Fuente: Bing Chat AI

LABORATORIO 7. SEGURIDAD DE RED

DEPARTAMENTO DE INFORMÁTCA. UNIVERSIDAD DE OVIEDO

Seguridad de Sistemas Informáticos | 2023 – 2024 (V4.0 *"S-81 Isaac Peral"*)

Contenido

Infraestructura de este laboratorio

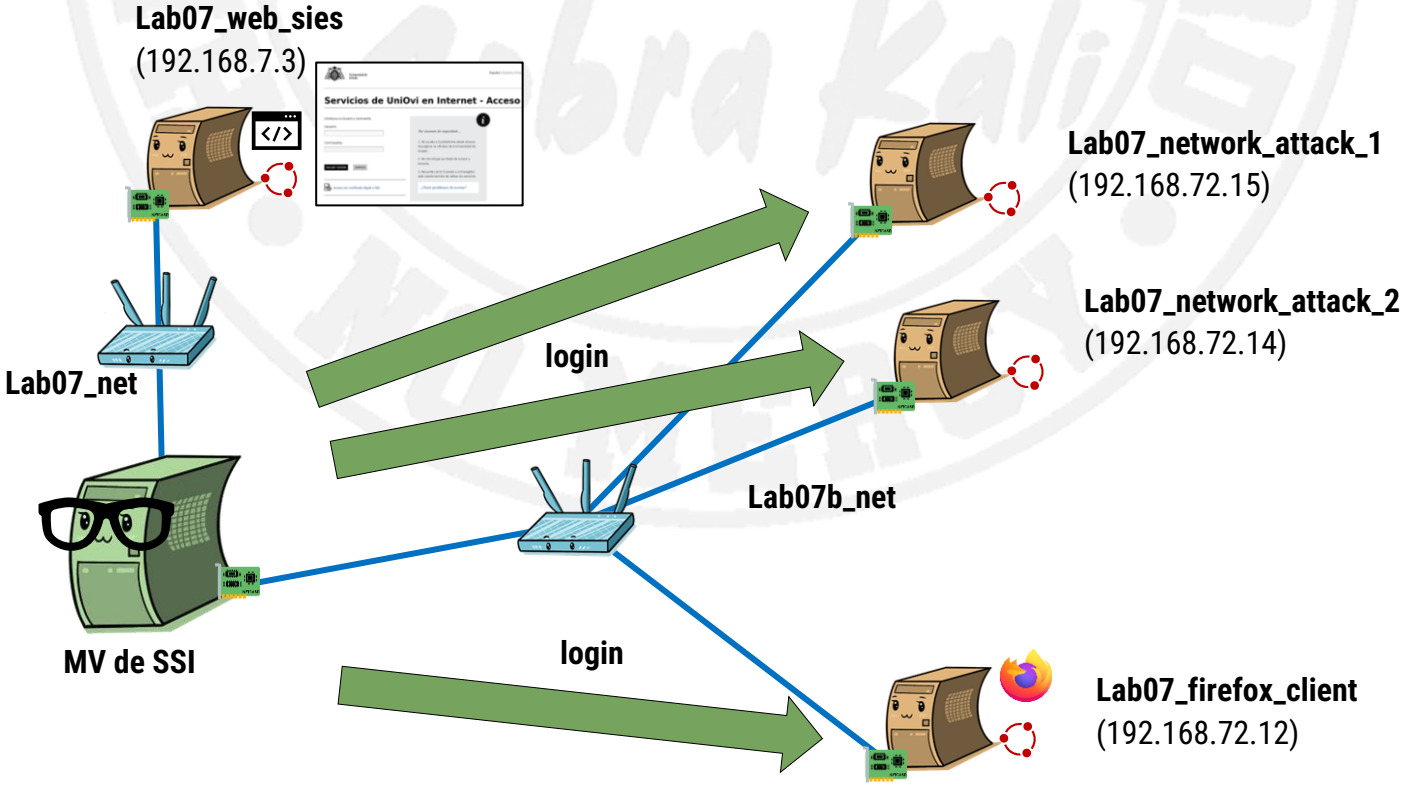
*(**NOTA:** Para evitar problemas, te recomendamos encarecidamente que ejecutes esta infraestructura de laboratorio **en la máquina virtual base en lugar de en sus versiones con hardening** que puedas haber creado en laboratorios anteriores. En la vida real, deberías hacer lo contrario, pero las implementaciones reales también tienen un tiempo de validación para garantizar que todo funcione como se espera después del hardening.*

***AVISO**: Algunos de vosotros habéis tenido problemas ejecutando Firefox desde un contenedor, algo que haremos en este laboratorio. Para evitarlos si te ocurren, sigue estos pasos:*

- *Abre el fichero* `build_lab07.sh`
- *Añádele la siguiente línea:* `xhost + local:root`
- *Construye el laboratorio como otras veces*

La infraestructura de este laboratorio está compuesta por contenedores comunicados a través de redes internas con la MV. La mayoría se usará de forma interactiva.

- Dos contenedores son contenedores de "ataque de red" y cuentan con las herramientas para probar el *firewall* y otro software de protección que vamos a ver en este laboratorio. La información que se escriba en el directorio `/etc/sysctl.d` se escribirá también en el directorio de la máquina virtual `volume_data/network_data_<N>/sysctl.d/`
- Otro contenedor está habilitado para ejecutar aplicaciones GUI y tiene *Firefox* instalado.
- Finalmente, el último contenedor es un servidor web con la página *front-end* de la intranet de Uniovi comunicada con la MV con una red privada especial (`192.168.7.0/24`) y con IP fija.



🔍 ***NOTA:*** ***Algunas de las operaciones de este laboratorio requieren conocer la dirección IP de los contenedores y las redes que están utilizando. Por favor, recuerda usar*** `ifconfig` ***dentro de los contenedores para conocer sus direcciones IP actuales***

Estos contenedores tienen instalado el *software* necesario para realizar las actividades del laboratorio. Sin embargo, varios de los productos que estamos a punto de utilizar son incompatibles con contenedores y, por lo tanto, **deben instalarse en la máquina virtual *Ubuntu*.** En este caso, los contenedores actuarán como "clientes atacantes" de la MV, por lo que podrás probar que funcionan sin necesidad de usar más elementos externos. Qué software está dirigido a los contenedores y cuál se va a instalar en las máquinas virtuales se indica claramente en cada actividad.

Por otro lado, también **tendrás que usar la "máquina *Wazuh*"** del laboratorio anterior para hacer un ejercicio. Recuerda que si no tienes en casa una máquina capaz de soportar la creación de una MV con estas características, te recomendamos que le **des prioridad a la actividad de *Wazuh* para hacerla en el laboratorio**, donde sí tienes hardware que soporte esa MV. Puedes incluso empezar por ella sin problemas.

🔍 ***NOTA****: Se ha detectado que a veces el build del laboratorio falla porque aparentemente el firewall arranca antes que el Docker y se produce un conflicto. Si es tu caso, haz esto para solucionarlo:*

- ***Para el firewall***
- ***Para el servicio de Docker***
- ***Arranca el servicio de Docker***
- ***Arranca finalmente el firewall***

Por otro lado, a veces el GUI del firewall no muestra correctamente la información más actual del estado de la red de la máquina. Esta información sí aparece si se maneja el firewall desde la línea de comandos, con el comando `firewall-cmd -list-all-zones`

Bloque 1: Protección de redes de un host. *Firewalls* PF

Ejercicio L7B1_FWZONES: Administración de zonas con *firewalld*

Descripción de la actividad / Aplicación práctica

Necesitas **separar tus interfaces de red en zonas** con `firewalld` para poder asignarles diferentes configuraciones

Resultados esperados

Esta actividad finalizará cuando se puedan **asignar zonas a las tarjetas de red en el *firewall*** y activar y probar que se puede acceder a un servicio en la zona "`work`".

Otra información necesaria para su realización

Esta actividad te enseñará cómo usar `firewalld` para realizar operaciones de seguridad de red típicas que ilustren partes de la SNI (*Secure Network Infrastructure)* de la teoría.

🔍 ***`firewalld` es un producto más potente que el `ufw` tradicional que viene con cada distribución de Ubuntu. Sirve tanto para distribuciones Debian como RedHat y además lo usáis en ASR.***

`firewalld` cumple con las mismas funcionalidades (y más) que `ufw`. No obstante, tener ambos habilitados a la vez causa problemas, por lo que primero **debemos asegurarnos de que `ufw` esté deshabilitado**: `sudo ufw disable`.

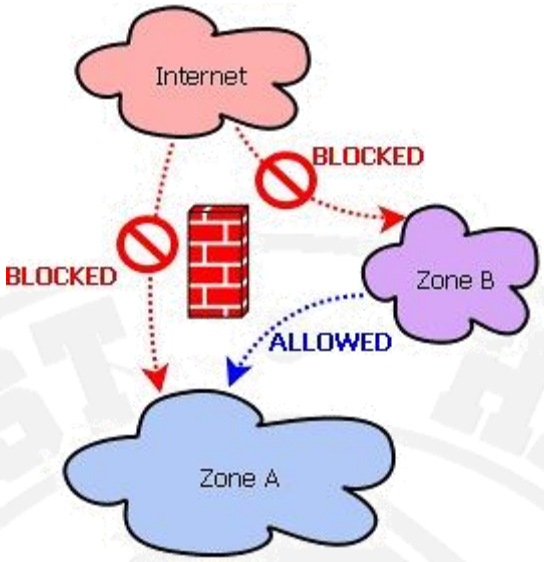
Una vez hecho esto, necesitamos instalar el *software* adecuado para usar y gestionar `firewalld`, consistente en el propio servicio de *firewall* y su GUI (`firewall-config`). Puedes instalar ambos de esta manera: `sudo apt install firewalld firewall-config`.

🔍 ***Una vez hecho esto, es muy importante reiniciar la máquina virtual.***

Una vez que la VM se haya reiniciado podremos trabajar con `firewalld`. Lo primero que debemos entender de este *firewall* es que distribuye las diferentes tarjetas de red que tengamos en la MV en **zonas.** Las zonas son conjuntos predefinidos de reglas que especifican qué tráfico se debe permitir en función del nivel de confianza en las redes a las que está conectado el equipo.

Puedes asignar cualquiera de las interfaces de red de la máquina que tiene `firewalld` a una zona. A partir de entonces, todas las máquinas conectadas a la misma red que cada una de esas tarjetas de red (asumiendo que mandan todo su tráfico a otras redes a través de dicha tarjeta, es decir, que la maquina con `firewalld` **actúa como puerta de enlace entre redes**), pertenecerán a dicha zona. Podrás así distribuir las máquinas por zonas, formando algo similar a lo que vimos en teoría con el SNI o este diagrama, que es una versión más sencilla.

🔍 ***En ambos casos se cumple un principio: el firewall hace de "policía" e "inspector de aduanas" de toda las conexiones, poniéndose siempre "en medio"***



Fuente: https://akm111.wordpress.com/2017/04/29/linux-firewalld-zones-and-services

A continuación se muestran las zonas que incorpora por defecto `firewalld`:

- **docker**: Una zona especial que inicialmente contiene todas las redes creadas por nuestros contenedores *Docker*. La tenemos porque se crea cuando se instala *Docker* en el sistema.
- **Zonas donde las conexiones entrantes están prohibidas y solo se permiten conexiones salientes**
  - **drop**: Todas las conexiones entrantes se eliminan sin ninguna notificación
  - **block**: Todas las conexiones entrantes son rechazadas (se notifica al cliente)
- **Zonas donde se permiten conexiones entrantes que previamente autoricemos nosotros.** Como puede verse, están pensadas para diferentes situaciones y redes donde se puede encontrar la máquina que tiene `firewalld` instalado
  - **public:** Para uso en **áreas públicas** no confiables. No confía en otros equipos de la red.
  - **external:** Para uso en **redes externas** con enmascaramiento NAT habilitado, cuando el sistema actúa como puerta de enlace o enrutador.
  - **internal:** Para uso en **redes internas**, cuando el sistema actúa como puerta de enlace o enrutador. Los otros sistemas de la red son generalmente de confianza.
  - **dmz**: Se utiliza para equipos ubicados en una **zona desmilitarizada o DMZ** (**Tema 5 de teoría**), que tendrán acceso limitado al resto de su red.
  - **work**: Utilizado para **máquinas del trabajo**. Otros equipos de la red son generalmente de confianza.
  - **home**: Utilizado para **máquinas domésticas**. Otros equipos de la red son generalmente de confianza.
- **Zonas donde no se restringe ninguna conexión**
  - **trusted**: Se aceptan todas las conexiones de red. Confía en todos los equipos de la red.

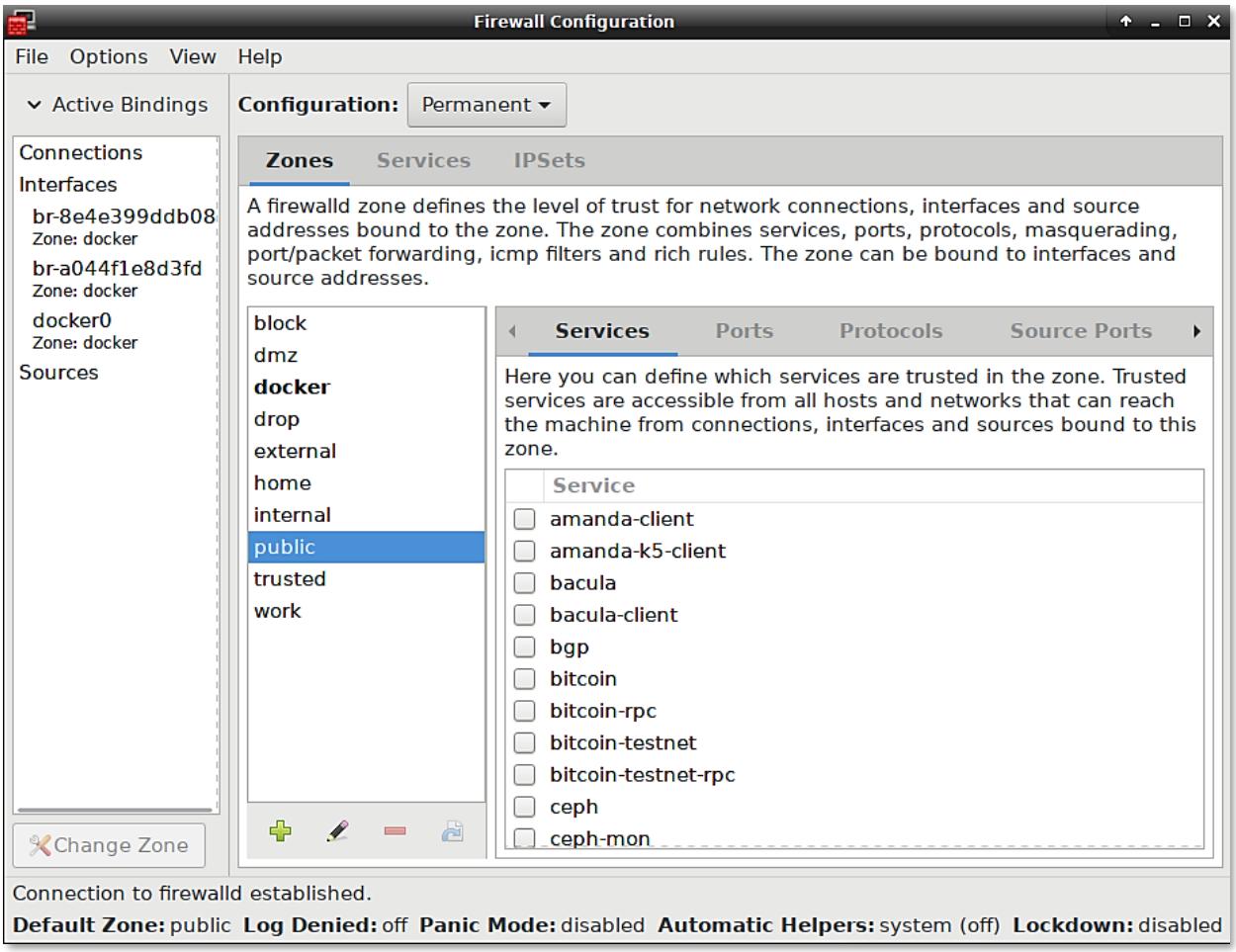
No tienes que entender todas estas zonas, pero necesitas saber a cambiar las interfaces entre ellas.

🔍 ***La parte buena de usar zonas es que una vez que habilitas un servicio o puerto para una zona, ¡lo habilitas para todas las interfaces que pertenecen a esa zona al mismo tiempo!***

Vamos a usar el GUI *Firewall Config* para administrar la mayoría de las operaciones del *firewall*, pero también puedes hacer las mismas cosas a través de la línea de comandos:

- https://computingforgeeks.com/install-and-use-firewalld-on-ubuntu/
- https://www.supportsages.com/everything-you-need-to-know-about-firewalld/

Para hacer eso, primero ejecuta `sudo firewall-config` (recuerda usar `sudo`, por favor, o te dará un error) y cambia el *ComboBox* de ***Configuration*** a ***Permanent***, como se muestra en esta imagen:



Desafortunadamente, la mayoría de vosotros verá que las tarjetas de red `eth0` y `eth1` no están asignadas a ninguna zona y no aparecen en la GUI. Podemos arreglar esto usando la línea de comandos, haciendo que aparezcan estas tarjetas de red, y pudiendo manipularlas usando la GUI a partir de ahora. Haz las siguientes operaciones para eso:

- Agregar `eth0` (la tarjeta que permite que la máquina virtual se conecte a Internet) a la zona "*public*": `sudo firewall-cmd --zone=public --change-interface=eth0`
- Agregar `eth1` (la tarjeta que permite que la máquina virtual se conecte a tu PC) a la zona "*work*": `sudo firewall-cmd --zone=work --change-interface=eth1`

Una vez hecho esto, asegúrate de que la zona `work` tenga habilitado el servicio `ssh` y prueba que `ssh` está funcionando desde tu PC.

Ejercicio L7B1_FWBLOCK: Bloqueo de direcciones IP y redes con *firewalld*

Descripción de la actividad / Aplicación práctica

Necesitas **bloquear IPs individuales o redes completas** para que no puedan acceder a tu máquina

Resultados esperados

Esta actividad finalizará cuando puedas **bloquear una IP o red** para acceder a un servicio en una zona de *firewall*.

Otra información necesaria para su realización

Uno de los usos más comunes de un *firewall* es permitir/bloquear el acceso desde ciertas IPs o redes a otras IPs y redes. Para practicar cómo hacer esto, sigue este procedimiento:

- Asegúrate de que puedes acceder desde tu PC a la máquina virtual a través de `ssh`
- Vete a la zona de "`work`" en la GUI del *firewall* y busca en la pestaña ***"Rich Rules"***.
- **Cree una nueva regla enriquecida** que **elimine (drop)** todas las conexiones al servicio `ssh` desde la IP `192.168.56.1` (normalmente la IP de tu PC en la red *host-only*) a la IP `192.168.56.10` (la IP de tu máquina virtual en la red *host-only*). Comprueba además todas las opciones que tiene la creación de reglas y piensa en sus posibilidades, especialmente las opciones de *limit, log* y *audit*.
- Prueba que ahora ya no puedes conectarte a través de `ssh`
- Modifica la regla anterior para bloquear toda la red `192.168.56.0/24`
- Prueba que aún no puedes conectarte a través de `ssh`
- Elimina la regla y prueba que ahora puede conectarte de nuevo.

Ejercicio L7B1_FWFORWARD: *Reenvío de puertos en firewalld*

Descripción de la actividad / Aplicación práctica

Necesitas **"esconder" dónde está realmente un servicio** (la IP y puerto de su máquina servidora real) redirigiendo su acceso a través de un *firewall*

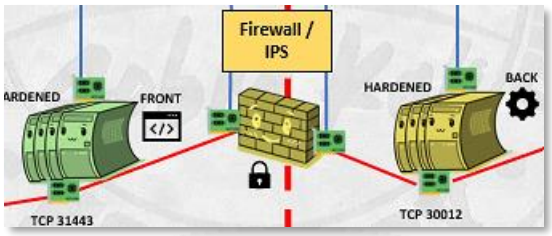
Resultados esperados

Esta actividad finalizará cuando puedas **comunicar un contenedor en la zona pública con un contenedor en la zona DMZ** para que se pueda acceder a un *front-end* web colocado en la DMZ desde la zona pública mediante el reenvío de puertos. Puedes contestar a estas preguntas:

- *¿Entiendes ahora como funciona en la práctica el aislamiento de redes y el papel que juega el firewall en él?*
- *¿Qué tendría que hacer para poder contactar directamente con la máquina que ofrece realmente el servicio si soy un atacante?*

Otra información necesaria para su realización

El objetivo de este ejercicio es emular la siguiente parte del SNI:



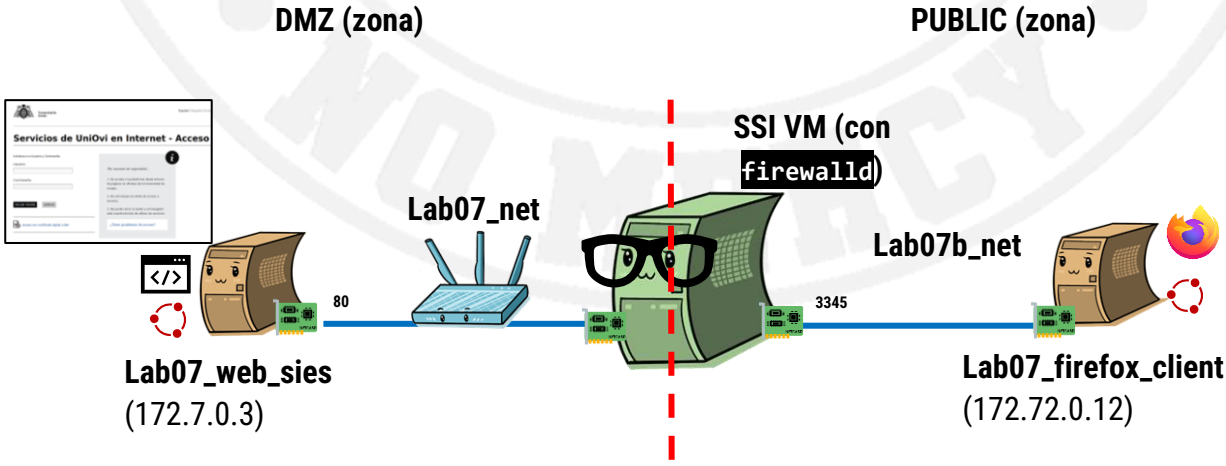
Tenemos dos máquinas situadas en diferentes zonas de red y, por lo tanto, no pueden comunicarse directamente. Entre ambas zonas hay un *firewall* que puede conectarse a ambas, y podemos indicar al *firewall* para que redirija una conexión realizada a uno de sus puertos en una de las zonas a otra máquina que está en una zona diferente: esto se llama **reenvío de puertos** (***port forwarding***).

> 🔍 ***La máquina en una de las zonas no sabrá quién es la máquina que atiende su solicitud en la otra zona, y la máquina que da el servicio nunca tendrá contacto directo con sus clientes: el firewall aísla a ambas.***

Para emular esto, vamos a utilizar los siguientes elementos de nuestra **infraestructura del Lab 7** (ver diagrama):

- Entra en sesión en el contenedor `lab07_firefox_client` y anota su IP (`ifconfig`)
- Vete a la VM y localiza la tarjeta que está conectada a la red que usa el contenedor de *Firefox* buscando la que tiene una IP compatible con la que anotaste en el primer paso. Su nombre debería ser algo como `br-5c8ea19b05fb`
- Usando la GUI del *firewall*, coloque esta tarjeta de red en la zona "*public*".
- Ahora, localiza la tarjeta de red que sirve al contenedor `lab07_web_sies` (la que tiene una IP en la red `172.7.0.0/16`), que debería ser `172.7.0.3`.
- Coloca esta tarjeta de red en la zona " *dmz*".

Este diagrama representa la estructura para este ejercicio:



Ahora has configurado dos contenedores en diferentes zonas y el *firewall* (tu máquina virtual) en el medio. Para poder comunicarlos mediante el reenvío de puertos, haz lo siguiente:

- Vete a la zona "*pública*" en la GUI y busca la pestaña "***Port Forwarding***" (dale a la flecha derecha para ver más pestañas)
- Crea una nueva regla de reenvío de puertos que redirija todas las conexiones al puerto `3345` al puerto `80` de la IP `172.7.0.3` (el contenedor `lab07_web_sies`). Si el firewall te pide que habilites el enmascaramiento, responde *Yes*.
- **Vuelve a cargar el *firewall* (*Options – Reload firewall*).**

Para probar esto, puedes ejecutar *Firefox* en la línea de comandos *bash* del contenedor `lab07_firefox_client`. La primera vez probablemente dará un error, **pero si lo reinicias, funcionará.** Ahora tienes un *Firefox* aislado de contenedor en la zona "pública". Navega a la IP de la máquina virtual en esta zona (si la IP del contenedor es `172.22.0.2`, la IP de la máquina virtual en esta red usa la misma red, pero termina en `1`) pero al puerto `3345` (Ejemplo: `172.22.0.1:3345`).

🔍 ***Si todo está bien, deberías ver esta página web, que es justo la que tiene alojada la máquina "del otro lado" del firewall, pero...¡ni sabes su IP ni el puerto real al que estás accediendo!***

Con esto, has conectado una máquina en una red con una máquina a otra red a través de un *firewall*. Piensa en lo que esto significa con respecto a la seguridad (quién puede acceder a la segunda máquina, qué servicios se están exponiendo ...) y cómo podrías replicar esto para conectar un *front-end* de una aplicación con su *backend* si ambos se colocan en diferentes zonas de red.

Bloque 2: Protección de red en conexiones salientes y entrantes

Ejercicio L7B2_DNSPI: Protección de conexiones salientes: Filtrado de DNS con *PiHole*

Descripción de la actividad / Aplicación práctica

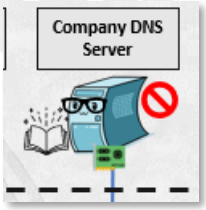
Necesitas **bloquear conexiones a dominios maliciosos** conocidos usando un DNS local que filtre las peticiones que tu máquina hace a dominios externos

Resultados esperados

Esta actividad finalizará cuando puedas **configurar el servidor de filtrado DNS *PiHole*** para probar cómo cambia su navegación una vez que uses sus funciones. Puedes contestar a las preguntas que se hacen a lo largo del ejercicio y esta otra: *¿Cuál crees que es la diferencia más significativa entre PiHole y el NextDNS que vimos en el **Laboratorio 1**?*

Otra información necesaria para su realización

Te recomendamos encarecidamente que crees una instantánea de la máquina virtual antes de realizar este ejercicio. El objetivo de este ejercicio es mostrarte la importancia de controlar tu propio servidor DNS para filtrar el tráfico saliente potencialmente malicioso. Esto corresponde a esta parte de la teoría *Secure Network Infrastructure* (SNI), **pero también lo puedes usar en casa** 😊 .



Haremos esto configurando un producto profesional, *PiHole* (https://pi-hole.net/), que te permite tener un servidor DNS que además filtra cualquier solicitud que reciba utilizando un enfoque ***blocklist***: cualquier conexión realizada a una IP presente en la lista de bloqueo se anulará.

PiHole permite la instalación en muchos sistemas, pero para este laboratorio lo hemos empaquetado en un contenedor. Vete a la carpeta `pihole` de los archivos de laboratorio y ejecuta el script `run_pihole.sh` (por favor, asegúrate de que *Apache 2* no esté instalado en la máquina virtual y haz `sudo apt-get remove apache2` si lo está). Una vez termine de ejecutarse el comando, navega a la URL `127.0.0.1/admin` en el navegador de la VM para ver la pantalla de estadísticas de bienvenida de *PiHole*. La contraseña de administrador es `test123...`

Te recomendamos que navegues ahora a cualquier página web con publicidad (como www.elmundo.es) y dejes esa web abierta. Ahora que *PiHole* está funcionando, para empezar a filtrar conexiones solo tenemos que cambiar el DNS de nuestra tarjeta de red `eth0` (la que se conecta a Internet) a `127.0.0.1` (*PiHole* se

ejecuta localmente). No te preocupes, no perderás la conexión a Internet, *PiHole* sabe qué hacer 😊. La siguiente imagen muestra los cambios que debes realizar para esto, **aunque también tendrás que cambiarlo en el fichero `/etc/resolv.conf`**:

```
 GNU nano 2.9.3              /etc/netplan/01-netcfg.yaml              Modified

1 network:
2   version: 2
3   ethernets:
4     eth0:
5       dhcp4: true
6       nameservers:
7                 addresses: [127.0.0.1]
8
```

Vuelve a cargar ahora el sitio web anterior. *¿Ves alguna diferencia? ¿Qué pasará ahora con cualquier dispositivo que utilice este servidor DNS?*

🛠 Ejercicio L7B2_DNSPIPLUS: Filtrado avanzado de conexiones salientes con *PiHole*

Descripción de la actividad / Aplicación práctica

Necesitas bloquear conexiones a aún más dominios maliciosos, **potenciando las capacidades de *PiHole***

Resultados esperados

Esta actividad finalizará cuando puedas **mejorar las listas de bloqueo de *PiHole*** para bloquear las conexiones a más tipos de sitios web maliciosos.

Otra información necesaria para su realización

Aunque *PiHole* filtra un gran número de sitios web maliciosos de serie, podemos darle más "potencia" utilizando las URL maliciosas recopiladas por ***The Block List Project*** (https://blocklistproject.github.io/Lists/). Sigue estas instrucciones para incorporar esta lista de IPs a *PiHole*:

- Copia el enlace de la lista más completa (***Everything***) en formato compatible con *PiHole* (https://blocklistproject.github.io/Lists/everything.txt).
- Agrega la URL a las listas de bloqueo de tu *PiHole* (***Login - Group Management-Adlists -Pega la URL de la lista en el campo "Dirección", agrega un comentario - Haz clic en "Add"***)
- Actualiza ***Gravity*** (***Tools-Update Gravity-Click "Update"***) para incorporar las nuevas URL a la lista de bloqueo. El proceso lleva un tiempo, **¡por favor espera y no navegues fuera de esta página!**

Navega de nuevo a cualquier web con la página web de estadísticas *de PiHole* abierta y observa cómo están ahora en funcionamiento listas de bloqueo mucho más grandes.

Ejercicio L7B2_FAIL2BAN: Protección de intentos de intrusión: *Fail2Ban*

Descripción de la actividad / Aplicación práctica

Necesitas **bloquear las conexiones de las máquinas** que tratan de acceder a tu servicio SSH sin éxito múltiples veces

Resultados esperados

Esta actividad finalizará cuando puedas proteger `ssh` utilizando el modo de protección agresivo de `fail2ban`. También puedes verificar que la protección funciona y revocar los bloqueos de IP creados por este programa. Puedes contestar a esta pregunta: *¿Qué tipo de ataque sobre SSH crees que bloquea esta herramienta especialmente?*

Otra información necesaria para su realización

Fail2Ban es una aplicación que **procesa los *logs* del sistema** para buscar indicios de ataques. Cuando se detectan, de acuerdo con las reglas que se le definan, `fail2ban` agrega una nueva regla al *firewall* que bloquea la IP desde la que se hizo el intento de ataque identificado. Esta regla bloquea la IP del atacante de forma temporal o permanente, dependiendo de la configuración.

🔍 ***También se puede notificar por correo electrónico a un usuario que un ataque está en marcha (instalando `sendmail`, pero no lo usaremos en este laboratorio).***

`fail2ban` se usa típicamente para **proteger** `ssh` pero tiene varias reglas predefinidas por defecto (llamadas ***jails*) que también funcionan con otros servicios, como *Apache 2 o Nginx* (servidores web).**

🔍 ***Técnicamente, cualquier servicio de red que genere un log puede ser protegido por `fail2ban` siempre que tenga configuradas las reglas adecuadas. Para hacer esto puedes crear tus propias reglas, pero eso excede el propósito de este laboratorio.***

Este programa no funciona dentro de contenedores, por lo que debes **instalarlo en tu máquina virtual.**

Protección de SSH mediante *Fail2Ban*

El uso de `fail2ban` con SSH requiere seguir estos pasos:

- Instálalo (el servicio se inicia automáticamente una vez instalado): `sudo apt install fail2ban`
- Asegúrate de que los servicios a proteger por `fail2ban` (SSH, HTTP, ...) no estén bloqueados por el *firewall* (¡no habría nada que proteger entonces! 😊).
- Modifica la configuración de `fail2ban` (archivo `/etc/fail2ban/fail2ban.conf`). Es **MUY** recomendable copiar este archivo a `fail2ban.local` y editar este nuevo archivo en lugar del `.conf` original. En este archivo puedes configurar cosas como el nivel de detalle del *log* (`loglevel`), dónde escribir la información de log (`logtarget`), etc. Usaremos la configuración predeterminada en este laboratorio.
- Modifica el archivo `/etc/fail2ban/jail.local` para habilitar o deshabilitar las *jails*. Para hacer eso, haz lo siguiente:
  - Si el archivo `jail.local` no existe, créalo copiando `jail.conf`

- **Al principio, solo la *jail* `sshd` está activa por defecto.** El resto debe activarse manualmente agregando `enabled = true` después de cada nombre de *jail* que quieras activar (el que se encuentra entre `[]`).
- Hay muchos otros comandos y parámetros (como IPs a ignorar), pero no los usaremos en este laboratorio, ya que queremos una implementación simple (pero efectiva) de `fail2ban`.
- Cuando se activa una *jail*, los parámetros `bantime`, `findtime` y `maxretry` definen cómo se desautoriza una IP. Un `bantime` negativo significa un **bloqueo permanente. Un host es bloqueado por una *jail*** si activa sus condiciones con éxito más de `maxretry` veces a lo largo de `findtime` segundos.
- Las *jails* disponibles se definen en una sección de este archivo. Como decíamos, `sshd` está habilitada por defecto, mientras que las demás deben activarse manualmente. Eso sí, esto solo se recomienda si realmente tenemos el servicio en marcha.

```
[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and det$
#mode   = normal
port    = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s


[dropbear]

port     = ssh
logpath  = %(dropbear_log)s
backend  = %(dropbear_backend)s


[selinux-ssh]

port     = ssh
logpath  = %(auditd_log)s


#
# HTTP servers
#

[apache-auth]

port     = http,https
logpath  = %(apache_error_log)s


[apache-badbots]
# Ban hosts which agent identifies spammer robots crawling the web
# for email addresses. The mail outputs are buffered.
port     = http,https
```

- Cada comportamiento de *jail* se define en un archivo diferente en el directorio `/etc/fail2ban/filter.d`. Cada uno de estos archivos contiene **las expresiones regulares que,** al coincidir con una entrada del *log*, activan la regla. También definen modos de operación y otros parámetros. No es necesario examinarlos en este laboratorio, esto es solo para que lo sepas 😊
- Para este laboratorio **habilitaremos el modo agresivo para `sshd` y garantizar el máximo modo de protección SSH.**

Comprobación del estado de *Fail2Ban*

`fail2ban-client` puede usarse para gestionar `fail2ban` a través de las siguientes opciones:

- `start`: inicia `fail2ban`
- `reload`: Recarga la configuración de `fail2ban`
- `reload JAIL`: Reemplaza la configuración actual de la *jail* especificada por la que está en el archivo que se pasa
- `stop`: Detiene `fail2ban`

- `status`: muestra el estado actual de `fail2ban` y las *jails* activas (incluidos sus nombres)
- `status JAIL`: muestra el estado de la *jail* especificada. Es una opción muy importante ya que también muestra cuántas veces se ha activado y las IPs bloqueadas actualmente.

Cada vez que actives nuevas *jails* en el archivo anterior **debes volver a cargar** `fail2ban` y verificar cuántas *jails* están activas.

Prueba de *Fail2Ban*

Probar `fail2ban` con SSH es muy fácil: solo tienes que hacer varios intentos de inicio de sesión con contraseñas incorrectas dentro del tiempo que has configurado. Se recomienda hacerlo desde uno de los contenedores de la **infraestructura del Lab 7.** Una vez que has sido bloqueado, **usa los comandos anteriores para comprobar que la IP aparece realmente como prohibida**.

Para desbloquear la IP, puedes esperar el tiempo que has especificado (¡si no, es infinito! 😊) o ejecutar: `fail2ban-client set <nombre de la jail> unbanip <ip>`

🛠 Ejercicio L7B2_PORTSENTRY: Protección de intentos de escaneo: *PortSentry* como *honeypot*

Descripción de la actividad / Aplicación práctica

Necesitas bloquear conexiones de atacantes que **están intentando escanearte**

Resultados esperados

Esta actividad finalizará cuando seas capaz de **proteger un host contra los intentos de escaneo** con `nmap` usando `portsentry`, y comprender el comportamiento del bloqueo que realiza, todo sin interferir con el *firewall* de la máquina. También debes **verificar que la protección funciona** y **deshacer los bloqueos** de IP creados por este programa.

Otra información necesaria para su realización

Aunque ya vimos un ejemplo en el primer laboratorio, los escaneos de puertos con *Nmap* se tratarán en profundidad el próximo laboratorio, y son una de las operaciones más comunes en cualquier actividad de *pentesting*. En este laboratorio veremos cómo detenerlos usando esta herramienta. ***PortSentry* puede detectar escaneos de puertos y bloquearlos**, por lo que la máquina que está escaneando no obtendrá información.

Existen varias formas de utilizar `portsentry`, pero vamos a usar **una de las más interesantes**: combinarlo con el *firewall* que desplegamos en una actividad anterior para **emular el comportamiento de un *Honeypot***. Para ello, **dejaremos algunos puertos abiertos a propósito en el *firewall***, aunque no haya ningún servicio que los escuche. Se detectarán los intentos de escaneo a través de estos puertos "falsamente abiertos " y la herramienta procederá a prohibir la máquina que hace el escaneo. Ten en cuenta que `portsentry` no puede escuchar en los puertos que tienen servicios reales que los están usando, ya que "ocupan" los puertos. Sigue estos pasos:

- Instala el servicio *PortSentry* en tu máquina virtual *Ubuntu* (no es compatible con contenedores): `sudo apt install portsentry`

- Comprueba que se está ejecutando: `sudo service portsentry status`
- Entiende los dos modos de **trabajo** de la herramienta (valores de los parámetros `TCP_MODE` y `UDP_MODE` en el archivo `/etc/default/portsentry`)
    - **Modo básico** (valores: `"tcp"`, `"udp"`): Escucha una lista estática de puertos predefinidos en el archivo de configuración. **No** utilizaremos este modo ya que la presencia de la herramienta se puede detectar fácilmente.
    - **Modo *stealth* avanzado** (valores: `"atcp"`, `"audp"`): Usa un *raw socket* para detectar escaneos y así la herramienta no puede ser detectada fácilmente: los puertos parecen tener el servicio "normal" en él, pero es `portsentry` quien está escuchando y detectando actividad sospechosa.
- Comprender las **posibles respuestas** a un intento de escaneo (valores de `BLOCK_UDP` y `BLOCK_TCP` en el archivo `/etc/portsentry/portsentry.conf`)
    - Solo hacer **log** del tráfico sospechoso en `/var/log/syslog` (valor: `"0"`) (**valor por defecto**)
    - **Bloquear a la máquina "infractora"** (valor: `"1"`)
    - **Ejecutar un programa** como respuesta (valor: `"2"`). Esto nos da mucha flexibilidad y permite respuestas "salvajes" a los intentos de escaneo 😉, pero no lo usaremos aquí.

Registrar y bloquear intentos de análisis

Lo primero que hay que hacer para habilitar `portsentry` es habilitar los siguientes servicios en la zona de *firewall* `docker`: SSH, FTP y Telnet.

🔍 ***Así tenemos un "cebo" para atraer los intentos de escaneo a nuestro servidor.***

Una vez hecho esto, abre el fichero `/etc/default/portsentry` y pon `TCP_MODE` a `"atcp"` y `UDP_MODE` a `"audp"` **para habilitar el modo *stealth avanzado***. Reinicia `portsentry` y haz un análisis `nmap` simple desde cualquiera de los contenedores de **infraestructura de Lab 7** a la máquina host (`nmap <IP>`). Una vez finalizado el análisis, comprueba el contenido de `/var/log/syslog` para ver si se han detectado los intentos de análisis.

Que se haga *log* con éxito de los intentos de escaneo es la condición para terminar de configurar `portsentry` según lo previsto: Abre `/etc/portsentry/portsentry.conf` y cambia los valores de los parámetros `BLOCK_UDP` y `BLOCK_TCP` a `"1"`. Reinicia `portsentry` y vuelve a realizar el mismo análisis.

Revertir bloqueos

Para revertir un bloqueo, es necesario hacer esto:

- Eliminar la IP de la máquina bloqueada de `/etc/hosts.deny`. **Estar en este archivo impide cualquier comunicación desde esta IP a nuestra máquina**.

```
  GNU nano 2.9.3                  /etc/hosts.deny

# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID

ALL: 192.168.8.100 : DENY
```

- Eliminar la regla del *firewall* que rechaza (`REJECT`) todo el tráfico del cliente infractor: `sudo route del -host <IP del cliente infractor> reject`
- Podemos usar `netstat -rn` para comprobar que se ha eliminado la regla.

Bloque 3. Detección de intrusiones

Ejercicio L7B3_IDSMALTRAIL: Instalación y prueba del IDS *Maltrail*

Descripción de la actividad / Aplicación práctica

Esta actividad te enseña a **instalar y usar un IDS llamado *Maltrail***, para que veas cómo funciona un IDS de red o NIDS.

Resultados esperados

Esta actividad se considerará completa cuando tengas **una instalación operativa de *Maltrail*** capaz de detectar incidencias como escaneos de puertos. Puedes contestar a esta pregunta: *¿Entiendes como un IDS bien configurado se puede usar para localizar actividades sospechosas que vengan de máquinas ajenas?*

Otra información necesaria para su realización

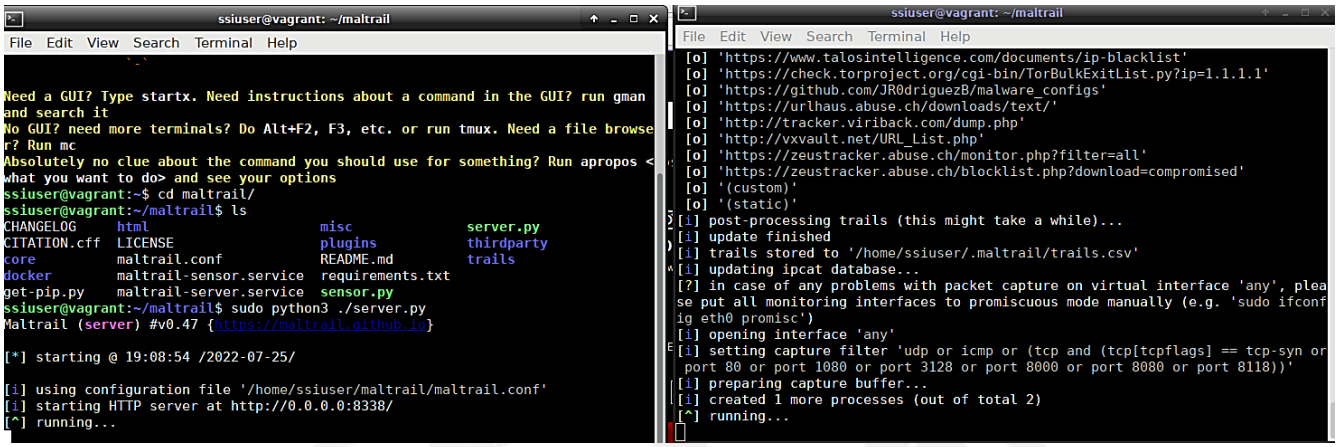
Snort (https://www.snort.org/) **es un *Intrusion Prevention System (IPS)* de código abierto de referencia**. El problema es que su instalación, puesta en marcha y administración **es compleja** (https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/011/074/original/Snort_3_on_Ubuntu_18_and_20.pdf), y por tanto es una inversión para infraestructuras grandes que requieran ese nivel de monitorización. Otra herramienta que puede usarse para el mismo fin es ***Suricata*** o bien una distribución de monitorización y seguridad profesional como ***SecurityOnion***. No obstante, ambos requieren muchos recursos y son complejos de manejar, al igual que *Snort*.

Debido a esto, en esta actividad vamos a ver **un IDS mucho más sencillo que tiene una interfaz de uso simple** y cumple con la misma función: *Maltrail*. *Maltrail* es una herramienta de detección de tráfico malicioso autónoma que **se ejecuta en un nodo de monitorización** por el que se supone que pasa todo el tráfico de una red (recuerda qué tipo de máquinas cumplen con esto en nuestra SNI de teoría, como puede ser un ***proxy* inverso**).

Este IDS usa listas de bloqueo públicas de direcciones web sospechosas y/o *malware*, junto con las direcciones estáticas recopiladas por diversos informes antivirus para detectar posibles amenazas. En caso de que se detecten, los detalles del evento se envían al servidor y se almacenan en un *log*. No obstante, esto ocurre si el componente llamado "**sensor**" se ejecuta en una máquina distinta que el componente llamado "**servidor**".

Para simplificar ambos componentes pueden residir en la misma máquina, como haremos en esta actividad.

Maltrail se puede instalar en cualquier máquina *Linux*, y para ello lo mejor es seguir su tutorial de instalación oficial: https://github.com/stamparm/maltrail, si bien uno de los paquetes de prerrequisitos listados en este tutorial no se encuentra en *Ubuntu*, y los prerrequisitos finales son: `sudo apt-get install git python3 python3-dev python3-pip libpcap-dev build-essential procps schedtool`. Terminada la instalación de ambos componentes en la misma máquina, y tal y como se indica en el tutorial, arrancamos el sensor y el servidor:



Y con esto ya podremos acceder a http://localhost:8338 para acceder al interfaz. Para probar el IDS basta con hacer un escaneo con `nmap` desde el *host* a la máquina virtual que tiene *Maltrail* operativo, y tras recargar su interfaz web debería detectar un posible escaneo de puertos como actividad maliciosa.

Ejercicio L7B3_WAZUHSCAN: *Wazuh* y ataques de red

Descripción de la actividad / Aplicación práctica

Consiste en ver **cómo reacciona *Wazuh*** ante un escaneo de red o un intento de acceso vía SSH fallido

Resultados esperados

Puedes ver cómo reacciona el XDR *Wazuh* de actividades anteriores cuando se somete a los casos cubiertos por ***Fail2Ban*** (intento de acceso repetido fallido mediante SSH) y ***PortSentry*** o ***Maltrail*** (escanear puertos con `nmap`). Puedes responder a estas preguntas:

- *¿Bloquea Wazuh estas actividades o solo informa de que las ha detectado?*
- *¿Detecta Wazuh las actividades mencionadas igual que el resto de las aplicaciones específicas o lo hace de manera menos evidente?*
- *¿Qué diferencia ves entre la información que te da Maltrail y Wazuh tras un escaneo con `nmap`?*
- *¿Entiendes por tanto mejor ahora la diferencia entre un NIDS y un HIDS?*

Otra información necesaria para su realización

Esta actividad es sencilla, puesto que solo tienes que **arrancar la "*MV Wazuh*"** de actividades anteriores y, desde tu host hacer un `nmap` e intentar **acceder varias veces vía SSH** con usuario / password erróneas a la MV de la asignatura. Hecho esto, examina la información capturada por *Wazuh* para poder responder a las preguntas realizadas en el ejercicio.

Bloque 4. Extremando la seguridad en las conexiones: VPNs

Ejercicio L7B4_VPN: Uso de VPNs tradicionales comerciales: *ProtonVPN*

Descripción de la actividad / Aplicación práctica

Consiste en usar la VPN comercial *ProtonVPN* en su versión gratuita para **familiarizarse con un servicio VPN comercial tradicional.**

Resultados esperados

Puedes usar *ProtonVPN* y conectarte a cualquier servidor remoto gratuito disponible, comprobando que la conexión a Internet sigue disponible y **que tu IP ha cambiado** por otra, geolocalizada en el país del servidor remoto. Puedes responder a las preguntas que se hacen en el enunciado del ejercicio.

Otra información necesaria para su realización

Lo primero que debes hacer es probar cual es tu IP actual y tu geolocalización en función de ella. Esto puedes hacerlo en varias páginas de Internet, pero una de las más típicas es esta: https://www.whatismyip.com/. Guarda o apunta el resultado para compararlo luego.

Si en el **laboratorio 4** te has decidido a usar *ProtonMail*, la cuenta que creaste entonces te servirá para este ejercicio. En caso contrario, puedes **hacerte una cuenta gratuita en *ProtonVPN*** (que te servirá también para *ProtonMail* si quieres usarlo 😊) en esta dirección: https://protonvpn.com/es-es.

🔍 ***En realidad serviría cualquier servicio VPN para hacer este ejercicio, pero nos hemos decantado por este por tener buena fama y ofrecer gratuitamente 200 y pico servidores en países de Europa, América y Asia para probarlo desde un dispositivo.***

Otra de las razones para optar por este servicio es que, aunque no podemos disfrutar de esas características en el plan gratuito, si que deja ver claramente como una VPN **puede ser un sistema de seguridad** no solo por la privacidad de las conexiones, sino evitando que navegues contenido malicioso. *¿Puedes localizar qué bloquea exactamente?*

Antes de empezar, es importante que tengas en cuenta una cosa. La versión probada en Diciembre de 2023 del cliente VPN de *ProtonVPN* se instala en *Ubuntu Server XFCE* (nuestra MV), pero por desgracia **no llega a conectar con ningún servidor**. Una versión *Desktop* de *Ubuntu* debería solucionar ese problema, pero no vamos a instalarla solo para este ejercicio. Por ello, la solución más sencilla es que uséis **un clon enlazado de vuestra MV *Windows*** de nuestra asignatura "hermana", ***Administración de Servidores y Redes***, puesto que el cliente *Windows* funciona sin ningún problema.

🔍 Os pedimos usar un clon enlazado para que no afectéis al trabajo de ASR y para que lo creéis a coste mínimo, ¡pero evidentemente sois libres de usar cualquier otra opción! 😊

No obstante, *ProtonVPN* tiene **clientes para una gran cantidad de sistemas** (*Android*, *iOS*, *Windows*, *MacOS*, *GNU/Linux*, *Chromebook* y *AndroidTV*), así que si el producto os convence podéis usarlo en cualquiera que más os convenga, ya que es lo mismo en todos.

Si ya tenéis experiencia con VPNs y tenéis un cliente compatible con ***OpenVPN***, también tiene ficheros de configuración para ellos. No obstante, en este ejercicio asumimos que no tenéis experiencia, y que usareis el cliente nativo de *ProtonVPN*. Una vez instalado el cliente, nos validamos en el con los datos de la cuenta en *ProtonMail*/*ProtonVPN* que ya tenemos/hemos creado. Si todo es correcto, **deberíamos ver esta pantalla**, donde se nos muestran los países a los que podemos conectarnos en el mapa y en la lista inferior izquierda. Seleccionamos uno (mejor el Europeo) y nos conectamos.

Hecho esto, contesta a las siguientes preguntas:

- *¿Cambia la IP en la pantalla de la VPN al conectarte?*
- *¿Si ahora navegas a la dirección para averiguar tu IP, proveedor de Internet y geolocalización anterior, cambia también?*
- *¿Entiendes que ahora eres esa máquina y usas el proveedor mostrado del país elegido para cualquier sitio al que te conectes? ¿Cuál crees que sería una forma de detectar que estás en otro país mientras navegas normalmente (y un posible problema de usabilidad)? ¿Para qué crees que sirve esto?*
- *¿Entiendes que todo el tráfico desde tu MV al servidor VPN al que te has conectado va cifrado usando la combinación de cifrado asimétrico y simétrico que vimos en el **laboratorio 4**? ¿Qué te garantiza esto entonces?*
- *A raíz de lo anterior, ¿comprendes por qué usar una VPN es una forma de estar seguro aunque uses una Wifi pública (cafetería, aeropuerto...)?*
- *¿Puedes localizar los sistemas de seguridad que incorpora la versión de pago y entender para qué servirían si pudieras activarlos? Concretamente:*
    - *Imagina que te has conectado vía VPN con la Wifi de un bar. ¿Cómo te podría salvar el kill switch de que te roben datos?*
    - *Imagina que tienes un servidor de Minecraft propio en el puerto 25565 y quieres jugar con tus amigos pero con la VPN activa. ¿Podrías usarlo a la vez que una ProtonVPN de pago con alguna de sus opciones? (recuerda el ejercicio L7B1_FWFORWARD)*

Ejercicio L7B4_TORBROWSER: Instalar y probar *ToR Browser*

Descripción de la actividad / Aplicación práctica

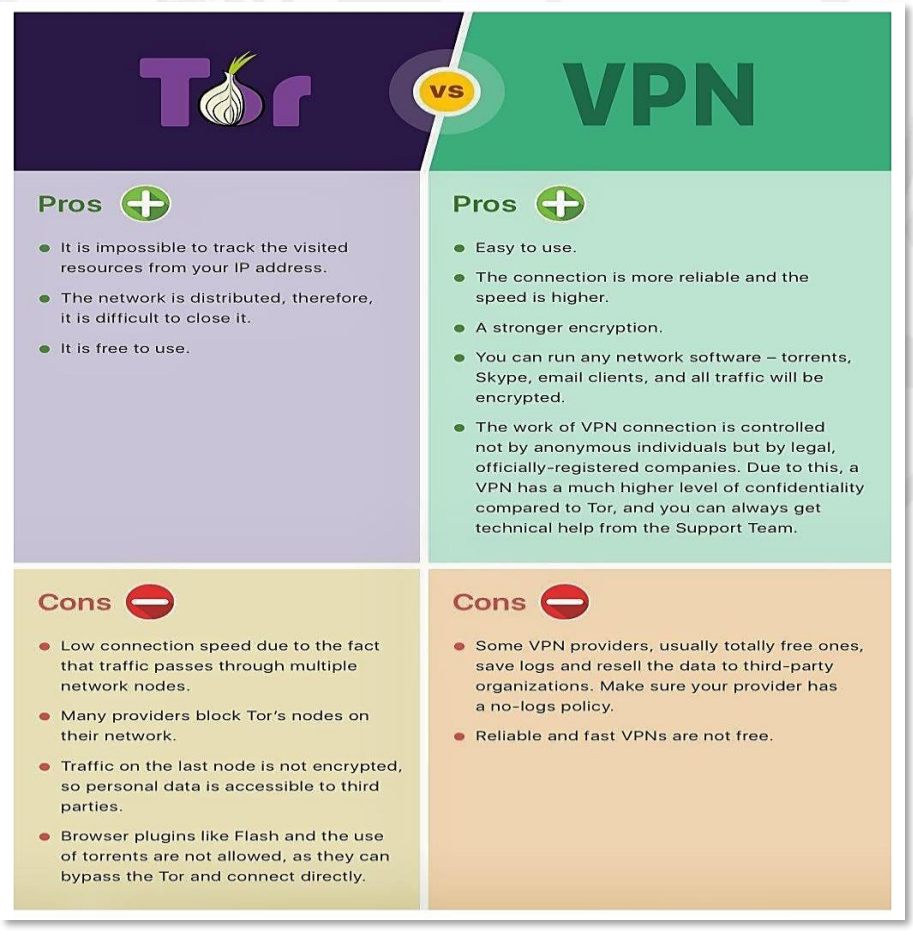
Consiste en **instalar el navegador *ToR Browser*** y usarlo para navegar por diferentes web para comprobar las ventajas y desventajas del **anonimato** que proporciona la red ToR, así como probar una dirección `.onion`

Resultados esperados

Puedes poner en marcha **una instalación funcional de *ToR Browser*** y usarla para navegar a través de la red ToR, entendiendo las ventajas y desventajas que supone hacerlo. Puedes contestar a estas preguntas y a las que se plantean durante el enunciado del ejercicio:

- *¿Sabes distinguir entre Deep y Dark web y lo que son ambas?*
- *¿Entiendes que es exactamente un enlace* `.onion` *y porque se usan para acceder a la dark web?*
- *¿Se resiente la velocidad de navegación?*
- *¿Hay algún efecto de navegar por webs escondiendo tu identidad y, por tanto, el país desde el que navegas realmente?*
- *¿Entiendes cuál es el propósito de un enlace* `.onion` *y por qué esto te puede llevar a sitios de la Deep y la Dark web?*

El objetivo final del ejercicio es que compruebes algunas cosas de este diagrama por ti mismo (**Fuente**: https://www.reddit.com/r/Tunisia/comments/xj3k6r/tor_vs_vpn/)



Otra información necesaria para su realización

Antes de hacer este ejercicio necesitas familiarizarte con 3 conceptos.

¿Qué es la *deep web*?

La "*Deep Web*" o "Web Profunda" es la parte de Internet **no indexada por motores de búsqueda tradicionales**. No encontrarás ninguno de sus contenidos en *Google*, *Bing*, etc. No importa por qué motivo no lo estén, simplemente no aparecen

🔍 ***No es todo webs ilegales (que obviamente también las hay), también aparecen lugares para consultar BBDD empresariales o gubernamentales, webs privadas de grupos u organizaciones (aunque basar su privacidad solo en no aparecer indexado es una mala decisión)***

En el sentido literal de la definición, *deep web* también es tu *Dropbox* o *OneDrive*, por ejemplo. O, en general, contenidos a los que un motor de búsqueda no puede acceder porque requieren unas credenciales de acceso. Cuando además de esas credenciales, se requiere **un método de acceso que requiera anonimato** de la persona que accede, entonces hablamos de la *Dark Web*.

🔍 ***Obviamente la Dark Web es parte de la Deep Web, ya que sus contenidos no están tampoco indexados***

Es importante mencionar que **no todo el contenido de la *Deep Web* es ilícito**. Muchos sitios legítimos y útiles se encuentran en la *Deep Web* porque requieren credenciales de inicio de sesión para acceder a ellos. Sin embargo, debido a las limitaciones y al anonimato total de la *Dark Web*, desgraciadamente suele ser un imán para la actividad delictiva.

¿Qué es la *dark web*?

La "*Dark Web*" o "Web Oscura" es una parte de la *World Wide Web* a la que **solo se puede acceder a través de un navegador especial** para acceder a los sitios web que usan la terminación de dominio `.onion`. Se caracteriza por un anonimato (teóricamente) absoluto y unas restricciones especiales de acceso. Este nivel de privacidad y seguridad atrae a individuos y grupos criminales, lo que ha llevado a la *dark web* a ser asociada con contenido prohibido y actividades turbias.

🔍 ***Aunque el mero uso de la Dark Web y las tecnologías correspondientes no es punible ni ilegal, desde el momento en que entras en mercados y foros de tráfico de armas, drogas o similares y estableces una relación comercial con los vendedores de estos sitios, estás cometiendo un delito. Navegar ahí no es en absoluto una broma, y no solo existe todo tipo de contenidos y servicios ilegales (y extremadamente perturbadores), sino exploits de navegadores, malware y otras amenazas***

Es importante mencionar que, a pesar de su reputación, la *Dark Web* **también se utiliza para fines legítimos**. Por ejemplo, puede ser un refugio para personas que viven bajo regímenes opresivos y buscan una forma segura de comunicarse o acceder a información. Sin embargo, por norma general, **no es un lugar para que los usuarios medios de internet se entretengan o naveguen**. El anonimato saca a relucir lo peor de la humanidad, y en ella se pueden encontrar auténticas atrocidades 😕.

🔍 ***En la asignatura preferimos usar una aproximación transparente acerca del tema. Existe y acceder a ella es fácil si tienes las herramientas. Pero no te engañes: lo peor de la humanidad está ahí, y te lo puedes encontrar hasta por descuido...***

¿Qué es un enlace `.onion`?

Un enlace `.onion` es un tipo de dirección web que corresponde a un servidor anónimo accesible a través de la red Tor mencionada en la teoría. **Estos enlaces son usados en la *Dark Web*** y son conocidos por su alto grado de anonimato.

> 🔍 ***Las direcciones .onion no funcionan como un DNS. Los navegadores web pueden acceder a sitios .onion usando programas proxy y enviando la solicitud a través de los servidores de la red ToR***

Estas direcciones son opacas y no nemotécnicas. Son combinaciones de caracteres alfanuméricos generados manualmente. El nombre "*onion*" (cebolla) hace referencia a la técnica de encaminamiento de cebolla, en inglés *onion routing*, **usada por Tor para lograr un alto grado de anonimato**.

> 🔍 ***Es importante tener en cuenta que al usar un proxy para acceder a sitios .onion, el operador del proxy puede espiar el tráfico y la IP origen. Si la intención es ser anónimo no se deberían usar proxies***

Pero para entender todo lo relativo a enlaces `.onion` y la red ToR es necesario que te familiarices con el ***ToR Browser***, que es el objetivo final de este ejercicio.

Probando *ToR Browser*

Puedes instalar *ToR Browser* en tu MV *Ubuntu* descargándoselo de aquí: https://www.torproject.org/download/ y siguiendo las instrucciones de instalación que están en este enlace: https://tb-manual.torproject.org/installation/. Conviene que te descargues el ejecutable y el fichero con su firma, y sigas el procedimiento de verificación de su firma que tienes aquí documentado: https://support.torproject.org/tbb/how-to-verify-signature/. *¿Entiendes los elementos que necesitas en función de lo que hemos visto en los **laboratorios 3 y 4**?*

Una vez extraigas y ejecutes el navegador, veras que lo primero que te permite es conectarte a ToR para navegar, ya que si no lo haces entonces navegarás normalmente. No obstante, **antes de hacerlo conviene que ajustes la configuración de la siguiente forma por si acaso**:

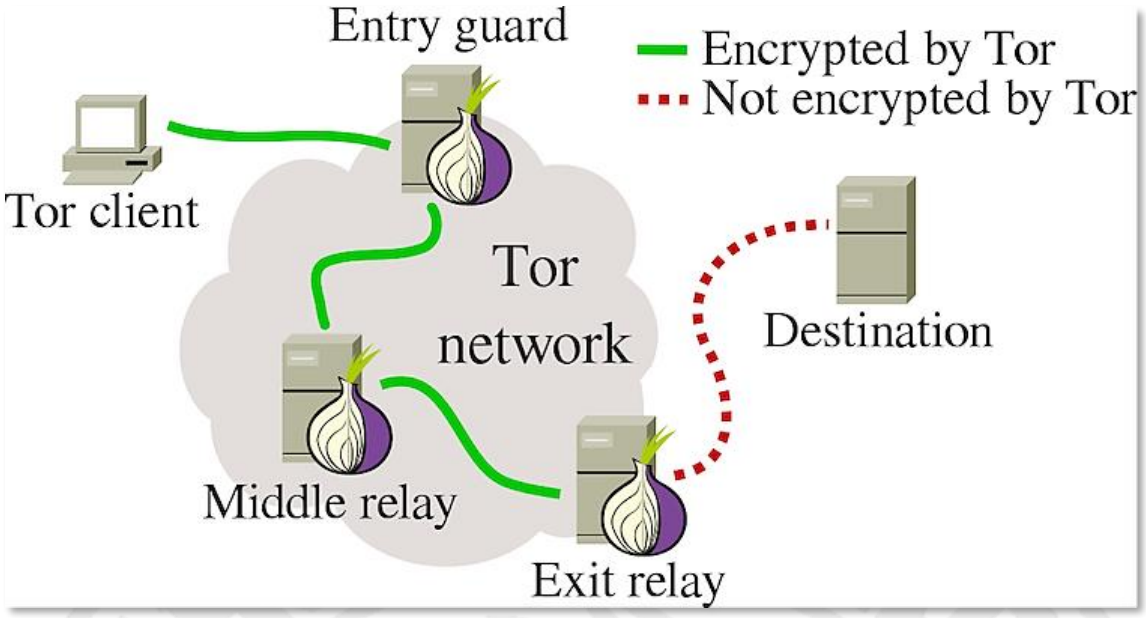
- Entra en la configuración del navegador y **ajústala a "*Safer*" o "*Safest*"**, dependiendo de si vas a acceder a webs conocidas o no.

> 🔍 ***Safest puede hacer que algunas webs no funcionen (bloquea JavaScript)***

- **Instálale el bloqueador de anuncios** del ejercicio **L2B4_NAVEGAR** del **Laboratorio 2**.

> 🔍 ***ToR Browser está construido con el motor de Firefox, por lo que usa sus mismos plugins.***

- Ahora conéctate a la red ToR y usa la dirección del ejercicio anterior para **saber tu IP y geolocalización**. *¿Dónde estás?* Abre el Firefox de la MV y haz lo mismo. *¿Qué ocurre? ¿Están todos los servicios de la MV enrutados a través de ToR o solo este navegador?*
- Si te digo que **la ruta** que se sigue para llegar hasta esa máquina (que se llama "**nodo de salida**") **cambia cada cierto tiempo** y que esos nodos intermedios solo conocen al anterior y al posterior en su "cadena", *¿Entiendes porque tu conexión es anónima dentro de la red ToR*? Mira esta imagen (**Fuente**: https://theconversation.com/tor-upgrades-to-make-anonymous-publishing-safer-73641) para hacerte mejor la idea, pero asume que la "ruta" desde tu máquina "*ToR client*" hasta el nodo de salida tiene normalmente **más de dos nodos** entre medias



- Si te digo además que **el nodo de salida también cambia a lo largo del tiempo**, *¿Entiendes el anonimato también desde el punto de vista del destino al que te conectas?*
- Prueba a navegar a la *Wikipedia*, un periódico, etc. *¿Puedes? ¿Se nota una disminución de la velocidad de la conexión?*
- Prueba ahora a navegar a la web de la Universidad de Oviedo o a la de cualquier banco. *¿Puedes? ¿Puedes acceder a esa página dese el Firefox de la MV?* Si te digo que los nodos de salida de la red ToR **son conocidos en todo momento**, *¿Por qué crees que pasa esto y uno funciona mientras el otro no?*
- *¿Crees que la razón de que esto pase tiene que ver con la relación **ToR – Dark web – Anonimato**?*
- *¿Qué pasa si intentas usar Google desde ToR Browser? ¿Entiendes por qué por defecto te muestra DuckDuckGo? ¿Qué característica principal tiene este buscador?*
- *¿Qué crees que pasaría en los casos anteriores **si activases la VPN de ProtonVPN y luego navegases vía ToR Browser**? ¿Funcionaría?*
- *¿Y si en lugar de eso navegases por ToR a una máquina remota (no vinculada a tu identidad) que fuera la que tiene la VPN activa? ¿Qué conseguirías en ambos casos? ¿Entiendes ahora la razón de que algunos delincuentes "okupen" máquinas ajenas?*
- *Si te digo que los proveedores de VPN tienen la obligación legal de guardar logs de quien se conecta a sus servicios desde cada sitio durante X tiempo para investigaciones policiales ¿Entiendes la diferencia entre usar una VPN y usar ToR ahora que sabes a grandes rasgos cómo funciona?*
- *¿Para qué crees que sirve la opción de reiniciar tu identidad de Tor Browser? ¿En qué tipo de PCs crees que sería más útil?*

Probando una dirección *.onion*

Antes hemos hablado de que las direcciones `.onion` se usan muchas veces para para **ocultar webs de delincuentes**, pero que también tienen usos para **evitar la censura** de acceso a determinados sitios y fredes sociales. Una compañía que lo permite es *Meta* con su dirección `.onion` de *Facebook* que puedes encontrar aquí: https://en.wikipedia.org/wiki/Facebook_onion_address.

Copia la dirección y navega a ella en tu *Tor Browser* conectado a ToR. *¿Qué sale? ¿Y si lo haces en un Firefox estándar? ¿Entiendes ahora como acceder a cualquier enlace* `.onion` *que tengas? ¿Entiendes también que, al igual que cualquier acortador de enlaces, **tienes que fiarte al 100% de la fuente** porque no hay forma de saber lo que hay realmente detrás de ella?*

🔍 ***No hace falta que entres en Facebook en absoluto, vale con llegar a la pantalla de login*** 😊

Ejercicio L7B4_VPNP2P: Uso de VPNs P2P: *ZeroTier*

Descripción de la actividad / Aplicación práctica

Consiste en usar la VPN P2P comercial *ZeroTier* en su versión gratuita para **familiarizarse con un servicio VPN P2P y las diferencias principales con una tradicional.**

Resultados esperados

Puedes **poner en marcha una VPN P2P *ZeroTier* entre dos MVs** y comprobar que ambas están comunicadas entre ellas como si fuera una red local. Puedes hacer esto **con dos MVs tuyas o con la de un compañero**. Puedes contestar a estas preguntas y a las que se plantean a lo largo del ejercicio:

- *¿Entiendes como este sistema te da el control de la VPN y no dependes tanto de un servidor como en el caso de ProtonVPN?*
- *¿Entiendes como puedes usar este servicio para conectarte de forma segura desde un PC del laboratorio a tu casa?*
- *¿Entiendes que aunque no tengas permisos de instalación de programas en un PC de un laboratorio, los tienes en la MV, y puedes comunicar por tanto la MV con el PC de tu casa sin problemas si ambos tienen cliente de ZeroTier?*
- *¿Se te ocurre como combinar el uso de un firewall con ZeroTier para que solo puedas acceder a los servicios desde la red que pones en marcha con ZeroTier?*
- *¿Entiendes ahora que puedes dejar servicios (Escritorio Remoto, carpetas compartidas...) en una máquina de tu casa sin exponer en Internet, solo accesibles para las máquinas detrás de tu router (no abres puertos en el router para ellos nunca, por tanto no son visibles "desde fuera"), y que gracias a ZeroTier puedes acceder a ellos desde cualquier parte?*

Otra información necesaria para su realización

En esta sección vamos a tratar de explicarte cómo aprender a usar **un servicio VPN P2P llamado *ZeroTier***, que permitiría acceder a cualquier dispositivo desde cualquier parte de una manera descentralizada (sin depender de un servidor central como *ProtonVPN*), segura y más sencilla de instalar que una VPN propia.

🔍 ***Esta VPN P2P es ideal para implementar teletrabajo en nuestra empresa futura o conectarnos a nuestra casa teniendo un control completo de todo el servicio que lo proporciona y, normalmente, de manera gratuita porque no necesitarás mucha infraestructura en estos casos***

Tradicionalmente, tal y como vimos con *ProtonVPN*, cuando se usa un servicio VPN se suele asumir que la necesidad de un cliente es **hacerse pasar por una máquina que está en otra localización**, que es la que realmente accede al servicio remoto, y de esa manera saltarse algún tipo de bloqueo regional o restricción que exista en función de donde el cliente se conecte. Este uso de las VPN tradicional y muy popular, pero no tiene que ver con el que vamos a ver ahora.

En el caso de una VPN P2P **no te conectas a un servidor de un tercero** (tu proveedor de VPN) para acceder a servicios remotos que te ofrezca otra compañía. Ahora el servicio remoto realmente es una máquina que

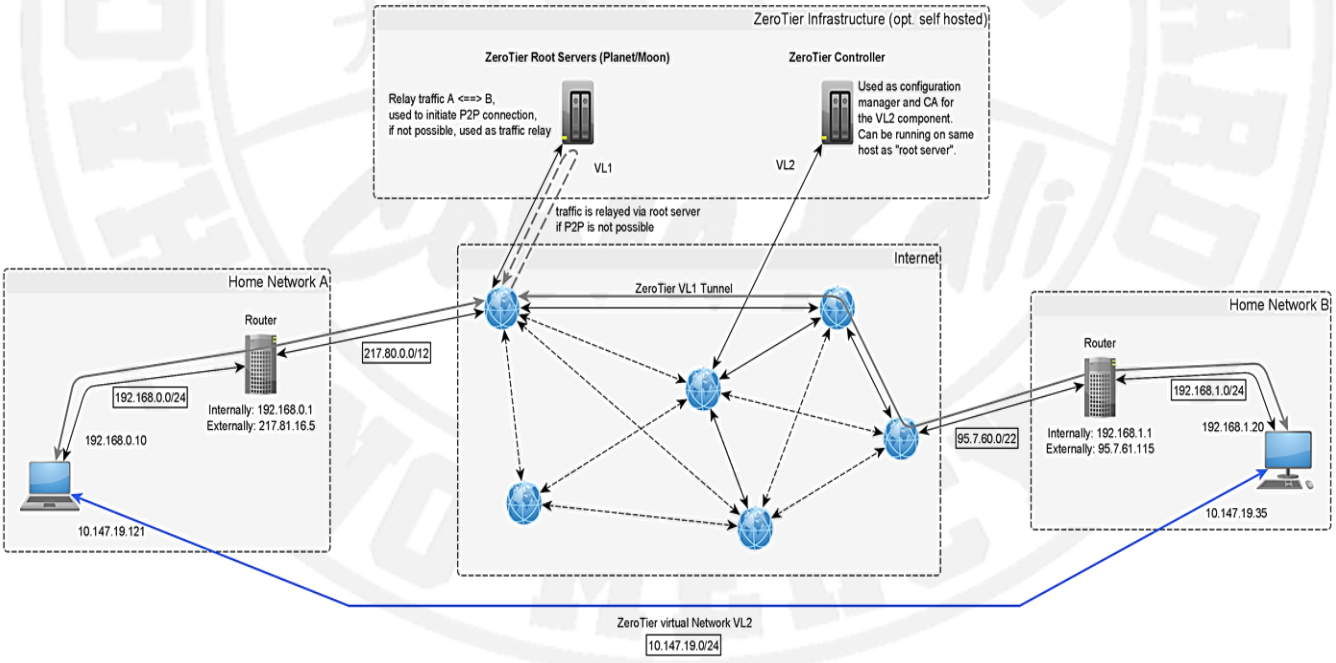
tienes en tu casa u oficina. Es decir, es una forma de conectarte **desde un dispositivo de tu propiedad,** o al que puedas acceder, **a otro dispositivo también de tu propiedad,** o al que tengas acceso, a través de Internet y de forma segura.

> 🔍 ***Y de poder hacer una red privada con todos estos dispositivos.***

Por otro lado, si usásemos un servicio VPN de terceros para acceder a una máquina nuestra, estaríamos cediéndole datos personales a dicho servicio, y además tendríamos que configurar esa máquina **para que sea visible a través de Internet** de una manera correcta para ese servicio VPN. No exponemos desde dónde nos conectamos pero si a lo que nos conectamos. Podríamos configurarla para solo permitir conexiones a través de esta VPN de terceros, pero es algo más complejo que excede el ámbito de este curso.

Si esto no nos interesa, la otra opción, que es **instalar nuestro propio servidor VPN** (Ej.: *Wireguard,* https://www.wireguard.com/) es técnicamente viable, pero también compleja y lenta especialmente si autorizamos muchos dispositivos. Es por tanto una solución posible, pero menos práctica en nuestro escenario concreto.

Servicios **VPN punto a punto** como ***TailScale*** o ***ZeroTier*** son los que nos resuelven este problema. Gracias a ellos podemos instalar su *software* **en cada uno de nuestros dispositivos que deseemos usar desde sitios remotos** y, de esta forma, en lugar de depender de un servidor central, formar una red de dispositivos que se pueden ver entre ellos como si estuvieran en una red local, independientemente del sitio del mundo en el que estén. La siguiente imagen (https://blog.rico-j.de/zerotier-one/) muestra como funcionan esta clase de VPNs



> 🔍 ***Esta es la principal ventaja de estas tecnologías que, al eliminar el punto central, nos quita problemas de disponibilidad y de rendimiento. También nos liberamos del problema de hacer visible el servicio hacia el exterior, porque habitualmente el cliente se encarga de hacer toda la gestión por nosotros***

Por tanto, tendríamos una forma sencilla de **compartir recursos que tenemos de una manera segura**, minimizando el riesgo de que alguien no autorizado sea capaz de acceder a ellos. Esto es así

porque, salvo que al servicio se le descubra algún tipo de vulnerabilidad, solo nosotros podremos establecer una conexión con él a través de internet, que será totalmente privada e inaccesible para gente no autorizada.

🔍 ***Por favor, nunca dejes un Escritorio Remoto visible en Internet. Esto tarde o temprano acabará haciéndote víctima de un ataque***

El servicio de VPN P2P *ZeroTier*

ZeroTier (https://zerotier.com/) es una empresa que da un servicio VPN P2P de *software* libre y código abierto (también llamado "*switch* universal") **más fácil de configurar** que una VPN tradicional. Sus servidores se encargan de poner en contacto a los dispositivos que van a comunicarse, y **son estos los que crean un túnel cifrado entre ellos**. A partir de ahí, mantienen **una comunicación cifrada punto a punto**. De este modo, además de aumentar la confidencialidad de la comunicación, se disminuye la latencia.

🔍 ***Es importante destacar que ZeroTier pone el peso de la comunicación en nuestros dispositivos, con las ventajas de privacidad que ello supone (nuestro tráfico no pasa por los servidores de ninguna compañía)***

Con *ZeroTier*, podrás **crear una VPN de hasta 25 dispositivos** de manera gratuita, más que suficiente para un uso personal. Crear VPNs más grandes requieren acogerse a un plan de pago (https://zerotier.com/pricing/). Además, es compatible con muchos dispositivos (*Windows*, *Mac*, *Linux*, *Android*, *iOS*...), por lo que podrás incorporar a la VPN y tener un acceso seguro a mucho *hardware* que ya tengas.

Instalación de *ZeroTier*

Para instalar *ZeroTier* lo primero que debemos hacer es crearnos una cuenta gratuita aquí: https://my.zerotier.com/login. El registro **exige una cuenta de correo electrónico real** porque requiere verificarla. Una vez registrados, veremos la pantalla de bienvenida. En ella veremos el botón "***Create A Network***" para crear una VPN, pero ya tendremos una creada de antemano que nos sirve perfectamente para hacer pruebas. Podemos crear varias VPN si queremos. *¿Para qué crees que serviría esto?*

Cada red de *ZeroTier* **tiene un identificador único de 48bits** (que será necesario usar luego para que los dispositivos se vinculen a la red creada. Además, se le asigna un nombre provisional aleatorio, pero que es conveniente cambiar por otro que tenga sentido para nosotros.

Es importante dejar la red como "PRIVATE" para que cualquier dispositivo que se una a la misma **tengas que aprobarlo** previamente. Las redes públicas no requieren aprobación para conectarse, y son para otros escenarios. En la sección ***Advanced*** hay más opciones, pero **con la configuración por defecto debería funcionar sin problemas**.

🔍 ***ZeroTier selecciona automáticamente un rango de IPs para la VPN que se va a crear. Fíjate que son direcciones IP privadas, como las que se mencionaron en el tema 1 de teoría***

En la sección ***Members*** es donde se autorizan los dispositivos que queréis tener activos en la VPN. También veréis los datos de la última vez que se conectó un dispositivo, entre otra información, lo cual es muy útil para detectar posibles intrusos o usos no autorizados. No obstante, inicialmente aparecerá una red sin dispositivos añadidos, como es lógico.

La sección "***Flow Rules***" **nos permite configurar un *firewall* y sus reglas**, lo que nos deja decidir el tipo de tráfico que queremos admitir o no en la red (protocolos, puertos, etc.). Esto es un uso avanzado, que nos permite **implementar una capa de seguridad extra** en nuestra VPN si fuera necesario. El proceso de configuración no es trivial y requiere estudiar cómo crear las reglas, pero tiene **ayuda integrada en la aplicación**. Para este ejercicio podemos dejar la configuración por defecto.

🔍 ***Piensa que si en una red limitamos el tráfico a un tipo de protocolo o puerto, porque es lo único que compartimos en ella, estamos eliminando de una sola vez varios posibles ataques a nuestros dispositivos***

Añadir dispositivos

Para añadir dispositivos a la red recién creada **tenemos que instalar el *software* que controla la VPN en cada uno de ellos**. Tenemos clientes para los dispositivos que mencionamos antes aquí: https://www.zerotier.com/download/. Para distribuciones *Linux* tenemos que instalar el *software* que haga de cliente usando alguno de los métodos indicados en la web de *ZeroTier*. Todo el proceso de instalación y de añadir cada cliente a la red **se realiza bajo línea de comandos**.

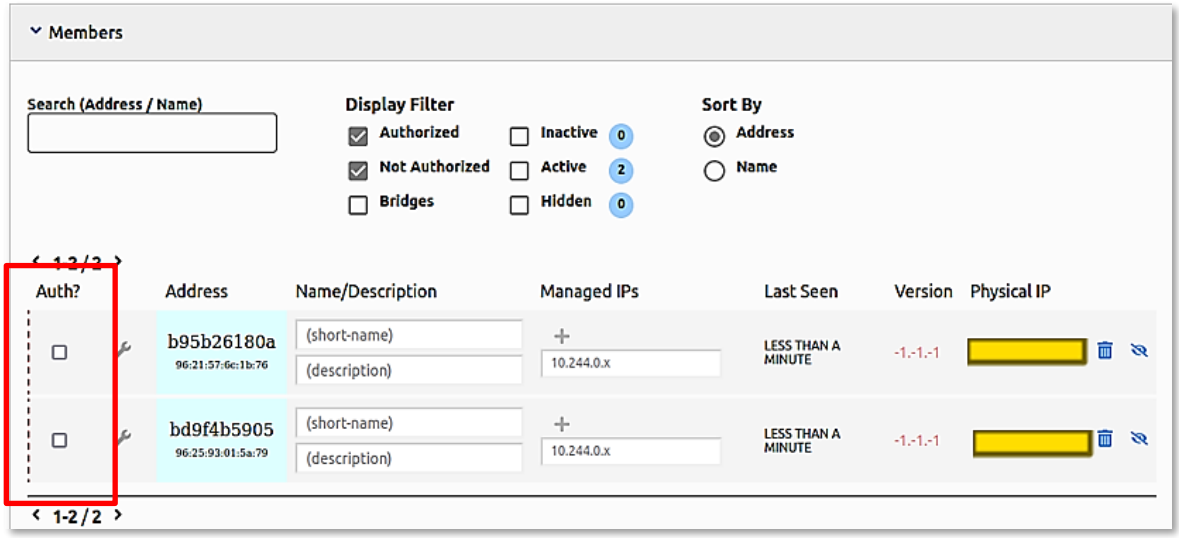
🔍 ***También hay una app de Android oficial, si queremos usar como cliente nuestro teléfono: https://play.google.com/store/apps/details?id=com.zerotier.one***

Una vez instalado el cliente, desde *Ubuntu*, en cada uno de los dispositivos que vayan a formar parte de la red debemos hacer previamente dos cosas :

- Asegurarnos de que está instalado ***curl***: `sudo apt-get install curl`
- Ejecutar `sudo zerotier-one -d` para arrancar el servicio en cada nodo que se vaya a añadir a la red

Ahora ya podemos añadir cada uno de los dispositivos que forman parte de la red ***con*** `sudo zerotier-cli join <el ID de red completo que nos salió en la web de ZeroTier>`. El resto de las opciones se pueden dejar como están. Como la red que creamos es privada y hemos usado el dispositivo para solicitar ser añadidos a ella, en la web de administración de *ZeroTier* nos aparecerán los **dispositivos que se han unido a nuestra VPN**. Es ahora cuando **debemos aceptar** que se conecten para poder seguir trabajando.

🔍 ***¿Cómo es posible que se hayan añadido los dispositivos sin más? Porque conocían nuestro ID de red. Si alguien lo averigua (no deberíamos revelárselo a nadie ajeno) podrá tratar de conectar un dispositivo, pero ese dispositivo solo podrá ver a los otros en la red si activamos su casilla de autorización, como veremos enseguida***



Si hacemos clic en la columna "***Auth?***", daremos acceso a los dispositivos que queramos y ya los podemos usar en la red, pudiéndose ver entre ellos de la misma forma que si estuvieran en una red local.

🔍 ***Conviene no obstante cambiarles el nombre para poder identificarlos fácilmente cuando la red crezca de tamaño.***

Al hacerlo, desde ese momento *ZeroTier* **asignará una IP** del rango privado que seleccionamos antes al dispositivo cliente y entonces podrá usarla. Para añadir más dispositivos a la VPN, y así poder establecer una conexión entre todos y probar que realmente una VPN punto a punto cumple con su labor, hay que repetir el proceso de instalación del cliente visto en cada dispositivo.

Para hacer las pruebas se puede probar a hacer un `ping` entre dispositivos usando la IP asignada por *ZeroTier* a cada uno de ellos, o bien intentar acceder a un servicio que un miembro de la red ofrezca a los otros (por ejemplo, una web, el escritorio remoto...). Si funciona, lo habremos hecho todo correctamente. Ni que decir tiene que, al poder elegir el rango de IPs, **podemos limitar el acceso a los servicios que ofrezcan los dispositivos solo a IPs de dicho rango con el *firewall*** de cada dispositivo (como el `firewalld` del bloque 1), lo que hace el acceso a los servicios aún más seguro.

🔍 ***ZeroTier puede funcionar dentro de máquinas virtuales que trabajen con una conexión NAT para salir Internet. Esto quiere decir que desde cualquier parte del mundo puedes conectarte a dentro de una MV tuya, por lo que no solo estás asegurando la conexión, sino que, además, si tu dispositivo está comprometido y el atacante puede usarlo para conectarse a tu VPN P2P, sus acciones estarán contenidas dentro de la MV correspondiente, limitando mucho su capacidad de causar problemas***

✍️ Insignias y Autoevaluación

NOTA: Tienes una versión de esta tabla de insignias en formato editable disponible en el *Campus Virtual*. Puedes usar este archivo para crear un documento en el formato que desees y tomar notas extendidas de tus actividades para crear un log de lo que has hecho. Recuerda que el material que elabores se puede llevar a los exámenes de laboratorio.

Nivel de Insignia	Desbloqueado cuando	¿Desbloqueado?
	Sabes cómo permitir o denegar servicios/puertos de un *firewall*	
	Sabes cómo agregar interfaces de red a las zonas de un *firewall*	
	Puedes permitir o denegar conexiones desde direcciones IP individuales o redes	
	Puedes responder a esta pregunta: *¿Qué tipo de ataques podría mitigar potencialmente que un firewall ponga un límite de peticiones/tiempo a un servicio?*	
	Puedes responder a esta pregunta: *¿Cuál crees que es el propósito de registrar / auditar intentos de conexiones explícitamente prohibidas en el firewall?*	
	Puedes responder a estas preguntas: *¿Por qué crees que es útil bloquear a las máquinas que realizan escaneos?* ***PISTA****: ¿Escaneas máquinas como parte de tu rutina normal?*	
	Entiendes cómo realizar reenvío de puertos y como conectar máquinas en distintos segmentos de red separados por *firewalls*	
	Entiendes por qué el bloqueo de IPs vía DNS es útil. Puede responder a esta pregunta: *¿Cuál es la ventaja de usar PiHole para bloquear IPs en lugar de un complemento de un navegador que bloquee IPs maliciosas?*	
	Puedes responder a estas preguntas: *¿Crees que* `fail2ban` *reemplaza a un firewall? ¿o más bien lo complementa?*	
	Puedes responder a esta pregunta: *¿Qué tipo de ataques previene* `fail2ban?`	
	Puedes responder a esta pregunta: *¿Crees que un bloqueo temporal es suficiente para prevenir un intento de ataque DoS?*	
	Puedes responder a estas preguntas: *¿Por qué crees que dejar los puertos abiertos a propósito puede ser una medida útil de detección de escaneos? Por lógica, ¿qué puertos dejarías abiertos de esta manera?*	
	Puedes responder a estas preguntas: *¿Qué pasa cuando una máquina está bloqueada por* `portsentry`*? ¿La máquina bloqueada puede ponerse en contacto con la que bloquea usando cualquier servicio o no?*	

	Puedes responder a estas preguntas: *¿Afectan los bloqueos de* `portsentry` *a la velocidad de escaneo? Si la respuesta es sí, ¿crees que esto también se puede utilizar como medida de seguridad?*	
	Puedes poner en marcha una conexión con un servicio VPN tradicional como *ProtonVPN* y usarlo para navegar por Internet de manera más anónima	
	Entiendes para qué sirve un NIDS y el tipo de cosas que detecta respecto al tráfico de red que pasa por él	
	Puedes responder a esta pregunta: *¿Cuántas cosas crees que puedes hacer para proteger las* conexiones `ssh` *desde el punto de vista de la red?* (piensa en combinar técnicas que vimos aquí y laboratorios anteriores)	
	Puedes protegerte de los intentos de ataque a `ssh` (y ataques similares a otros servicios que hacen) con *jails* predefinidas de `fail2ban`.	
	Puedes habilitar una "trampa de escaneo de puertos" en una máquina sin entrar en conflicto con el comportamiento normal del *firewall*. Puedes responder a esta pregunta: *si puedes recopilar las IP bloqueadas, ¿qué crearías con ellas?*	
	Comprendes los conceptos básicos de la red ToR, los enlaces `.onion` y las ventajas y desventajas de usarla, así como la diferencia con una VPN tradicional	
	Puedes construir una red de dispositivos tuyos con una VPN P2P con *ZeroTier* y gestionarlos de forma segura, así como distinguir estas VPN de las tradicionales	
	**Protección contra las redes oscuras**: Tu conocimiento sobre la seguridad de red te permite implementar un servidor fortificado capaz de resistir los intentos de ataque de red más comunes contra los servicios típicos y como usuario final. Además, te permite usar la red de forma anónima, incluyendo usar servicios tuyos de forma que solo tu puedas verlos y acceder a ellos en circunstancias normales	