

EVASION DE RESTRICCIONES EN LA RED

Presentación realizada por:

Jesus Osorio

Harold Gaviria

Juan David Guzman

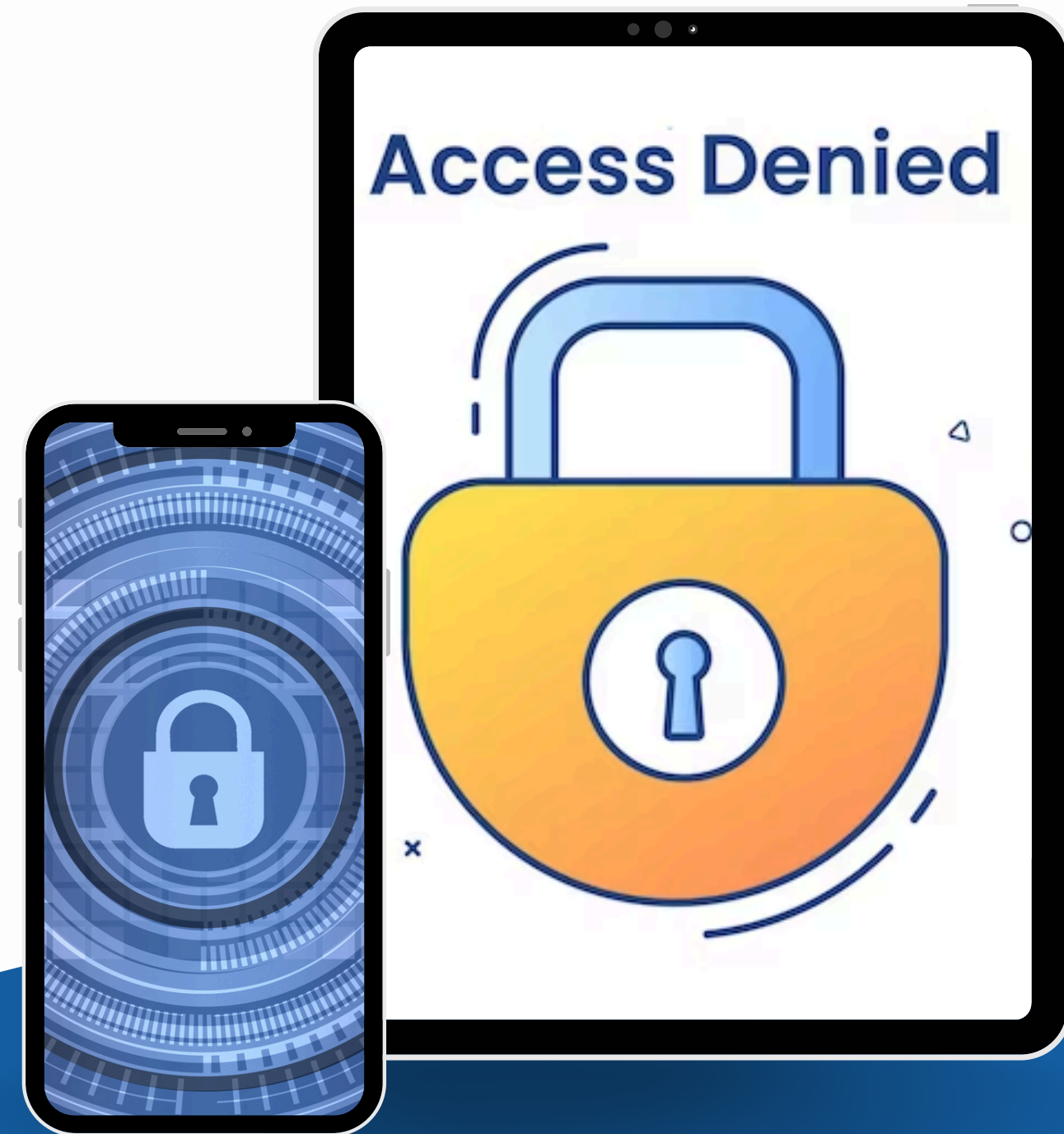
Norman Pabón Gallo

Cristian David Chacón



INTRODUCCIÓN

La evasión de restricciones en redes se refiere a las técnicas usadas para eludir controles de seguridad y acceder a recursos o información restringida. Estas técnicas incluyen el uso de VPN, proxies y túneles SSH, y buscan sortear barreras como cortafuegos y filtros de contenido. Mientras algunos usuarios lo hacen para acceder a contenido bloqueado, otros pueden tener fines maliciosos, como el robo de datos. Esta práctica plantea desafíos importantes para la seguridad de las redes, requiriendo de estrategias avanzadas para detectar y prevenir accesos no autorizados.

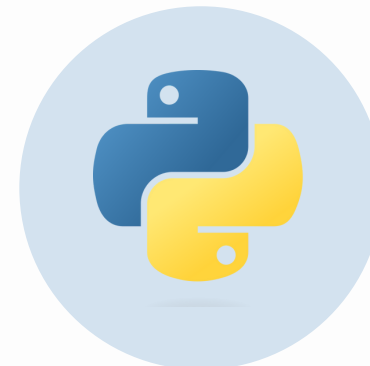


Tecnologías a usar



Sshuttle

Servidor proxy transparente que funciona como una VPN básica. Evita la creación de reenvíos de puertos individuales para cada host/puerto en la red remota. Supera limitaciones y lentitudes del reenvío de puertos de openssh y evita el mal rendimiento del TCP-over-TCP de PermitTunnel. Redirige el tráfico a través de SSH. Soporta túneles DNS. No requiere permisos admin. Compatible con Linux y MacOS.



Python

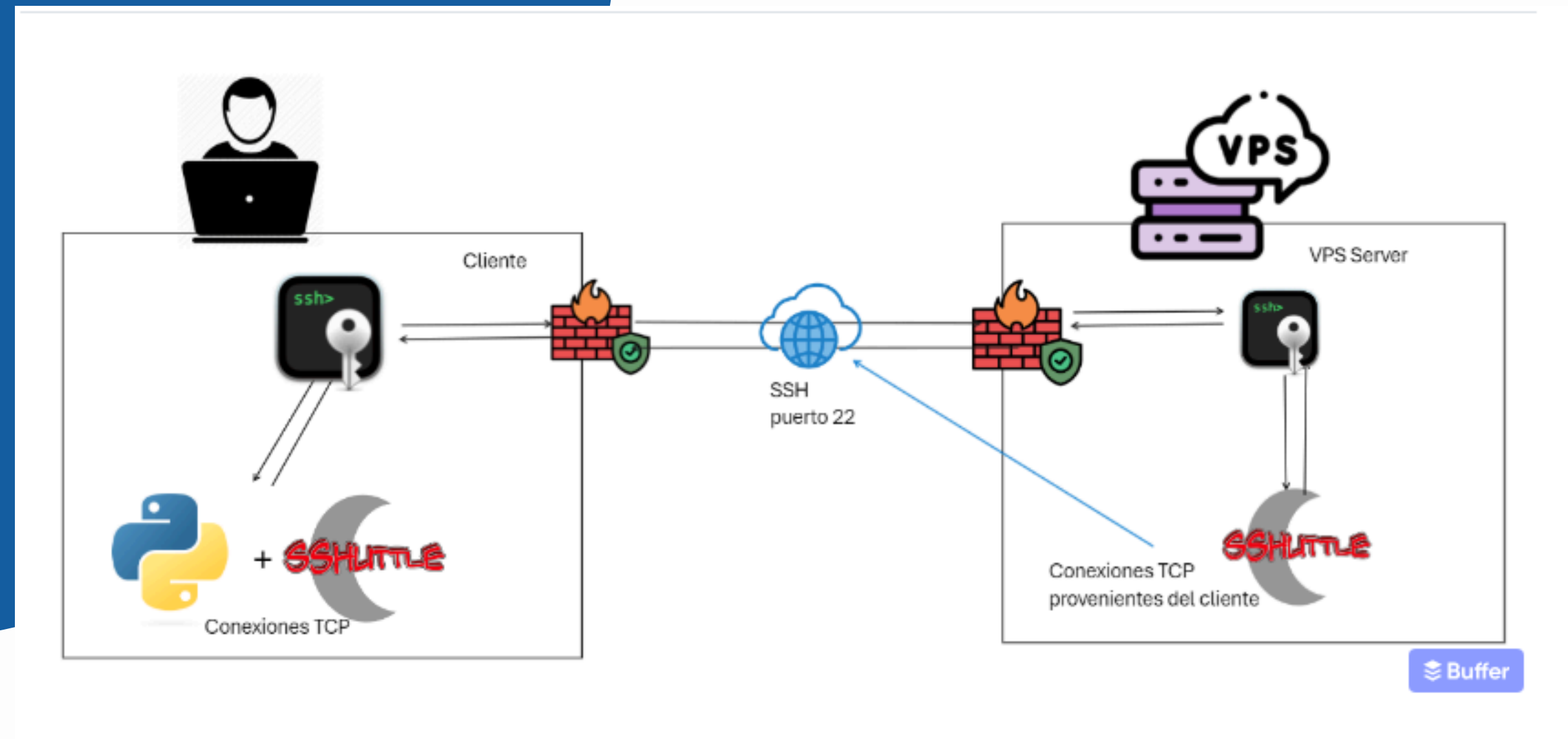
Automatiza la configuración, mantenimiento y ejecución de SSHUTTLE, facilitando el establecimiento de conexiones seguras y optimizando el flujo de trabajo. Se pueden escribir scripts que configuren conexiones SSH y manejen túneles de manera eficiente. Gestiona y monitorea conexiones TCP creadas por SSHUTTLE; proporciona una capa adicional de control y flexibilidad.



Secure Shell (ssh)

Protocolo que permite conexiones seguras a máquinas remotas mediante cifrado y autenticación. Usado para administración remota, transferencia segura de archivos y ejecución de comandos. SSHUTTLE y Python colaboran para automatizar y gestionar túneles SSH, redirigiendo el tráfico TCP del cliente a través de firewalls hacia un servidor VPS. Esto ofrece seguridad, automatización y flexibilidad a conexiones eficientes sin la complejidad de VPNs tradicionales.

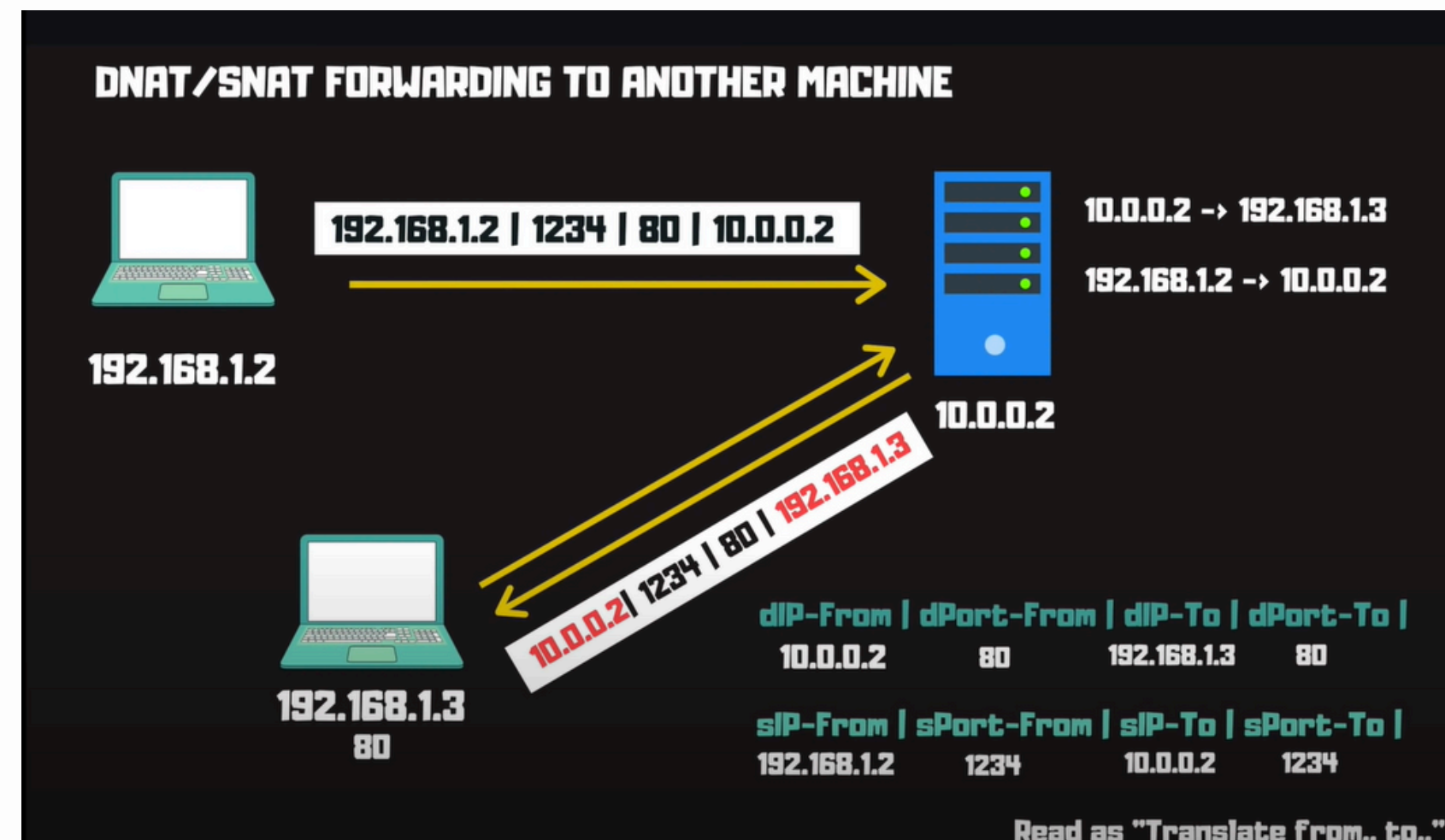
Evadiendo restricciones con Sshuttle



Sshuttle

Es un proyecto de código abierto que nos sirve como una especie de proxy, el cual aprovecha la herramienta de software iptables, la cual está disponible en todas las distros de Linux y permite configurar las tablas de enrutamiento, cadenas y reglas de red.

Con sshuttle, primero se establece una conexión cifrada haciendo uso del protocolo SSH, aprovechando que muchos firewalls no bloquean dicho protocolo, pues se suele utilizar bastante en entornos de empresariales.



SSHUTTLE

Sshuttle una vez tiene conexión por SSH, redirige las peticiones hechas con protocolo TCP haciendo uso de iptables a la máquina remota a la que estemos conectados (Servidor con sshuttle).

VPN

Una Red Privada Virtual (VPN, por sus siglas en inglés) es una tecnología que permite crear una conexión segura y encriptada sobre una red menos segura, como Internet. como podría ser, estar conectado por medio de una red pública.

Mediante esta conexión, generalmente cifrada, podemos enrutar el tráfico de nuestro dispositivo hacia la red en la que se encuentre el servidor. De esta manera, mientras el servidor vpn al que nos conéctenos sea de confianza, nuestro tráfico estará seguro de posibles ataques de captura de paquetes que se puedan dar en la red pública por parte de un actor malicioso.

Cabe aclarar que actualmente la mayoría de las conexiones se encuentran cifradas en la capa de aplicación, en el caso de la web, esta se cifra mediante SSL.



VENTAJAS Y DESVENTAJAS

VPN

Seguridad: Ofrece cifrado robusto y autenticación, protegiendo los datos del usuario.

Velocidad: Puede reducir la velocidad de conexión debido al cifrado y la latencia con el servidor. Adicionalmente, puede llevar a caídas de conexión

PROXY

Simplicidad: Fácil de configurar para navegación web básica.

Seguridad limitada: Los proxies, especialmente los HTTP, no cifran el tráfico, lo que puede ser un riesgo de seguridad.

Shuttle

Seguridad: Utiliza SSH, que proporciona un alto nivel de cifrado y autenticación.

Requiere conocimientos técnicos: No es tan intuitivo como algunas soluciones de proxies sin embargo es más fácil la configuración en comparación a un VPN

Comparativa

Característica	Shuttle	VPN	Proxy
Nivel de seguridad	Alto (SSH)	Alto (cifrado robusto)	Bajo(HTTP)
Privacidad	Moderada	Alta	Baja a Moderada
Flexibilidad	Alta (aplicaciones específicas)	Moderada (todo el tráfico)	Moderada (depende del tipo)
Velocidad	Variable (posible reducción)	Buena cuando se usa protocolo UDP	Varia segun la disponibilidad del servidor y recursos

Saltando bloqueos de páginas web con sshuttle



Configuracion del servidor

configuracion de la máquina virtual en azure :

- Crear una maquina virtual con Ubuntu 22.x
- Abrir el puerto 22 para el SSH para realizar la conexion
- Instalar python3
- Instalar sshuttle mediante apt



Maquina de azure

- Ubuntu 22.04
- 1 vCPU
- 1 GiB
- Costo 0.0104 USD/hr
- Costo mensual 2.496 USD por 8 horas diarias usando la maquina en 1 mes



Azure Virtual Machine

Configuración de sshuttle

Despues de haber realizado la configuracion del servidor descargamos archivo SshuttleServer.sh del repositorio, le damos permisos de ejecucion y ejecutamos el script, este nos va a dar ciertas opciones, seleccionamos la primera opcion que va a realizar la instalacion de shuttle para el servidor VPN

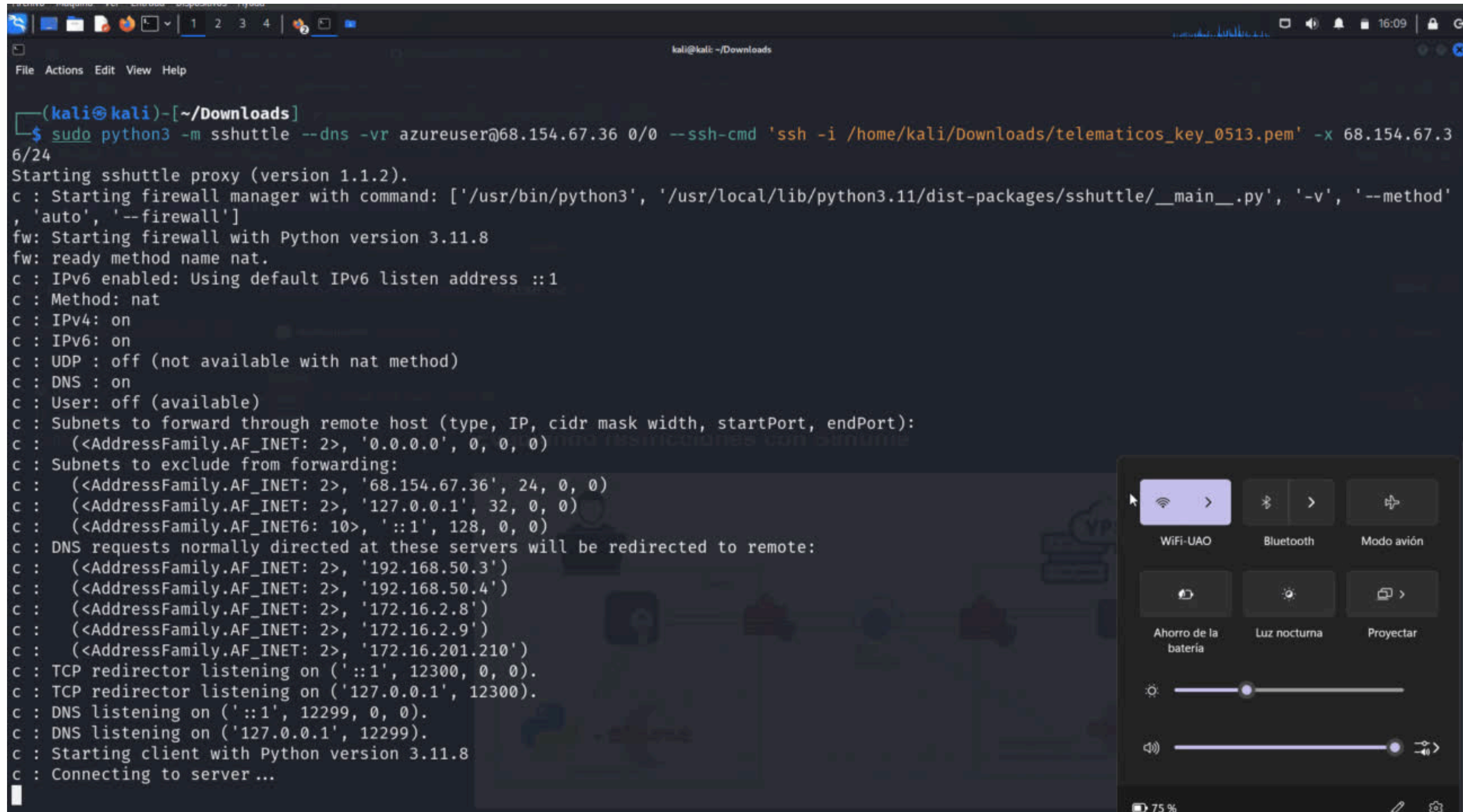


Configuracion Cliente

En nuestra maquina linux tenemos que tener python3 luego realizamos este comando para la instalacion de sshuttle **"python3 -m pip install sshuttle"** y despues ejecutamos este otro comando **"sudo python3 -m sshuttle --dns -vr azureuser@68.154.67.36 0/0 --ssh-cmd 'ssh -i ./telematicos.pem' -x 68.154.67.46/24"** para conectarnos al servidor VPN



Demo



```
(kali@kali)-[~/Downloads]
$ sudo python3 -m sshuttle --dns -vr azureuser@68.154.67.36 0/0 --ssh-cmd 'ssh -i /home/kali/Downloads/telematicos_key_0513.pem' -x 68.154.67.36/24
Starting sshuttle proxy (version 1.1.2).
c : Starting firewall manager with command: ['/usr/bin/python3', '/usr/local/lib/python3.11/dist-packages/sshuttle/__main__.py', '-v', '--method', 'auto', '--firewall']
fw: Starting firewall with Python version 3.11.8
fw: ready method name nat.
c : IPv6 enabled: Using default IPv6 listen address ::1
c : Method: nat
c : IPv4: on
c : IPv6: on
c : UDP : off (not available with nat method)
c : DNS : on
c : User: off (available)
c : Subnets to forward through remote host (type, IP, cidr mask width, startPort, endPort):
c :   (<AddressFamily.AF_INET: 2>, '0.0.0.0', 0, 0, 0)
c : Subnets to exclude from forwarding:
c :   (<AddressFamily.AF_INET: 2>, '68.154.67.36', 24, 0, 0)
c :   (<AddressFamily.AF_INET: 2>, '127.0.0.1', 32, 0, 0)
c :   (<AddressFamily.AF_INET6: 10>, '::1', 128, 0, 0)
c : DNS requests normally directed at these servers will be redirected to remote:
c :   (<AddressFamily.AF_INET: 2>, '192.168.50.3')
c :   (<AddressFamily.AF_INET: 2>, '192.168.50.4')
c :   (<AddressFamily.AF_INET: 2>, '172.16.2.8')
c :   (<AddressFamily.AF_INET: 2>, '172.16.2.9')
c :   (<AddressFamily.AF_INET: 2>, '172.16.201.210')
c : TCP redirector listening on ('::1', 12300, 0, 0).
c : TCP redirector listening on ('127.0.0.1', 12300).
c : DNS listening on ('::1', 12299, 0, 0).
c : DNS listening on ('127.0.0.1', 12299).
c : Starting client with Python version 3.11.8
c : Connecting to server ...
```