Evadiendo restricciones de red en entornos con políticas estrictas

Norman A. Pabon - 2201424. Harold Gaviria-2160215. Jesus Osorio-2190700
Juan Guzman-2205483. Cristian Chacon-0000000

Universidad autonoma de Occidente, departamento de telecomunicaciones
Cali, Colombia

Abstract— Nowadays, it is considered a normal practice for institutions, both educational and governmental, to block Internet resources to prevent access to unauthorized sites, whether for piracy, possible malware or entertainment content. These blocks are usually done through the entity's firewall, adding domain rules and/or deep packet inspection to block packets of certain protocols, such as BitTorrent. However, it is common for these blocks to consist of a series of rules automated by companies, which block resources necessary for learning, as well as freedom of expression and access to information, a case that is common in China with its large firewall that seeks to limit access to information to the civilian population.

Keywords—freedom, expression, network, blockade, vpn, isp.

Resumen: Hoy en día, se considera una práctica normal, los bloqueos realizados a recursos de internet por parte de las instituciones, tanto educativas como gubernamentales, con la finalidad de evitar el acceso a sitios no permitidos, ya sea por piratería, posible malware o contenido de entretenimiento. Estos bloqueos usualmente se realizan mediante el firewall de la entidad, agregando reglas de dominios y/o inspección profunda de paquetes, para bloquear paquetes de ciertos protocolos, como es el caso del BitTorrent. Sin embargo, es común que estos bloqueos consistan en una serie de reglas automatizadas por compañías, que bloquean recursos necesarios tanto para el aprendizaje, así como la libertad de expresión y acceso a la información, caso que es común en China con su gran firewall que busca limitar el acceso de información a la población civil.

Palabras clave: libertad, expresión, red, bloqueos, vpn, isp.

Introducción:

El bloqueo de recursos de red no necesarios es una práctica extendida en el entorno empresarial, pues nos permite controlar los recursos a los que tienen acceso los dispositivos conectados a la red, permitiendo que, en entornos, por ejemplo, bancarios, los computadores solamente puedan acceder a los recursos justos y necesarios, como la página del banco, el servidor NTP para la sincronización de las fechas y otros recursos únicos del área. Prohibiendo el acceso a redes sociales, para evitar desde la distracción por parte de los empleados hasta el posible fraude.

En los entornos educativos, los bloqueos suelen ser menos restrictivos, se suelen hacer uso de etiquetas de dominios y con estas, se agregan ciertas etiquetas a una lista negra (lista de bloqueo), algunos ejemplos de etiquetas son: videojuegos, piratería, hacking, malware…

Así mismo, bloquean también ciertos protocolos de red, como los usados para la comunicación P2P, la cual es ampliamente usada para la distribución de contenido pirata.

Los DPI (inspección profunda de paquetes):

Los DPI suelen utilizare en los firewalls como parte de los sistemas de detección de intrusiones. La detección de intrusiones suele ser benéfica para interceptar y prevenir la propagación de virus, gusanos y spyware en la red. Sin embargo, puede tener un impacto negativo, por ejemplo, a la hora de la censura financiada por estados.

Algunas técnicas usadas para la inspección profunda de paquetes son:

Detección por firma / patrones: se analiza cada paquete de red contra una base de datos con los ataques de red conocidos, en busca de patrones maliciosos conocidos. Este tipo de detección funciona solo con las amenazas conocidas y requiere de una actualización continua de las bases de datos.

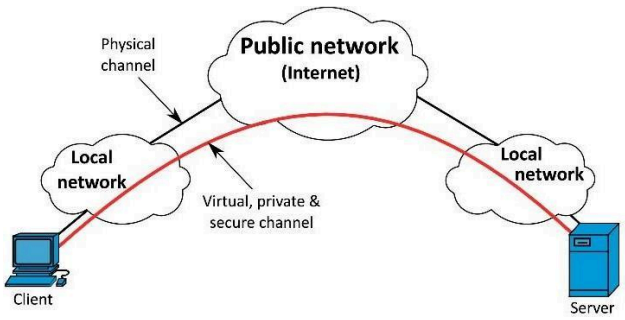
Anomalía de protocolo: funciona con denegación por defecto, donde el firewall determina que trafico de red puede “pasar” según las definiciones de los protocolos de red.

Inspección de Https: recientemente, con la estandarización de uso de comunicaciones seguras mediante cifrado con SSL/HTTPS, los firewalls modernos ofrecen inspección de paquetes https, donde ellos desencriptan el tráfico https para analizarlo, es una estrategia muy parecida a la usada por los ataques de hombre en medio (MiTM).

Los VPN (Virtual prívate network):

Una VPN (Virtual Prívate Network), es un mecanismo para crear una conexión segura entre un cliente (como nuestro equipo= y una red de manera segura estando en un entorno de red inseguro, 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.



Para evitar estos bloqueos de red, los individuos pueden hacer uso de vpn. Un vpn puede resumirse como establecer una conexión a un servidor remoto para enrutar el tráfico de red del dispositivo conectado a este servidor y que este servidor (vpn) haga los llamados a los servicios requeridos por los usuarios.

El tráfico del cliente al VPN va cifrado de punto a punto, dificultando las reglas de bloqueo de dominio. Pues, al estar cifrado de extremo a extremo, solo el cliente y el VPN saben el contenido que se está solicitando. Esto permite acceder a contenido que usualmente se encuentra bloqueado para ciertas regiones, como podría ser el catálogo de Netflix.

Entre los protocolos más usados por los servidores VPN tenemos los siguientes: OpenVPN, WireGuard, L2TP/IPsec, IKEv2/IPsec, PPTP and SSTP.

Configuración del servidor:

A la hora de decidir usar un VPN, tenemos varias opciones comerciales, por ejemplo: nordvpn, protonvpn, lastvpn… Entre otras, las cuales se han hecho populares gracias a la cantidad de anuncios que nos salen ofreciendo estos servicios como una solución milagrosa “anti-hackers”. Sin embargo, estas opciones suelen estar en un rango de precio mensual de 6 a 12USD/mensuales.

Pese a que nos ofrecen varios servidores de vpn alrededor del mundo para conectarnos, hay que tener en cuenta que las direcciones IPs públicas de estos servidores, son las mismas para todos los usuarios que tenga la compañía prestadora del servicio de vpn. Permitiendo así, que esas direcciones IP sean marcadas rápidamente como servidores VPN y bloqueadas por los firewalls.

Para motivos de investigación, configuraremos un servidor VPN, haciendo uso de una máquina virtual alojada en Azure, puntualmente usaremos una maquina B1s, que nos ofrece las siguientes especificaciones por 7.5USD/mes:

1 Core de 3-4GHz 1Gb de

memoria RAM 30Gb de

SSD

OS: Ubuntu 22.04 LTS

El tener nuestro propio servidor vpn, no solo nos ofrece una mayor seguridad de la información, si no que nos permite tener una dirección IP publica propia, la cual no será categorizada como VPN mediante reglas de firewall. También, al ser el servidor propio, podemos aprovechar los recursos de máquina que nos sobren para montar otros servicios de red sobre esta máquina virtual, como podría ser un servicio de correo electrónico, de archivos o de paginas web.

Ingreso a servidor remeto en Azure desde terminal:

Asignamos los permisos a las llaves SSH, esto es el equivalente a asignar permiso chmod 600 en sistemas Linux, es decir, permisos de solo lectura y escritura para el propietario del archivo. Esto es obligatorio por seguridad al usar el SSH con llaves RSA:

> icacls .\private.key /inheritance:r

> icacls .\private.key /grant:r "%username%":"(R)"

Nos paramos en la carpeta donde se encuentra la llave privada RSA para conectarnos al servidor:

- cd "~/Downloads/Proyecto bypass firewall"

Establecemos la conexión SSH con el equipo:

- Ssh -i ./telematicos_key_-513.pem azureuser@68.154.67.36

Al ingresar nos saldrá la pantalla de bienvenida, donde se nos especifica la versión actual del OS del servidor, en este caso Ubuntu 20.04.6 LTS.

Configuración del VPN con OpenVPN:

Para configurar el servidor VPN haremos uso del protocolo OpenVPN, por motivos prácticos usaremos el siguiente script automatizado para la configuración del OpenVPN:

- wget https://git.io/vpn -O openvpn-ubuntu- install.sh

Contestamos las preguntas que nos salgan en pantalla, según el tipo de autenticación, encriptación, puerto y protocolo de transporte que vayamos a utilizar.

Para la configuración de nuestro servidor OpenVPN, haremos uso del protocolo de transporte TCP sobre el puerto 443, pese a que el UDP es el protocolo de transporte recomendado para servidores VPN por temas de

rendimiento, se hace uso del protocolo TCP por los beneficios que tiene a la hora de recuperar paquetes, adicionalmente se usa el protocolo 443, para saltar ciertas reglas de firewall, donde usualmente se permite el tráfico TCP por este puerto, debido a que es el más usado para el protocolo de aplicación HTTPS.

Una vez realizada la configuración, el script nos va a preguntar para la creación de un cliente para el servidor Vpn, en este caso crearemos un usuario denominado "NormanClient" el cual no va a requerir autenticación con clave, si no la llave RSA generada.

```
Tell me a name for the client.
The name must consist of alphanumeric character. It may also ir
Client name: NormanClient

Do you want to protect the configuration file with a password?
(e.g. encrypt the private key with a password)
   1) Add a passwordless client
   2) Use a password for the client
Select an option [1-2]: 1

* Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020
```

Si nos dirigimos al escritorio en la máquina virtual, veremos que se nos ha generado un archivo de configuración de cliente OpenVpn. En este caso, con el nombre del cliente "NormanClient" y con extensión .ovpn. Si exploramos el contenido del archivo de configuracion del cliente, veremos que tiene la especificación de que se va a realizar una conexión TCP, la dirección ipv4 del servidor (172.200.178.153) y el puerto de conexión (443). También se puede observar las opciones de cifrado del protocolo TLS y el inicio del certificado en cuestión.

```
PrivateIOT@IotClass:~$ cat NormanClient.ovpn
client
proto tcp-client
remote 172.200.178.153 443
dev tun
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
verify-x509-name server_gVs8TcEPBTToMkmf name
auth SHA256
auth-nocache
cipher AES-128-GCM
tls-client
tls-version-min 1.2
tls-cipher TLS-ECDHE-ECDSA-WITH-AES-128-GCM-SHA256
ignore-unknown-option block-outside-dns
setenv opt block-outside-dns # Prevent Windows 10 DNS leak
verb 3
<ca>
-----BEGIN CERTIFICATE-----
MIIB1zCCAX2gAwIBAgIUaZU8QFRxAMs1TpFlGE54/KsFztYwCgYIKoZIzj0
```

conexión al servidor OpenVPN:

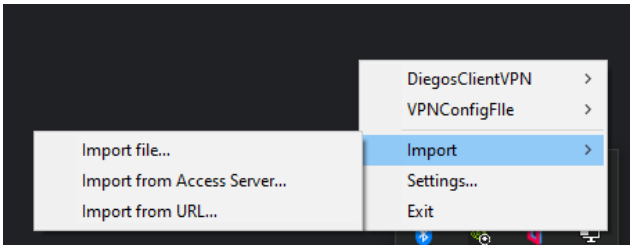
El procedimiento para establecer la conexión con el servidor OpenVPN varia según la plataforma desde la cual nos encontremos, por ejemplo, en el caso de Debian, el cliente de OpenVPN viene por defecto con el resto de paquetes base de la distro. Si nos encontraremos en Windows, haría falta descargar el cliente de OpenVPN "openvpn connect", el cual es gratuito y lo podemos encontrar desde su página principal en : ***https://openvpn.net/client/client-connect-vpn-for-windows/***

Para conectarnos desde Debian con la interfaz KDE Plasma, abrimos la configuracion de conexiones del sistema, le picamos al simbolo de agregar una nueva conexion y buscamos entre los VPNs la opcion de "Importar conexion VPN", luego escogemos el archivo correspondiente a nuestra configuracion del cliente y este quedara guardado en nuestras conexiones.

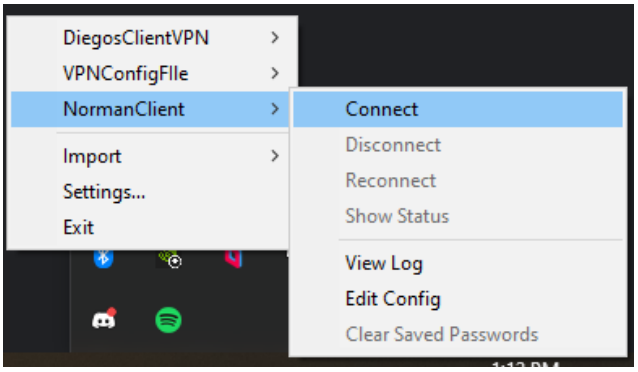
Conexión desde Windows:

Una vez instalado el cliente "OpenVPN connect", contaremos con una aplicación llamada "OpenVPN GUI"

Al abrirla, en la barra de tareas nos saldrá un icono de un computador con un candado, si le presionamos encima nos dará la opcion para importar nuestro archivo de configuración del VPN.

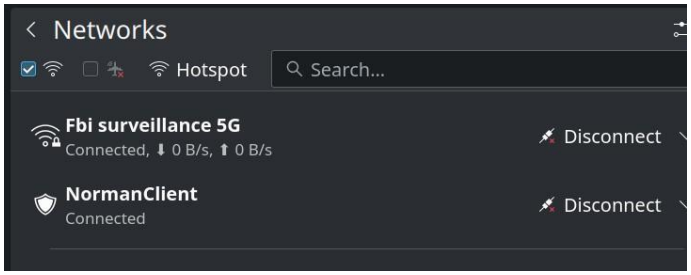


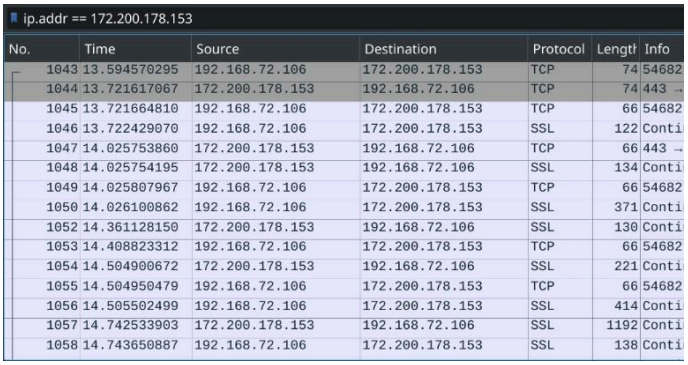
Una vez hemos buscado e importado nuestra configuración del vpn, al dirigirnos de nuevo al icono nos saldrá el perfil de conexión que hemos importado, brindándonos la opcion de conectarnos al servidor VPN.



Estableciendo conexión con el VPN desde un entorno sin restricciones

Una vez hayamos importado el perfil de conexión para el cliente, nos aparece la opcion de conectarnos con a ese vpn desde las redes del equipo. En este caso probaremos la conexión fuera del Wifi de la universidad para evitar que alguna regla de red nos impida probar el servicio de vpn que acabamos de montar, en este caso usamos una red wifi denominada “Fbi surveillance”, con la cual podemos apreciar que pudimos establecer conexión con nuestro servidor



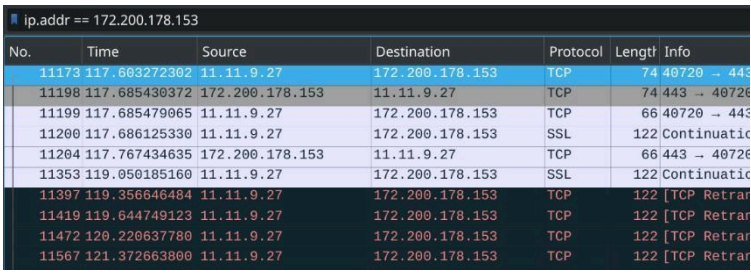

ip.addr == 172.200.178.153

No.	Time	Source	Destination	Protocol	Lengt	Info
1043	13.594570295	192.168.72.106	172.200.178.153	TCP	74	54682
1044	13.721617067	172.200.178.153	192.168.72.106	TCP	74	443 →
1045	13.721664810	192.168.72.106	172.200.178.153	TCP	66	54682
1046	13.722429070	192.168.72.106	172.200.178.153	SSL	122	Conti
1047	14.025753860	172.200.178.153	192.168.72.106	TCP	66	443 →
1048	14.025754195	172.200.178.153	192.168.72.106	SSL	134	Conti
1049	14.025807967	192.168.72.106	172.200.178.153	TCP	66	54682
1050	14.026100862	192.168.72.106	172.200.178.153	SSL	371	Conti
1052	14.361128150	172.200.178.153	192.168.72.106	SSL	130	Conti
1053	14.408823312	192.168.72.106	172.200.178.153	TCP	66	54682
1054	14.504900672	172.200.178.153	192.168.72.106	SSL	221	Conti
1055	14.504950479	192.168.72.106	172.200.178.153	TCP	66	54682
1056	14.505502499	192.168.72.106	172.200.178.153	SSL	414	Conti
1057	14.742533903	172.200.178.153	192.168.72.106	SSL	1192	Conti
1058	14.743650887	192.168.72.106	172.200.178.153	SSL	138	Conti

Al observar nuestros paquetes capturados haciendo uso de Wireshark, vemos que se establece la conexion TCP entre el cliente y el servidor, seguido de la conexion SSL.

Estableciendo conexión desde un entorno con restricciones de VPN

Si intentamos establecer la conexión con nuestro servidor VPN desde la red de la universidad, la cual tiene restricciones del protocolo VPN con el FortiGuard, podemos ver que intenta establecer la conexión, pero luego de excederse el tiempo máximo de espera, no puede resolver. Pero ¿que lleva a que falle la conexión? Para contestar la pregunta, nos toca revisar los paquetes de red enviados del cliente al VPN


ip.addr == 172.200.178.153

No.	Time	Source	Destination	Protocol	Lengt	Info
11173	117.603272302	11.11.9.27	172.200.178.153	TCP	74	40720 → 443
11198	117.685430372	172.200.178.153	11.11.9.27	TCP	74	443 → 40720
11199	117.685479065	11.11.9.27	172.200.178.153	TCP	66	40720 → 443
11200	117.686125330	11.11.9.27	172.200.178.153	SSL	122	Continuatio
11204	117.767434635	172.200.178.153	11.11.9.27	TCP	66	443 → 40720
11353	119.050185160	11.11.9.27	172.200.178.153	SSL	122	Continuatio
11397	119.356646484	11.11.9.27	172.200.178.153	TCP	122	[TCP Retran
11419	119.644749123	11.11.9.27	172.200.178.153	TCP	122	[TCP Retran
11472	120.220637780	11.11.9.27	172.200.178.153	TCP	122	[TCP Retran
11567	121.372663800	11.11.9.27	172.200.178.153	TCP	122	[TCP Retran

Al capturar los paquetes de red desde Wireshark, podemos observar que inicialmente el cliente intercambia paquetes TCP con el servidor, pero luego se corta la transmisión y queda con “error de retransmisión”. Este error se presenta cuando hay un error en la transmisión y los paquetes se “caen” o "tumban", en otras palabras. Para nuestro caso, este error nos da la idea de que los paquetes están siendo detenidos por el firewall de Fortinet, al no estar aprobado en las reglas definidas por la organización.

Para este tipo de bloqueos, las alternativas suelen ir de la mano con ofuscar el protocolo de vpn usado, sin embargo, esto es poco práctico frente a firewalls con inspectores profundos de paquetes avanzados. Pues además de reglas e identificación de protocolos por sus huellas digitales, también hacen seguimiento de las conexiones de los clientes, entonces al detectar un tráfico de red alto entre todo el tráfico de un cliente y una dirección ipv4, lo interpretan como una conexión a un VPN.

Otras alternativas consisten en hacer uso de proxys intermediarios, entre la conexión del cliente y el servidor VPN, esto podría lograrse con un Proxy SOCK5 o herramientas más dirigidas a este ámbito, como ShadowSocks, el cual a diferencia de los proxys que dependen de SOCK5, utiliza cifrado en sus conexiones, mejorando la seguridad del protocolo y dificultando la inspección de los paquetes de red.

Una alternativa a una extensa configuración: SSHUTTLE

En este escenario bajo las políticas de red que estan activas, podemos verificar que tenemos acceso a nuestro servidor a través del protocolo SSH, permitiéndonos hacer uso de este protocolo para enrutar nuestros paquetes de salida. Si fuésemos a usar el protocolo SSH nativo de nuestro servidor, este solo nos permitirá enrutar un puerto de red a la vez.

Para esto existe una herramienta opensource creada por la comunidad con la finalidad de implementar el enrutamiento de todo el tráfico del dispositivo por SSH, la herramienta en cuestión se llama sshuttle.

Entre los beneficios de sshuttle, esta que no requiere permisos de administrador (root) para ser ejecutado y puede ser instalado desde el gestor de paquetes de Python (PIP) o bien, si nos encontramos en una distro como Debian, con el gestor de paquetes APT. El lado negativo, es que sshuttle solo funciona de manera nativa (sin configuración adicional) en Linux y Macos, para poder hacer uso de la herramienta desde Windows toca hacer uso de Vagrant, sin embargo, eso no está en el alcance de este documento.

Instalación en cliente:

Como sshuttle fue diseñado para trabajar sobre el servicio de SSH, entonces no requerimos configuración adicional, puesto que, si ya tenemos un servicio SSH activo y funcionando, es todo lo que necesitan

python3 -m pip install sshuttle

En nuestro caso específico, que nos identifiquemos en la máquina virtual por medio de la llave privada generada, debemos usar el siguiente comando:

- sudo python3 -m sshuttle --dns -vr user@yourserver.com 0/0 --ssh-cmd 'ssh -i /your/key/path.pem' -x ipVps/subnet

Quedando para nuestro contexto de la siguiente manera:

- 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

explicación del comando y banderas de Sshuttle:

0/0: Nos permite especificar que redirija todas las ipv4 con todas las subnets. El comando también podría escribirse como 0.0.0.0/0 para la misma finalidad. Si quisiéramos incluir las direcciones ipv6 usamos la flag ::/0.

--dns: Captura el trafico al dns local y los redirecciona al servidor remoto para resolver allá la petición dns.

--vr: Habilita modo verbose y recibe el usuario, servidor y puerto a conectar.

-x: Excluye la subnet especificada del forwading. En este caso, excluimos la dirección ipv4 del servidor para prevenir bucles de transmisión.

--ssh-cmd: Permite pasarle parámetros adicionales a la conexión por SSH, como en nuestro caso que le pasamos al cliente SSH la llave privada para conectarnos al servidor.

Saltando bloqueos de páginas web con sshuttle:

Una de las páginas bloqueadas en nuestro contexto, es la página Scihub. Scihub, para los que no lo conozcan, es una biblioteca virtual con polémica detrás, pues en ella puedes encontrar múltiples papeles de investigación que normalmente estan detrás de un muro de pago, requiriendo una suscripción a servicios como Springer para poder acceder a ellas. El papel que cumple Scihub es importante en el mundo académico, aunque tenga su polémica por temas de derechos de autor, este a su vez permite que personas de lugares del mundo con menos recursos, tengan papeles investigativos al alcance de sus manos sin necesidad de adquirir las costosas licencias.

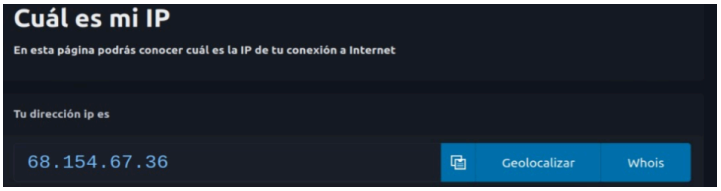
Esta página por defecto está bloqueada en varias reglas de red, por ejemplo, cuando intentamos acceder a la página desde la red inalámbrica que esta con bloqueo, nos redirige a la siguiente página:

Como se puede apreciar, nuestra petición ha sido denegada por el firewall y nos bloqueado la conexión con la página web, redirigiéndonos a una notificación de que la página web se encuentra no permitida.

Haciendo uso de la herramienta para seguridad web BurpSuite, podemos ver los métodos http usados para el llamado de la página web, vemos que inicialmente hacemos la petición para obtener la página sci-hub.ru, sin embargo, automáticamente se nos redirige a la página estándar de bloqueo hosteado por el firewall de Fortinet. devolviéndonos el mensaje que presenciamos anteriormente de “página web no permitida”.

Al ver el intercambio de paquetes HTTP entre nuestro cliente y la página web, podemos apreciar que solo se comunica con la dirección ip 68.154.67.36, al hacer una búsqueda en la página cual-es-mi-ip.net, nos damos cuenta de que la ip con la que intercambiamos paquetes pertenece a Azure. Esto, al documentarse

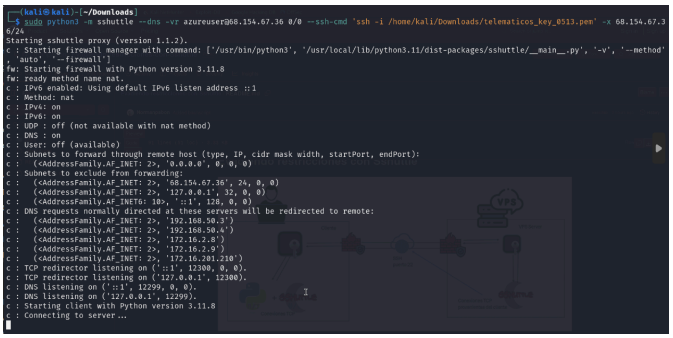
acerca de las normas de red que rigen en el Firewall de Fortinet, este está integrado con Azure para las corporaciones, siguiendo este orden de ideas, el Fortinet se encarga de revisar nuestra petición http en el servidor de Azure asociado al firewall de corporación y decide si deja o no deja pasar el paquete de red.

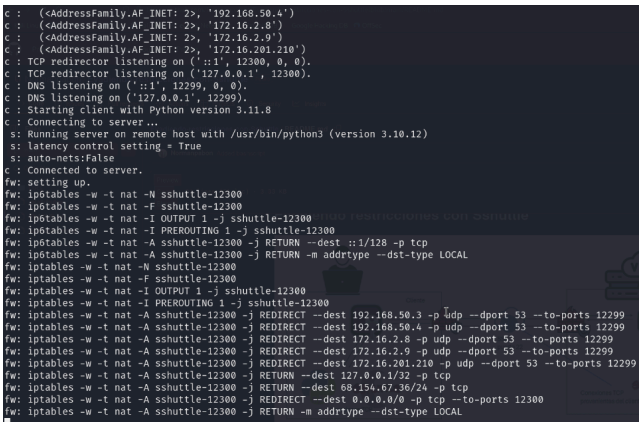


Ejecutando Sshuttle:

Ya que corroboramos que tenemos una regla de red activa respecto al sitio Scihub, ejecutemos el programa "Sshuttle" el cual instalamos anteriormente y conectémonos a nuestro servidor remoto:

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





Una vez verificamos en la consola que tenemos conexión con el servidor, podemos proceder a hacer nuestra nueva petición a la página de SciHub.

Como podemos observar, ahora nuestra petición de la página si ha obtenido el resultado esperado, si revisamos el historial de peticiones http, obtuvimos correctamente la página sin el redireccionamiento generado por el firewall de Fortinet

Si nos dirigimos al log de Sshuttle generado desde la terminal donde tenemos el servicio corriendo, podemos apreciar el redireccionamiento de paquetes que se está llevando a cabo a través de este aplicativo hacia nuestro servidor en Azure:

Conclusiones:

Con las pruebas que realizamos, podemos ver que los Firewalls que cuentan con inspector profundo de paquetes, cada vez se estan volviendo mas avanzados. Brindando una mayor seguridad a la costa de la privacidad de los usuarios de la red.

En versiones anteriores de los firewalls que contaban con estas herramientas, no eran capaces de detectar con tanta facilidad los protocolos de VPN y detener las conexiones en el momento, como pudimos darnos cuenta, el cambiar el puerto de conexión al 443 usado por https, el protocolo de transporte a TCP y el cifrado a SSL no fue suficiente para poder saltarnos el bloqueo de red.

Para este tipo de casos, hay métodos de ofuscado fuera de los alcances de este documento, como seria hacer uso del protocolo obfs4 creado por el proyecto Tor de la red cebolla para evitar bloqueos en el trafico de la red Tor con firewalls como el de China. De esta manera, el cliente se conectaría al servidor VPN mediante el proxy creado por obfs4, agregando la complejidad de que ahora el cliente debe tener adicional al software para establecer la conexión al VPN, otro para establecer la conexión con el proxy Obfs4.

Por otro lado, si el bloqueo que queremos saltarnos está asociado a una página web, podemos sacarle ventaja al servicio de Sshuttle, que en nuestro caso nos permitió acceder a la pagina de SciHub la cual también se encontraba bloqueada.

La mayor desventaja que encontramos hasta ahora con sshuttle, es que el único protocolo de transporte que soporta es TCP, si bien hay muchas páginas web que funcionan aun sobre TCP, la mayoría son UDP por términos de rendimiento. Igualmente, al ser TCP, también tenemos la ventaja del uso de este protocolo de transporte, por protocolos de aplicación altamente usados para el intercambio de archivos, como lo es el protocolo BitTorrent.

El mayor beneficio que nos brinda sshuttle respecto a las otras opciones, es que el único requerimiento para el cliente es tener instalado Python, pues como se mencionó en un inicio, sshuttle está disponible desde el gestor de paquetes de Python, PIP.

Referencias

1. https://es.wikipedia.org/wiki/Inspecci%C3% B3n_profunda_de_paquete
2. https://www.techtarget.com/searchnetworkin g/definition/deep-packet-inspection-DPI
3. https://2019.www.torproject.org/docs/plugg able-transports
4. https://www.fortinet.com/lat/products/public-cloud-security/gcp
5. https://sshuttle.readthedocs.io/en/stable/man page.html
6. https://thematrix.dev/setup-go-shadowsocks- on-ubuntu/