxes** Contraseña: **osboxes.org**

Una vez descargadas las máquinas virtuales de OSboxes, descomprimir cada una en un subdirectorio:

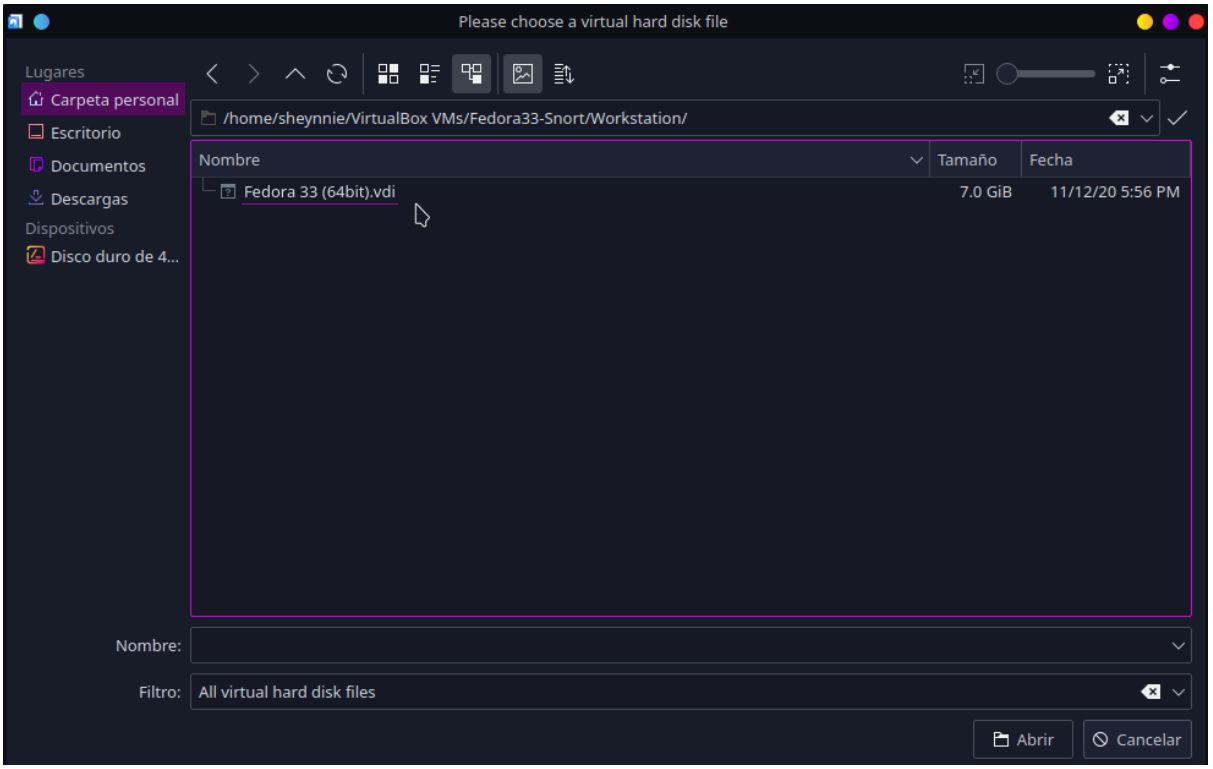
```
localhost:/home/sheynnie/Downloads # p7zip -d Fedora-33-Workstation-VB_64bit.7z
```

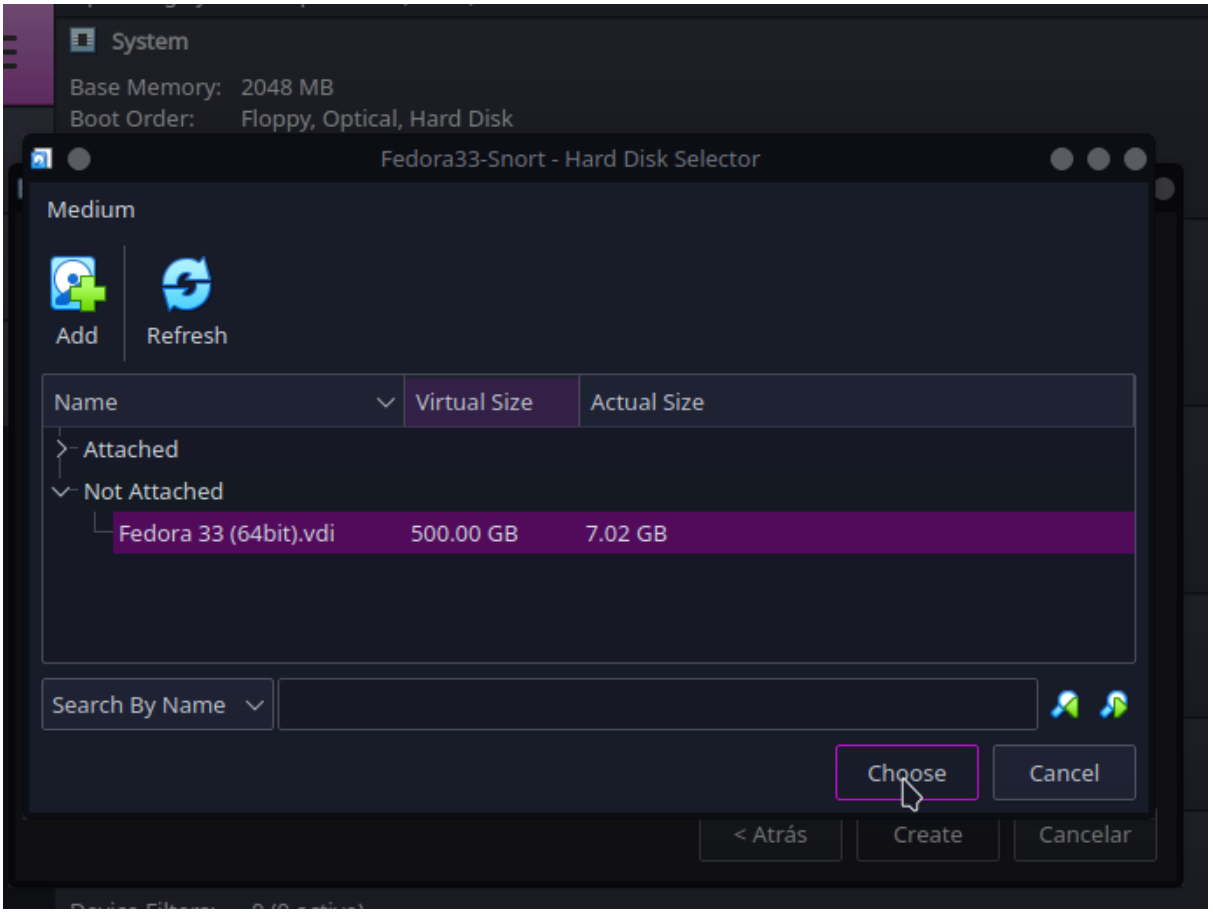
Abrir virtualbox y crear 2 maquinas virtuales con las siguientes características: Mínimo 2048 mb de memoria



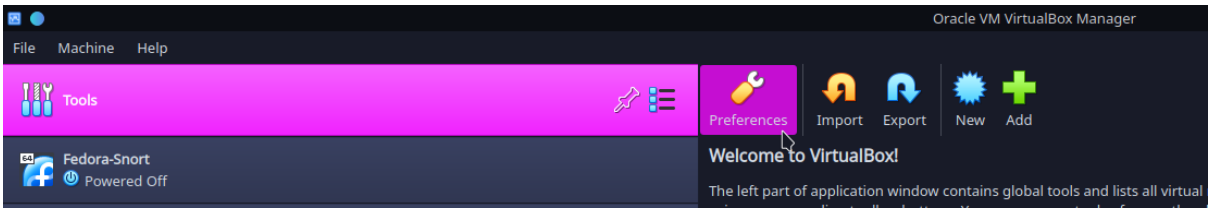
Asignar el archivo .vdi previamente extraído como disco duro existente de la máquina:



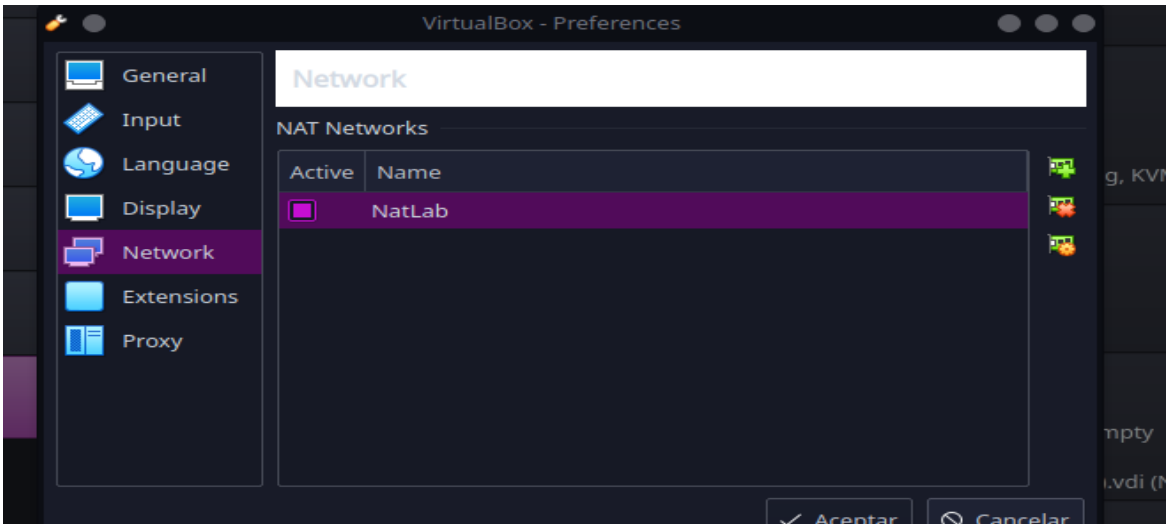




Dar click al botón crear para finalizar.

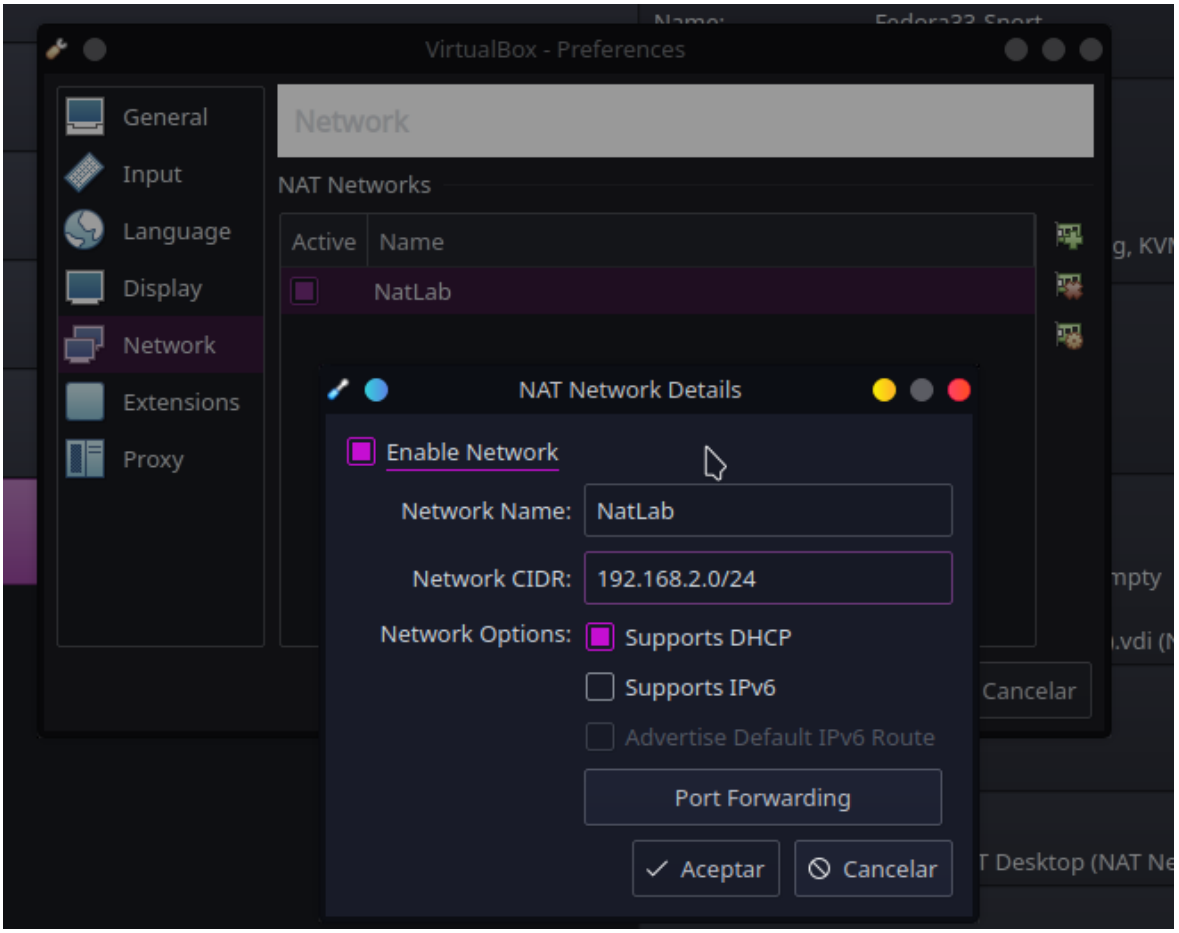
Una vez creada, desde virtualbox seleccione la opción herramientas -> Preferencias



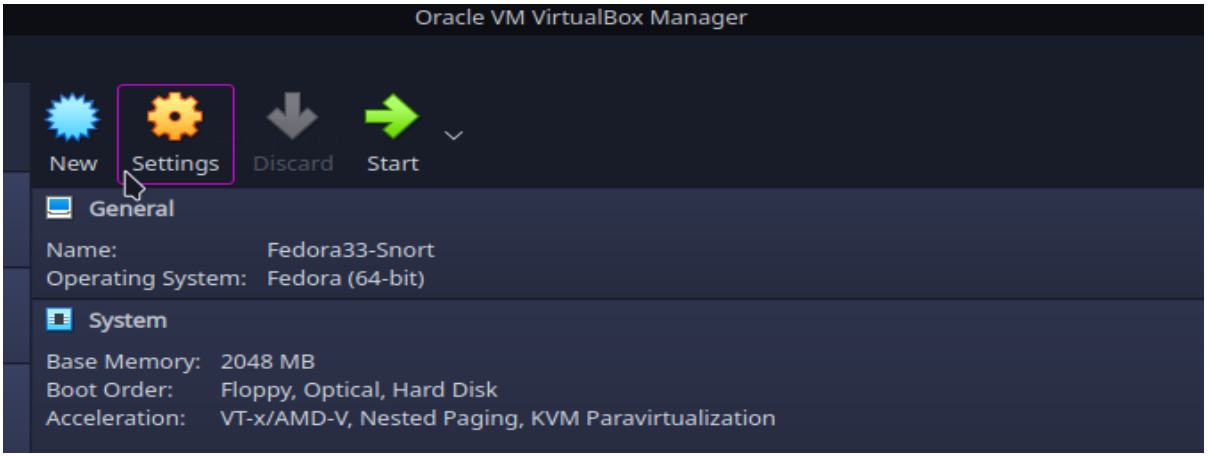

Seleccionar la opción “Red” y dar click al botón con símbolo “+” para añadir una nueva red.

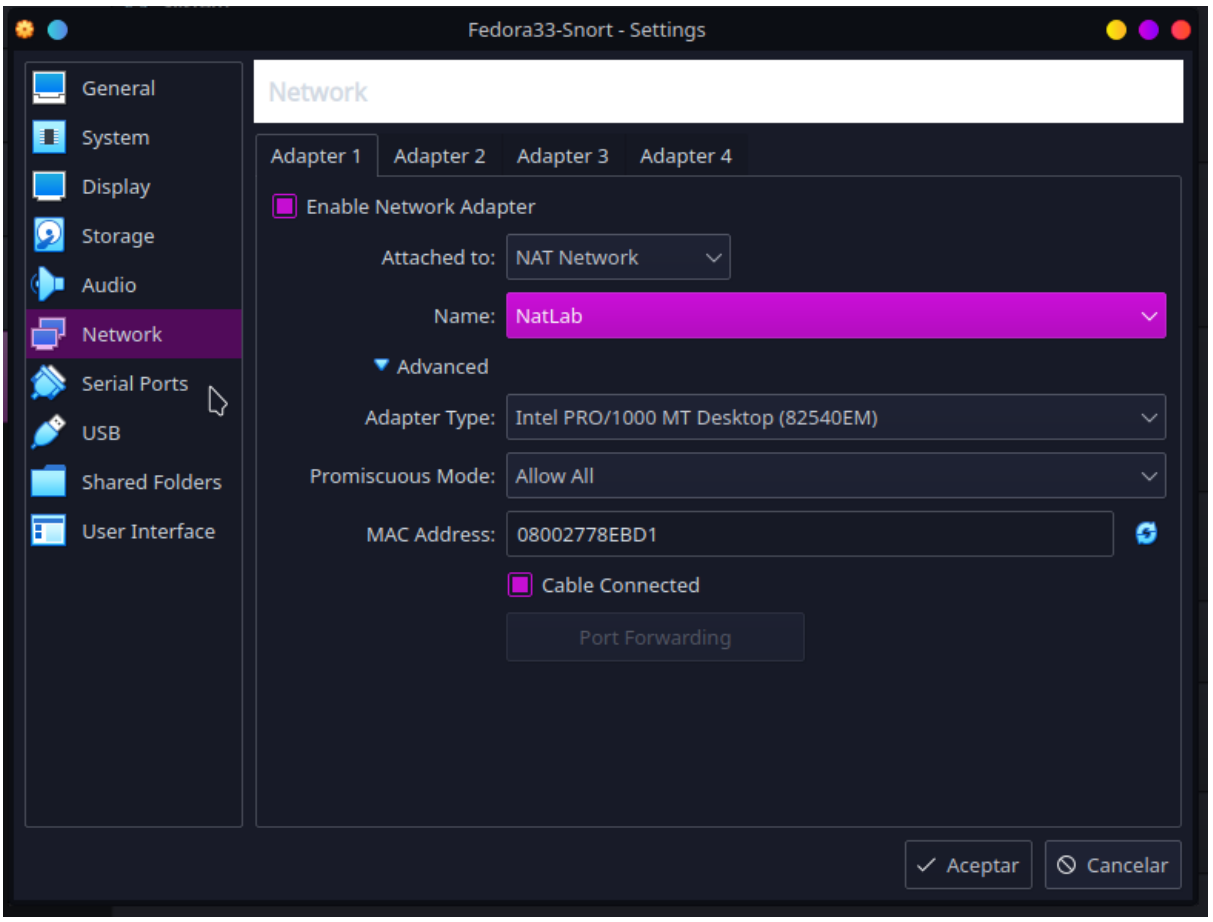


Nota: Si queremos ver los detalles como rango de IP con el cual se creó la red NAT podemos visualizarlo dandole click al botón con el símbolo de configuración (A pesar de que no es necesario editar ningún parámetro en este apartado se recomienda desmarcar el soporte para DHCP de forma que para pruebas posteriores no nos cambie la IP de las máquinas).




Una vez creada la red NAT, nos dirigimos a cada una de las máquinas creadas para enlazarlas a esta red. Seleccione la máquina y de click al botón configuración->red y en la sección del primer adaptador, asigne la “red NAT” creada previamente:






Con estas configuraciones ya podemos iniciar las máquinas.

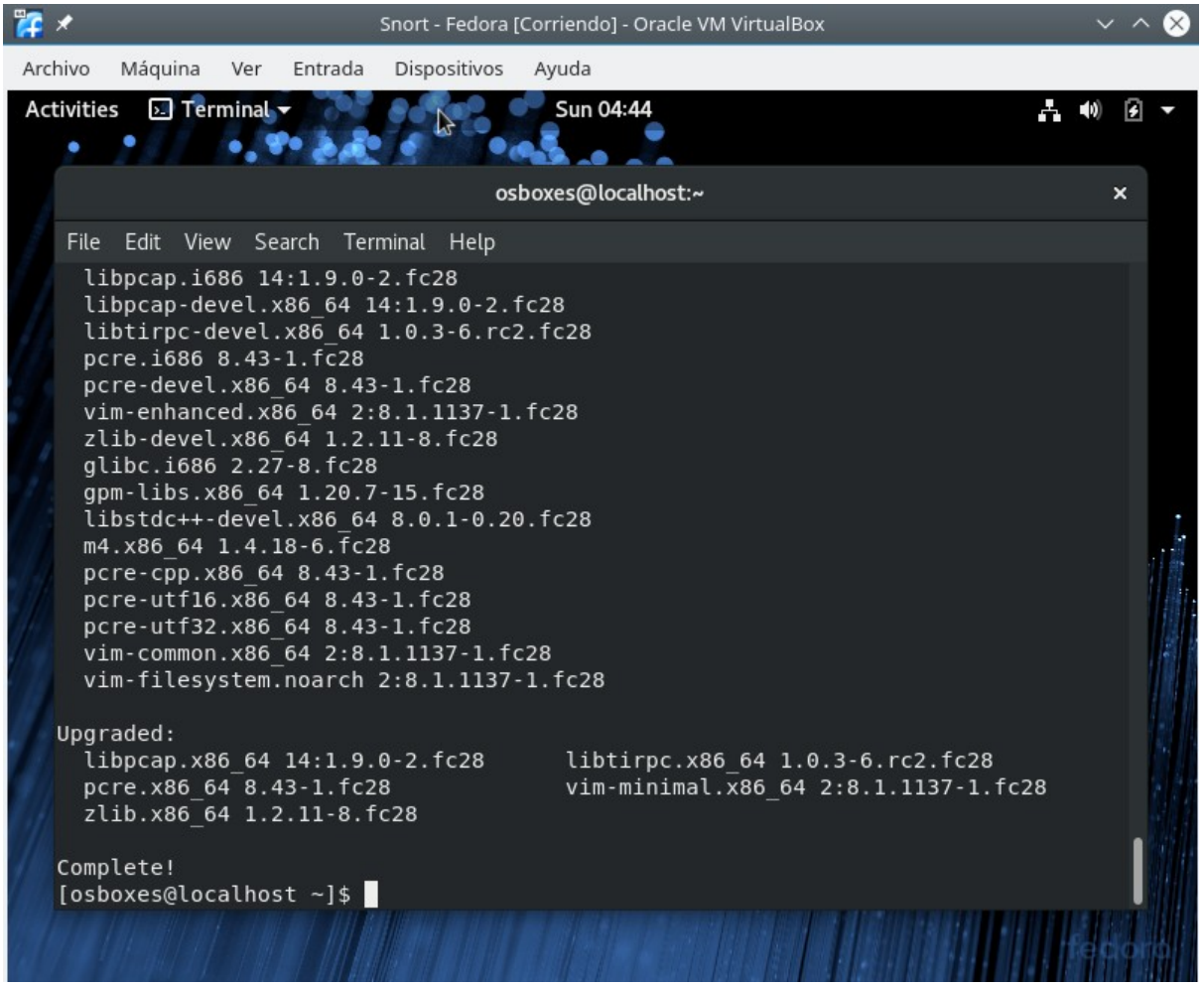
Notas:

- Ambas máquinas deben quedar bajo la misma red NAT para que puedan visualizarse entre sí.
- Para la máquina FEDORA se debe habilitar en esta misma sección el modo promiscuo en: “permitir todo” para que pueda visualizar todo el tráfico de la red ya que en esta se instalará el IDS Snort.
- Se recomienda actualizar las máquinas:
    - Para actualizar Fedora, desde la terminal ingrese el comando: “***sudo yum update***”.
    - Para OpenSUSE, desde la terminal ingrese el comando “***sudo zypper update***”.

# Parte 1 - Configuración del IDS Snort en la máquina Fedora.

Instalar las librerías necesarias desde la terminal:
Aplicaciones → buscador→terminal
Ingresamos lo siguiente para instalar las librerías:

***sudo yum install pcre.* gcc git vim flex bison libpcap.* libdnet.* zlib.* libtirpc.* luajit autoconf openssl libtool perl lwp.* perl-libwww-perl perl-GD perl-Crypt-SSLeay perl-LWP-Protocol-https tcpdump zlib-devel libpcap-devel pcre-devel libdnet-devel libtirpc-devel***

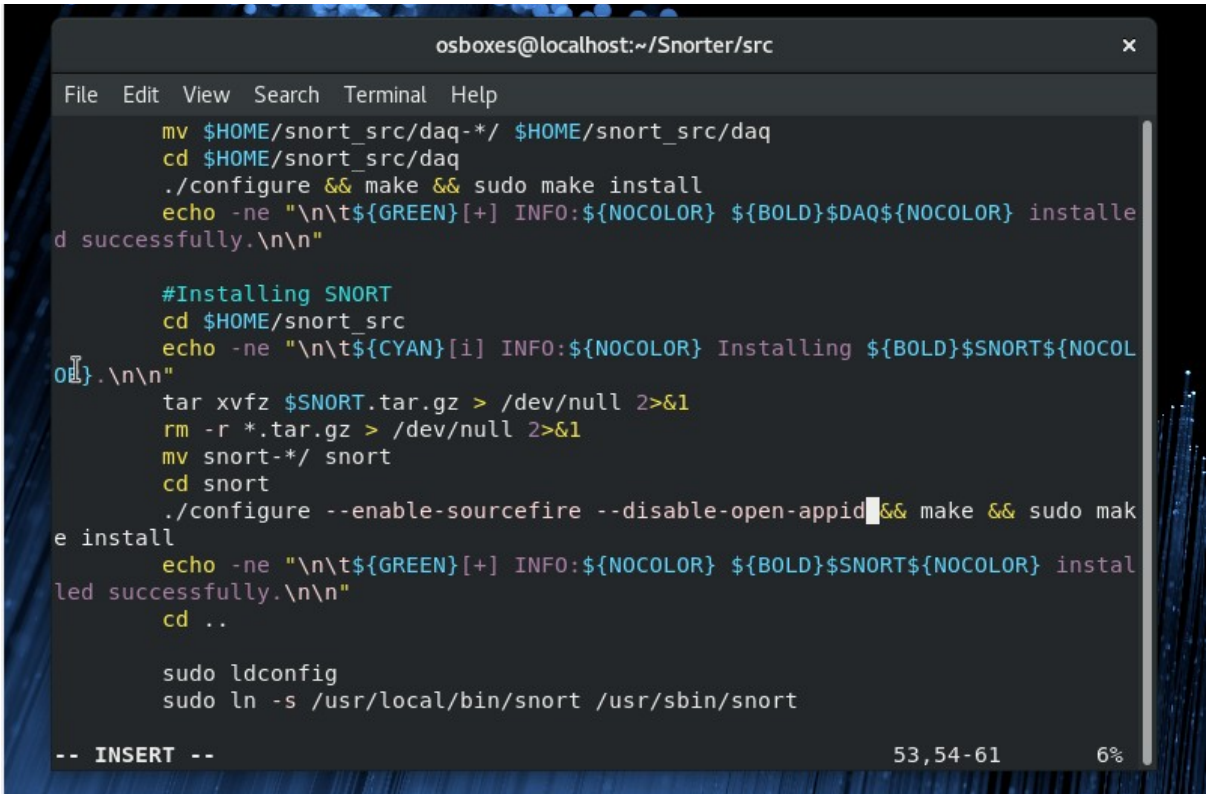



Nota: Los entornos linux son Case sensitive por lo que se deben respetar las mayúsculas y minúsculas.

Ahora, realizaremos el clon del repositorio github de Snorter en nuestro entorno local con el comando: ***git clone https://github.com/joanbono/Snorter.git***

```
[osboxes@localhost ~]$ git clone https://github.com/joanbono/Snorter.git
Cloning into 'Snorter'...
remote: Enumerating objects: 526, done.
remote: Total 526 (delta 0), reused 0 (delta 0), pack-reused 526
Receiving objects: 100% (526/526), 6.52 MiB | 6.63 MiB/s, done.
Resolving deltas: 100% (214/214), done.
[osboxes@localhost ~]$ cd Snorter
[osboxes@localhost Snorter]$ ls
_config.yml  doc  img  LICENSE  package-lock.json  README.md  src
[osboxes@localhost Snorter]$ cd src
[osboxes@localhost src]$ ls
SnorterDock  Snorter.sh  Snorter_Ubuntu-14.04.sh
[osboxes@localhost src]$
```

Posicionarse en la carpeta Snorter/src y editar la línea 53 del archivo snorter.sh con el comando ***sudo vim Snorter.sh***. Una vez en el editor agregaremos el parámetro ***–disable-open-appid*** como se muestra a continuación:

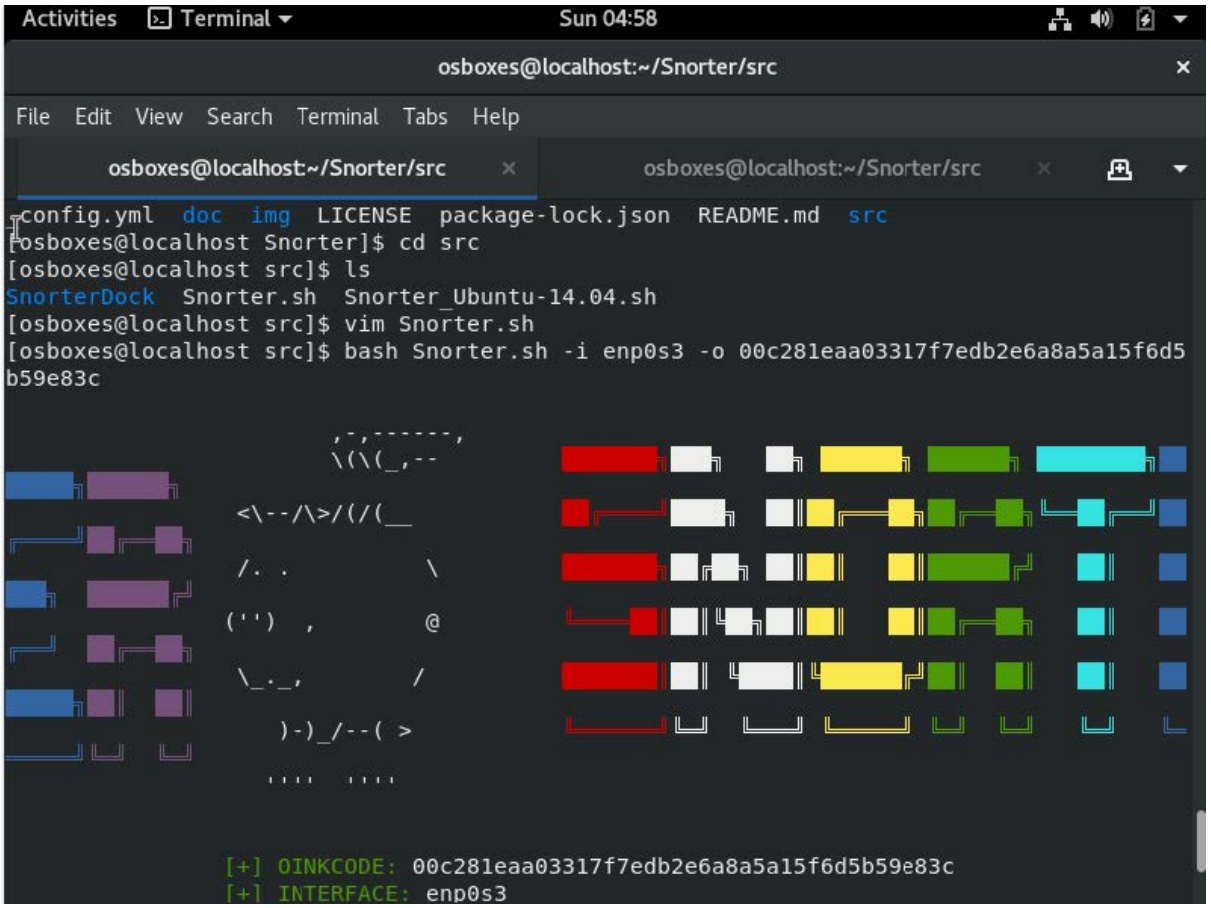



Para salir del editor presionamos la tecla escape ***ESC*** seguido de **:wq!**
Luego, ejecutamos los siguientes comandos para asociar los comandos de instalación de Fedora en el script de snorter:
***sed -i 's/apt-get/yum/g' Snorter.sh***
***sed -i 's/--force-yes//g' Snorter.sh***

```
[osboxes@localhost ~]$ cd Snorter/src
[osboxes@localhost src]$ ls
SnorterDock  Snorter.sh  Snorter_Ubuntu-14.04.sh
[osboxes@localhost src]$ sed -i 's/apt-get/yum/g' Snorter.sh
[osboxes@localhost src]$
```

Una vez hecho esto ejecutamos snorter:

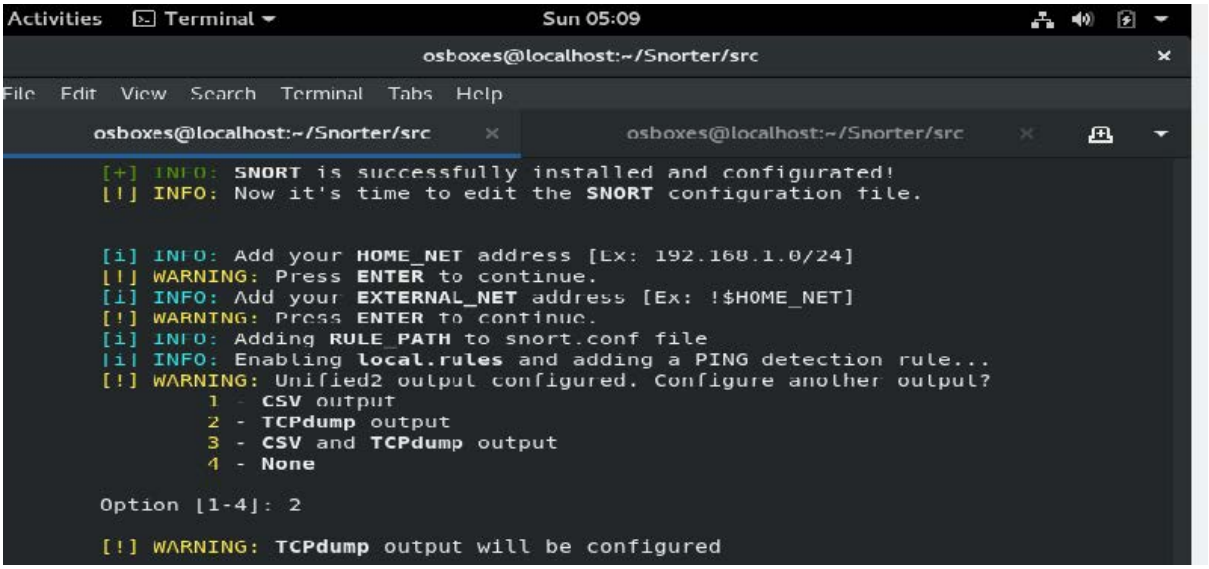
***sudo bash Snorter.sh -o <'oinkcode'> -i 'interfaz de red'***



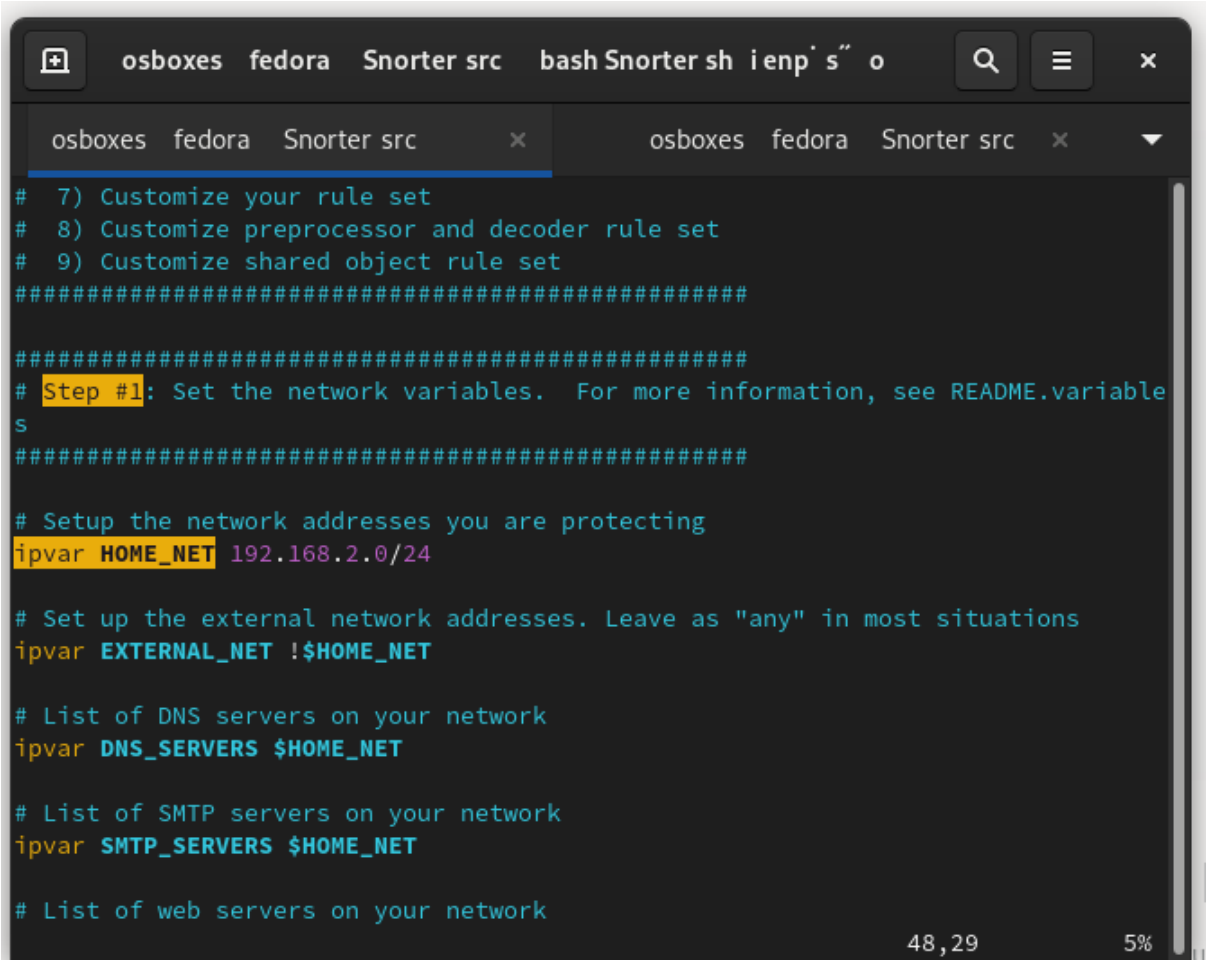

Nota:

- El comando -o indica que ingresaremos el oinkcode asociado, en caso de no tenerlo, ejecutemos Snorter sin este comando.
- Es recomendable ejecutar snorter con un oinkcode generado al registrarnos en Snort: https://snort.org/users/sign_up. El oinkcode es una llave asociada a nuestro registro que funciona como api para descargar las reglas de snort. Para el taller no es necesario el oinkcode, se utilizarán las reglas comunitarias.
- Con el comando -i indicamos cuál será la interfaz de red sobre la cual queremos que Snort realice el monitoreo (Por ejemplo: eth1, enp0s3). Puedes ver tus interfaces de red desde la terminal ejecutando el comando ***ip address***.

Una vez se realice la revisión y descarga de librerías se presentará la interfaz para configuración:




Primero, para la configuración de snort se deben modificar los valores de la red interna (la que indicamos al crear la red NAT de virtualbox) y la red externa (Todo aquello que no sea la red interna).




Recuerda salir del editor con la tecla escape ***ESC*** seguido de **:wq!**
Para la segunda pregunta de la configuración elegimos la opción ***2*** (cualquiera de las 3 primeras es válida).

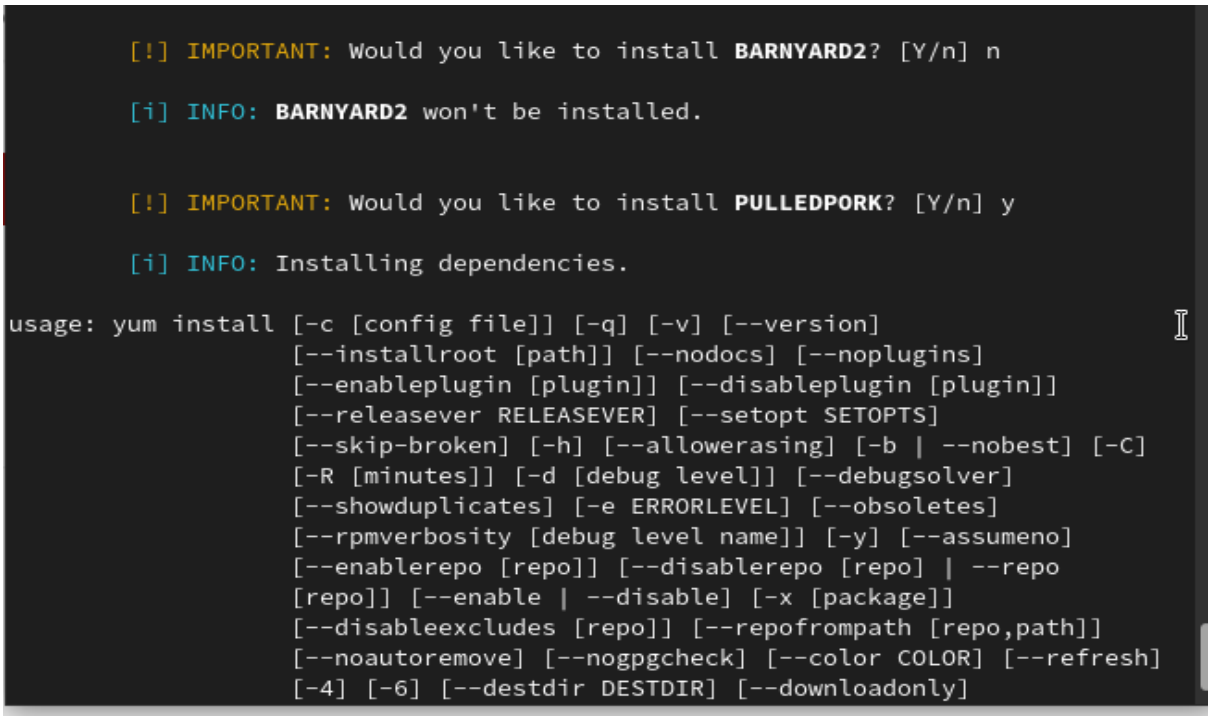
```
[i] INFO: Enabling local.rules and adding a PING detection rule...
[!] WARNING: Unified2 output configured. Configure another output?
        1 - CSV output
        2 - TCPdump output
        3 - CSV and TCPdump output
        4 - None

Option [1-4]: 2

[!] WARNING: TCPdump output will be configured
```

Después de esto, se ejecutará Snort en modo de prueba para validar la configuración.
Anteriormente Snorter escribió una regla en el archivo de reglas locales que alertará cualquier tráfico ICMP generado, la que usará en el proceso de prueba de snort.

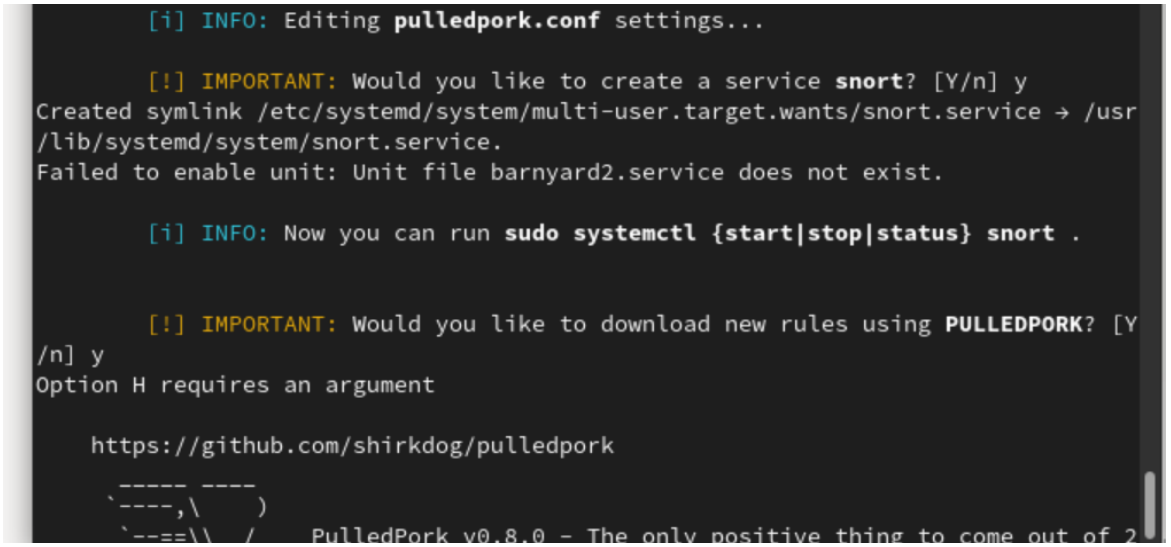
```
osboxes@fedora:~/Snorter/src — sudo bash Snorter.sh -i enp0...
osboxes@fedora:~/Snorter/src — ...   osboxes@fedora:~/Snorter/src
Snort exiting

        [!] WARNING: Attempting to test ICMP rule in enp0s3. Send a PING to your
 SNORT machine. Press Ctrl+C once and wait few seconds to stop the process...

        [!] WARNING: Press ENTER to continue.
01/24-01:01:03.987751  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.6 -> 192.168.2.8
01/24-01:01:03.988052  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.8 -> 192.168.2.6
01/24-01:01:04.991672  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.6 -> 192.168.2.8
01/24-01:01:04.991710  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.8 -> 192.168.2.6
01/24-01:01:06.088852  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.6 -> 192.168.2.8
01/24-01:01:06.088905  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.8 -> 192.168.2.6
^C*** Caught Int-Signal
01/24-01:01:07.137595  [**] [1:10000001:1] PING ATTACK [**] [Priority: 0] {ICMP}
 192.168.2.6 -> 192.168.2.8
snort: no process found

        [!] IMPORTANT: Would you like to install BARNYARD2? [Y/n]
```

Nota: En este apartado, podemos probar snort lanzando un comando ping desde otra máquina que esté dentro de la red NAT hacia la máquina con snort.
Presionamos enter y veremos las alertas generadas en tiempo real. Luego podemos salir de este modo presionando ***Ctrl+C*** para continuar con las configuraciones.

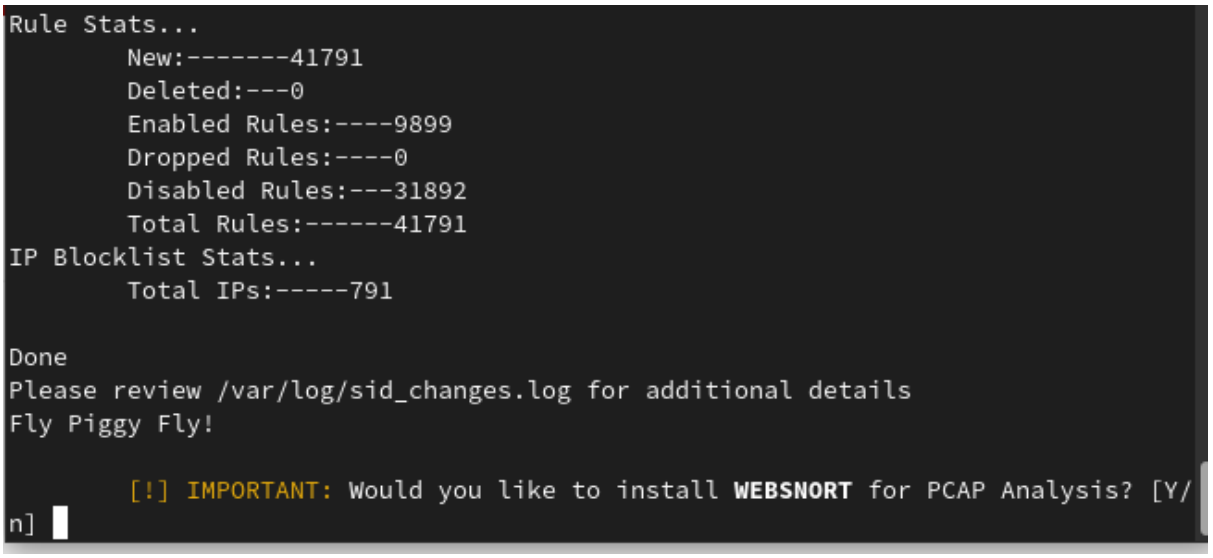
Ingresamos la opción ***n*** para no instalar Barnyard2 (fuera de alcance del taller ya que requiere otras configuraciones y dependencias).
Ingresamos la opción ***y*** para instalar Pulledpork.



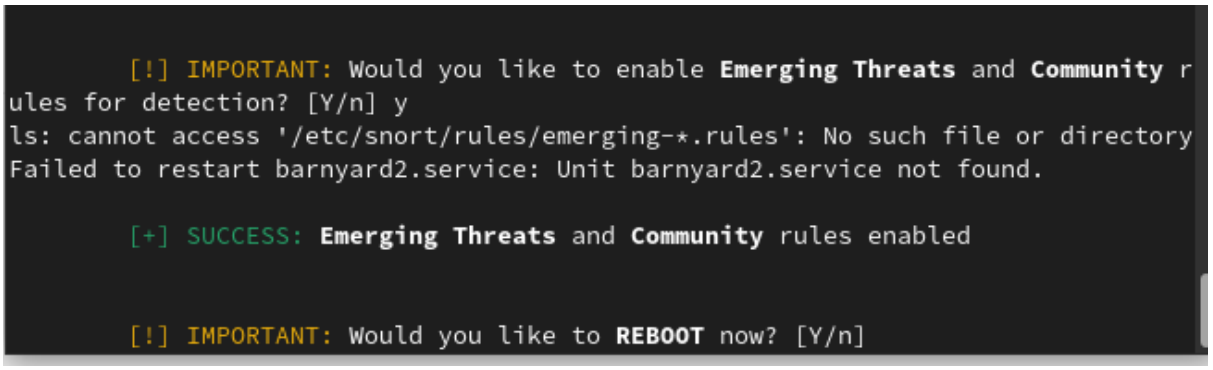

Seleccionamos la opción ***n*** para no habilitar las reglas de Emerging Threats y la opción ***y*** para la creación del servicio snort y también para la opción de descargar nuevas reglas utilizando Pulledpork.



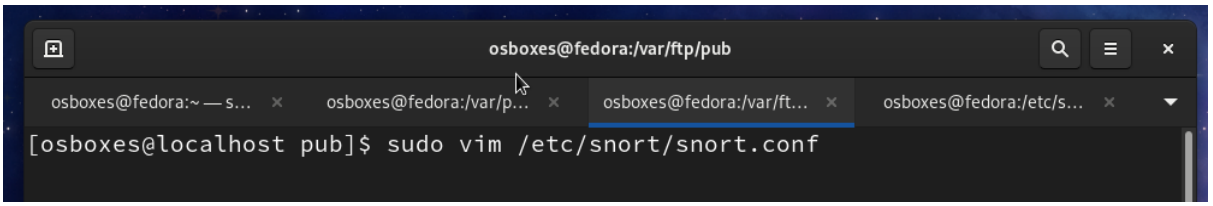
Instalamos websnort con la opción ***y***



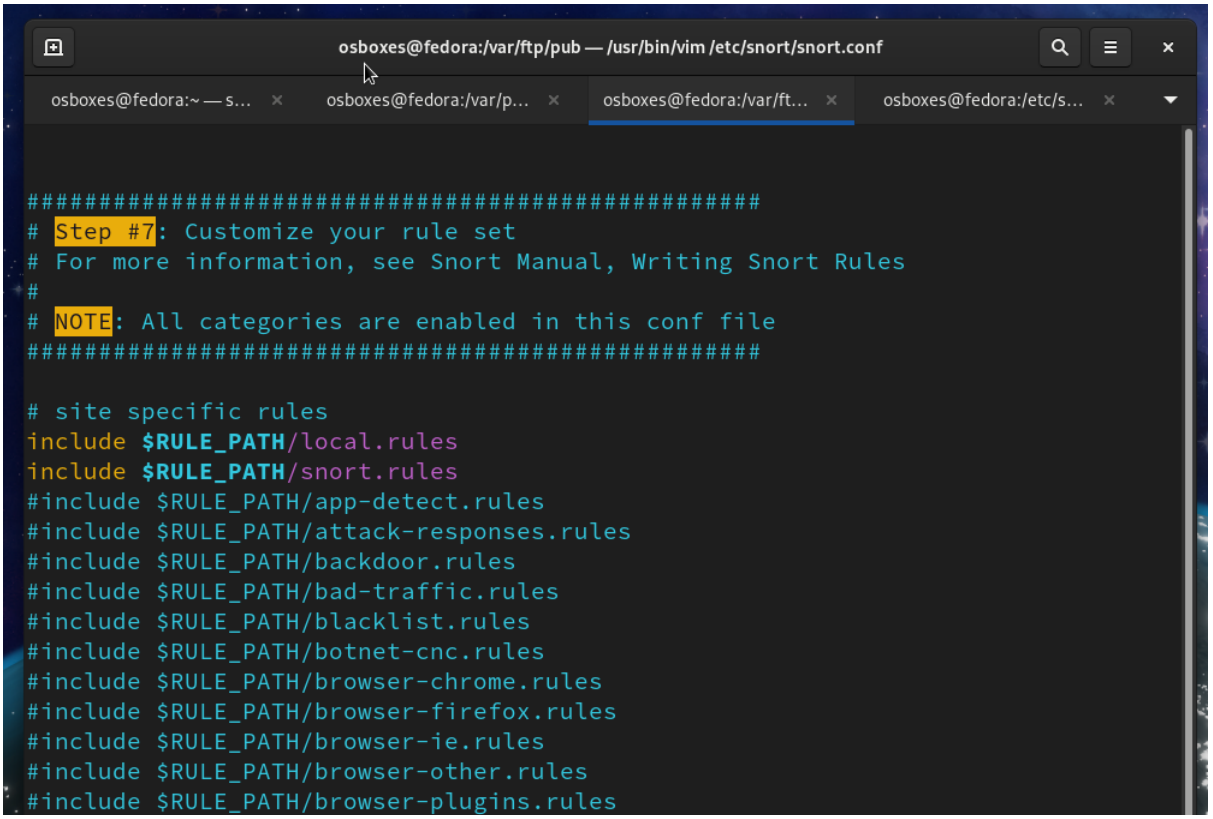

Habilitamos la descarga de las reglas comunitarias y luego reiniciamos la máquina virtual para aplicar todos los cambios con la opción ***y***.



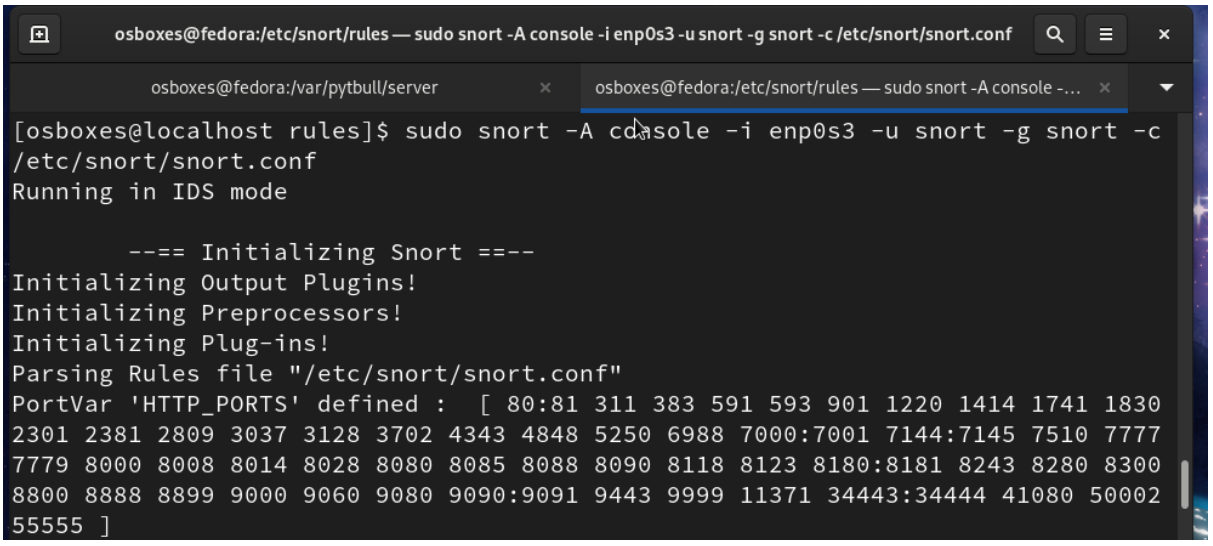

Una vez reiniciamos, nos dirigimos al archivo de configuración de snort para incluir la ruta al archivo de reglas de snort:
***sudo vim /etc/snort/snort.conf***




y debajo de la ruta de reglas locales (línea 547) agregamos la linea ***include $RULE_PATH/snort.rules*** y bajo la línea 690 agregamos la inclusión de las reglas comunitarias: ***include $RULE_PATH/community.rules***

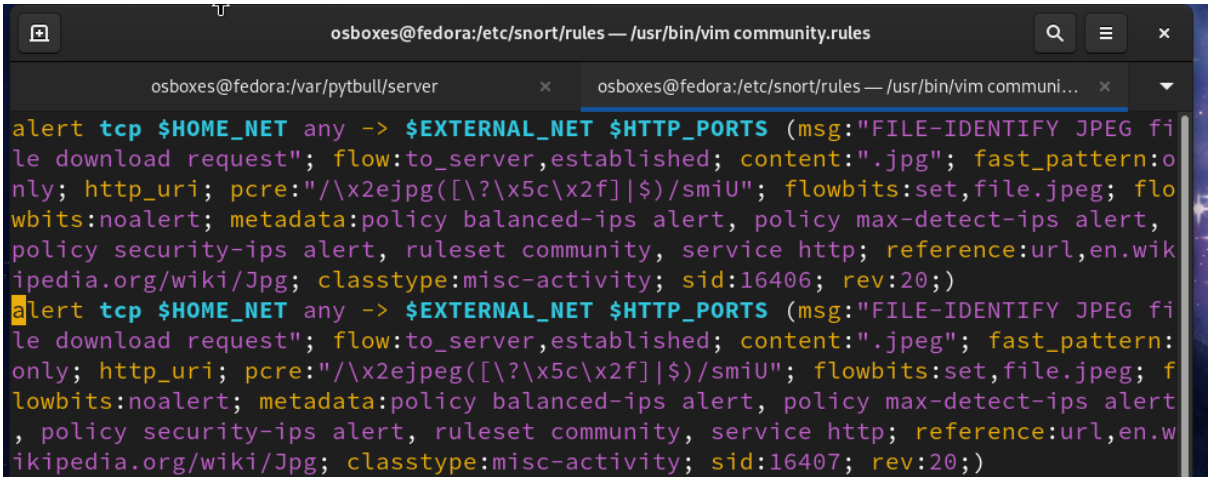



Ahora si, podemos ejecutar snort y dejarlo en modo escucha para ver las alertas en la consola con el comando: ***sudo snort -A console -i enp0s3 -u snort -g snort -c /etc/snort/snort.conf***



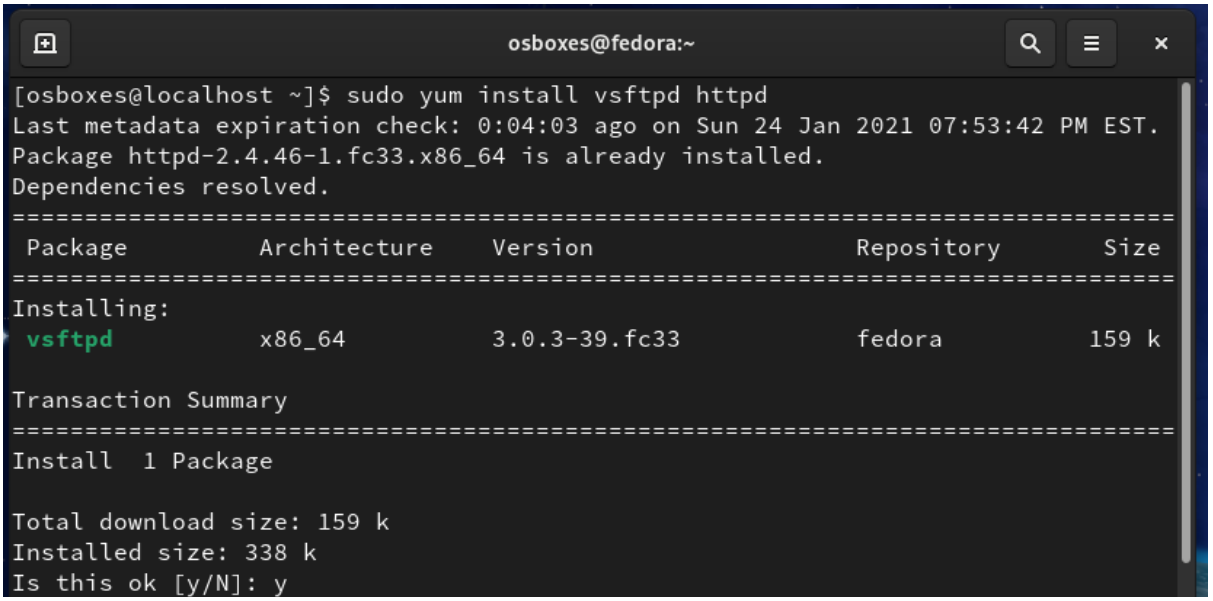
```
          Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
          Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
          Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
          Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
          Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
          Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
          Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
          Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
          Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
          Preprocessor Object: SF_POP  Version 1.0  <Build 1>
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
          Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
          Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
          Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
          Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
          Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
Commencing packet processing (pid=8535)
```

Nota: los archivos de reglas de snort los encuentras en la ruta: /etc/snort/rules.

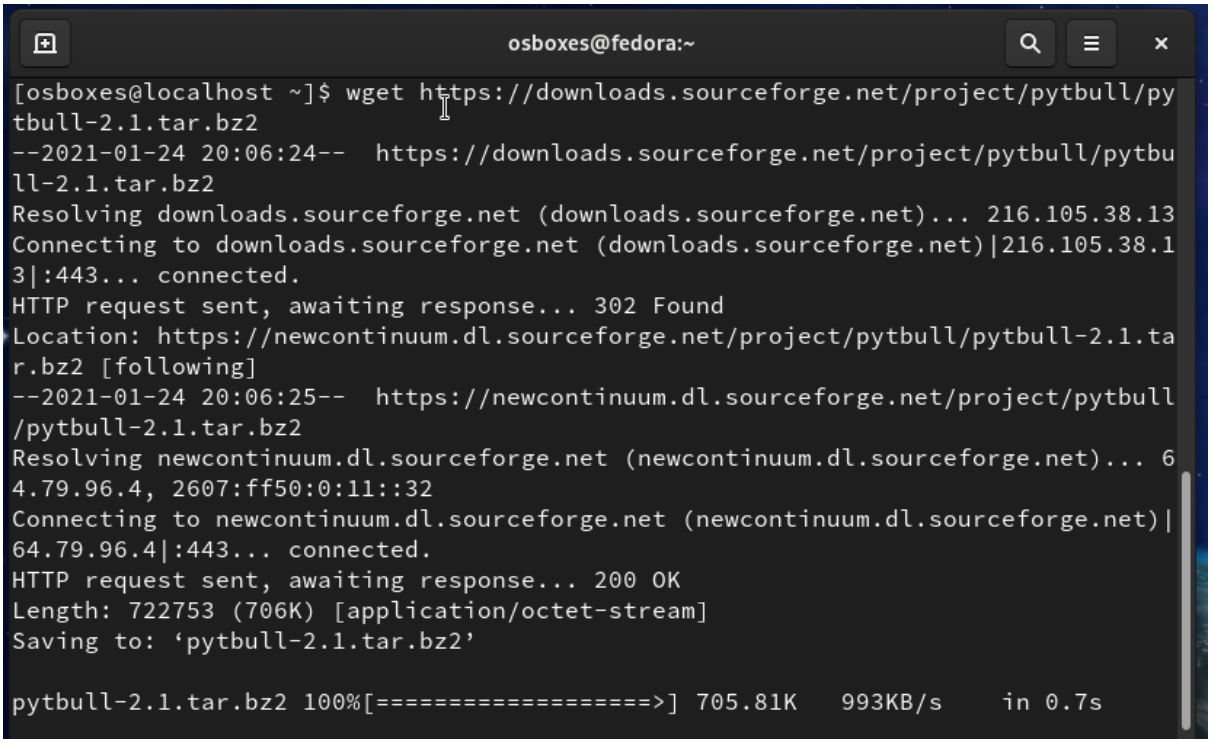


# Parte 2 - Configuración del servidor Pytbull en la máquina Fedora.

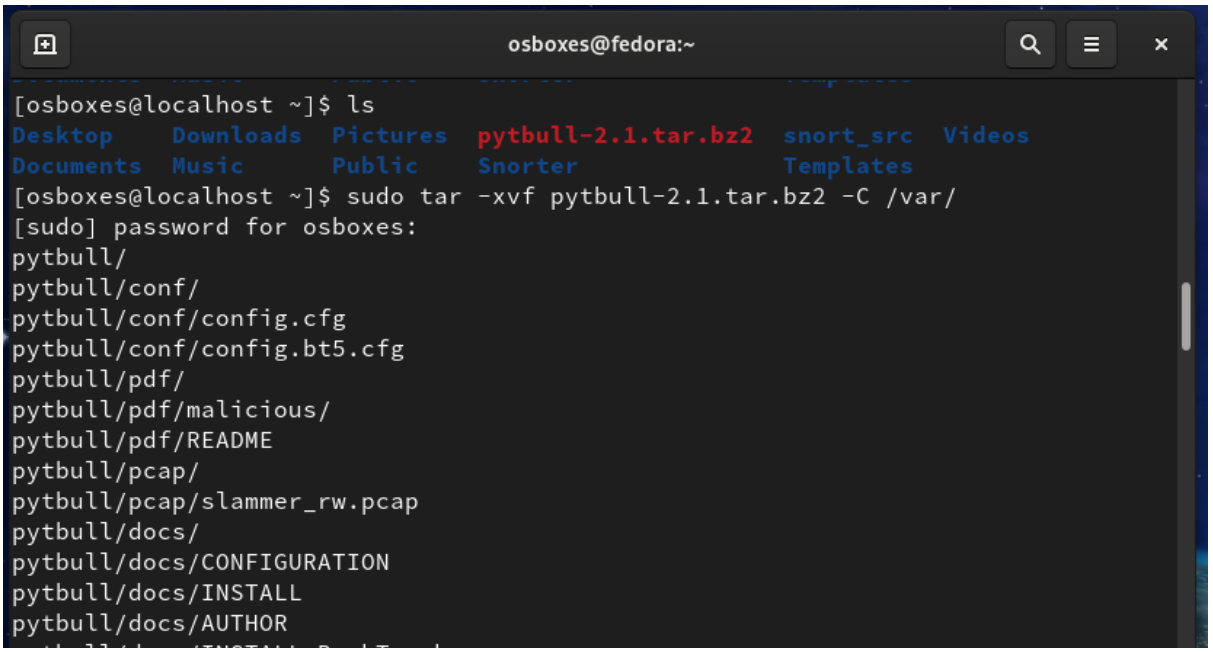
Nuevamente desde la consola instalamos las librerías que necesitaremos ahora para el servidor Pytbull:
***sudo yum install vsftpd httpd python2***




Descargar pytbull con el comando: ***wget***
***https://downloads.sourceforge.net/project/pytbull/pytbull-2.1.tar.bz2***

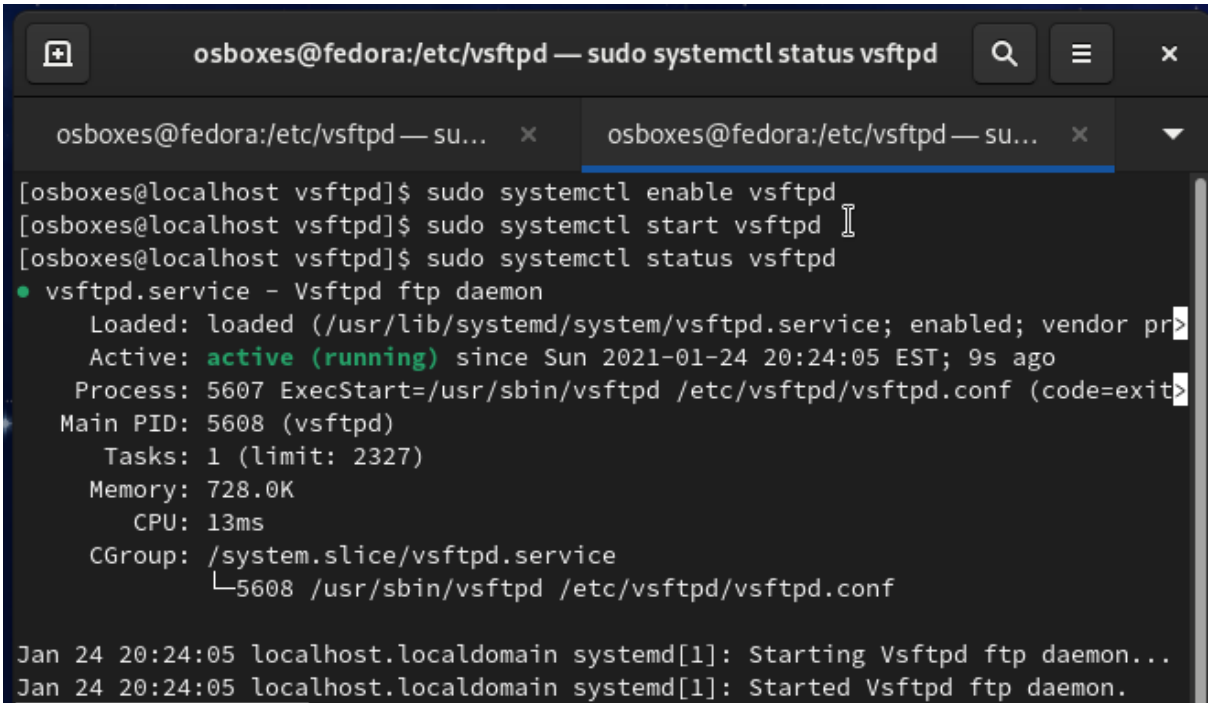


Extraer los archivos a la carpeta var: ***tar -xvf pytbull-2.1.tar.bz2 -C /var/***



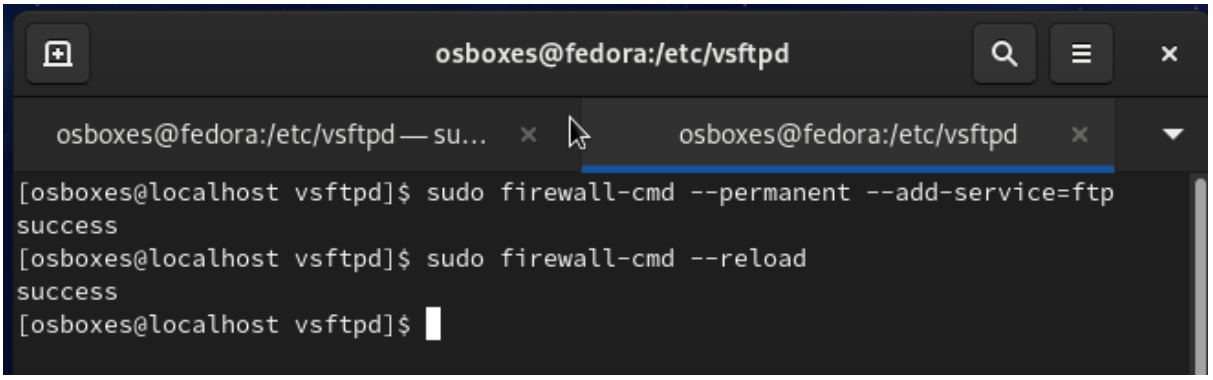

Habilitar e iniciar los servicios para los servidores web (httpd), ftp (vsftpd) y ssh (sshd):

***sudo systemctl enable vsftpd***
***sudo systemctl start vsftpd***
***sudo systemctl enable sshd***
***sudo systemctl start sshd***
**sudo systemctl enable httpd**
**sudo systemctl start httpd**

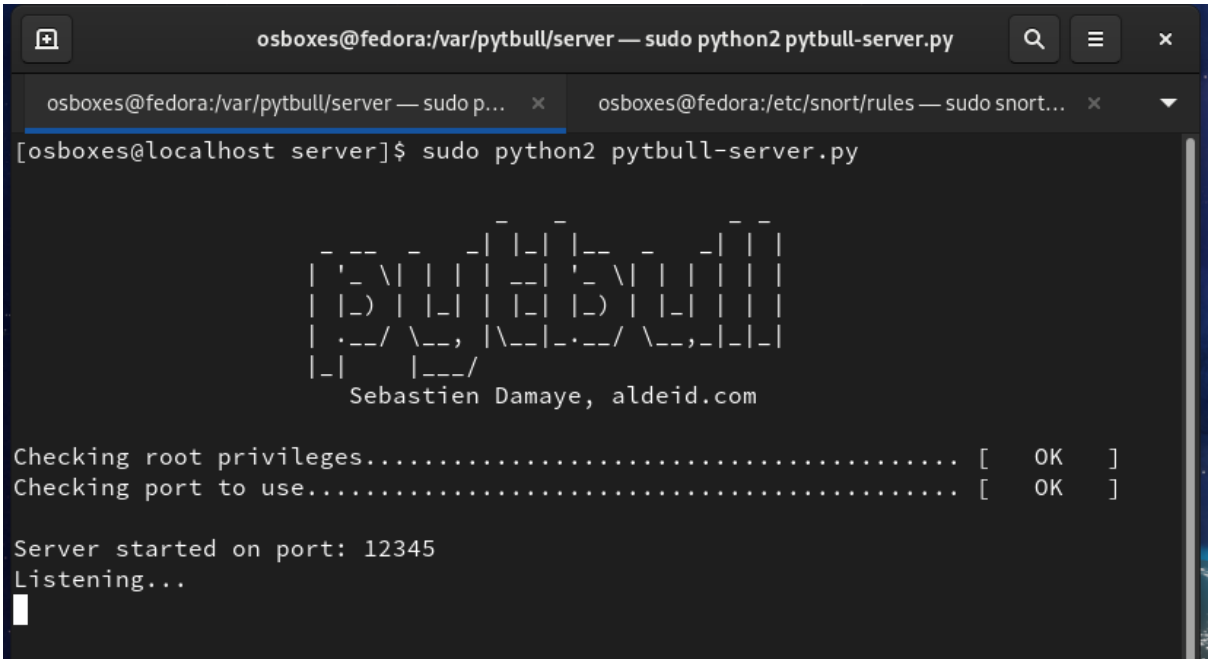



Nota: podemos verificar el status de los servicios (deben quedar activos) por ejemplo: ***sudo systemctl status httpd***

Ahora, añadimos las reglas  al firewall para permitir la comunicación de los servicios:
***sudo firewall-cmd --permanent --add-service=ftp***
***sudo firewall-cmd --permanent --add-service=ssh***
***sudo firewall-cmd --permanent --add-service=http***
***sudo firewall-cmd --permanent --add-port=21/tcp***
***sudo firewall-cmd --permanent --add-port=22/tcp***
***sudo firewall-cmd --permanent --add-port=80/tcp***
Luego reiniciamos el firewall:
***sudo firewall-cmd --reload***

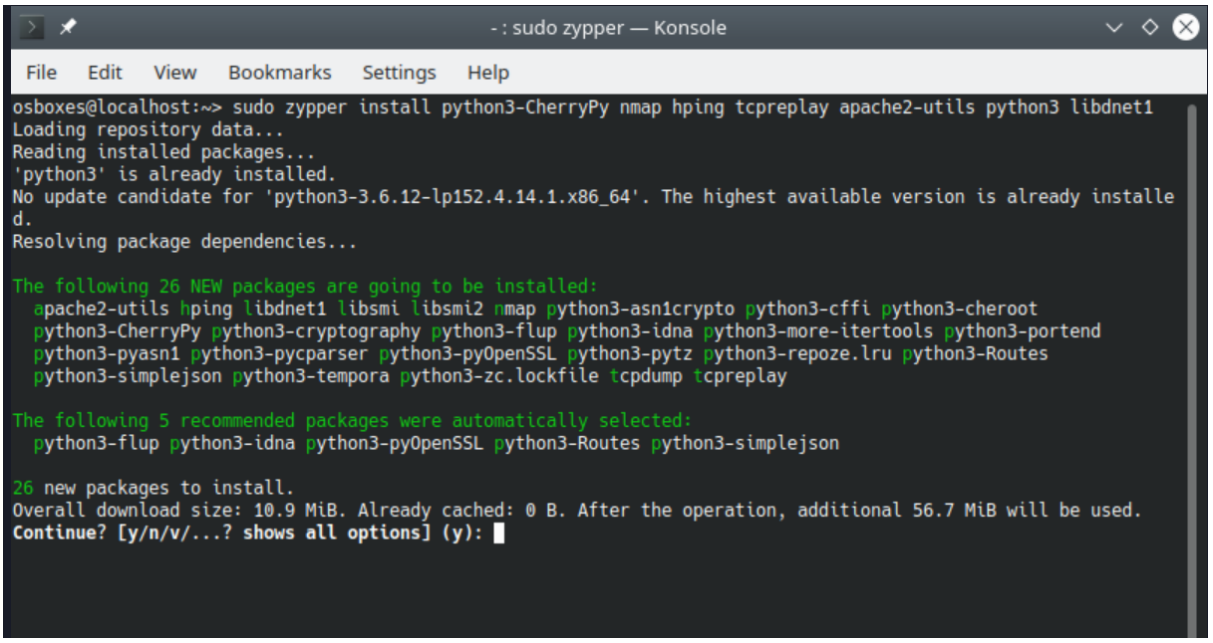



Nos dirigimos a /var/ftp/pub y creamos un archivo de alerta, el cual llamaremos desde el cliente pytbull:
***cd /var/ftp/pub***
***sudo touch snort.alert***
Nos dirigimos a la ruta /var/pytbull/server y ejecutamos el servidor de pytbull y lo dejamos en modo escucha:
***cd /var/pytbull/server***
***sudo python2 pytbull-server.py***

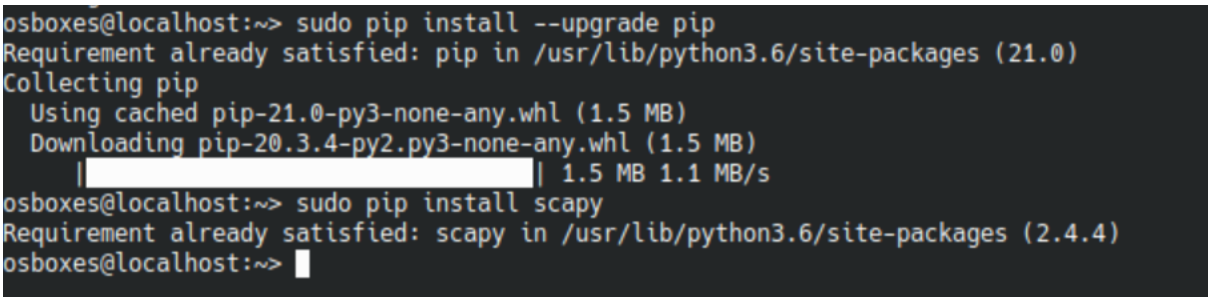


# Parte 3 - Configuración del cliente Pytbull en la máquina OpenSUSE.

Instalamos las librerias necesarias:
***sudo zypper install python2 python2-pip nmap hping tcpreplay apache2-utils python3 libdnet1 libssl45, libcrypto43 y libssh-devel python2-feedparser***

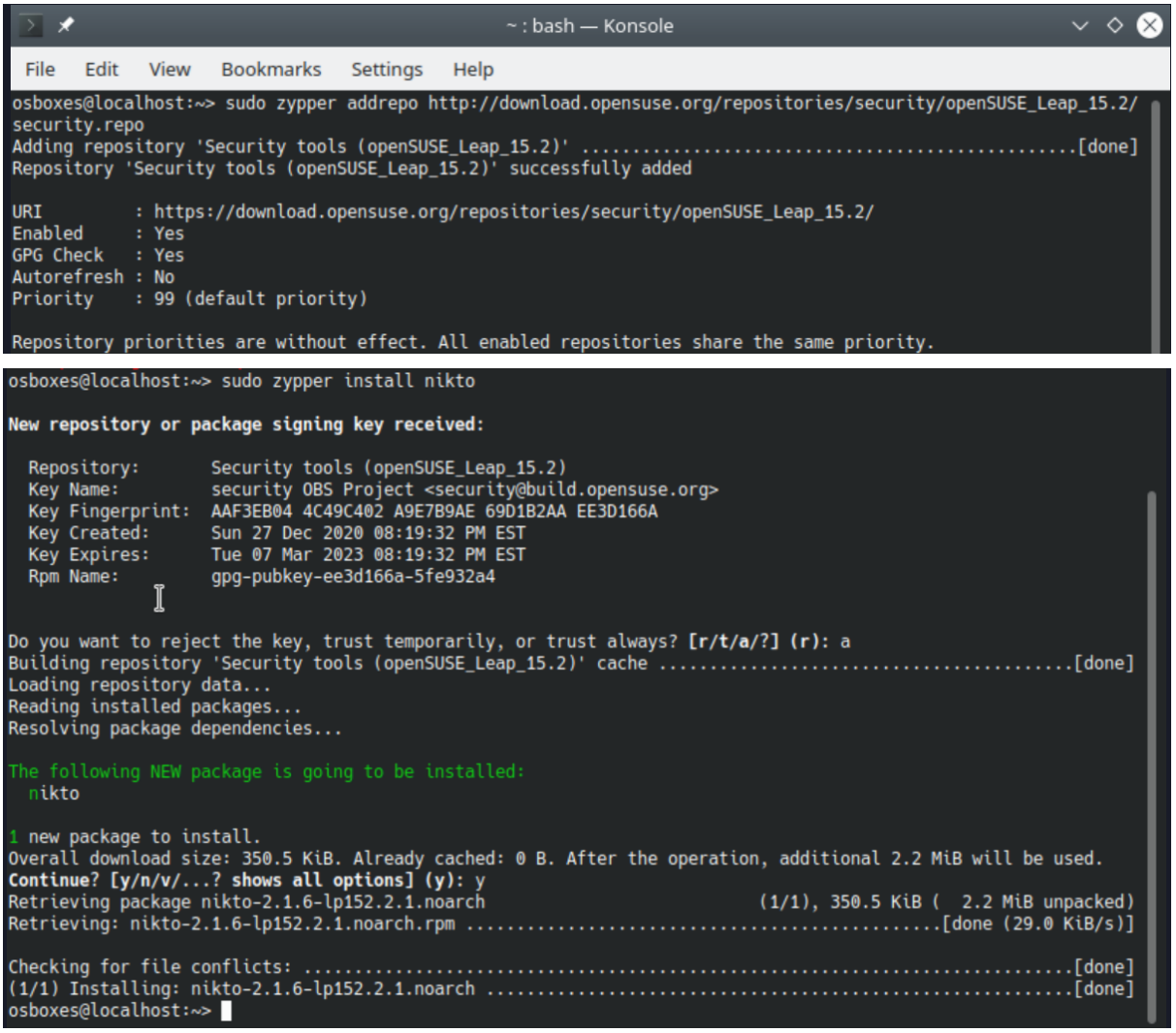



Ahora haremos un upgrade al comando pip para luego instalar las librería sscapy y cherrypy:
***sudo pip install --upgrade pip***
***sudo pip install scapy***
***sudo pip install cherrypy***



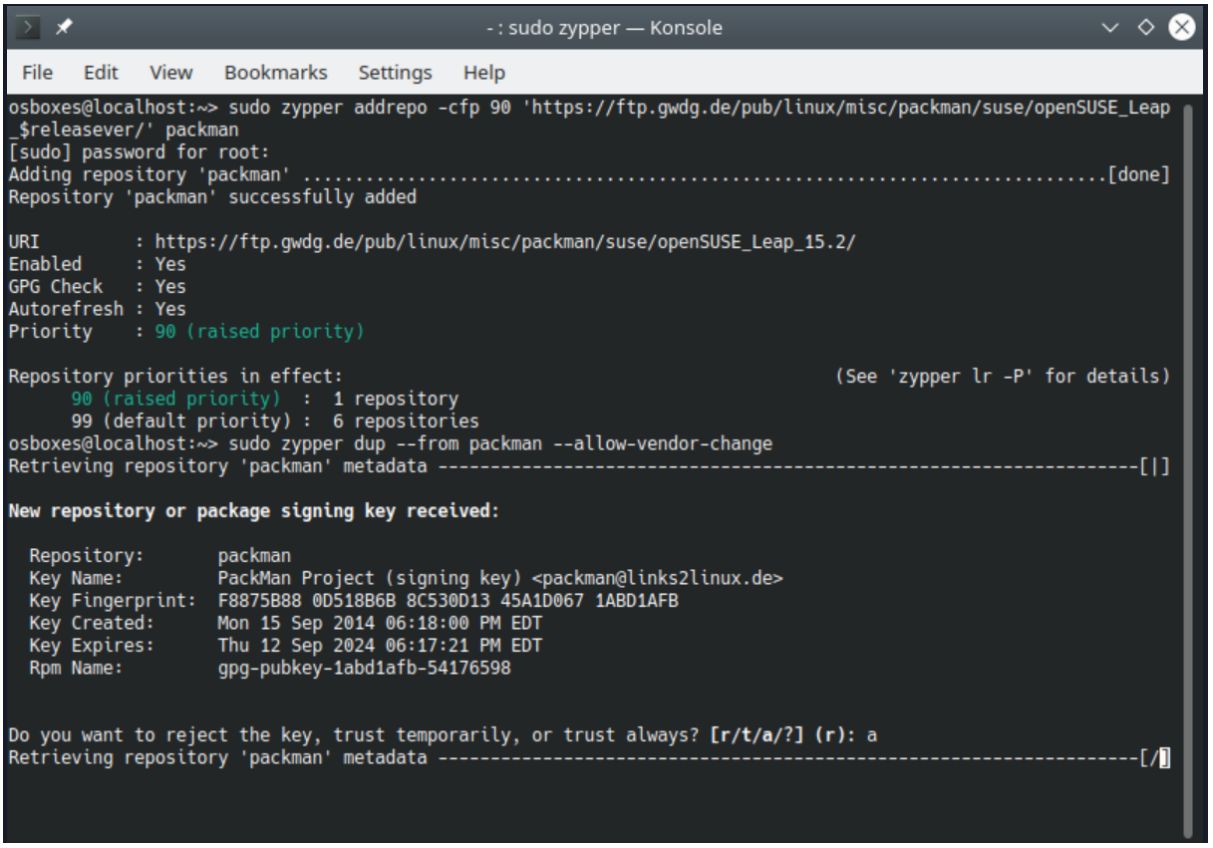

Agregamos el repositorio de seguridad que nos permitirá descargar nikto:
***zypper addrepo https://download.opensuse.org/repositories/security/openSUSE_Leap_15.2/security.repo***

***sudo zypper install nikto***

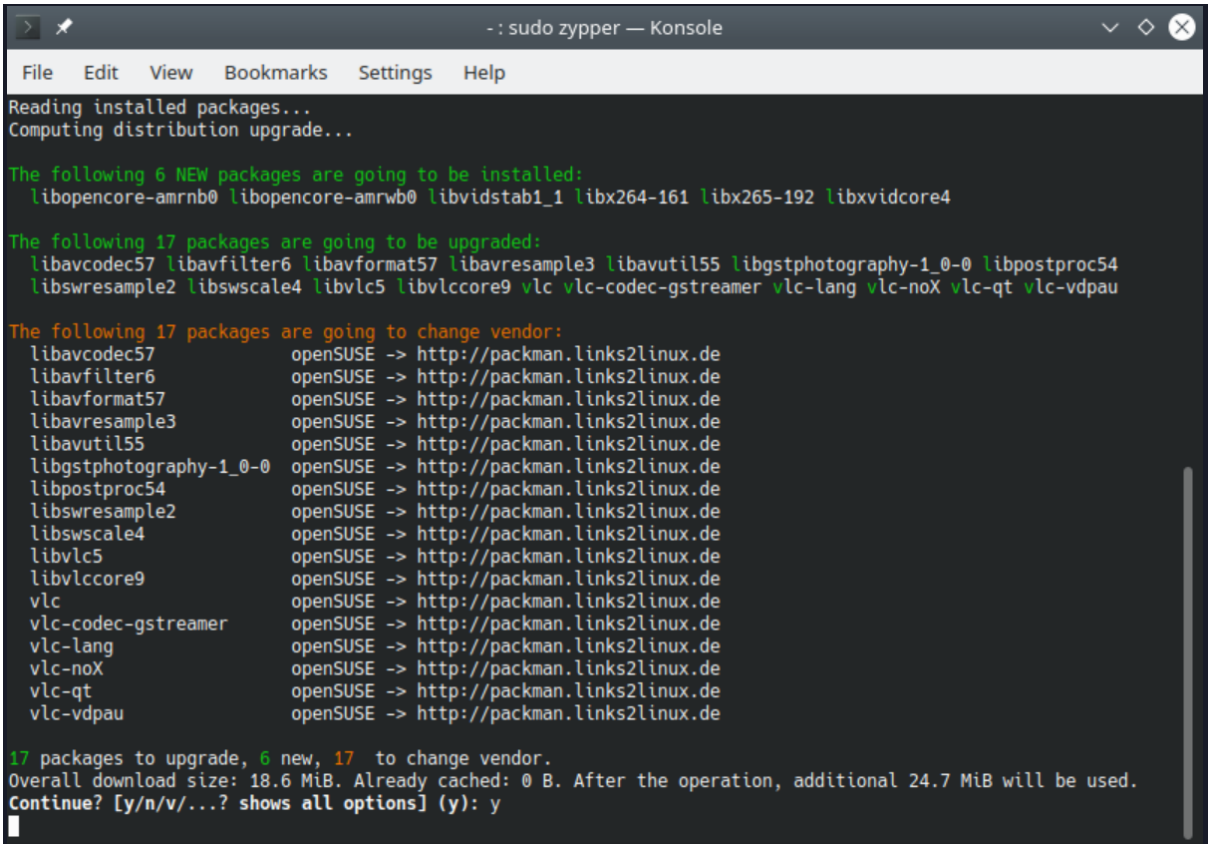



Ahora agregaremos el repositorio packman para descargar ncrack:

***zypper ar -cfp 90 'https://ftp.gwdg.de/pub/linux/misc/packman/suse/openSUSE_Leap_$releasever/' packman***




Indicamos la opción ***y*** para continuar




Procedemos a instalar ncrack con: ***sudo zypper install ncrack***

```
The following NEW package is going to be installed:
  ncrack

1 new package to install.
Overall download size: 630.6 KiB. Already cached: 0 B. After the operation, additional 1.6 MiB will be used.
Continue? [y/n/v/...? shows all options] (y): y
Retrieving package ncrack-0.6.0+git.20180620-pm152.2.7.x86_64            (1/1), 630.6 KiB (  1.6 MiB unpacked)
Retrieving: ncrack-0.6.0+git.20180620-pm152.2.7.x86_64.rpm ..............................[done (207.6 KiB/s)]

Checking for file conflicts: ..................................................................[done]
(1/1) Installing: ncrack-0.6.0+git.20180620-pm152.2.7.x86_64 ..................................[done]
osboxes@localhost:~>
```

Descargar pytbull con el comando: ***wget https://downloads.sourceforge.net/project/pytbull/pytbull-2.1.tar.bz2***
Extraemos la carpeta y la movemos al directorio /opt/ con: ***sudo tar -xvf pytbull-2.1.tar.bz2 -C /opt/***

```
Downloads : bash — Konsole
File  Edit  View  Bookmarks  Settings  Help
osboxes@localhost:~/Downloads> sudo tar -xvf pytbull-2.1.tar.bz2 -C /opt/
pytbull/
pytbull/conf/
pytbull/conf/config.cfg
pytbull/conf/config.bt5.cfg
```

Desde el script de pytbull, agregaremos dos líneas para importar la librería datetime la cual es utilizada por varios de los tests a ejecutar:
***sudo vim /opt/pytbull/pytbull***

```
osboxes@localhost:/opt/pytbull> sudo vim /opt/pytbull/pytbull
```

Ya dentro del archivo bajamos a la línea 368 y agregamos una nueva línea con lo siguiente:
***import datetime***
Repetimos este paso en la línea 482 del archivo y guardamos los cambios con ***ESC :wq!***

```
            except:
                print "[ Failed ]"
                print "\n***ERROR: Please setup reverse shell on remote server first or
use --mode=gateway!"
                sys.exit(0)
            print "[   OK   ]"

        for payload in payloads:
            # Perform test & write report
            str = "TEST #%s - %s" % (self.testnum, payload[0])
            print str[:62].ljust(65,'.'),
            import datetime
            test_dt_start = datetime.datetime.now()

            if self._mode=="standalone":
                # Send cmd to execute on server side (wget file)
                s.send("wget %s" % os.path.join(self.config.get('PATHS', 'urlpdf'), payl
oad[0]))
                # Issue 3450032 - Synchronisation issue. The server has to instruct
                # the client that the file has been successfully downloaded before it
                # goes to next file
                s.recv(1024)
            else:
                self.downloadFile(self.config.get('PATHS', 'urlpdf'), payload[0])

            # Sleep before getting alerts
            time.sleep(int(self.config.get('TIMING', 'sleepbeforegetalerts')))
-- INSERT --                                                    482,28        68%
```

Ahora nos ubicamos en la siguiente ruta:
***cd /opt/pytbull/conf*** y editamos el archivo de configuración config.cfg para ir ingresando los datos:
***sudo vim config.cfg***

```
[CLIENT]
ipaddr                  = 192.168.2.9
iface                   = eth1
useproxy                = 0
proxyhost               =
proxyport               =
proxyuser               =
proxypass               =

[PATHS]
db                      = data/pytbull.db
urlpdf                  = https://github.com/sebastiendamaye/public/raw/master/infected/
pdfdir                  = pdf/malicious
pcapdir                 = pcap
tempfile                = /tmp/pytbull.tmp
alertsfile              = /var/ftp/pub/snort.alert
#alertsfile              = /var/log/suricata/fast.log
```

En la estructura del archivo, la cual está dividida por secciones, ingresaremos la información correspondiente:

- En la sección client, solo ingresaremos la ipaddr en la cual tenemos instalado nuestro cliente pytbull y la iface que indica la interfaz de salida que usará pytbull para enviar los payloads.
- En la sección PATHS, en las variables alertfiles indicamos la dirección donde se almacenan nuestros archivos de alerta, en este caso indicaremos la siguiente ruta: /var/ftp/pub/snort.alert
  Es importante mantener solo una ruta alertfiles activa, la otra se comenta con un #.
  Nota: Para efectos del taller se creó anteriormente el archivo snort.alert para evitar el error de pytbull client al intentar acceder al archivo de alerta, ya que dicha configuración no está dentro del alcance del taller. su objetivo es mostrar en el reporte de pytbull las pruebas que el IDS detectó correctamente. Para ello se debe configurar el acceso FTP a la ruta de logs de snort con los permisos correspondientes.

En la sección FTP agregamos el protocolo FTP, puerto 21 y las credenciales de nuestra máquina osboxes para establecer la conexión:

```
[FTP]
ftpproto                = ftp
ftpport                 = 21
ftpuser                 = osboxes
ftppasswd               = osboxes.org
```

Las secciones Timing y Server no requieren modificación.

En la sección ENV, modificamos las rutas por defecto y agregamos las siguientes para nikto (***/usr/bin/nikto***), niktoconf (***/etc/nikto.conf***) y ncrack (***/usr/bin/ncrack***) como se muestra en la imagen:

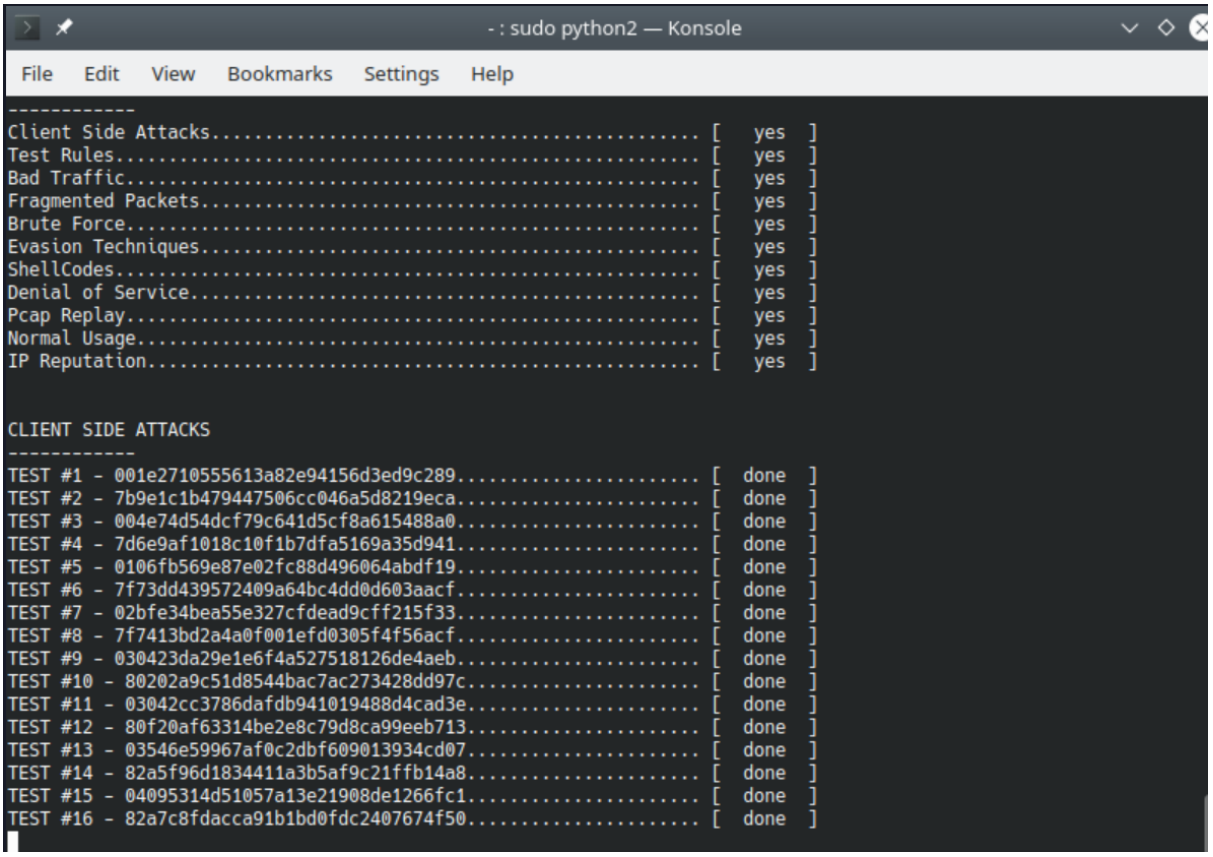
```
[ENV]
sudo                = /usr/bin/sudo
nmap                = /usr/bin/nmap
nikto               = /usr/bin/nikto
niktoconf           = /etc/nikto.conf
hping3              = /usr/sbin/hping3
tcpreplay           = /usr/bin/tcpreplay
ab                  = /usr/bin/ab
ping                = /bin/ping
ncrack              = /usr/bin/ncrack
ncrackusers         = data/ncrack-users.txt
ncrackpasswords     = data/ncrack-passwords.txt
-- INSERT -- W10: Warning: Changing a readonly file                    16,53         Top
```

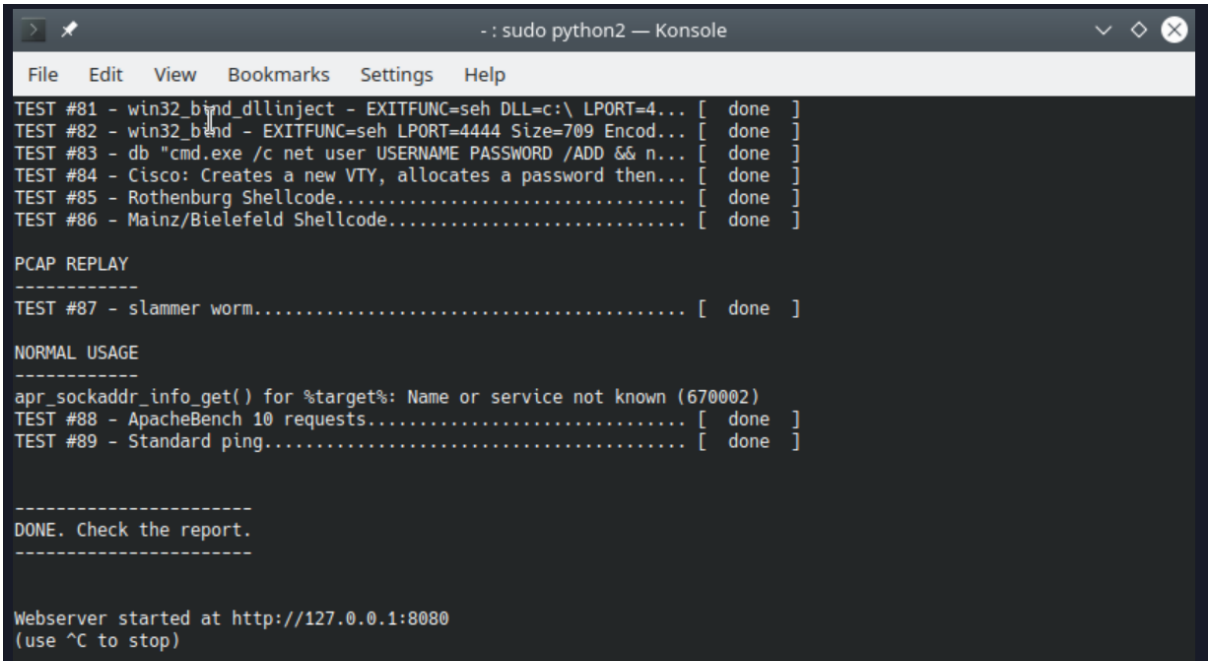
El apartado de TESTS puede modificarse para habilitar (valor 1) o deshabilitar (valor 0) los tests que queramos realizar. Para este caso deshabilitamos los testRules, denialOfService y ipReputation.

```
[TESTS]
clientSideAttacks       = 1
testRules               = 0
badTraffic              = 1
fragmentedPackets       = 1
bruteForce              = 1
evasionTechniques       = 1
shellCodes              = 1
denialOfService         = 0
pcapReplay              = 1
normalUsage             = 1
ipReputation            = 0
```

Una vez realizadas las configuraciones, procedemos a ejecutar las pruebas desde pytbull client: ***sudo python2 ./pytbull -t 192.168.2.8***

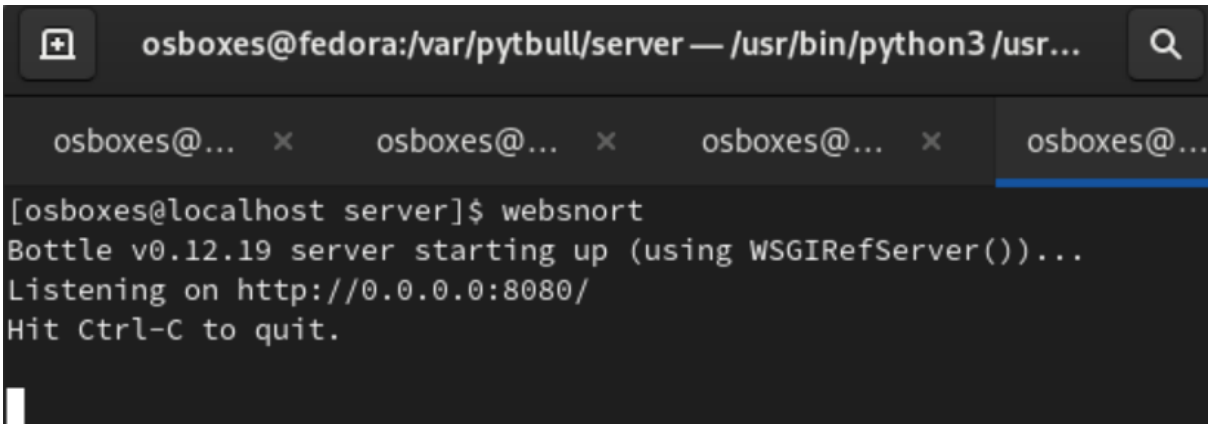
```
osboxes@localhost:/opt/pytbull> sudo python2 ./pytbull -t 192.168.2.8

                    I          _   _           _ _
                 _ __  _   _| |_| |__   _   _| | |
                | '_ \| | | | __| '_ \| | | | | |
                | |_) | |_| | |_| |_) | |_| | | |
                | .__/ \__, |\__|_.__/ \__,_|_|_|
                |_|    |___/
                   Sebastien Damaye, aldeid.com

What would you like to do?
1. Run a new campaign (will erase previous results)
2. View results from previous campaign
3. Exit
Choose an option: 1

(standalone mode)

+-------------------------------------------------------------------------+
| pytbull will set off IDS/IPS alarms and/or other security devices       |
| and security monitoring software. The user is aware that malicious      |
| content will be downloaded and that the user should have been           |
| authorized before running the tool.                                     |
+-------------------------------------------------------------------------+
Do you accept (y/n)? y
```


- : sudo python2 — Konsole
File Edit View Bookmarks Settings Help
Client Side Attacks..... [ yes ]
Test Rules..... [ yes ]
Bad Traffic..... [ yes ]
Fragmented Packets..... [ yes ]
Brute Force..... [ yes ]
Evasion Techniques..... [ yes ]
ShellCodes..... [ yes ]
Denial of Service..... [ yes ]
Pcap Replay..... [ yes ]
Normal Usage..... [ yes ]
IP Reputation..... [ yes ]
CLIENT SIDE ATTACKS
TEST #1 - 001e2710555613a82e94156d3ed9c289..... [ done ]
TEST #2 - 7b9e1c1b479447506cc046a5d8219eca..... [ done ]
TEST #3 - 004e74d54dcf79c641d5cf8a615488a0..... [ done ]
TEST #4 - 7d6e9af1018c10f1b7dfa5169a35d941..... [ done ]
TEST #5 - 0106fb569e87e02fc88d496064abdf19..... [ done ]
TEST #6 - 7f73dd439572409a64bc4dd0d603aacf..... [ done ]
TEST #7 - 02bfe34bea55e327cfdead9cff215f33..... [ done ]
TEST #8 - 7f7413bd2a4a0f001efd0305f4f56acf..... [ done ]
TEST #9 - 030423da29e1e6f4a527518126de4aeb..... [ done ]
TEST #10 - 80202a9c51d8544bac7ac273428dd97c..... [ done ]
TEST #11 - 03042cc3786dafdb941019488d4cad3e..... [ done ]
TEST #12 - 80f20af63314be2e8c79d8ca99eeb713..... [ done ]
TEST #13 - 03546e59967af0c2dbf609013934cd07..... [ done ]
TEST #14 - 82a5f96d1834411a3b5af9c21ffb14a8..... [ done ]
TEST #15 - 04095314d51057a13e21908de1266fc1..... [ done ]
TEST #16 - 82a7c8fdacca91b1bd0fdc2407674f50..... [ done ]

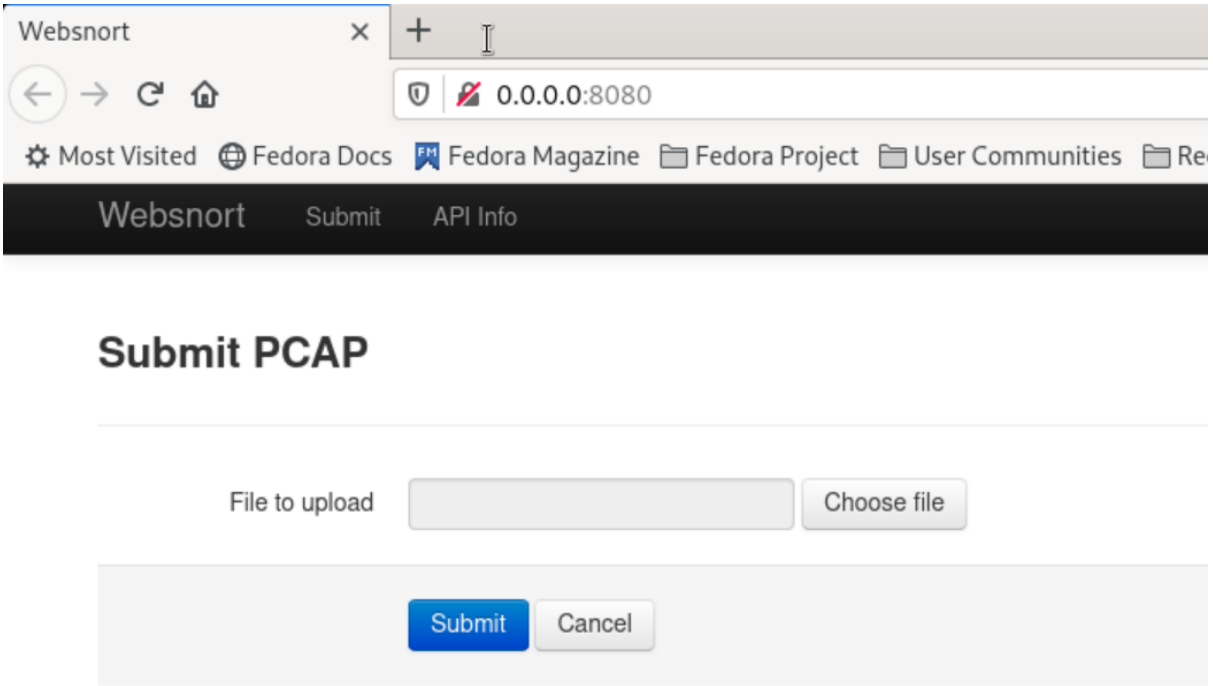

- : sudo python2 — Konsole
File Edit View Bookmarks Settings Help
TEST #81 - win32_bind_dllinject - EXITFUNC=seh DLL=c:\ LPORT=4... [ done ]
TEST #82 - win32_bind - EXITFUNC=seh LPORT=4444 Size=709 Encod... [ done ]
TEST #83 - db "cmd.exe /c net user USERNAME PASSWORD /ADD && n... [ done ]
TEST #84 - Cisco: Creates a new VTY, allocates a password then... [ done ]
TEST #85 - Rothenburg Shellcode..... [ done ]
TEST #86 - Mainz/Bielefeld Shellcode..... [ done ]
PCAP REPLAY
TEST #87 - slammer worm..... [ done ]
NORMAL USAGE
apr_sockaddr_info_get() for %target%: Name or service not known (670002)
TEST #88 - ApacheBench 10 requests..... [ done ]
TEST #89 - Standard ping..... [ done ]
DONE. Check the report.
Webserver started at http://127.0.0.1:8080
(use ^C to stop)

Luego de terminada la ejecución de los tests de pytbull, podemos regresar a la máquina snort a visualizar las alertas generadas utilizando websnort:
Asignamos los permisos correspondientes a la carpeta de logs de snort para que pueda utilizarlos websnort:
***sudo chmod -R 775 /var/log/snort***

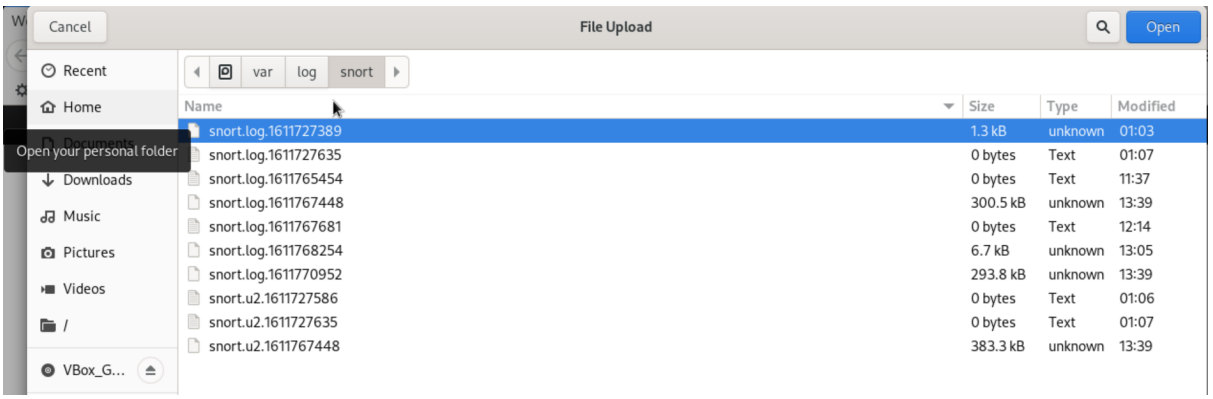

Ejecutamos el comando ***websnort*** desde la consola:



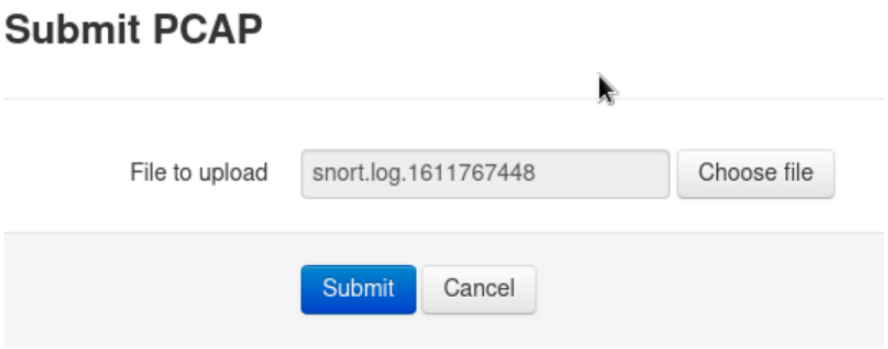

Y nos dirigimos a la ruta indicada: ***http://0.0.0.0:8080*** donde seleccionamos "***choose file***" para cargar un archivo de alerta:

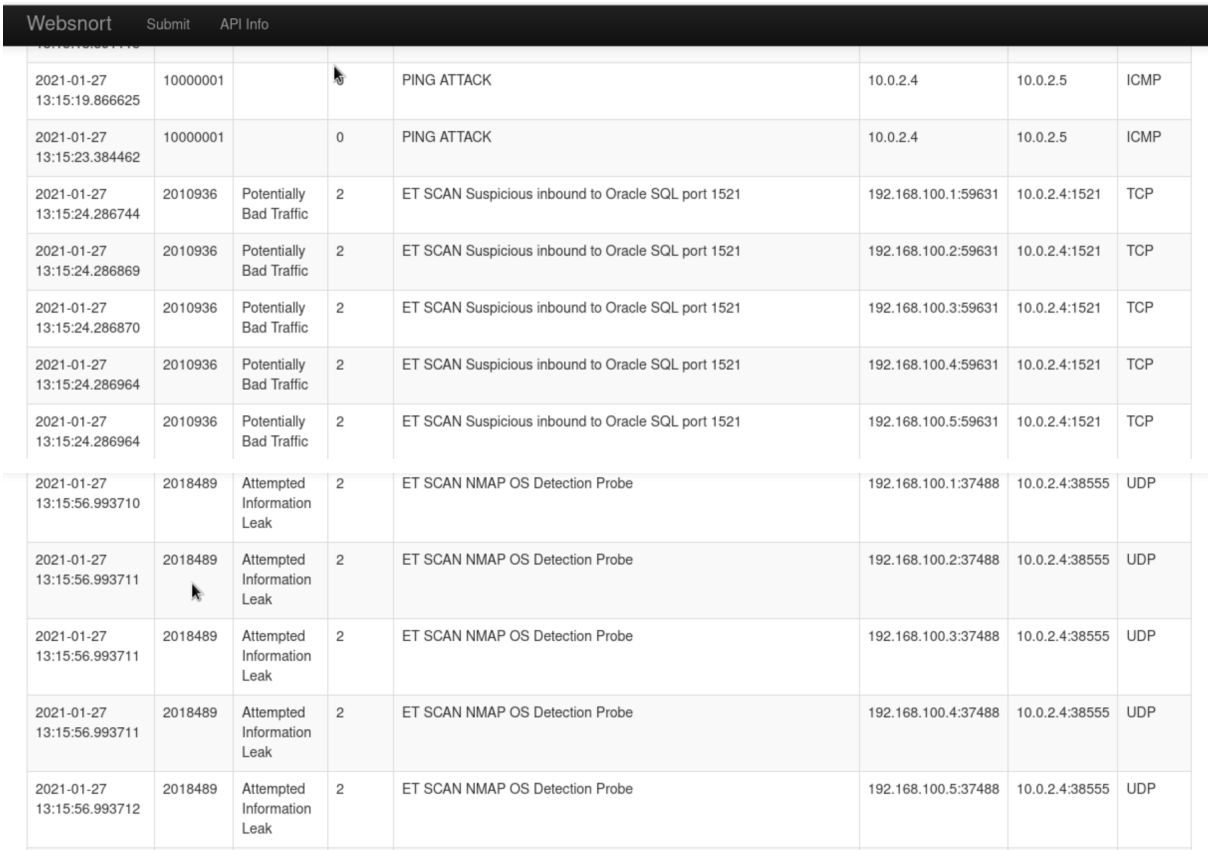



Seleccionamos el archivo de log a visualizar:




Seleccionamos submit y luego podremos visualizar las alertas correspondientes:




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2021-01-27 13:15:19.866625 | 10000001 | | 0 | PING ATTACK | 10.0.2.4 | 10.0.2.5 | ICMP |
| 2021-01-27 13:15:23.384462 | 10000001 | | 0 | PING ATTACK | 10.0.2.4 | 10.0.2.5 | ICMP |
| 2021-01-27 13:15:24.286744 | 2010936 | Potentially Bad Traffic | 2 | ET SCAN Suspicious inbound to Oracle SQL port 1521 | 192.168.100.1:59631 | 10.0.2.4:1521 | TCP |
| 2021-01-27 13:15:24.286869 | 2010936 | Potentially Bad Traffic | 2 | ET SCAN Suspicious inbound to Oracle SQL port 1521 | 192.168.100.2:59631 | 10.0.2.4:1521 | TCP |
| 2021-01-27 13:15:24.286870 | 2010936 | Potentially Bad Traffic | 2 | ET SCAN Suspicious inbound to Oracle SQL port 1521 | 192.168.100.3:59631 | 10.0.2.4:1521 | TCP |
| 2021-01-27 13:15:24.286964 | 2010936 | Potentially Bad Traffic | 2 | ET SCAN Suspicious inbound to Oracle SQL port 1521 | 192.168.100.4:59631 | 10.0.2.4:1521 | TCP |
| 2021-01-27 13:15:24.286964 | 2010936 | Potentially Bad Traffic | 2 | ET SCAN Suspicious inbound to Oracle SQL port 1521 | 192.168.100.5:59631 | 10.0.2.4:1521 | TCP |
| 2021-01-27 13:15:56.993710 | 2018489 | Attempted Information Leak | 2 | ET SCAN NMAP OS Detection Probe | 192.168.100.1:37488 | 10.0.2.4:38555 | UDP |
| 2021-01-27 13:15:56.993711 | 2018489 | Attempted Information Leak | 2 | ET SCAN NMAP OS Detection Probe | 192.168.100.2:37488 | 10.0.2.4:38555 | UDP |
| 2021-01-27 13:15:56.993711 | 2018489 | Attempted Information Leak | 2 | ET SCAN NMAP OS Detection Probe | 192.168.100.3:37488 | 10.0.2.4:38555 | UDP |
| 2021-01-27 13:15:56.993711 | 2018489 | Attempted Information Leak | 2 | ET SCAN NMAP OS Detection Probe | 192.168.100.4:37488 | 10.0.2.4:38555 | UDP |
| 2021-01-27 13:15:56.993712 | 2018489 | Attempted Information Leak | 2 | ET SCAN NMAP OS Detection Probe | 192.168.100.5:37488 | 10.0.2.4:38555 | UDP |