INFORME PRÁCTICA 6

Julio César Velásquez Cárdenas 1397896
David Sánchez González 1401641
Asignatura: Gestió i Administració de Xarxes

Ejercicio 1

a) Configurar cluster con Docker Swarm

Para crear el Swarm utilizaremos las máquinas A y D, ya que son las dos que ya disponen de una interfaz Internet, por lo que no necesitaríamos hacer nada más. En nuestro caso, la máquina A será el *manager* y la máquina D será un *worker*.

Comenzamos inicializando el Swarm en la máquina A, para ello ejecutamos el siguiente comando:

```
root@master-1-8:~# docker swarm init --advertise-addr 10.10.10.53
Swarm initialized: current node (7nskduzt2h14gyl444zm8zt8e) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1o14xa2eeyjtotdisdfucs3uclc2jbs1dn960zumdjv2g6lzjr-0ajhhej57jg8qy1zljgsp59tw 10.10.10.
53:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Con ello ya tenemos la máquina A como *manager* del Swarm, a continuación seguimos las instrucciones para añadir el *worker*, la máquina D.

```
root@extra1-1-8:~# docker swarm join --token SWMTKN-1-1o14xa2eeyjtotdisdfucs3uclc2jbs1dn960zumdjv2g6lzjr-0ajhhej57jg8qy1zljgs
p59tw 10.10.10.53:2377
This node joined a swarm as a worker.
```

b) Mostrar un *index.html* diferente en cada máquina

Creamos los Dockerfile tal como se indica en el enunciado, quedando de la siguiente manera para las máquinas A y D respectivamente. Añadiremos también la variable de entorno *DEBIAN_FRONTEND=noninteractive* para evitar que el script de creación del contenedor se quede atascado intentando pedir un input.

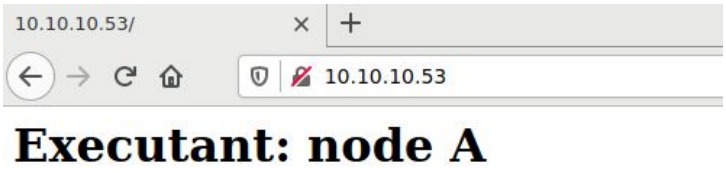
```
FROM ubuntu

ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install -y apache2
RUN echo "<html><body><h1>Executant: node A</h1></body></html>" > /var/www/html/index.html

EXPOSE 80
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

```
FROM ubuntu

ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install -y apache2
RUN echo "<html><body><h1>Executant: node D</h1></body></html>" > /var/www/html/index.html

EXPOSE 80
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

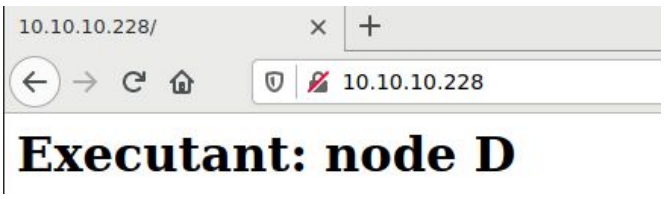
A continuación creamos los contenedores de ambas máquinas, para las cuales el comando será el mismo para ambas, *docker build --tag=apache2-server* . en el directorio que contiene el Dockerfile, en ambos casos */root/docker-apache*.

Una vez hecho esto, ejecutamos el contenedor permitiendo que el tráfico se dirija al puerto 80, necesario para el protocolo HTTP. En el caso de la máquina A, ha sido necesario parar el servidor Apache utilizado en prácticas anteriores.

```
root@master-1-8:~/docker-apache# systemctl stop apache2
root@master-1-8:~/docker-apache# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
apache2-server      latest              e40054d29b02        14 minutes ago      213MB
ubuntu              latest              f643c72bc252        6 weeks ago         72.9MB
root@master-1-8:~/docker-apache# docker run -d -p 80:80 apache2-server
c231eef1f6c6835b2339cc9f0a35675f29e16d401a5738829b88685c00df9e36
```



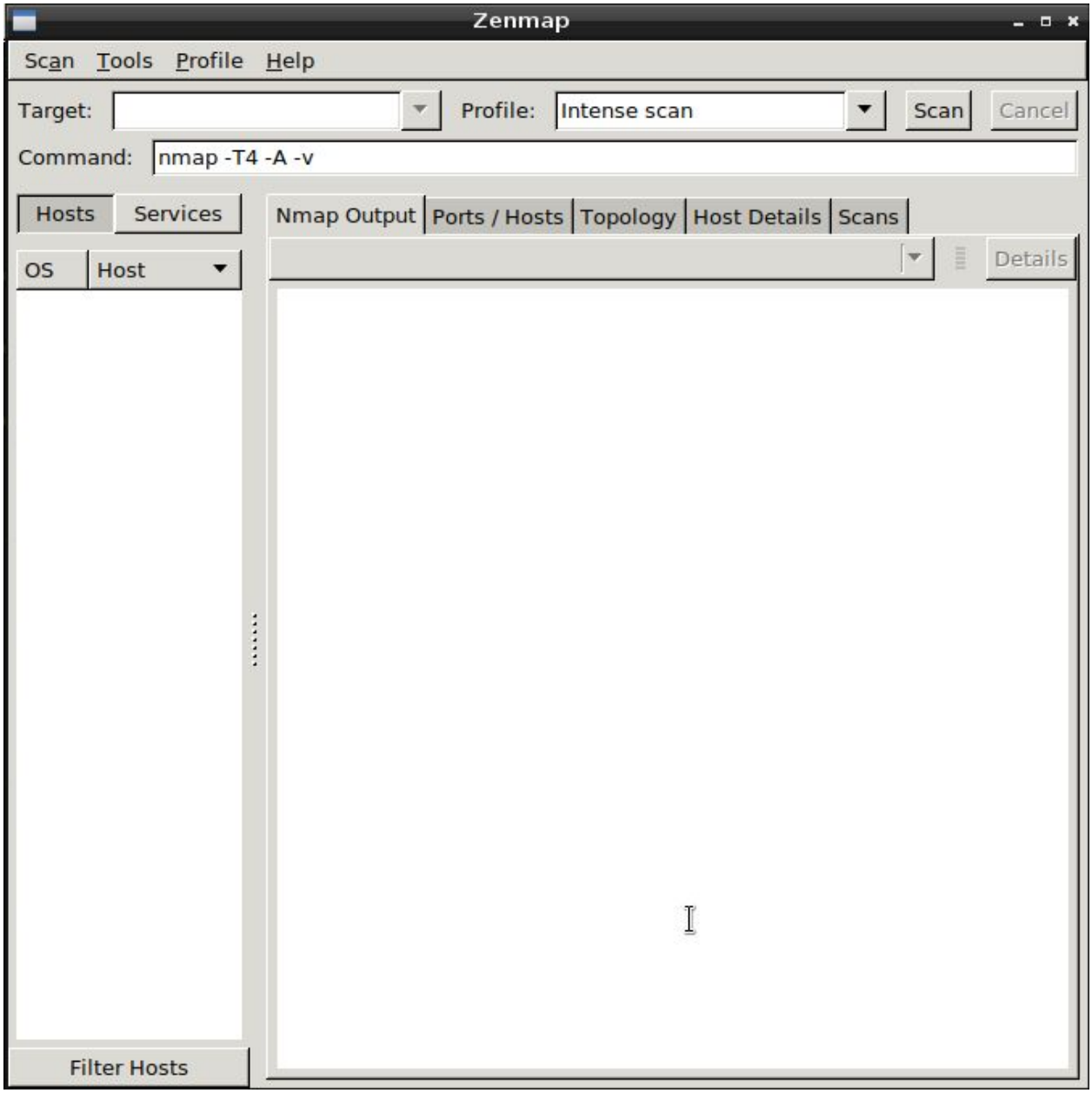

```
root@extra1-1-8:~/docker-apache# docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
apache2-server            latest              cef03c0a69d0        13 minutes ago      213MB
<none>                    <none>              b2dc680b2d82        17 minutes ago      99.4MB
<none>                    <none>              5224e9fc601e        20 minutes ago      99.4MB
<none>                    <none>              c2ad9319188f        54 minutes ago      99.4MB
ganglia                   latest              66329c97964e        4 weeks ago         263MB
<none>                    <none>              8cc4bd8a40bb        4 weeks ago         263MB
ubuntu                    latest              f643c72bc252        6 weeks ago         72.9MB
gcr.io/cadvisor/cadvisor  v0.38.0             129037340334        8 weeks ago         163MB
root@extra1-1-8:~/docker-apache# docker run -d -p 80:80 apache2-server
b0b3cfb316772b4ca29b827186ea50889423549e1e8b95d04a21179278da94a1
```



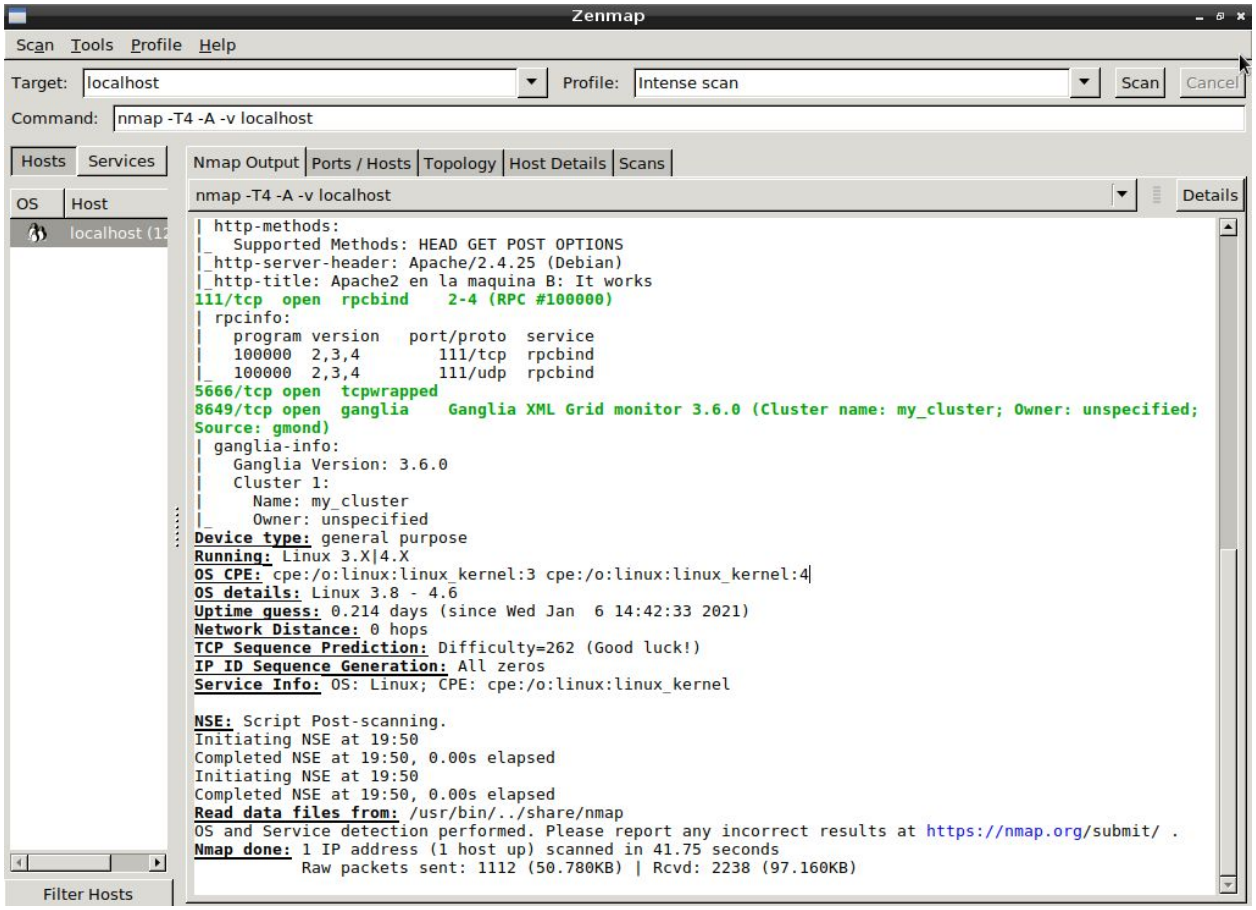
Ejercicio 2

a) Análisis Zenmap

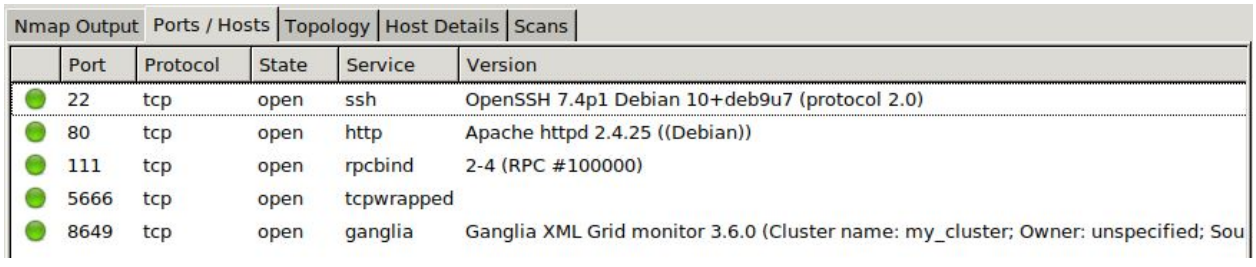
Instalamos Zenmap en las máquinas B y C usando el comando apt-get install zenmap. Iniciamos el programa y nos mostrará la siguiente interfaz.



Probaremos diferentes scans sobre estas dos máquinas con diferentes *profiles.* Aquí podemos ver un scan con el profile *Intense* en la máquina B.

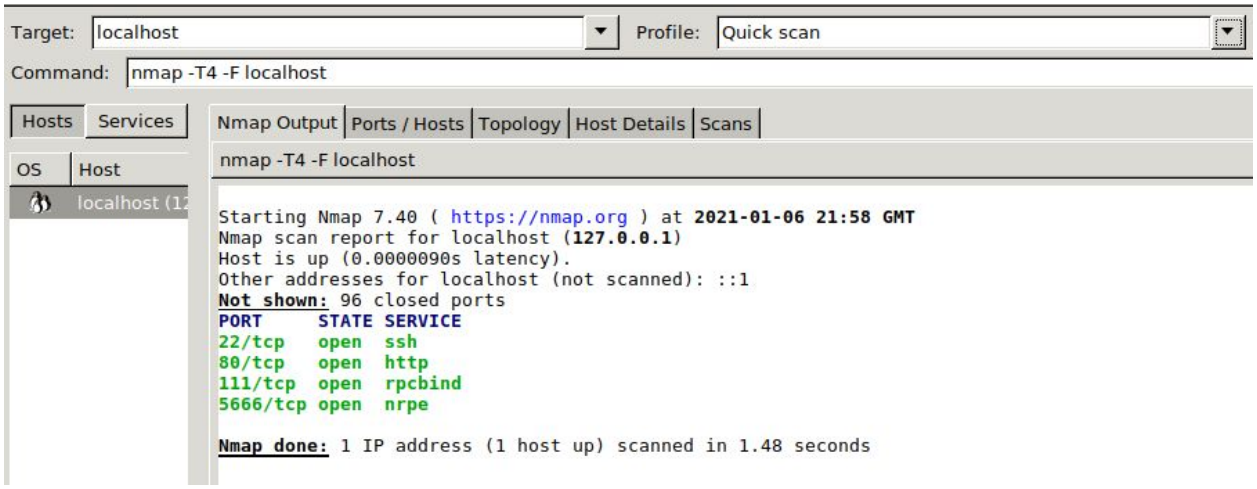


También podemos ver el comando que ha sido usado, en este caso sería el *nmap -T4 -A -v localhost* y en la pestaña de Ports/Hosts se puede observar los puertos actualmente abiertos que contiene algún servicio.

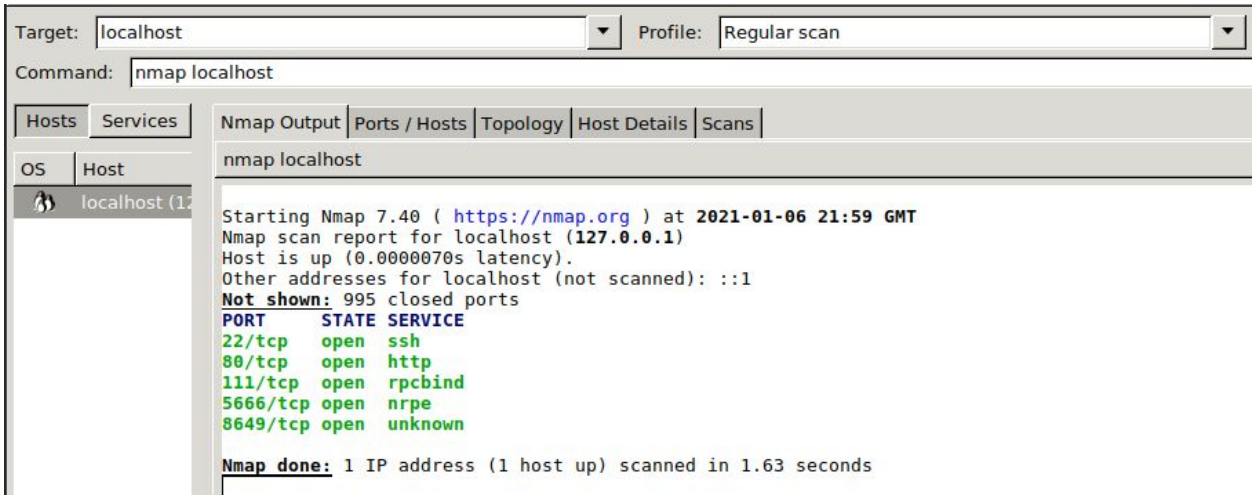


Estos son los demás escaneos realizados sobre esta máquina.

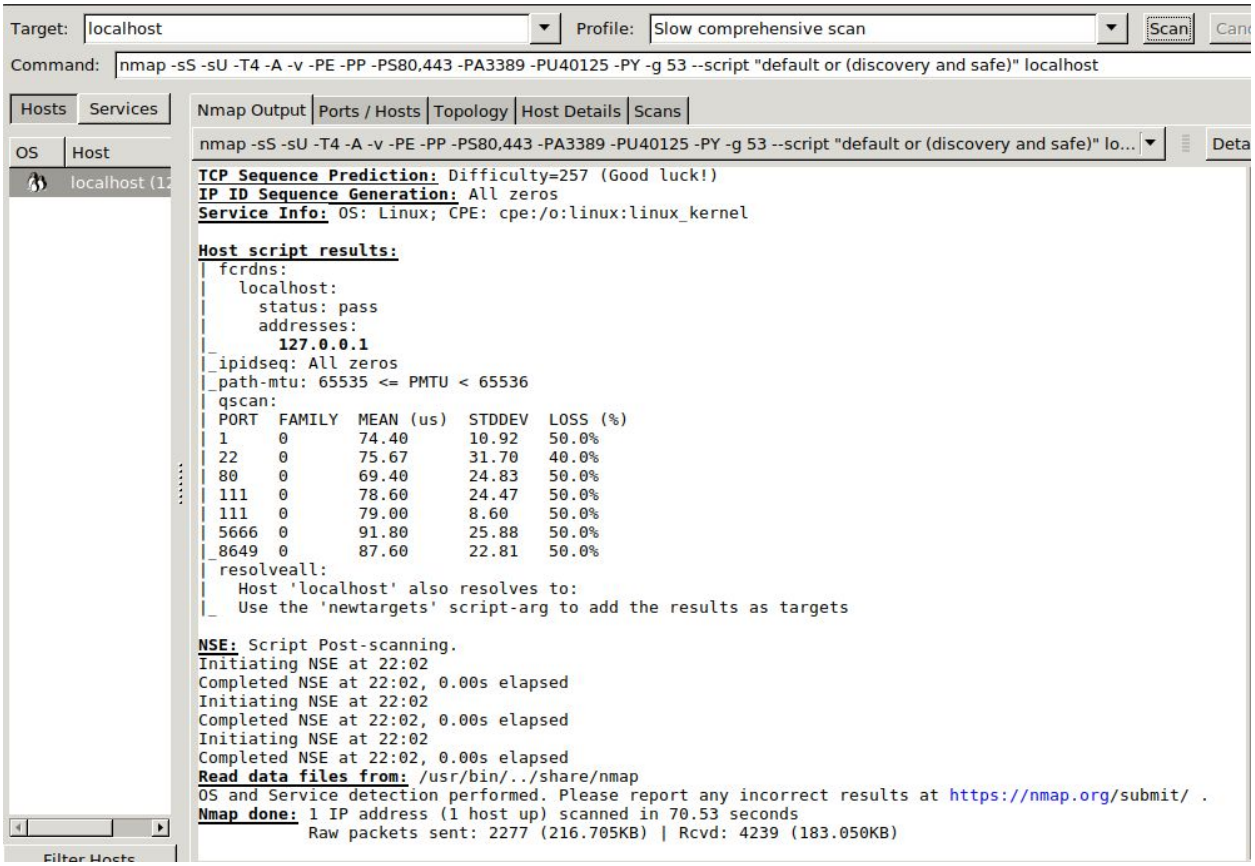
Quick



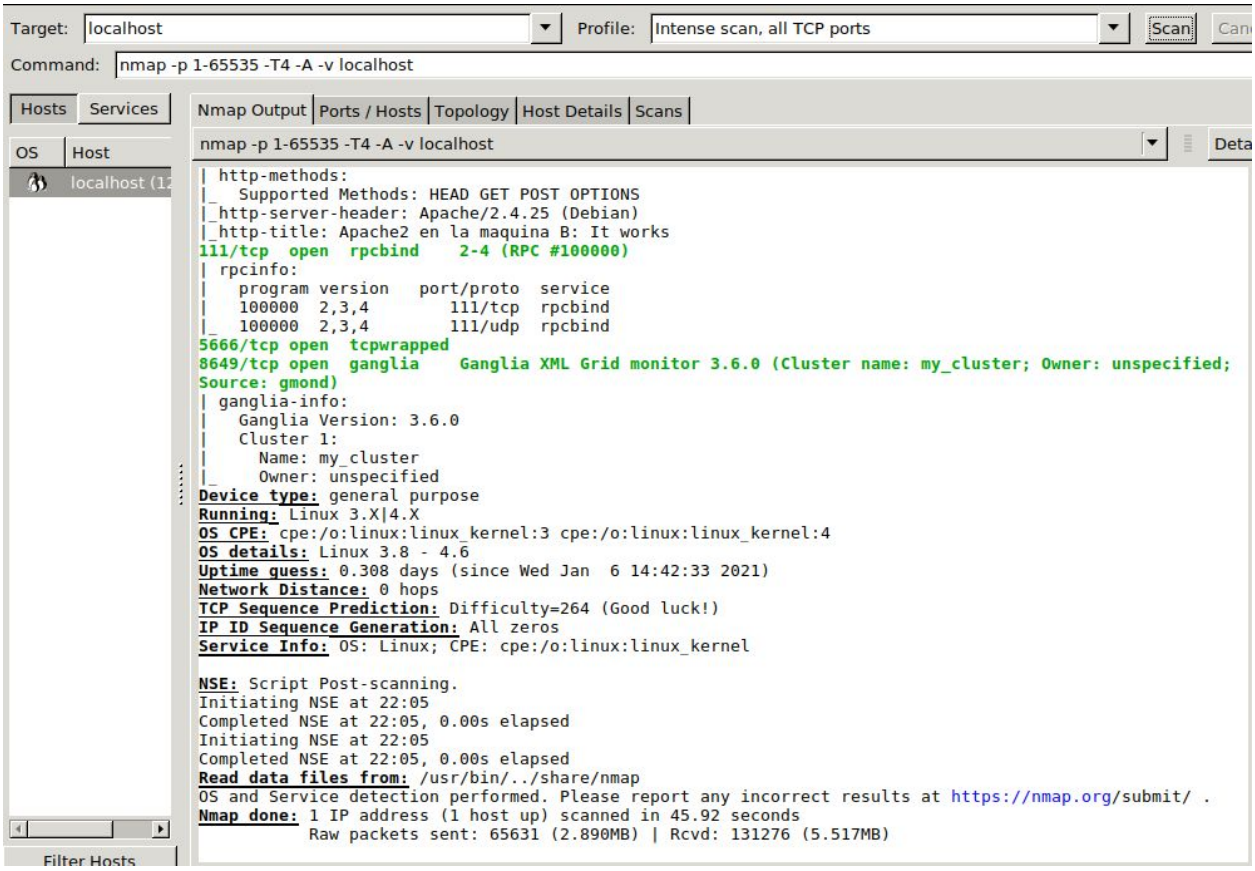
Regular



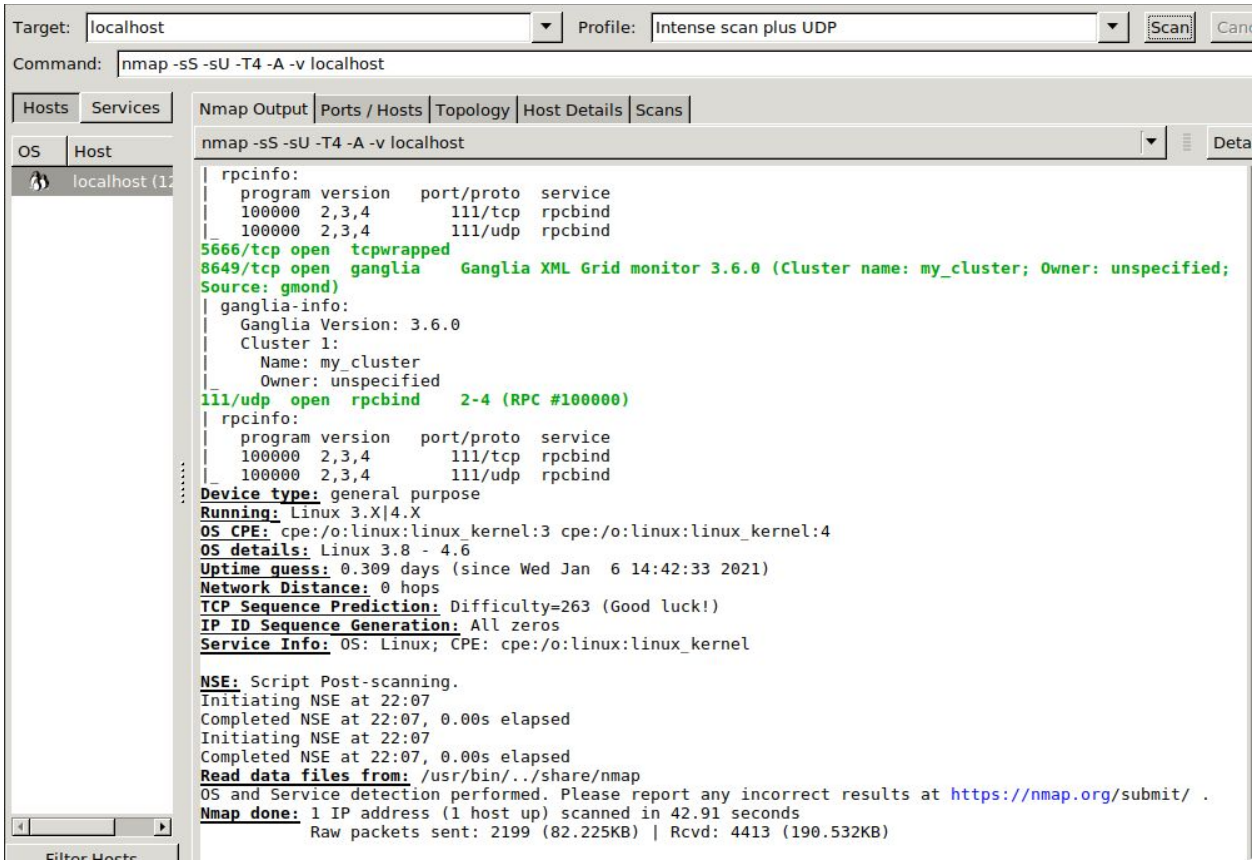
Slow


Target: localhost
Profile: Slow comprehensive scan
Scan
Command: nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 --script "default or (discovery and safe)" localhost
Hosts
Services
OS
Host
localhost (1
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 --script "default or (discovery and safe)" lo...
TCP Sequence Prediction: Difficulty=257 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Host script results:
| fcrdns:
| localhost:
| status: pass
| addresses:
|_ 127.0.0.1
|_ipidseq: All zeros
|_path-mtu: 65535 <= PMTU < 65536
| qscan:
| PORT FAMILY MEAN (us) STDDEV LOSS (%)
| 1 0 74.40 10.92 50.0%
| 22 0 75.67 31.70 40.0%
| 80 0 69.40 24.83 50.0%
| 111 0 78.60 24.47 50.0%
| 111 0 79.00 8.60 50.0%
| 5666 0 91.80 25.88 50.0%
|_8649 0 87.60 22.81 50.0%
| resolveall:
| Host 'localhost' also resolves to:
|_ Use the 'newtargets' script-arg to add the results as targets
NSE: Script Post-scanning.
Initiating NSE at 22:02
Completed NSE at 22:02, 0.00s elapsed
Initiating NSE at 22:02
Completed NSE at 22:02, 0.00s elapsed
Initiating NSE at 22:02
Completed NSE at 22:02, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 70.53 seconds
Raw packets sent: 2277 (216.705KB) | Rcvd: 4239 (183.050KB)

Con TCP

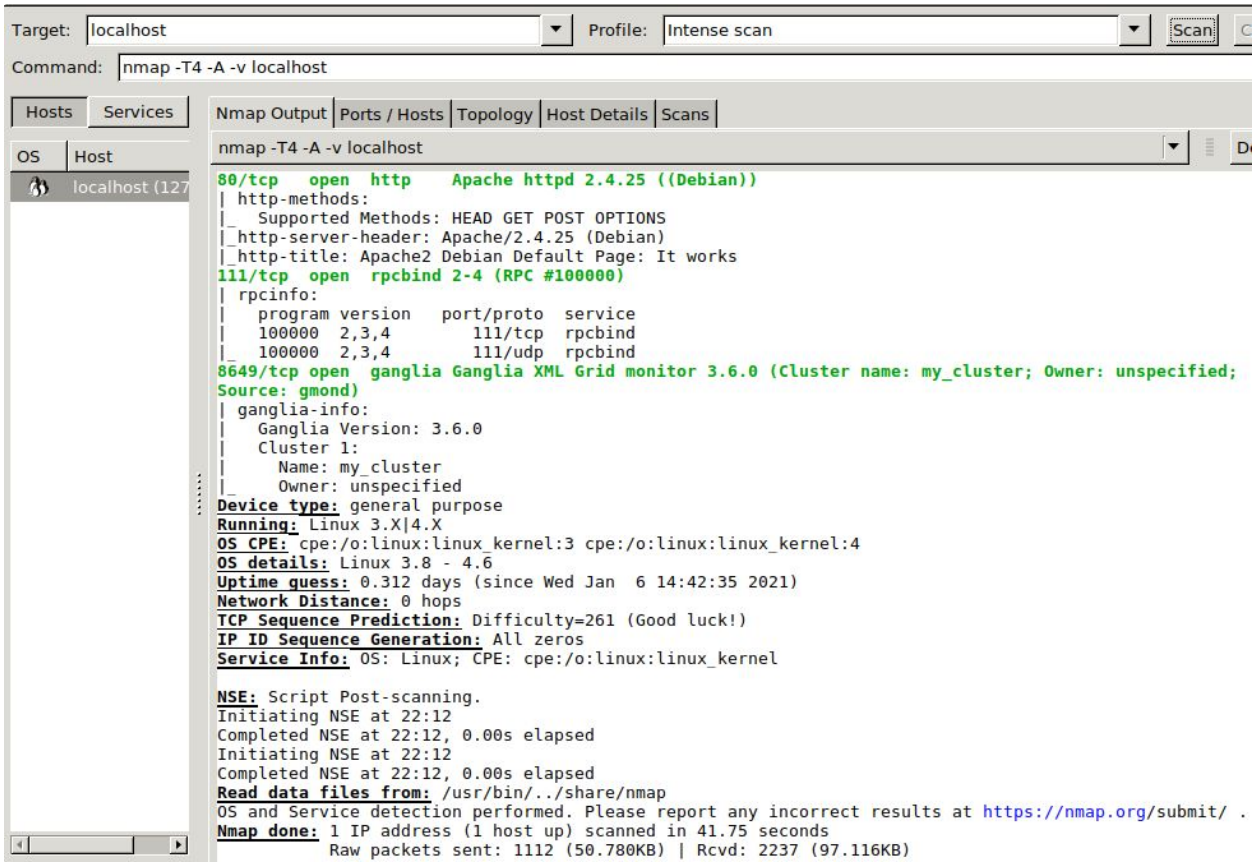

Target: localhost
Profile: Intense scan, all TCP ports
Scan
Command: nmap -p 1-65535 -T4 -A -v localhost
Hosts
Services
OS
Host
localhost (12
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
nmap -p 1-65535 -T4 -A -v localhost
| http-methods:
|_ Supported Methods: HEAD GET POST OPTIONS
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Apache2 en la maquina B: It works
111/tcp open rpcbind 2-4 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2,3,4 111/tcp rpcbind
|_ 100000 2,3,4 111/udp rpcbind
5666/tcp open tcpwrapped
8649/tcp open ganglia Ganglia XML Grid monitor 3.6.0 (Cluster name: my_cluster; Owner: unspecified;
Source: gmond)
| ganglia-info:
| Ganglia Version: 3.6.0
| Cluster 1:
| Name: my_cluster
|_ Owner: unspecified
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.6
Uptime guess: 0.308 days (since Wed Jan 6 14:42:33 2021)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=264 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
NSE: Script Post-scanning.
Initiating NSE at 22:05
Completed NSE at 22:05, 0.00s elapsed
Initiating NSE at 22:05
Completed NSE at 22:05, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.92 seconds
Raw packets sent: 65631 (2.890MB) | Rcvd: 131276 (5.517MB)
Filter Hosts

Con UDP

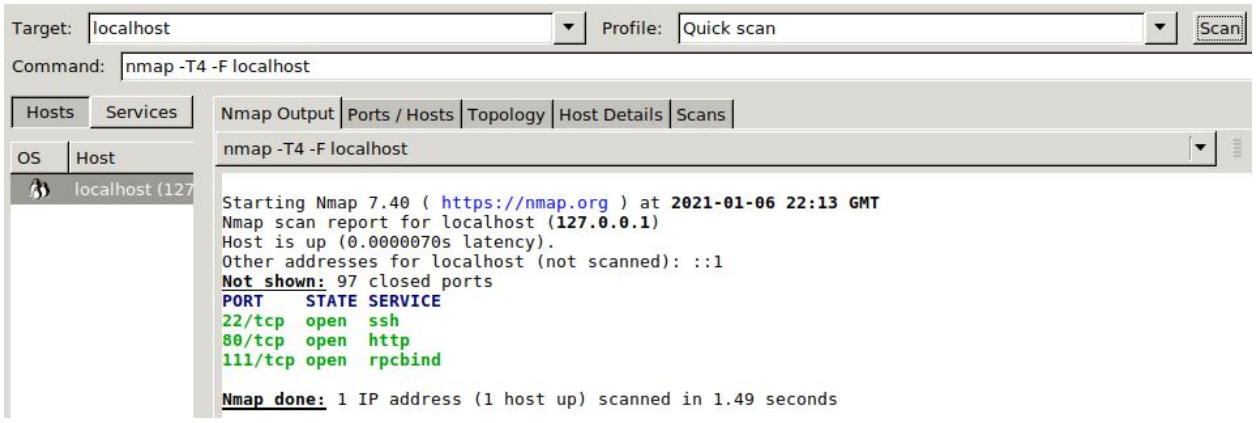

Target: localhost
Profile: Intense scan plus UDP
Scan
Command: nmap -sS -sU -T4 -A -v localhost
Hosts
Services
OS
Host
localhost (1
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
nmap -sS -sU -T4 -A -v localhost
| rpcinfo:
| program version port/proto service
| 100000 2,3,4 111/tcp rpcbind
|_ 100000 2,3,4 111/udp rpcbind
5666/tcp open tcpwrapped
8649/tcp open ganglia Ganglia XML Grid monitor 3.6.0 (Cluster name: my_cluster; Owner: unspecified;
Source: gmond)
| ganglia-info:
| Ganglia Version: 3.6.0
| Cluster 1:
| Name: my_cluster
|_ Owner: unspecified
111/udp open rpcbind 2-4 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2,3,4 111/tcp rpcbind
|_ 100000 2,3,4 111/udp rpcbind
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.6
Uptime guess: 0.309 days (since Wed Jan 6 14:42:33 2021)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=263 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
NSE: Script Post-scanning.
Initiating NSE at 22:07
Completed NSE at 22:07, 0.00s elapsed
Initiating NSE at 22:07
Completed NSE at 22:07, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 42.91 seconds
Raw packets sent: 2199 (82.225KB) | Rcvd: 4413 (190.532KB)

Y ahora repetiremos lo mismo sobre la máquina C.

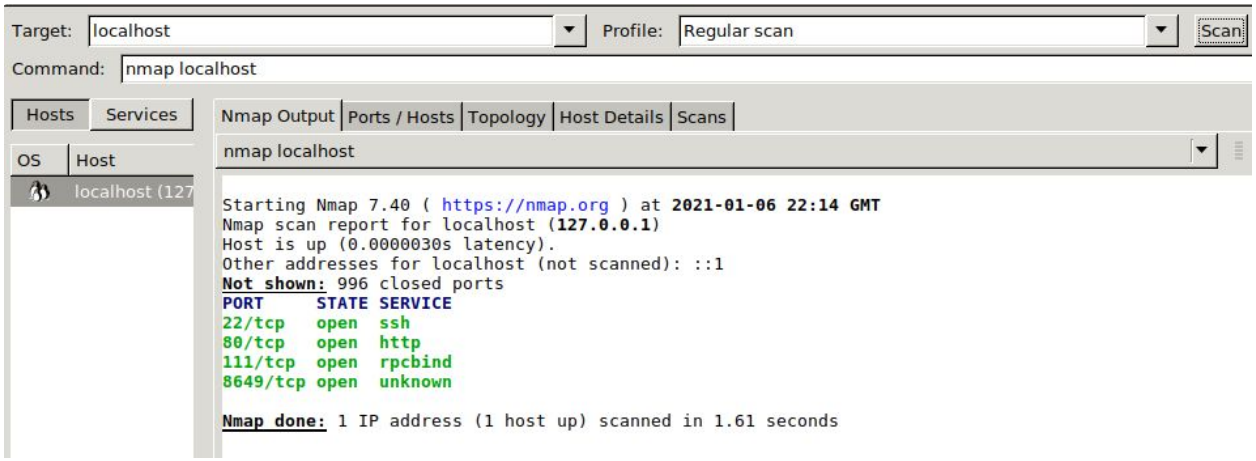
Intense



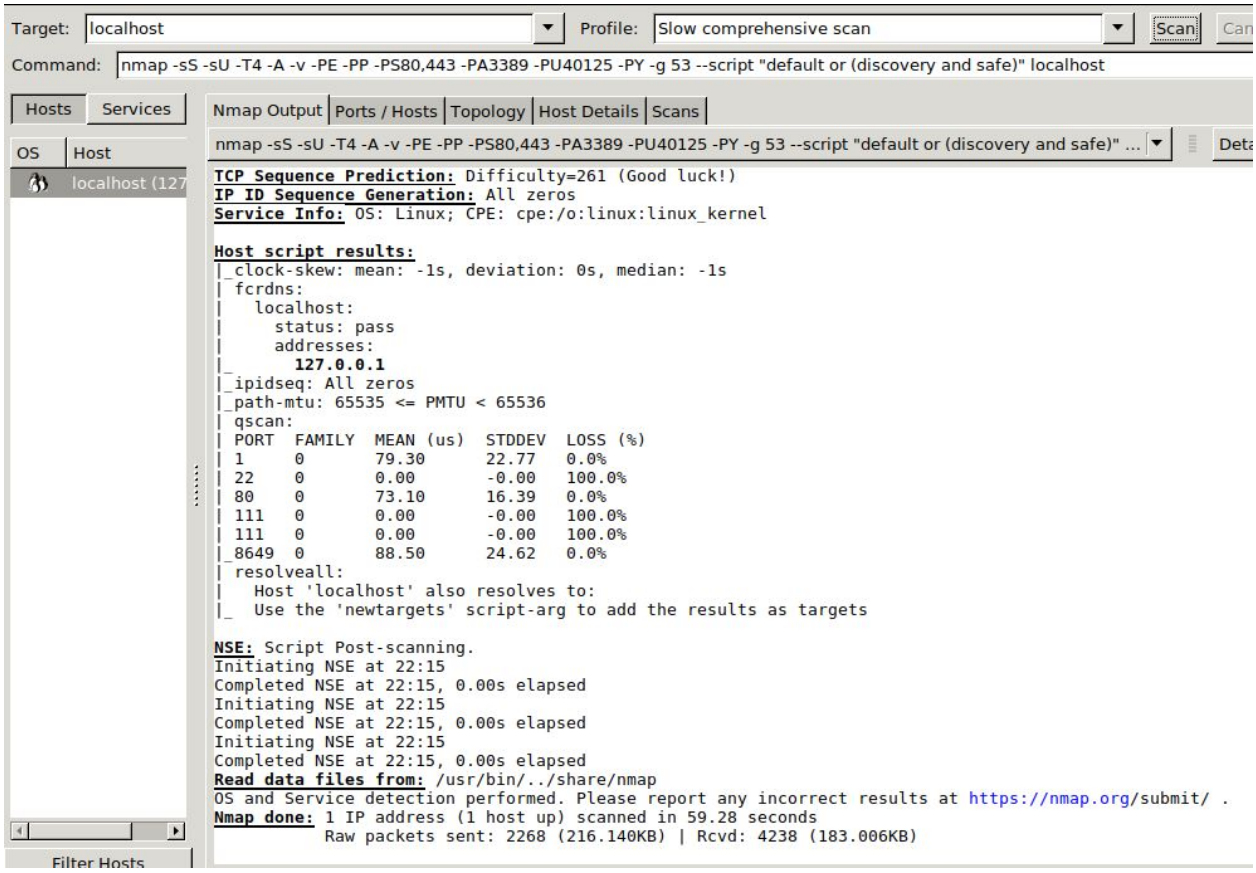
Quick



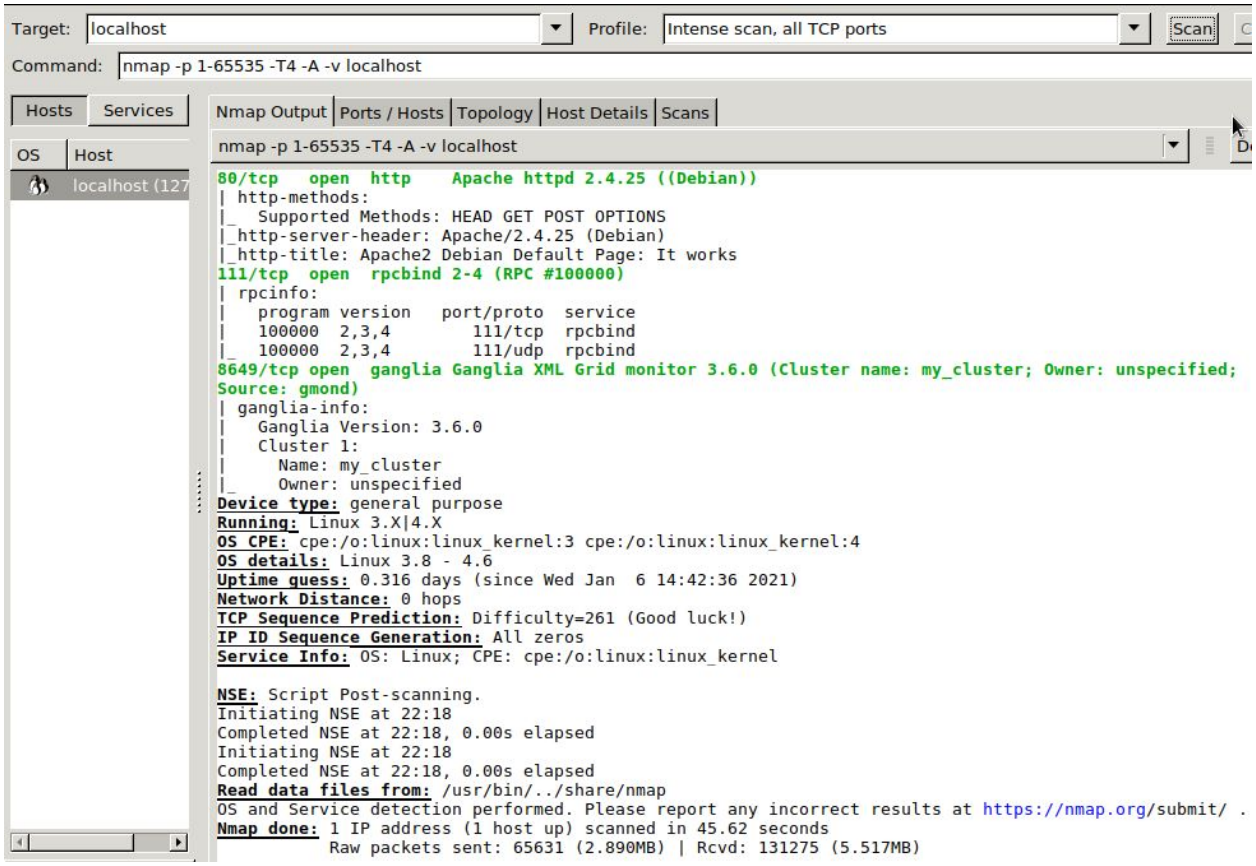
Regular



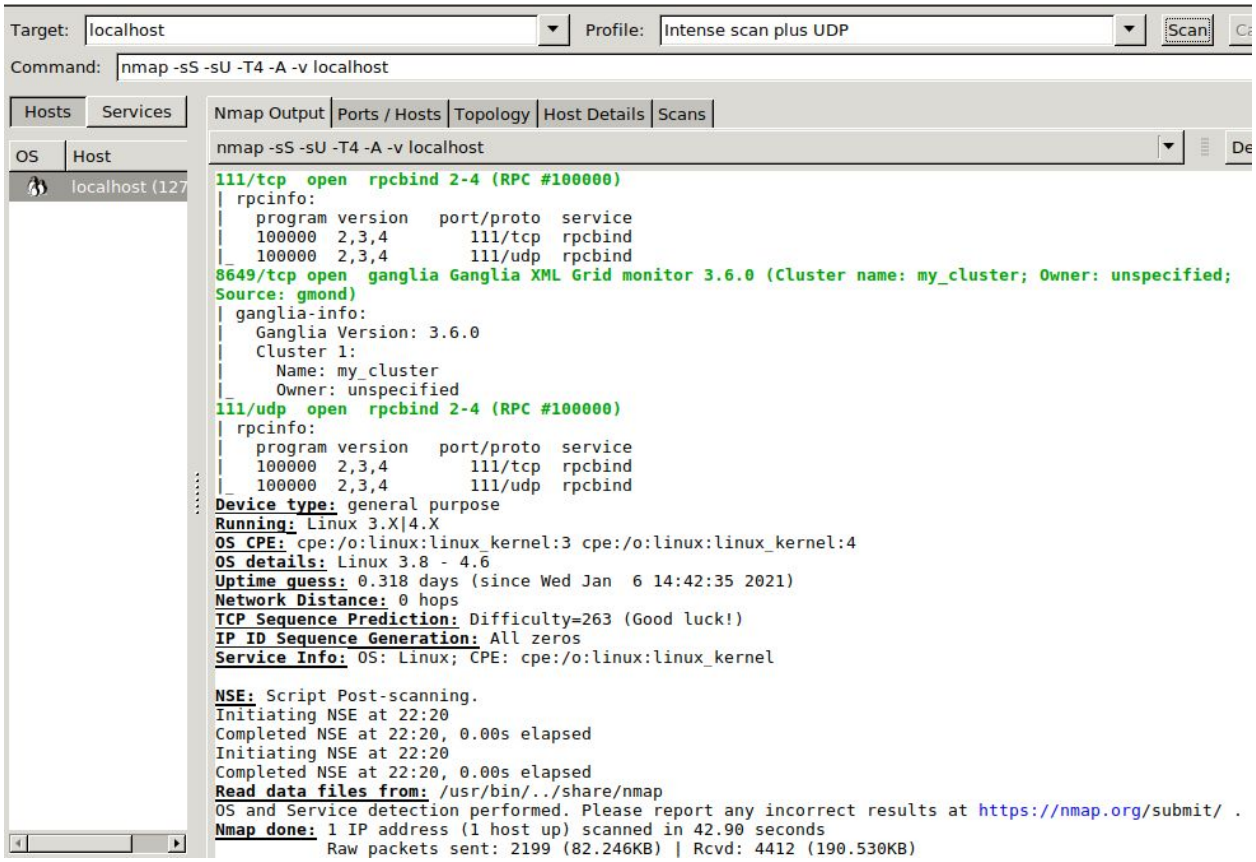
Slow



Con TCP


Target: localhost
Profile: Intense scan, all TCP ports
Scan
Command: nmap -p 1-65535 -T4 -A -v localhost
Hosts
Services
OS
Host
localhost (127
Nmap Output
Ports / Hosts
Topology
Host Details
Scans
nmap -p 1-65535 -T4 -A -v localhost
80/tcp open http Apache httpd 2.4.25 ((Debian))
| http-methods:
|_ Supported Methods: HEAD GET POST OPTIONS
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Apache2 Debian Default Page: It works
111/tcp open rpcbind 2-4 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2,3,4 111/tcp rpcbind
|_ 100000 2,3,4 111/udp rpcbind
8649/tcp open ganglia Ganglia XML Grid monitor 3.6.0 (Cluster name: my_cluster; Owner: unspecified;
Source: gmond)
| ganglia-info:
| Ganglia Version: 3.6.0
| Cluster 1:
| Name: my_cluster
|_ Owner: unspecified
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.6
Uptime guess: 0.316 days (since Wed Jan 6 14:42:36 2021)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
NSE: Script Post-scanning.
Initiating NSE at 22:18
Completed NSE at 22:18, 0.00s elapsed
Initiating NSE at 22:18
Completed NSE at 22:18, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.62 seconds
Raw packets sent: 65631 (2.890MB) | Rcvd: 131275 (5.517MB)

Con UDP



Vemos como los escaneos tipo Quick y Regular suelen ser bastante más cortos y rápidos que los otros si bien nos dan muchos menos detalles que los otros profiles. Son escaneos básicos que sirven para saber si hay algún puerto abierto y con qué servicio. Su única diferencia es en el nombre de puertos escaneados. Los otros tipos de escaneo son mucho más lentos pero más exhaustivos. Se analizan todos los puertos y se detallan mucho más los que están abiertos como por ejemplo dándonos la información sobre la versión del servicio o nombres usados en estos como el nombre del clúster de Ganglia o la página de inicio del Apache.

b) Análisis Zenmap sobre D

Instalamos Zenmap en D y realizaremos el siguiente escaneo. Este comando se basa en usar una máquina zombie como si fuera el atacante que envía un paquete SYN a un puerto y espere a recibir un paquete ACK. En este caso como se puede observar no se ha podido realizar el ataque. Esto puede ser debido a, según la documentación oficial, que la máquina escogida para ser zombie no sirva o pueda ser usada. En esta

documentación se propone usar un comando para escanear máquinas en una red que nos pueda servir de zombie pero al poder suponer un delito, hemos preferido no continuar con esto y tan solo explicar el qué hubiese pasado en un caso de ataque correcto.

```
root@slave3-1-8:~# nmap -P0 -sI 1.1.1.1:1234 10.10.10.53

Starting Nmap 7.40 ( https://nmap.org ) at 2021-01-06 22:34 GMT
Idle scan zombie 1.1.1.1 (1.1.1.1) port 1234 cannot be used because it has not returned any of our probes -- perhaps it is do
wn or firewalled.
QUITTING!
root@slave3-1-8:~#
```

c) PSAD

Instalamos Psad en las máquinas A, B y C y después de la instalación modificaremos el archivo rsyslog.conf para añadir una línea más.

```
kern.info                       |/var/lib/psad/psadfifo
```

Reiniciamos el rsyslog y cambiaremos también el archivo psad.conf para su configuración.

```
### Machine hostname
HOSTNAME                        host;
```

```
HOME_NET                        NOT_USED;
EXTERNAL_NET                    any;
```

```
### define a set of ports to ignore (this is useful particularly
### for port knocking applications since the knock sequence will
### look to psad like a scan).  This variable may be defined as
### a comma-separated list of port numbers or port ranges and
### corresponding protocol,  For example, to have psad ignore all
### tcp in the range 61000-61356 and udp ports 53 and 5000, use:
### IGNORE_PORTS        tcp/61000-61356, udp/53, udp/5000;
IGNORE_PORTS                    udp/53, udp/5000;
```

```
### If "Y", enable automated IDS response (auto manages
### firewall rulesets).
ENABLE_AUTO_IDS                 Y;
```

Reiniciamos psad y añadimos las siguientes reglas a la iptables.

```
root@master-1-8:/etc/psad# iptables -A INPUT -j LOG
root@master-1-8:/etc/psad# iptables -A FORWARD -j LOG
root@master-1-8:/etc/psad#
```

Ahora si usamos el comando psad -S podemos ver un log en el que se nos detallan unos cuantos datos.

```
[+] Version: psad v2.4.3

[+] Top 50 signature matches:
      "ICMP Echo Reply" (icmp),  Count: 1,  Unique sources: 1,  Sid: 408

[+] Top 25 attackers:
      20.20.20.200    DL: 2, Packets: 24, Sig count: 1

[+] Top 20 scanned ports:
      tcp 8649  5 packets
      tcp 22    1 packets

      udp 8649  125 packets
      udp 67    29 packets
      udp 1947  4 packets
      udp 53216 3 packets
      udp 17500 3 packets
      udp 62976 2 packets

[+] iptables log prefix counters:
        [NONE]

    Total protocol packet counters:
        icmp: 1 pkts
         tcp: 6 pkts
         udp: 166 pkts

[+] IP Status Detail:

SRC:  20.20.20.200, DL: 2, Dsts: 2, Pkts: 24, Total protocols: 2, Unique sigs: 1, Email alerts: 2

    DST: 10.10.10.53
        Scanned ports: UDP 8649, Pkts: 23, Chain: INPUT, Intf: ens4
        Total scanned IP protocols: 1, Chain: INPUT, Intf: ens4
    DST: 20.20.20.44
        Total scanned IP protocols: 1, Chain: INPUT, Intf: ens4
        Signature match: "ICMP Echo Reply"
            ICMP, Chain: INPUT, Count: 1, Sid: 408

    Total scan sources: 1
    Total scan destinations: 2

[+] These results are available in: /var/log/psad/status.out
```

Podemos ver como nos muestra un ranking de las diferentes IPs que han realizado algún tipo de ataque a esta máquina. Se puede ver como la IP 20.20.20.200, que corresponde a nuestra máquina C, ha realizado unos cuantos ataques. También se puede ver los puertos que han sido los más escaneados. Esta herramienta nos puede servir bastante cuando se quiere saber cuán vulnerable es un sistema.

d) Labrea

El *honeypot* es una herramienta usada en la seguridad de redes que simula una máquina para que sea el objetivo de ataques para poder detectarlos e identificar al atacante. Este tipo de herramienta entra en lo que son consideradas las defensas activas del sistema, ya que protegen o evitan los ataques mientras estén funcionando.

Instalamos Labrea sobre la máquina D y usaremos el siguiente comando para ponerla en marcha.

```
root@slave3-1-8:/etc/labrea# labrea --switch-safe --verbose -v --no-resp-excluded-ports --log-bandwidth --exclude-resolvable-
ips --foreground --log-to-stdout --max-rate 2000000 --init-file labrea.conf --device ens3 -z --dry-run
```

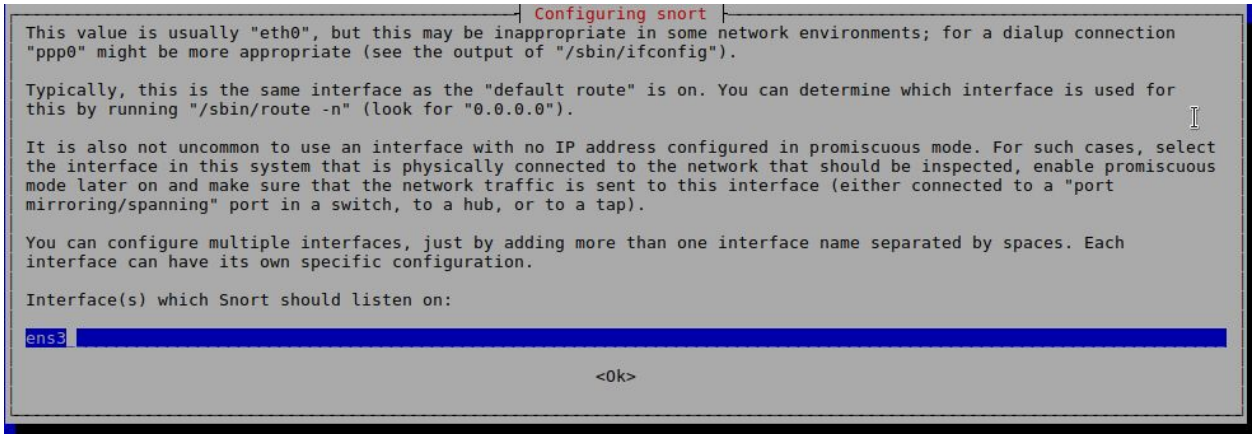
Esto es lo que nos muestra por pantalla al ejecutar el comando anterior.

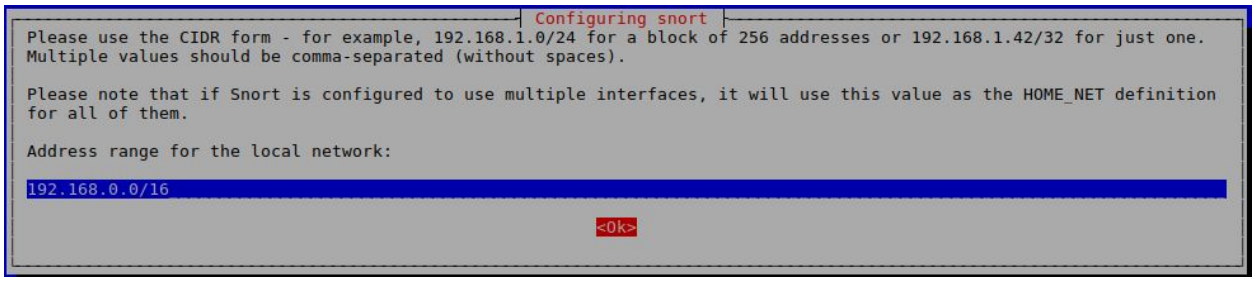
```
Thu Jan  7 22:08:38 2021  LaBrea will attempt to capture unused IPs.
Thu Jan  7 22:08:38 2021  Full internal BPF filter: arp or (ip and ether dst host 00:00:0F:FF:FF:FF)
Thu Jan  7 22:08:38 2021  LaBrea will log to stdout
Thu Jan  7 22:08:38 2021  Logging will be very verbose.
Thu Jan  7 22:08:38 2021  LaBrea will attempt to operate safely in a switched environment
Thu Jan  7 22:08:38 2021  Ports will be firewalled. No RST will be returned for excluded ports.
Thu Jan  7 22:08:38 2021  Connections will be captured in persist state up to 2000000 Kb/sec
Thu Jan  7 22:08:38 2021  Bandwidth use will be logged every minute.
Thu Jan  7 22:08:38 2021  Initiated on interface: ens3
Thu Jan  7 22:08:38 2021  Host system IP addr: 10.10.11.7, MAC addr: 02:00:0a:0a:0b:07
Thu Jan  7 22:08:38 2021  ...Processing configuration file
Thu Jan  7 22:08:38 2021  ... End of configuration file processing

Thu Jan  7 22:08:38 2021  LaBrea will exclude resolvable IP addresses.
Thu Jan  7 22:08:38 2021  IP excluded via DNS: 10.10.10.1
Thu Jan  7 22:08:39 2021  Network number: 10.10.10.0
Thu Jan  7 22:08:39 2021  Netmask: 255.255.254.0
Thu Jan  7 22:08:39 2021  Number of addresses LaBrea will watch for ARPs: 511
Thu Jan  7 22:08:39 2021  Range: 10.10.10.0 - 10.10.11.255
Thu Jan  7 22:08:39 2021  Throttle size set to WIN 3
Thu Jan  7 22:08:39 2021  Rate (-r) set to 3
Thu Jan  7 22:08:39 2021  Test mode run complete... LaBrea is exiting.
Thu Jan  7 22:08:39 2021  Labrea exiting...
Thu Jan  7 22:08:39 2021  100/0 packets (received/dropped) by filter
```

Al intentar realizar un ataque desde A con el Zenmap no parece que cambie nada así que lo más probable es que haya un error en la configuración o estemos ejecutando el comando incorrectamente.

f) Snort

Instalamos Snort sobre la máquina A. Nos aparecerán unos mensajes en los que introduciremos lo siguiente.





Aun con estas configuraciones, no estamos seguros de porque al iniciar snort usando el comando snort -v nos dice que está viendo el tráfico de la interfaz docker_gwbridge en vez de la ens3 que es donde está conectada a internet y de donde llegarian los ataques de D. Y estos, asumiendo que son realizados con éxito y captados por snort, aparecen en la terminal.

```
root@master-1-8:~# snort -v
Running in packet dump mode

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

        --== Initialization Complete ==--

           -*> Snort! <*-
  ,,_      Version 2.9.7.0 GRE (Build 149)
 o"  )~    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 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.8

Commencing packet processing (pid=17255)
```