Universitat Oberta de Catalunya

uoc.edu

Técnicas de evasión de elementos de red

Fernando Modamio Miguel
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
M6.151 - Hacking

Manuel Jesús Mendoza Flores
Víctor García Font

29 de diciembre de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Técnicas de evasión de elementos de red*
Nombre del autor:	*Fernando Modamio Miguel*
Nombre del consultor/a:	*Manuel Jesús Mendoza Flores*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	12/2020
Titulación:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*M6.151 - Hacking*
Idioma del trabajo:	*Castellano*
Palabras clave	*Técnicas evasión firewalls*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo de este trabajo ha sido la evasión de firewalls de capa 4, firewalls de nueva generación y firewalls de entornos cloud mediante técnicas de tunelización y Domain Fronting.

La metodología llevada a cabo ha sido comenzar a trabajar con el firewall más simple, iptables, que viene incluido en el kernel de Linux. Una vez estudiadas, entendidas y aplicadas las técnicas sobre este firewall de capa 4, se ha sustituido por un firewall de nueva generación del fabricante Fortinet, se han comprendido sus capacidades y se han configurado protecciones para que fueran eficientes contra las técnicas anteriores. Por último, se ha desplegado un entorno IaaS, con la protección de red gestionada por Azure Firewall y se han replicado una vez más las técnicas anteriores.

Como resultado, el firewall iptables fue vulnerable a todas las técnicas, el Fortigate bloqueó la mayoría y el Azure Firewall apenas fue eficaz frente a estas.

Las conclusiones obtenidas han sido que los entornos protegidos por firewalls de capa 4 no son seguros ante estas técnicas, que los protegidos por NGFW sí lo son (aunque no son capaces de detener una de las técnicas), pero requieren configuración experta y que el firewall de Azure no cumple los requisitos de seguridad de los firewalls de nueva generación.

Abstract (in English, 250 words or less):

This work's objective has been firewall evasion 4-layer, new generation firewalls and cloud environments firewalls through tunneling techniques and Domain Fronting.

The methodology has been start working with the simplest firewall, iptables, wich is included on Linux's kernel. Once the techniques about this 4-layer firewall had been studied, understood and applied, it has been replaced by a Fortinet new generation firewall, its capacities had been comprehended and the protections have been configured for them to be efficient against the previous techniques.

As a result, the iptables firewall was vulnerable to all techniques, the Fortigate blocked most of them and Azure Firewall was barely effective against them.

The conclusions obtained were that environments protected by 4-layer firewalls aren't safe against this techniques, but the ones protected by NGFW are (although they are not capable of block one of the them), though they require an expert's configuration and that Azure Firewall doesn't meet the security requirements of new generation firewalls.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día la industria de la seguridad informática es un valor al alza. Tradicionalmente las medidas de seguridad de las compañías se han centrado en securizar el perímetro mediante zonas desmilitarizadas DMZ, elementos en red y mediadas de protección de los puntos finales mediante motores de antivirus.

Debido al cambio de paradigma de los últimos años en el que la presencia de los activos de las compañías ha pasado de estar localizados en oficinas y datacenters a estar presentes alrededor del globo mediante servicios y aplicaciones cloud, han aparecido nuevas técnicas de protección como proxies en la nube, CASB o EDR. Sin embargo, las compañías siguen viendo imprescindible la seguridad perimetral de sus oficinas.

Las soluciones de cortafuegos han avanzado hasta los denominados NGFW o firewalls de nueva generación incorporando inspección por estado de paquete, inspección a nivel de aplicación, filtrado web por categoría o sandboxing.

El cambio on-premises – cloud hace preguntarse si los antiguos métodos de defensa perimetral (a pesar de sus avances tecnológicos) ¿siguen siendo efectivos en la realidad actual en la que el teletrabajo parece imponerse al trabajo presencial y en el que el mantenimiento de datacenters parece ser fácilmente externalizado? ¿Convendría a una pequeña oficina, sin servicios publicados en Internet, disponer de equipos de seguridad perimetral?¿Cómo de fácil sería vulnerar estos elementos?

En una oficina en la que hay implantado un sistema de control de navegación, un usuario podría evadirlo mediante el uso de un proxy público (siempre y cuando el acceso a este no haya sido bloqueado por el propio control), conectándose a una VPN que enrute todo el tráfico por la misma o simplemente utilizando su móvil como hotspot. En los dos primeros casos, una configuración robusta lo hubiera evitado. La tercera podría haberse evitado con una solución de NAC.

En este proyecto se espera evadir la seguridad de la mayor parte de estos elementos que podrían encontrarse en una red corporativa principalmente centrándose en el elemento más ampliamente utilizado: el firewall.

1.2 Objetivos del Trabajo

1. Evasión de firewall de código abierto.
2. Evasión de firewall comercial.
3. Evasión de firewall en infrastructura cloud.
4. Evasión de censura en la navegación.

1.3 Enfoque y método seguido

En la parte de evasión de firewall se va a trabajar con el firewall de código abierto iptables. El objetivo es demostrar la eficacia de las técnicas de SSH tunneling, DNS tunneling e ICMP tunneling para burlar la política de seguridad configurada en el equipo.

Se han valorado otras técnicas de evasión como IP fragmentation o Source Port Manipulation, pero finalmente se ha escogido investigar las diferentes técnicas de tunneling y Domain Fronting. Esto es debido a que, en el caso de IP fragmentation, sería necesario desplegar un firewall stateless o de "filtrado sin estado". La introducción de la inspección de estado del paquete data por parte de un firewall data del año 1994 y desde entonces todos los equipos fueron adoptando esta característica. Sobre la técnica de Source Port Manipulation, solo es útil para realizar un barrido de puertos abiertos con la herramienta nmap, saltándose el bloqueo del firewall.

Si los resultados son satisfactorios se sustituirá por un equipo comercial para ver sí estas técnicas son efectivas en estos. Se trabajará con una versión de prueba de alguno de los siguientes fabricantes: Fortigate, Palo Alto Networks o Check Point.

La elección se basará en la facilidad de obtener una licencia completa de cada uno de estos, la versión de prueba on-premises-VM del fabricante Fortigate no inspecciona SSL, pero la versión de prueba en AWS es completa, aunque había que pagar al proveedor del servicio Cloud por la utilización de la máquina. La versión de prueba del fabricante Palo Alto no dispone de las licencias de URL filtering ni Applications and Threats aunque de igual modo se puede desplegar en AWS de forma completa.
La versión de prueba de Check Point es completa en un despliegue on-premises en máquina virtual durante 15 días.

Por último, se va a preparar un entorno en la nube de Azure. La motivación es la siguiente. El modelo tradicional de CPD alojado físicamente en la entidad está llegando a su fin. La apuesta por el modelo de nube se ha ido imponiendo en los últimos años y ya quedan pocas empresas que no estén trabajando en un modelo híbrido. Entre las ventajas están los costes en infraestructura, escalado y mantenimiento de los equipos físicos. Pero, ¿qué pasa con la

seguridad? La seguridad sigue un modelo de responsabilidad compartida entre el cliente y el proveedor de servicios cloud. Aduciendo a la reducción de costes, estos últimos ofrecen soluciones de seguridad propias, a coste 0 o más baratas que las soluciones de los anteriores proveedores de seguridad on-premises. Si en el propio CPD se tenía una salida a Internet protegida por un firewall con inspección de capa de aplicación y su correspondiente licenciamiento, ¿cumplirá con los requisitos de seguridad el firewall suministrado por el proveedor de servicios cloud?

1.4 Planificación del Trabajo

Antes de realizar una planificación del trabajo es inevitable enumerar los recursos necesarios para la realización del proyecto y el listado de tareas a realizar durante la ejecución de este.

Por la parte de los recursos, en cada uno de los entornos a probar, serán necesarias como mínimo tres máquinas: la máquina que lanzará el ataque, la máquina destino y la máquina que contiene el elemento de seguridad en red. La máquina que lanza el ataque, en el caso del firewall levantará el túnel contra la máquina destino. La máquina destino será el otro extremo del túnel en el firewall. La máquina restante implementará el firewall.

Se procurará que todas ellas sean máquinas virtuales en el mismo sistema anfitrión, separadas por redes virtuales. En el caso de que por cuestiones de rendimiento/requisitos sea necesario incluir una segunda máquina física, será necesario contar con un switch para conectarlas en red.

Para probar las técnicas de evasión contra sistemas comerciales, será necesario conseguir una licencia de prueba de alguno de los principales fabricantes lo más completa posible.

En el caso de que se decida reproducir las evasiones en un entorno cloud, será necesario contar con una cuenta en alguno de los proveedores de servicios de nube.

A continuación, se muestra la lista de tareas planeada:

1 Evasión de iptables
- 1.1 Desarrollo del laboratorio y pruebas de conectividad
- 1.2 Implementación reglas SSH y evasión por medio de tunel.
- 1.3 Implementación reglas DNS y evasión por medio de túnel.
- 1.4 Implementación reglas ICMP y evasión por medio de túnel.

2 Evasión de Fortigate FWF40F
- 2.1 Desarrollo del laboratorio y pruebas de conectividad
- 2.2 Implementación reglas SSH y evasión por medio de túnel.

2.3 Implementación reglas DNS y evasión por medio de túnel.
2.4 Implementación reglas ICMP y evasión por medio de túnel.
2.5 Implementación reglas de filtrado web y evasión por medio de Domain Fronting.

3 Evasión de firewall cloud
3.1 Despliegue del entorno y pruebas de conectividad
3.2 Implementación reglas SSH y evasión por medio de túnel.
3.3 Implementación reglas DNS y evasión por medio de túnel.
3.4 Implementación reglas ICMP y evasión por medio de túnel.
3.5 Implementación reglas de filtrado web y evasión por medio de Domain Fronting.

En la ilustración 1 se muestra un diagrama de Gantt con la planificación de las tareas en el tiempo.

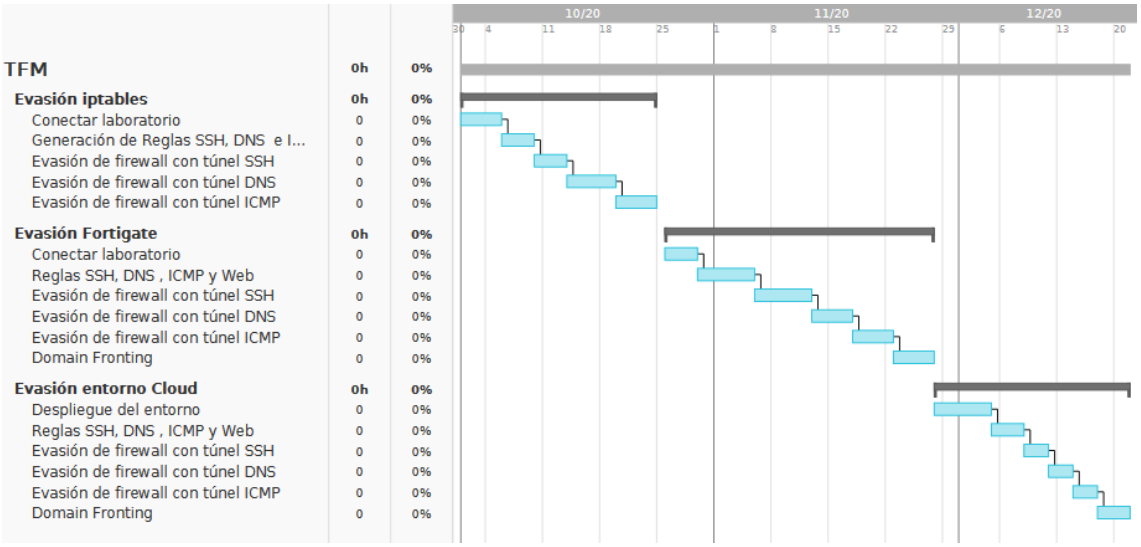


Ilustración 1 - planificación de tareas

1.5 Breve sumario de productos obtenidos

Al final de este trabajo se habrán obtenido tres entornos configurados. El primero de ellos será vulnerable a las técnicas de tunelización.

El segundo de ellos será un entorno más robusto en el que las técnicas de tunelización serán bloqueadas en gran medida por una solución comercial de firewall.

El tercero será un entorno cloud protegido por un firewall perimetral, que gestionará el acceso a la gestión de los dispositivos y que intentará contener las técnicas de evasión mediante tunelización.

1.6 Breve descripción de los otros capítulos de la memoria

Estado del arte: revisión de las últimas técnicas de evasión de sistemas de seguridad en red.

Diseño del laboratorio: en este capítulo se presentará el escenario de trabajo en el que se ejecutarán las técnicas de evasión y se detallará cómo y para qué se ha desplegado cada elemento.

Desarrollo del proyecto: en este capítulo se detallará el proceso de evasión y los resultados obtenidos.

Conclusiones: conclusiones extraídas de la experiencia realizada durante el desarrollo del proyecto.

Glosario: términos referidos en esta memoria de los cuales es necesario dar un pequeño detalle o explicación.

Bibliografía: catálogo de elementos de consulta visitados durante la realización del presente documento.

Anexos: configuración del firewall iptables en el momento del tunneling.

2. Estado del arte

2.1 Introducción

Para la evasión de sistemas de cortafuegos, los métodos más comunes son la tunelización, la fragmentación de IP y la manipulación de los puertos de origen.

Sobre la manipulación de puertos de origen, también conocida como *port spoofing,* se han consultado las siguientes webs en las que explican cómo funciona este ataque:

https://www.note4tech.com/post/how-to-bypass-firewall-using-nmap

En este caso el atacante hace uso de la herramienta nmap para realizar un escaneo de puertos indicando un puerto de origen “habitualmente abierto” como el 21, el 80 o el 443.

https://blog.gigamon.com/2019/06/04/port-spoofing-the-hidden-danger-to-your-network/

En la entrada de este blog el autor explica el supuesto en el que un atacante crea un cliente SSH corriendo en un puerto “habitualmente” abierto en una máquina dentro de la red corporativa, para después lanzar una conexión contra ese equipo y pivotar hacia el resto.

Estos ataques serían detectables por los modernos NGFW con inspección de capa 7. Esta detección no filtra únicamente por puerto, sino que es capaz de identificar el protocolo que intenta correr sobre este. Seguramente iptables podría verse comprometido mediante esta técnica.

Sobre la fragmentación de IP, en esta web se explica cómo con un tamaño suficientemente grande de datos a transportar, el paquete es fragmentado y debidamente dirigido a un puerto permitido, procesado, aceptado y reensamblado. Después del primer fragmento que iba dirigido a un puerto permitido, podría haber una segunda cabecera que apuntara a un nuevo destino o a un puerto bloqueado. Al ser reensamblado después del veredicto de permitir/denegar, el paquete llegaría a su destino.

https://resources.infosecinstitute.com/bypassing-packet-filters-with-ip-fragmentation-overlapping/#gref

El problema de este ataque es que únicamente funciona con firewalls “stateless” o sin inspección de estado. No funcionaria en iptables.

En las conferencias del ingeniero informático y pentester Candan Bolukbas subidas a su canal de la plataforma YouTube se explica cómo implementar distintos tipos de túneles para burlar la seguridad de firewalls de nueva generación. No explica la configuración de los firewalls que evade, pero demuestra mediante capturas de paquetes cómo la interfaz a la que está conectada el equipo detecta los paquetes enviados cómo DNS, ICMP o SSH, dependiendo del tipo de tunelizado realizado.

https://www.youtube.com/channel/UC7NmOUOJn0Vr5BrfzjgBD2A

También va a ser objeto de este trabajo el estudio de la técnica de Domain Fronting. Esta técnica es utilizada para burlar la censura gubernamental en países donde el estado actúa como censor en Internet, evitando el acceso a páginas que puedan resultarles "conflictivas". Son numerosos los gobiernos que intentan controlar la información que los ciudadanos podemos compartir, de tal modo que podrían bloquear ciertos dominios sospechosos de disidencia. El caso más flagrante fue el de Kazajistán en 2015, que empezó a descifrar todo el tráfico SSL de los ciudadanos del país con lo que sus datos personales como credenciales de cuentas de correo, redes sociales, etc. quedaron a merced del gobierno kazajo. El plan quedó invalidado cuando los principales fabricantes de navegadores se pusieron de acuerdo para no aceptar el uso del certificado que debía ser instalado en ellos para romper el túnel SSL. A pesar de ello, el gobierno del país sigue intentando implantar su modelo de "transparencia total" de los usuarios en Internet.

Se ha de decir que Domain Fronting no sirve para evadir Man-In-The-Middle como el del ejemplo expuesto, pero sí otros bloqueos menos evidentes de censura en Internet que se llevan a cabo gubernamentalmente por parte de los proveedores de servicio en Internet.

En los siguientes apartados se va a comentar esta técnica de evasión de censura además de las técnicas de tunelizado para evasión de sistemas de cortafuegos que van a ser objeto de estudio.

Durante estos se hará alusión a MITRE ATT&CK. MITRE es una organización sin ánimo de lucro que provee investigación y soporte en el ámbito de la ingeniería de sistemas. Su misión es "resolver problemas por un mundo más seguro". MITRE ATT&CK es una división dentro de MITRE que se encarga de catalogar y analizar las técnicas, tácticas como formas de amenaza en el mundo de las redes y sistemas.

MITRE ATT&CK proporciona una matriz con una clasificación de las técnicas utilizadas por los atacantes para comprometer una red empresarial. Las técnicas que van a estudiarse en este proyecto son calificadas por MITRE ATT&CK como técnicas de *Command&Control*.

En la siguiente tabla se muestran las 16 técnicas clasificadas en esta categoría.

MITRE ATT&CK: Command & Control			
Application Layer Protocol	Communication Trough Removable Media	Data Encoding	Data Obfuscation
Dynamic Resolution	Encrypted Channel	Fallback Channels	Ingress Tool Transfer
Multi-Stage Channels	Non-Application Layer Protocol	Non-Standard Port	Protocol Tunneling
Proxy	Remote Access Software	Traffic Signaling	Web Service

Tabla 1 – MITRE ATT&CK: C&C

Las técnicas aplicaremos están clasificadas dentro de las categorías que aparecen en color verde en la tabla 1.

2.2 Túnel SSH

El protocolo *Secure Shell* sirve para conectarse a una terminal de un sistema de forma segura a través de una red no segura mediante modelo cliente-servidor: el servidor SSH acepta conexiones de clientes en el puerto 22 TCP por defecto de clientes que deberán autenticarse a través de contraseña o certificado. Los datos de la conexión viajan cifrados, así que si son interceptados no se verán en claro por el atacante. También se puede utilizar para transferir archivos o para tunelizar tráfico de aplicaciones.

Esta última característica puede ser aprovechada para implementar comunicaciones seguras para aplicaciones que no la soportan de forma nativa realizando reenvíos de puertos o Port Forwarding, de modo que, si el tráfico de datos de la aplicación es interceptado, el atacante no podrá leer en claro el contenido. Este comportamiento puede ser aprovechado en el sentido opuesto por parte de un atacante para exfiltrar datos o descargar malware, evadiendo el análisis de los sistemas de seguridad de red, como los NGFW con capacidades de antivirus por firmas o reglas de Data Loss Prevention. También puede ser utilizado para ocultar la fuente de un ataque a base de encadenar túneles SSH de modo que rastrear la conexión inicial se convierta en una tarea imposible.

MITRE ATT&CK califica el SSH tunneling/SSH Port Forwarding como una técnica de tunelización de protocolos para ser utilizado como Command And Control.

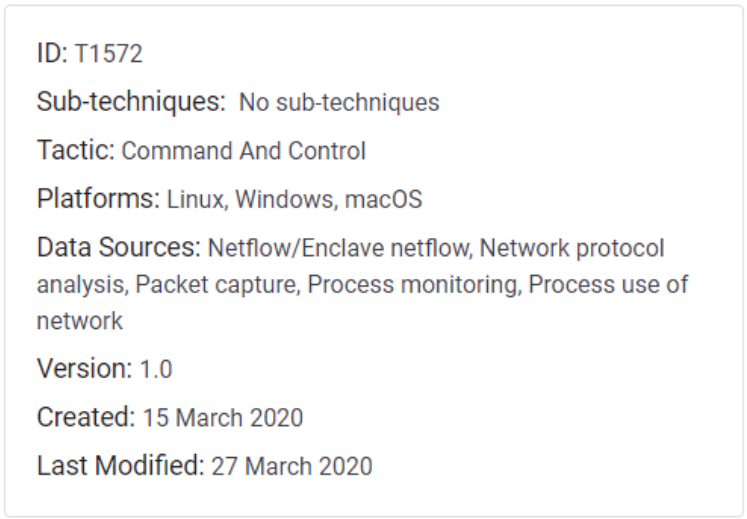


Ilustración 2 – MITRE SSH tunnel

Para implementar un túnel SSH no será necesario utilizar una herramienta específica, cualquier cliente permitirá realizar el Port Forwarding. En sistemas Windows se podrá hacer uso de PuTTY.

https://www.ssh.com/ssh/tunneling/

2.3 Túnel DNS

El protocolo *Domain Name Service* es uno de los más importantes protocolos para el acceso a Internet. Es el encargado de traducir nombres de dominio en direcciones IP basadas en sucesiones de números difíciles de recordar para el ser humano. Utiliza una base de datos distribuida para realizar las resoluciones. Cada host tiene configurado la dirección del servidor de nombres al que realizar las peticiones para la resolución nombre-IP: a la petición de una URL se le responde con una IP en el caso de que se haya encontrado un registro en la base de datos o un error.

El DNS tunneling consiste en utilizar el protocolo de nombres de dominio para transferir datos en vez realizar resoluciones, camuflando estos dentro de campos o encabezados propios del protocolo, haciéndolos pasar como tráfico legítimo. Este ataque se utiliza para evadir sistemas de cortafuegos para navegar libremente, exfiltrar datos, o descargar malware o claves de encriptación para ransomware.

MITRE ATT&CK califica el DNS tunneling como una subtécnica contra protocolos de capa de aplicación (capa 7 en el modelo OSI) utilizada para Command and Control.

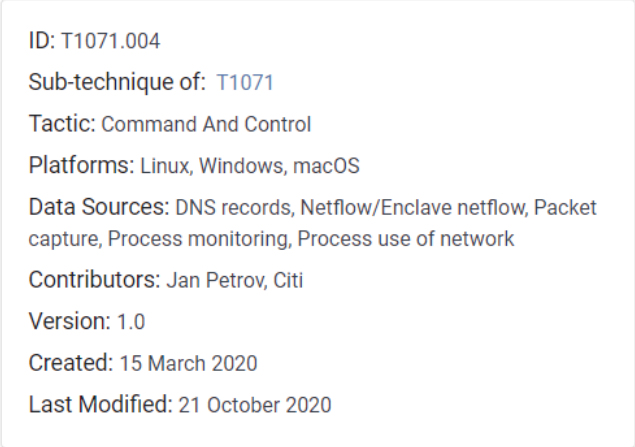


Ilustración 3 – MITRE DNS tunnel

En los entornos corporativos la resolución DNS se realiza internamente, es decir que la dirección configurada en los equipos es una IP de la red interna del cliente. A pesar de ello, muchos tienen configuradas reglas permisivas hacia Internet y no monitorizan ese tráfico debido a que se considera un protocolo confiable y crítico para el funcionamiento de la organización.

Para la mitigación de este ataque MITRE y los principales fabricantes recomiendan inspeccionar el tráfico DNS en busca de patrones sospechosos o mediante firmas de IPS/IDS.

Existen multitud de herramientas para levantar túneles DNS. Se van a comentar algunas de ellas

- **Dns2tcp:** Una herramienta sencilla de tunelización DNS que será utilizada durante la demostración práctica.
  https://tools.kali.org/maintaining-access/dns2tcp

- **Dnscat2**: Esta alternativa intenta diferenciarse de las demás especializándose en comunicaciones de Command and Control. Puede cargar y descargar archivos, incluso iniciar una Shell en el equipo infectado. Además, cifra el tráfico en ambos sentidos.
  https://github.com/iagox86/dnscat2.

- **Iodine:** Esta herramienta permite levantar un túnel DNS para evadir filtrado de firewall permitiendo un ancho de banda de sin límite para subida y de hasta 1 Mb/s en bajada.
  https://github.com/yarrick/iodine

2.4 Túnel ICMP

El protocolo ICMP es un protocolo de gestión con el que dos máquinas intercambian mensajes de estado o error a través de IP. Es muy útil para diagnosticar problemas de red, aunque se recomienda controlar los

dispositivos que lo implementan debido a que puede ser utilizado por atacantes para en la fase de descubrimiento de activos.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31
Tipo								Código								Checksum															
Datos																															

Tabla 2 - Datagrama ICMP

El protocolo ICMP no es utilizado para transportar datos, a pesar de que dentro de los paquetes aparece un campo para estos. En los mensajes de error este campo se utiliza para contener una copia de todo el encabezado y de los 64 primeros bits de datos del paquete que causó el error.

ICMP tunneling consiste en crear un canal de comunicación entre dos equipos a través del campo de datos de los paquetes ICMP. Dentro de este canal podrán viajar datagramas de otros protocolos que podrían estar prohibidos en el firewall. Resulta muy útil para realizar exfiltraciones de datos, evitando reglas de Data Loss Prevention.

MITRE ATT&CK califica el ICMP tunneling como una técnica contra protocolos de no nivel de aplicación utilizada para *Command And Control*.

Ilustración 4 - MITRE ICMP tunnel

Se van a presentar una serie de herramientas capaces de levantar túneles ICMP entre dos hosts.

- **Icmpsh**: esta herramienta es capaz de levantar un túnel ICMP entre un servidor y un cliente sin requerir privilegios administrativos. El objetivo de este ataque es el cliente, que deberá correr un sistema operativo de la familia Windows.
  https://github.com/inquisb/icmpsh

- **Ptunnel**: esta herramienta establece conexiones TCP a través del protocolo ICMP. En su documentación describe como lanzar una Shell inversa a través de SSH.
  https://www.mit.edu/afs.new/sipb/user/golem/tmp/ptunnel-0.61.orig/web/

- **Icmptunnel**: otra herramienta de tunelización ICMP. Se diferencia del resto, porque está especialmente implementada para engañar a firewalls con inspección de estado y NAT, que bloquean peticiones ICMP cuando detectan irregularidades. Puede configurarse para limitar el tamaño de los paquetes y que así parezcan menos sospechosos.
  https://github.com/jamesbarlow/icmptunnel

- **Hanstunnel**: esta herramienta es de fácil instalación, ya que crea la interfaz de túnel con la configuración de red preparada (IP, máscara de subred, puerta de enlace…) para la conexión con el peer.
  http://code.gerade.org/hans/

2.5 Domain Fronting

Esta técnica enmascarará el dominio al que se intenta acceder utilizando una red de distribución de contenidos (CDN). La CDN alojará tanto el dominio enmascarado como el dominio "falso" con el que se engañará al firewall. El dominio enmascarado se colocará en el encabezado TLS y el dominio falso en el encabezado Host de HTTP. La petición DNS se hará con el dominio falso y devolverá una IP de la CDN. A continuación, se realizará la petición HTTP contra la IP devuelta. La CDN redirigirá el tráfico hacia el dominio enmascarado.

Durante este proceso se hace uso de TLS, ESNI y SNI.

TLS, o *Transport Secure Layer*, es un protocolo de cifrado para securizar conexiones entre clientes y servidores. Las conexiones TLS comienzan con un *handshake*, en el que los participantes acuerdan las claves de cifrado asimétrico que van a utilizar durante la comunicación.

El *Server Name Indication* (SNI) le dice a un servidor web qué certificado TLS utilizar en una conexión cliente servidor. Esto hace posible el funcionamiento de las CDN, que alojan diferentes sitios web bajo una misma dirección IP.

Mediante *Encrypted Server Name Indication* (ESNI) se garantiza que en caso de intercepción del handshake TLS no se pueda ver en claro el dominio que pertenece la página que se está intentando visitar, cifrando la SNI. Como es necesario enviar la SNI antes de que se termine el handshake TLS, la manera en que se pueda enviar cifrada es agregar una clave pública al registro DNS, de modo que durante la resolución del nombre de dominio el cliente consigue la clave pública con la que cifra la SNI.

MITRE ATT&CK califica Domain Fronting como una subtécnica de proxy como táctica de Command & Control. Un caso de uso de C&C sería enmascarar una conexión contra el servidor de control, que previamente ha sido incluido en una regla de lista de negra, haciendo creer al firewall que se está intentando navegar contra un dominio lícito.

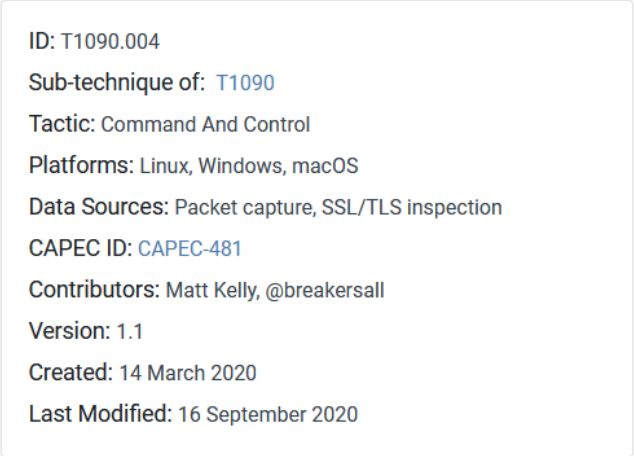


Ilustración 5 - MITRE Domain Fronting

Respecto a las herramientas que se pueden utilizar para Domain Fronting, hemos estudiado Noctilucent.

- **Noctilucent**: publicada en agosto de 2020, esta herramienta es capaz de explotar la técnica de Domain Fronting utilizando TLS 1.3 en la CDN de Cloudflare. En abril de 2018 las CDN de Google y de AWS dejaron de ser vulnerables a esta técnica. Se va a utilizar esta herramienta en el punto 4.6.
  https://github.com/SixGenInc/Noctilucent

3. Diseño del laboratorio

En esta sección se van a presentar los tres escenarios con los que se ha trabajado. Se ha intentado replicar en todos un conjunto de elementos similar: un firewall como elemento central de comunicación, dos subredes, una conexión a Internet y un servidor en la nube. En cada uno de los escenarios el firewall se intercambia por otro, en concreto se van a tratar tres: un iptables, un Fortigate y un Azure FW.

3.1 Laboratorio con iptables

Este es el escenario base del que han ido surgiendo el resto. Un firewall iptables, incluido en el kernel de Linux. Un firewall que acepta inspección de estados del paquete y reglas de NAT, pero sin funcionalidades de inspección a capa 7 del modelo OSI.

El laboratorio consiste en tres subredes: connected, protected e isolated. La subred connected 192.168.1.0/24 tiene conexión directa con Internet. No es una DMZ, ya que no hay ningún servicio accesible desde el exterior. La puerta de enlace es la .1 y corresponde con un router residencial de acceso a Internet.

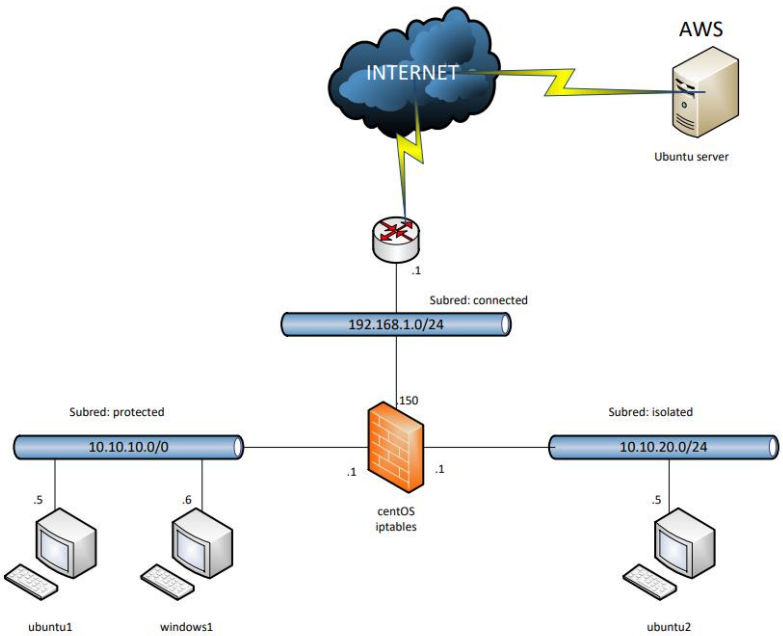


Ilustración 6 - mapa de red iptables

La subred protected 10.10.10.0/24 tiene acceso a Internet a través de una regla de NAT. En ella hay desplegada una VM corriendo Ubuntu 20, la llamada ubuntu1 en la IP .5 y un servidor Windows 2019, de nombre windows1 con la IP .6.

La subred isolated 10.10.20.0/24 estará en principio aislada del resto. En ella hay desplegada una VM corriendo Ubuntu 20, llamada ubuntu2, en la .5.

El firewall estará desplegado en una VM centOS. Tendrá una interfaz conectada a cada subred.

También se hará uso del servicio EC2 de máquinas virtuales en la nube de Amazon Web Services.

3.2 Laboratorio con Fortigate FWF40F

A continuación vamos a probar las técnicas de evasión contra una solución comercial del fabricante Fortinet. Como hemos dicho, vamos a construir un entorno muy parecido al anterior, pero sustituyendo el iptables por un Fortigate modelo FWF40F con licencias de WebControl, AppControl e Intrusión Prevention.

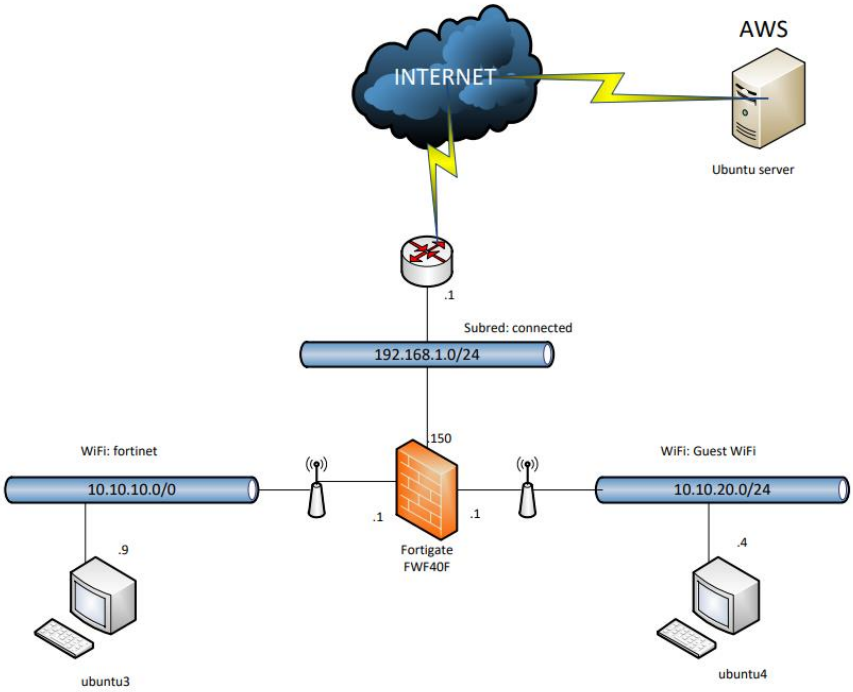


Ilustración 7 - mapa de red con Fortigate

Las máquinas estarán conectadas al firewall por redes WiFI. Aparte de esto y de que ahora podremos inspeccionar los paquetes a nivel de aplicación, todo debe comportarse exactamente igual.

3.3 Laboratorio con Azure FW

Ahora se va a comentar el entorno de laboratorio en Azure. Respecto a la arquitectura, de nuevo será similar, con dos subredes protegidas mediante un Azure Firewall. Tendremos otra vez la subred protected, que podrá salir a Internet y otra llamada isolated, que tendrá limitada su comunicación con la otra subred. Se utilizarán máquinas Linux con la

distribución Ubuntu como hasta ahora para realizar las técnicas de evasión.

En la plataforma de Azure se crea un *Resource group* llamado "Tunneling", para así tener aislado el entorno. Dentro de este grupo de recursos existirá una *Virtual Network* llamada "tunneling-vn", esta contedrá tres subredes, "AzureFirewallSubnet" con direccionamiento 10.10.0.0/24, que alojará el firewall, "protected" con direccionamiento 10.10.10.0/24 e isolated con 10.10.20.0/24.

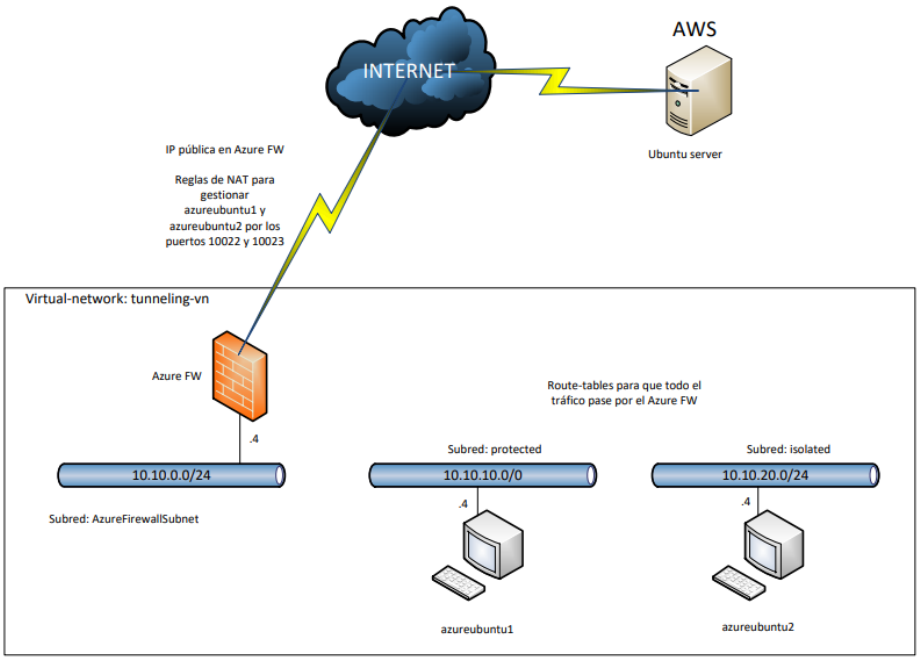


Ilustración 8 - mapa de red en Azure

Las redes virtuales funcionan de manera similar a las redes físicas, pero en este caso va a haber una diferencia. Mediante la creación de *route tables* vamos a asignar el Azure Firewall como puerta de enlace de las subredes protected e isolated. El Azure Firewall va a tener una IP de la subred AzureFirewallSubnet por lo que en redes tradicionales no sería posible ser la puerta de enlace para elementos de otras subredes, sin embargo, por defecto dentro de una Virtual Network todo "se ve" con todo sin hacer uso de routers. Por tanto también tendremos que crear rutas para llegar de la subred protected a la isolated y viceversa a través del Azure Firewall para que puedan aplicarse las reglas.

Azure Firewall tendrá una IP pública conectada a Internet y mantendremos el servidor en AWS para utilizarlo en las técnicas de tunelización DNS.

Una vez realizada la configuración, tendremos en Resource Group los elementos que pueden verse en la ilustración 9.

Name	Type
azureFW	Firewall
azurefw-ip	Public IP address
azureubuntu1	Virtual machine
azureubuntu1-nsg	Network security group
azureubuntu177	Network interface
azureubuntu1_disk1_d78a1102dfa94f64be968eae53dbd593	Disk
azureubuntu1_key	SSH key
azureubuntu2	Virtual machine
azureubuntu2-nsg	Network security group
azureubuntu2125	Network interface
azureubuntu2_disk1_7222f2f384a7467fa82356c4b1cc0845	Disk
azureubuntu2_key	SSH key
isolated_route_table	Route table
protected_route_table	Route table
tunneling-vn	Virtual network

Ilustración 9 - elementos en entorno Azure

Estos son: el Azure FW, la IP pública del firewall, la red virtual, las tablas de rutas para las dos subredes y dos máquinas, cada una con una interfaz de red, un disco, un grupo de seguridad (no aplica en este escenario) y una clave .pem para acceder a ellas por SSH.

Además, Azure nos proporciona un esquema topológico de la red virtual creada. Podemos verlo en la ilustración 10.

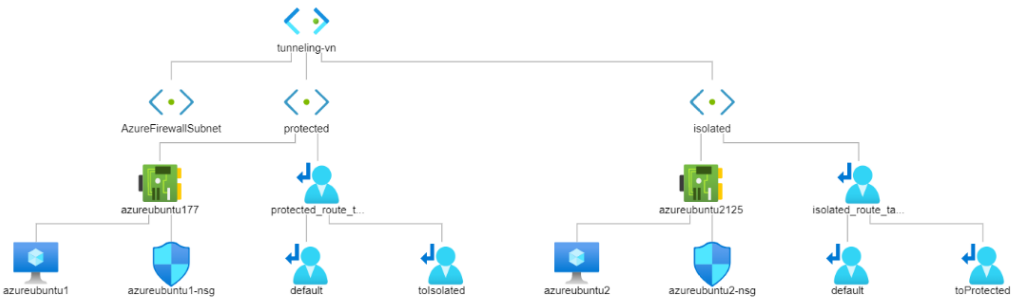


Ilustración 10 - topología de la Virtual Network

De la Virtual Network “tunneling-vn” cuelgan tres subredes. En la primera estará el Azure FW con salida a Internet propia.

En la segunda, con la misma nomenclatura que hasta ahora, llamada protected, se encuentra la interfaz de red de azureubuntu1 y una tabla de rutas con dos rutas estáticas: default para salir a Internet y toIsolated para ir la tercera subred.

De igual modo en la tercera o protected, tendremos la interfaz de red de azureubuntu2 y la tabla de rutas equivalente: salida a Internet (que estará bloqueada por reglas de firewall) y para ir a la subred protected a través del Azure FW.

Una vez definidos los tres escenarios podemos empezar a trabajar en las técnicas descritas en el apartado 2.

4. Desarrollo del proyecto

4.1 Túnel SSH

La primera técnica a probar es probablemente la más fácil de explotar. El protocolo SSH es ampliamente utilizado para la gestión remota de máquinas Linux y por ello es necesario permitirlo a través de los equipos cortafuegos. Las buenas prácticas proponen separar la red de gestión de la de servicio no todos los administradores las aplican y en muchas ocasiones encontraremos tráfico SSH permitido entre subredes.

Vamos a establecer una conexión SSH entre dos máquinas pasando por el firewall iptables, que va a tener permitido este tráfico. La máquina que actuará de servidor ssh va a estar en la subred protected y va a tener permitido el acceso a Internet. El cliente va a tener permitido únicamente el tráfico SSH contra la subred protected, sin embargo, durante la conexión se va a realizar port forwarding que dejará un puerto escuchando en el cliente y cuyas peticiones serán adelantadas a través del túnel hacia el servidor.

4.1.1 Definición del entorno y comprobación de política de seguridad

El entorno corresponderá al laboratorio con iptables, las máquinas que estarán implicadas serán ubuntu1, en la subred protected y ubuntu2 en la subred isolated y el cortafuegos iptables en la máquina centos.

Las reglas del iptables permitirán salir a navegar a las máquinas de la subred protected (ubuntu1) mientras que las máquinas en la subred isolated tendrán denegada la navegación. Adicionalmente se creará una regla de NAT para traducir las IP de la subred protected con la IP de la máquina centos en la subred connected. La justificación de esta red es que permite que el router de salida a Internet enrute los paquetes a través del iptables, sin necesidad de crear una ruta estática.

Además las conexiones SSH estarán permitidas desde la red isolated a la red protected. Esto posibilita la creación del túnel SSH.

En la máquina ubuntu1 se ha configurado el servidor SSH para que permita el forwarding incluyendo las opciones *AllowTcpForwarding yes* y *GatewayPort yes* dentro del fichero de configuración /etc/ssh/sshd_config.

Para resumir y concretar el entorno, tendremos:

1. Reglas de salida a Internet de la máquina ubuntu1.
2. Regla de NAT para la vuelta de los paquetes a ubuntu1.
3. Regla SSH desde ubuntu2 a ubuntu1.
4. Servidor SSH en máquina ubuntu1 que permite Port Forwarding.

4.1.2 Levantamiento del túnel SSH

Desde la terminal de ubuntu2 (10.10.20.5) lanzamos el comando **ssh -f -N -D 1080 $USER@10.10.10.5**. El parámetro -f sirve para lanzar la conexión en segundo plano, -N para utilizar una sesión dedicada al *port forwarding* o reenvío de puertos y -D para especificar el reenvío de puertos, lo que levantará un socket en la máquina para que actúe como servidor SOCKS. 1080 será el puerto donde se levantará el socket. El tráfico destinado a ese puerto se enrutará a través del túnel.

En cuanto lanzamos el comando en la máquina ubuntu2, el puerto 1080 se queda levantado esperando conexiones como podemos ver en la ilustración 11.

```
fer@fer-VirtualBox2:~$ ssh -f -N -D 1080 fer@10.10.10.5
fer@10.10.10.5's password:
fer@fer-VirtualBox2:~$ netstat -ano | grep 1080
tcp        0      0 127.0.0.1:1080          0.0.0.0:*               LISTEN
off (0.00/0/0)
tcp6       0      0 ::1:1080                :::*                    LISTEN
off (0.00/0/0)
```

Ilustración 11 - puerto SOCKS a la escucha

Se confirma que el túnel está preparado para cursar tráfico.

4.3.3 Comprobación de resultados

Para comprobar que el túnel está funcionando vamos a configurar las opciones de proxy del navegador de la máquina ubuntu2 para que envíe el tráfico de navegación a través del servidor SOCKS en el puerto 1080.

La dirección del proxy será localhost, 127.0.0.1, el puerto, como hemos dicho, el 1080. Es importante que también se proxifiquen las peticiones DNS, de otro modo no podremos navegar utilizando nombres de dominio.

network.proxy.socks	127.0.0.1	✎
network.proxy.socks_port	1080	✎
network.proxy.socks_remote_dns	true	⇌
network.proxy.socks_version	5	✎

Ilustración 12 - configuración del proxy SOCKS en el navegador

Una vez configurado el proxy de este modo, podemos comprobar que tenemos salida a Internet en la máquina ubuntu2. Si desde la máquina centos inspeccionamos los paquetes capturándolos mediante el programa Wireshark, en la interfaz de la subred isolated veremos la conexión SSH con ubuntu1.

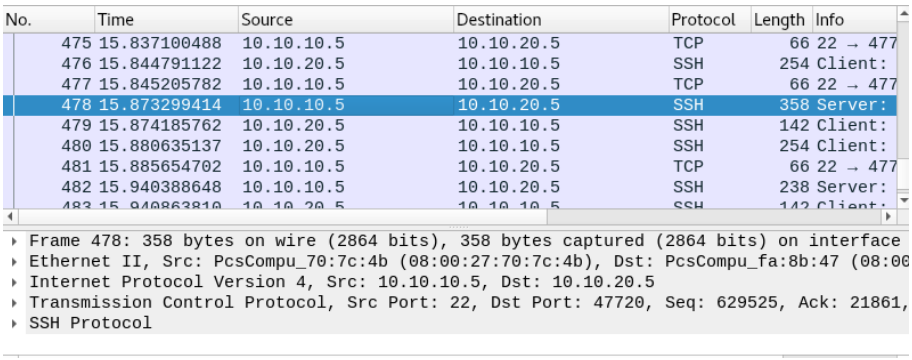


Ilustración 13 - captura de paquetes en túnel SSH

Si capturamos paquetes en la interfaz de la máquina centos en la interfaz de la red protected veremos las peticiones de la máquina ubuntu2 como si fueran de ubuntu1. En la ilustración 14 se muestra captura de tráfico DNS con origen 10.10.10.5, ubuntu1, cuando la petición ha comenzado en la máquina sin salida a Internet ubuntu2.

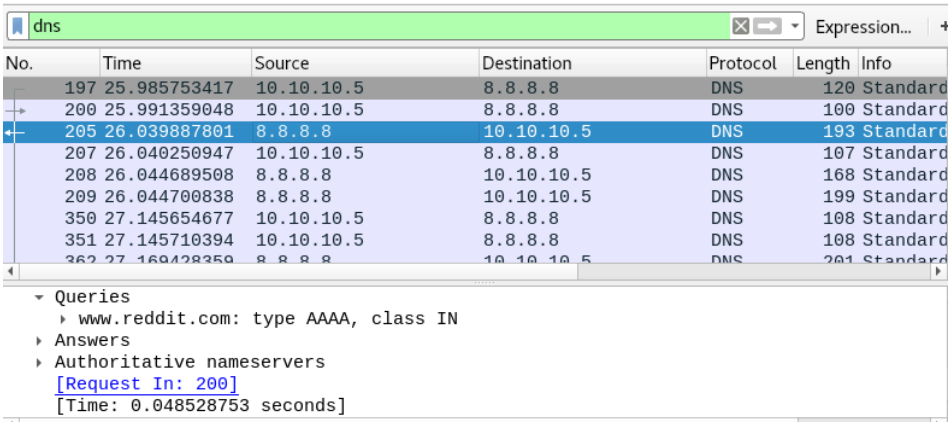


Ilustración 14 - peticiones DNS a través del túnel SSH

Estamos cursando tráfico DNS, HTTP y HTTPS a través del túnel utilizando un servidor proxy SOCKS a pesar de que la política de seguridad del firewall iptables deniega este tráfico por defecto. Gracias al túnel se ha evadido la seguridad.

4.3.4 Caso de uso: túnel SSH para lanzamiento conexión de escritorio remoto

En esta ocasión se va a levantar una sesión RDP a través de un túnel SSH. Mediante la herramienta plink (la versión por CLI de PuTTY) se realizará portforwarding del puerto 12345 TCP hacia el 3389 TCP de la máquina víctima windows1 situada en la red protected.

4.3.4.1 Definición del entorno y comprobación de política de seguridad.

Para esta prueba se va a denegar todo el tráfico excepto el que va desde la subred protected a la subred isolated por el puerto 22 TCP, que es el puerto donde estará escuchando el servidor SSH de la máquina ubuntu2.

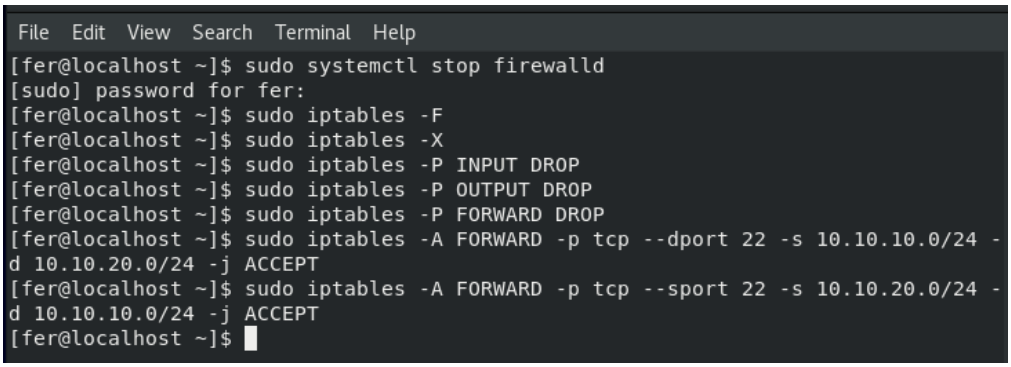


Ilustración 15 - configuración iptables ssh

La máquina windows1 va a tener la IP 10.10.10.6/24 dentro de la subred protected. La máquina ubuntu2 tendrá la configuración vista en anteriores puntos, la IP 10.10.20.5/24 dentro de la subred isolated, con la máquina centos como puerta de enlace predeterminada corriendo el firewall iptables.

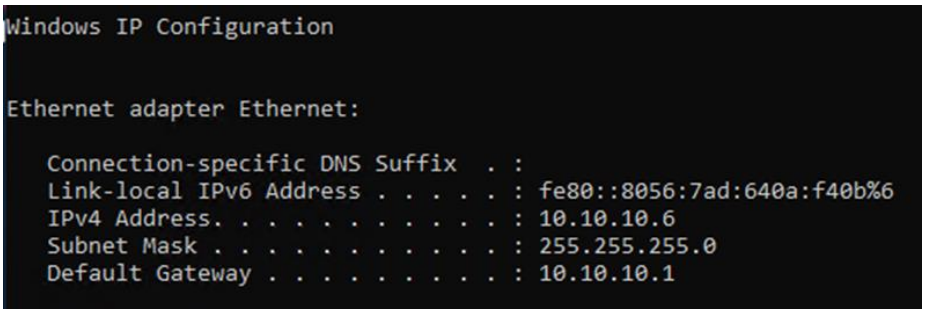


Ilustración 16 - configuración de red windows1

En la ilustración 17 se puede comprobar que no existe comunicación entre windows1 y ubuntu2 por ICMP ni por otro protocolo con la excepción de SSH.

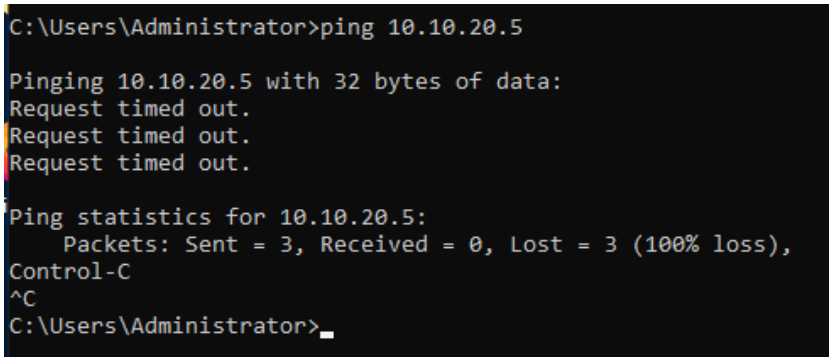


Ilustración 17 - ping fallido

En la máquina ubuntu2 se ha configurado el servidor ssh para que permita el forwarding incluyendo las opciones *AllowTcpForwarding yes* y *GatewayPort yes* dentro del fichero de configuración /etc/ssh/sshd_config.

```
#AllowAgentForwarding yes
AllowTcpForwarding yes
GatewayPorts yes
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
```

Ilustración 18 - configuración ssh server

4.3.4.2 Levantamiento del túnel

El siguiente paso es lanzar la conexión SSH contra ubuntu2 indicando usuario@ip, contraseña, puerto, -4 para forzar IPv4, -2 para usar SSH protocolo v2, -N para no levantar una shell, -C para habilitar compresión y -R para describir el Port Forwading: redirigirá el puerto 12345 de la máquina destino a través del túnel hacia el puerto 3389 de la máquina local, en el que corre el servicio RDP.

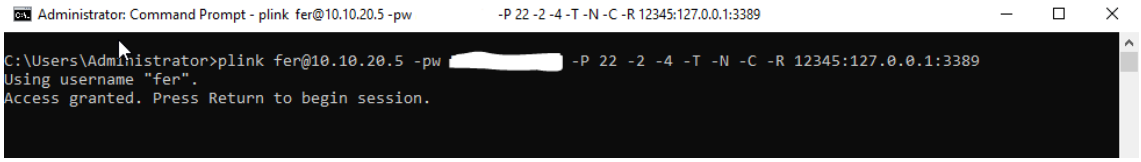


Ilustración 19 - ssh port forwarding con plink

Una vez establecida la sesión SSH, el puerto 12345 TCP de la máquina ubuntu2 quedará a la espera de conexiones entrantes. El tráfico será redirigido a través del túnel hacia el puerto 3389 de la máquina windows1.

```
fer@fer-VirtualBox:~$ netstat -ano | grep LIST
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
 off (0.00/0/0)
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN
 off (0.00/0/0)
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
 off (0.00/0/0)
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
 off (0.00/0/0)
tcp6       0      0 ::1:631                 :::*                    LISTEN
 off (0.00/0/0)
tcp6       0      0 :::12345                :::*                    LISTEN
 off (0.00/0/0)
tcp6       0      0 :::22                   :::*                    LISTEN
 off (0.00/0/0)
```

Ilustración 20 - puerto 12345 tcp a la escucha

4.3.4.3 Comprobación resultados

Se procede a lanzar la conexión de escritorio remoto contra la dirección de ubuntu2, al puerto 12345. Se ha creado el usuario remote en la máquina windows1 y se le ha permitido conectarse mediante RDP. La conexión es redirigida a través del túnel y aceptada por la máquina windows1.

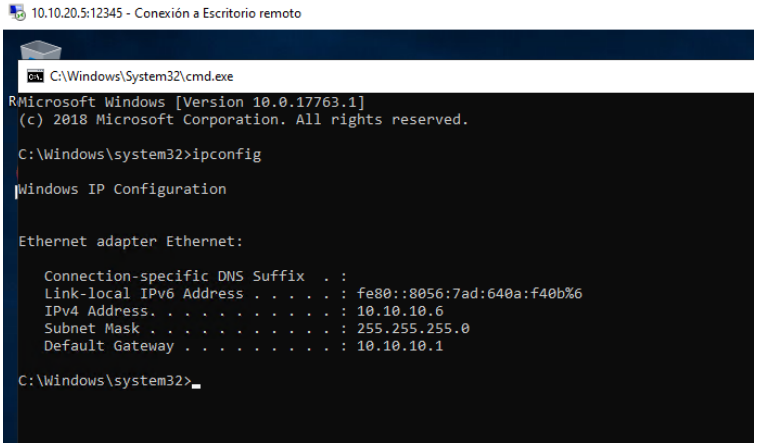


Ilustración 21 - RDP completado

Al igual que sucedía en la ilustración 13, mediante una captura de paquetes podemos ver que el tráfico RDP es enmascarado dentro del túnel SSH.

4.2 Túnel DNS

Como hemos dicho en el apartado 2.3, el protocolo DNS es uno de los imprescindibles para el funcionamiento de Internet. Resultaría imposible recordar todas las direcciones IP de los sitios que visitamos habitualmente. La mayoría de las entidades cuentan con un servidor DNS interno pero muchas otras permiten la resolución de nombres contra servidores DNS públicos.

Para establecer el túnel DNS será necesario utilizar un dominio DNS y una máquina con direccionamiento público escuchando en el puerto 53 UDP. Se va a hacer uso de la herramienta *dns2tcp*, del tipo cliente-servidor, que tunelizará las peticiones TCP en peticiones DNS. El servidor correrá en la instancia de AWS y el cliente se lanzará desde la máquina *ubuntu1*, en la subred protected. Está tiene como puerta de enlace predeterminada la máquina *centos*, corriendo el firewall *iptables*.

4.2.1 Dominio

Para la configuración del dominio es necesario crear un registro A que apunte a la dirección de la máquina publicada y un registro NS que apunte al registro A. De este modo indicaremos que el DNS autoritativo será el valor al que apunta dns2tcp.*****.es, donde estará corriendo el servidor que levantará el túnel.

Tipo	Nombre	Valor	TTL	
A	dns2tcp	15.237.84.244	600 segundos	
CNAME	www	@	1 hora	
CNAME	_domainconnect	_domainconnect.gd.domaincontrol.com	1 hora	
NS	@	ns61.domaincontrol.com	1 hora	
NS	@	ns62.domaincontrol.com	1 hora	
NS	tunnel	dns2tcp.[redacted].es	600 segundos	
SOA	@	Nombre del servidor principal: ns61.domainc...	1 hora	

Ilustración 22 - configuración dominio

4.2.2 Instancia AWS

Para la máquina publicada en Internet se ha optado por desplegar una instancia de Ubuntu Server en AWS.

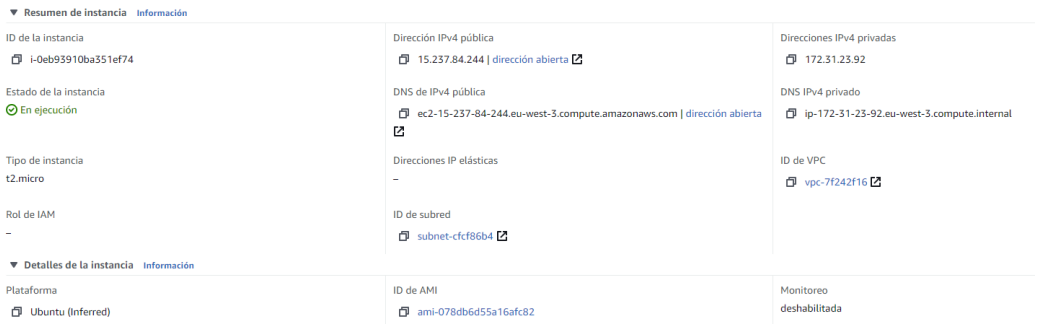


Ilustración 23 - instancia servidor DNS tunnel

Respecto al firewall de AWS, es necesario incluir a la instancia en un grupo de seguridad que tenga abiertos los puertos 53 y 22.

Intervalo de p...	Protocolo	Origen	Grupos de seguridad
22	TCP	0.0.0.0/0	launch-wizard-2
53	TCP	0.0.0.0/0	launch-wizard-2
53	UDP	0.0.0.0/0	launch-wizard-2
-1	ICMP	0.0.0.0/0	launch-wizard-2

Ilustración 24 - configuración AWS firewall

4.2.3 Definición del entorno local

El entorno local estará formado por las subredes *protected* con el direccionamiento 10.10.10.0/24, *connected* en la 192.168.1.0/24, las máquinas *ubuntu1* en la subred protected, *centos* en ambas subredes y el router de salida a Internet en la subred connected.

La máquina *centos* tiene la IP 10.10.10.1 en la interfaz enp0s3. También tendrá salida a Internet a través de la interfaz enp0s9.

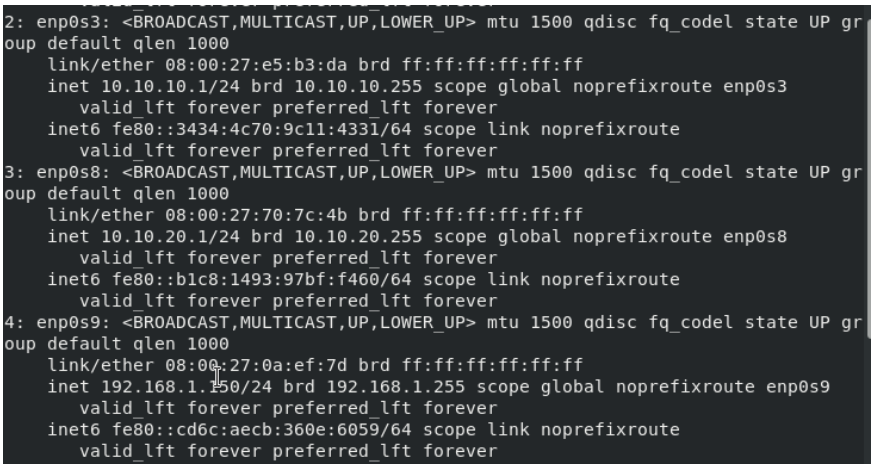


Ilustración 25 - interfaces centOS

El firewall *iptables* se ha configurado con la siguiente política de seguridad:

1. La política por defecto de las cadenas INPUT, OUTPUT y FORWARD es DROP.
2. Se permite el tráfico FORWARD con puerto origen o destino 53 UDP.

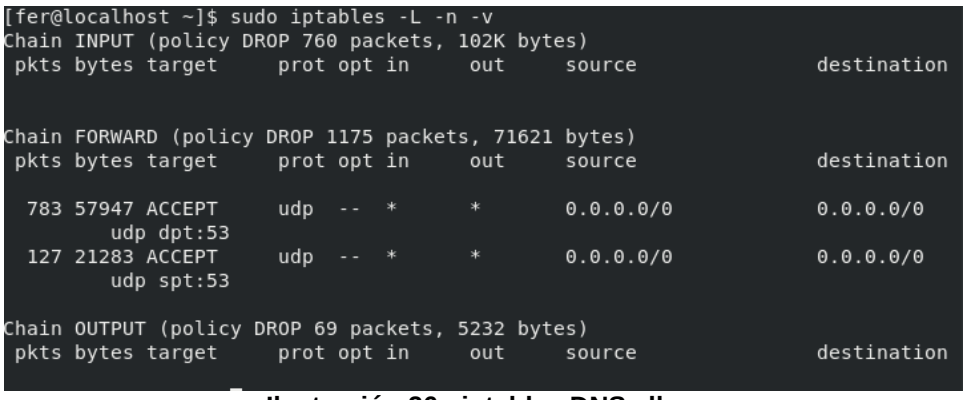


Ilustración 26 - iptables DNS allow

Para que la máquina ubuntu1 pueda conectarse a Internet, se ha configurado una regla de NAT para que el router de la interfaz enp0s9 con la IP 192.168.1.1 sepa enrutar los paquetes de vuelta por la IP del centOS, en la 192.168.1.150. Otra opción hubiera sido configurar una ruta estática en el router para que supiera enrutar los paquetes de vuelta hacia la subred *protected*.

```
[fer@localhost ~]$ sudo iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 2031 packets, 182K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 63 packets, 4540 bytes)
 pkts bytes target     prot opt in     out     source               destination
  133 11019 SNAT       all  --  *      enp0s9  10.10.10.0/24        0.0.0.0/0
        to:192.168.1.150

Chain OUTPUT (policy ACCEPT 69 packets, 5232 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Ilustración 27 - NAT salida a Internet

La máquina ubuntu1 tendrá configurada la IP 10.10.10.5, el servidor DNS de Google (8.8.8.8) y la puerta de enlace predeterminada.

```
fer@fer-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4e:8f:61 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.5/24 brd 10.10.10.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::6c10:b446:2049:34d5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Ilustración 28 - configuración de red de ubuntu1

Por tanto, ubuntu1 podrá hacer peticiones DNS contra la IP 8.8.8.8 pero no podrá comunicarse con otras subredes por ningún otro protocolo.

```
fer@fer-VirtualBox:~$ nslookup
> google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 216.58.201.174
Name:   google.com
Address: 2a00:1450:4003:80b::200e
> twitter.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   twitter.com
Address: 104.244.42.193
Name:   twitter.com
Address: 104.244.42.129
> exit

fer@fer-VirtualBox:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2041ms
```

Ilustración 29 - resolución DNS y ping fallido

Como resumen del entorno, esto es lo que necesitamos antes de levantar el túnel:

1. Servidor conectado a Internet, permitiendo entrada al puerto 53 UDP.

2. Dominio apuntando a IP pública del servidor.
3. Máquina ubuntu1 con permiso del iptables para realizar consultas DNS al puerto 53 UDP.

4.2.4 Levantamiento del túnel DNS

Para levantar el túnel vamos a utilizar la herramienta *dns2tcp*. Esta herramienta tunelizará todas las peticiones TCP en peticiones DNS.

El servidor tiene una lista de recursos especificados en un archivo de configuración. Cada recurso es un servicio que escucha las conexiones TCP. La configuración del lado del servidor se realizará mediante el fichero /etc/dns2tcpd.conf. En este fichero se va a definir la IP en la que escuchará el túnel (0.0.0.0 para escuchar en todas), el puerto en el que va a levantar el servicio, el dominio que va a utilizar para la conexión, los recursos publicados a través del túnel y la contraseña.

```
ubuntu@ip-172-31-23-92:~$ cat /etc/dns2tcpd.conf
listen = 0.0.0.0
port = 53
# If you change this value, also change the USER variable in /etc/default/dns2tcpd
user = nobody
chroot = /tmp
domain = tunnel.[redacted].es
resources = ssh:127.0.0.1:22 , smtp:127.0.0.1:25
key = vpn12345
```

Ilustración 30 - configuración dns2tcp server

Lanzamos el servidor dns2tcp especificando la IP del único interfaz de la máquina y el fichero de configuración.

```
ubuntu@ip-172-31-23-92:~$ sudo dns2tcpd -i 172.31.23.92 -F -d2 -f /etc/dns2tcpd.conf
17:43:01 : Debug options.c:97   Add resource ssh:127.0.0.1 port 22
17:43:01 : Debug options.c:97   Add resource smtp:127.0.0.1 port 25
17:43:01 : Debug socket.c:54    Listening on 172.31.23.92:53 for domain tunnel.[redacted].es
Starting Server v0.5.2...
17:43:01 : Debug main.c:132     Chroot to /tmp
17:43:01 : Debug main.c:142     Change to user nobody
```

Ilustración 31 - lanzamiento dns2tcp

Desde la máquina protegida, lanzamos la conexión al servidor, pidiendo que escuche conexiones SSH en el puerto 3333.

```
fer@fer-VirtualBox:~$ dns2tcpc -z tunnel.[redacted]es -k vpn12345 -l 3333 -r ssh
No DNS given, using 127.0.0.53 (first entry found in resolv.conf)
Listening on port : 3333
```

Ilustración 32 - lanzamiento cliente dns2tcp

A continuación lanzamos una conexión SSH a localhost en el puerto 3333. Se indican las credenciales para la conexión a la instancia de AWS. Se utiliza la opción -D 1080 para realizar una redirección de puertos hacia el 1080, actuando el cliente SSH como servidor SOCKS. En ese puerto configuraremos el proxy de la máquina protegida para redirigir las peticiones por el túnel.

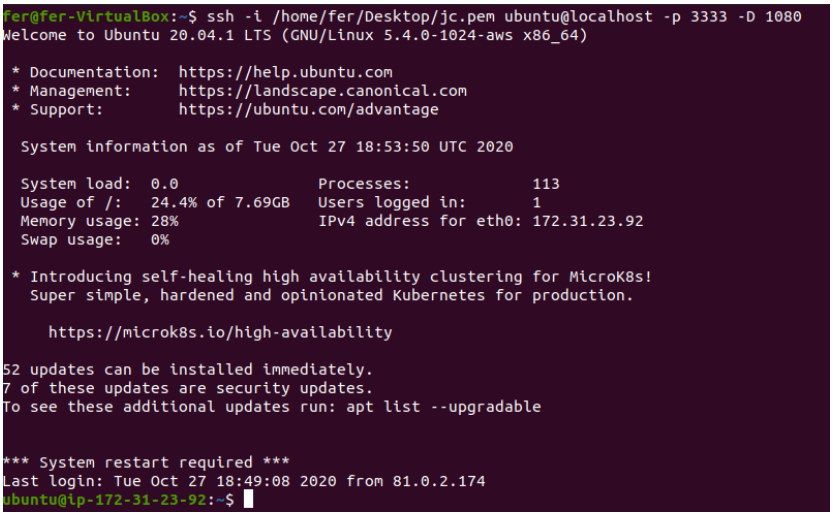


Ilustración 33 - túnel SSH sobre DNS con forwarding hacia el 1080/tcp

Una vez lanzamos la conexión SSH, en el servidor se aprecia como empiezan a manejarse paquetes.

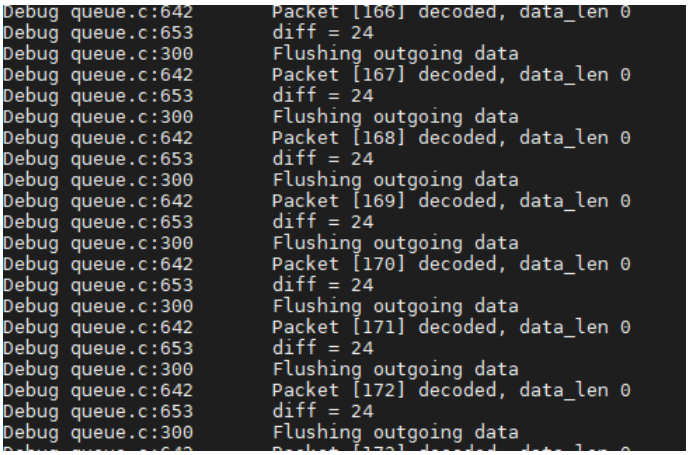


Ilustración 34 - paquetes DNS en el servidor

Se procede a realizar una captura de paquetes en la interfaz enp0s3. Sólo aparecen peticiones DNS desde la 10.10.10.5 hacia 8.8.8.8.

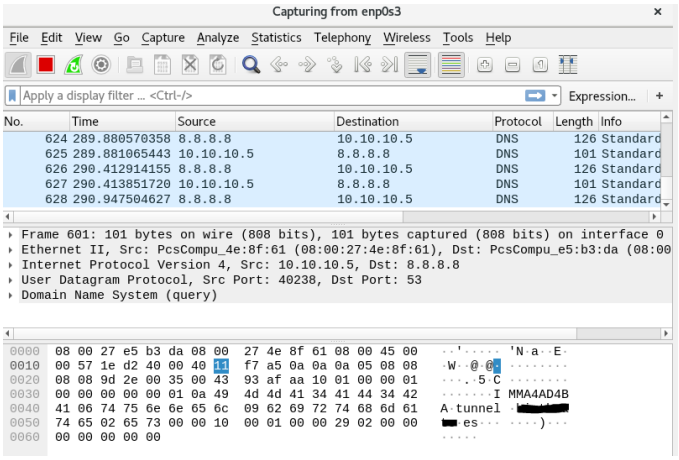


Ilustración 35 - inspección paquete DNS con Wireshark

Si se examinan las respuestas DNS se aprecia como a través de registros TXT se está codificando el tráfico.

Ilustración 36 - detalle contenido paquete DNS

Se adjuntan capturas demostrando la capacidad de navegar por Internet atravesando el iptables, una vez configurado el proxy del navegador por el puerto 1080.

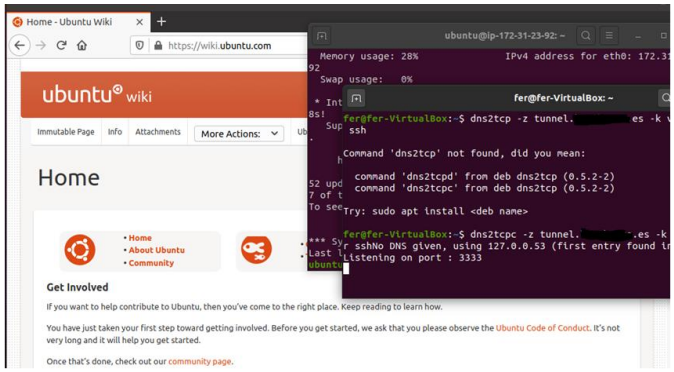


Ilustración 37 - navegación a través de proxy SOCKS

4.3 Túnel ICMP

La manera recurrente de hacer troubleshooting de red a través de lanzar “pings” está ampliamente extendida entre administradores de redes, sistemas e incluso usuarios avanzados. Cuando se tiene un problema de conexión la primera pregunta que uno se hace es “¿llegaré por ping?” Si antes llegaba y ahora no tiene que ser un problema de mi adaptador o quizá el otro extremo sea el que tiene un problema o puede que haya una regla de firewall que me lo impida. Si llego por ping puedo pensar que no es un tema de mi adaptador o la IP que tenga configurada, hay comunicación pero no por el puerto deseado.

Ciertamente es útil para hacer diagnostico, pero puede suponer un problema de seguridad. ¿Es necesario que los usuarios tengan la capacidad de diagnosticar problemas de red? ¿No sería mejor que esta utilidad fuera de uso exclusivo de los administradores? A continuación se va a demostrar porque esta afirmación es necesaria.

Para establecer el túnel ICMP vamos a hacer uso de la herramienta *hanstunnel.* Esta herramienta funcionará en modo cliente-servidor, la máquina ubuntu1 funcionará como server y ubuntu2 se conectará a ella a través de una interfaz de túnel creada por la herramienta de tunelización. Esta interfaz estará fuera del ámbito del firewall, aunque físicamente estará desplegada dentro de las interfaces de red conectadas al iptables.

4.3.1 Definición del entorno y comprobación de política de seguridad

El servidor correrá en la máquina *ubuntu1*, en la subred protected. Está tiene como puerta de enlace predeterminada la máquina *centos*, corriendo el firewall *iptables*.

```
[root@localhost fer]# iptables -L -n -v
Chain INPUT (policy DROP 1261 packets, 159K bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy DROP 1905 packets, 118K bytes)
 pkts bytes target     prot opt in     out     source               destination

    5   420 ACCEPT     icmp --  *      *       10.10.20.0/24        10.10.10.0/24

    4   336 ACCEPT     icmp --  *      *       10.10.10.0/24        10.10.20.0/24


Chain OUTPUT (policy DROP 141 packets, 10584 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Ilustración 38 - iptables ICMP allow

La configuración de red de la máquina ubuntu1 antes de la ejecución de la herramienta de tunelización es la siguiente: la interfaz virtual enp0s3, con la dirección IP 10.10.10.5/24 y la interfaz de loopback.

```
fer@fer-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.5  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::6c10:b446:2049:34d5  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4e:8f:61  txqueuelen 1000  (Ethernet)
        RX packets 801  bytes 77875 (77.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1975  bytes 155572 (155.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2087  bytes 161561 (161.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2087  bytes 161561 (161.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

fer@fer-VirtualBox:~$
```

Ilustración 39 - interfaces ubuntu1

De igual modo, la máquina ubuntu2 tiene dos interfaces configuradas, la enp0s3 con la IP 10.10.20.5/24 y la de loopback.

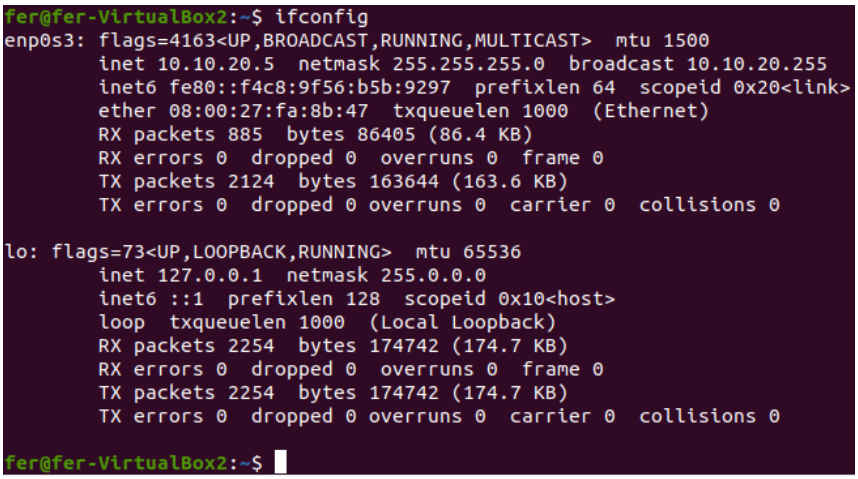


Ilustración 40 - interfaces ubuntu2

Antes de comprobar la política de seguridad, recapitulemos para entender la situación del entorno. Tenemos:

1. Máquina ubuntu1 pudiéndose comunicar con ubuntu2 mediante el protocolo ICMP.
2. Máquina ubuntu2 pudiéndose comunicar con ubuntu1 mediante el protocolo ICMP.
3. El resto de comunicaciones de ambas máquinas están denegadas.

En las ilustraciones 41 y 42 se puede comprobar que gracias a las reglas configuradas en la ilustración 38 no se dispone de acceso SSH a la máquina ubuntu2, pero sí que se permite hacer ping contra esta.

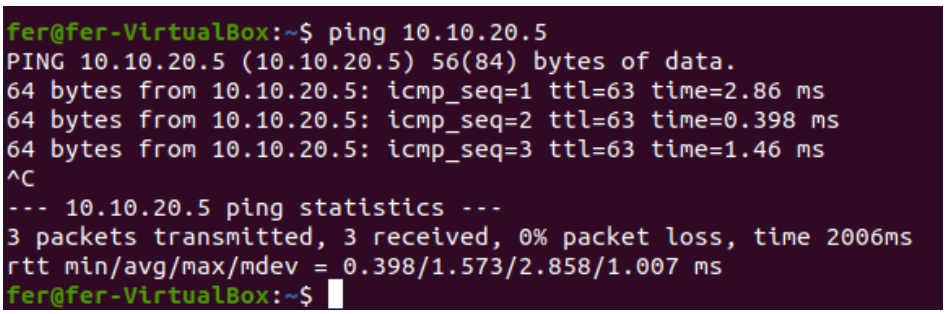


Ilustración 41 - ping allow

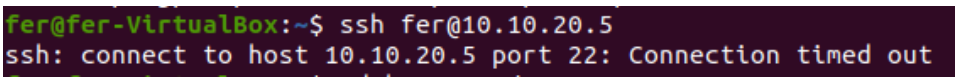


Ilustración 42 - ssh deny

El ping lanzado desde ubuntu1 a ubuntu2 puede ser capturado mediante la herramienta de análisis de red Wireshark para comprobar un paquete ICMP antes del lanzamiento de la herramienta de tunelización.

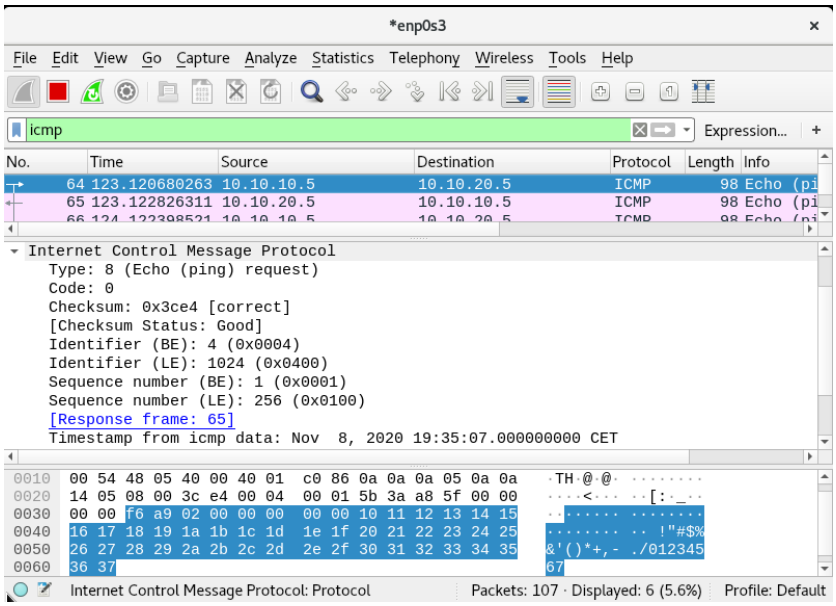


Ilustración 43 - paquete ICMP sin modificar

4.3.2 Levantamiento del túnel ICMP

Se procede a lanzar la herramienta hanstunnel desde ubuntu1 que ejercerá como servidor, para ello se elige que el direccionamiento del túnel será 10.10.30.0 y una contraseña para impedir las conexiones no autorizadas. Se puede observar en la ilustración 44 que una vez que se lanza la herramienta se crea una nueva interfaz de túnel, tun0 con una IP dentro del rango seleccionado, la dirección 10.10.30.1.

```
root@fer-VirtualBox:/home/fer/hans-1.0# ./hans -s 10.10.30.0 -p vpn12345
./hans: opened tunnel device: tun0
./hans: detaching from terminal
root@fer-VirtualBox:/home/fer/hans-1.0# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.5  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::6c10:b446:2049:34d5  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4e:8f:61  txqueuelen 1000  (Ethernet)
        RX packets 832  bytes 80045 (80.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2406  bytes 187160 (187.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2814  bytes 217373 (217.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2814  bytes 217373 (217.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1467
        inet 10.10.30.1  netmask 255.255.255.0  destination 10.10.30.1
        inet6 fe80::f872:89e3:19e1:6f9c  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 48 (48.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 44 – servidor hanstunnel e interfaz tun0 en ubuntu1

El siguiente paso será lanzar el cliente de hanstunnel en la máquina ubuntu2. Se va a indicar a la herramienta la IP donde está corriendo el servidor y la contraseña configurada en este.

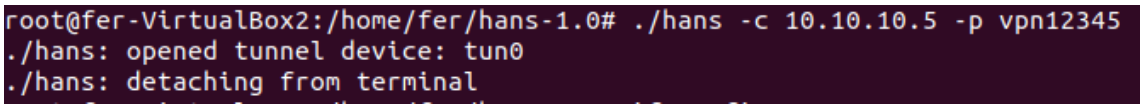


Ilustración 45 - lanzamiento cliente hanstunnel

Como se aprecia en la ilustración 46, la herramienta crea la interfaz de tunnel y le asigna una IP del rango, en este caso la dirección 10.10.30.100.

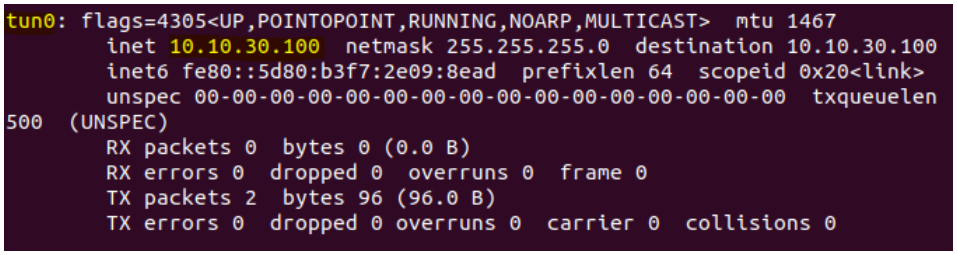


Ilustración 46 - interfaz tun0 en ubuntu2

4.3.3 Comprobación de resultados

Una vez levantado el túnel se puede lanzar una conexión SSH entre ambas máquinas, haciendo bypass de las reglas configuradas en el firewall iptables, siempre utilizando la IP de la interfaz creada por la herramienta de tunelización.

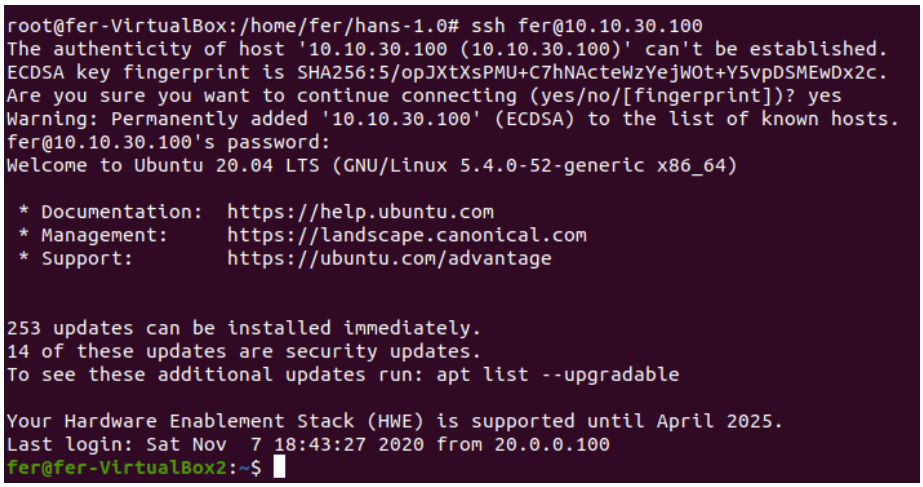


Ilustración 47 - conexión SSH sobre ICMP

De nuevo mediante Wireshark capturamos los paquetes entre ambas máquinas. Estos son identificados como paquetes ICMP, significativamente mayores a los paquetes lícitos.

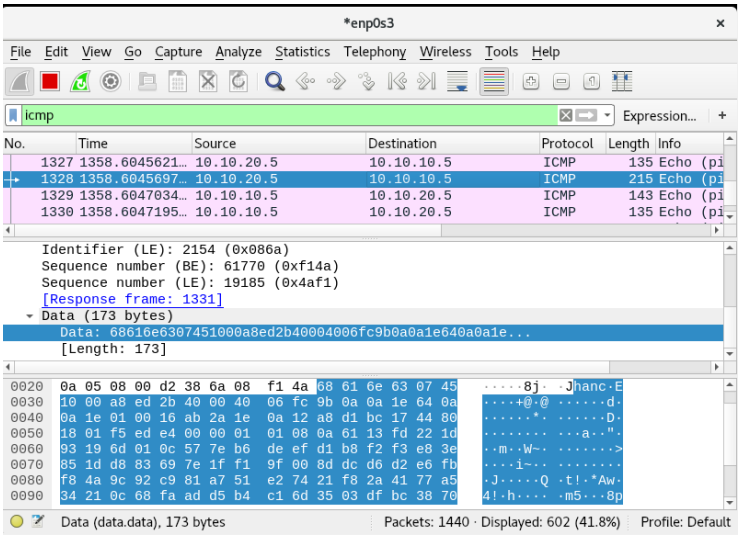


Ilustración 48 - inspección paquetes ICMP tunelizados

4.4 Tunelización sobre solución comercial

Vamos ahora a replicar las técnicas anteriores en el entorno del punto 3.2, sustituyendo el firewall iptables por el Fortigate FWF40F. Ahora podremos aplicar inspección a capa 7, esto es a nivel de aplicación: las reglas, además de basadas en interfaces, IP de origen y destino y puertos TCP/UDP también podrán tener asociados perfiles de inspección que permitan o denieguen basándose en la aplicación que genera dicho tráfico.

4.4.1 SSH tunnel sobre Fortigate

Empezamos con el levantamiento de un túnel SSH haciendo pasar el tráfico a través del Fortigate FWF40F. Las máquinas implicadas serán ubuntu3 y ubuntu4, del laboratorio con Fortigate. La primera de estas estará en la red WiFi con SSID “fortinet” y la segunda en “Guest WiFi”.

1. Sin perfiles de seguridad.
2. Con perfil de aplicación de alto riesgo (inspección de capa7) y firmas de IPS y perfil de inspección SSH por defecto.
3. Con perfil de inspección SSH bloqueando Port Forwarding.

4.4.2.1 SSH tunnel sin perfiles de seguridad

Vamos a comprobar que sin perfiles de seguridad Fortigate funciona como un iptables con interfaz gráfica. La política de seguridad será la siguiente: desde Guest WiFi, ubuntu4 no podrá salir a Internet, pero si se podrá conectarse por SSH a ubuntu3. Por otra parte, ubuntu3 podrá salir a navegar, ya que tendrá permitidos los protocolos HTTP, HTTPS y DNS.

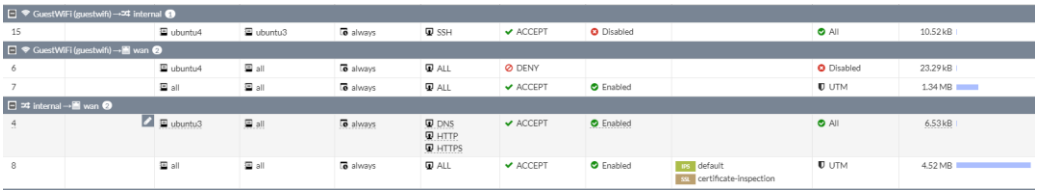


Ilustración 49 – reglas Fortigate SSH sin perfiles

Lanzamos el comando con el Port Forwarding, reenviando por el 1080 TCP desde ubuntu4 a ubuntu3, configuramos el proxy y comprobamos que navegamos a través de él.

Herramienta	Acción Fortigate
SSH+port forwarding	✅

Tabla 3 – Fortigate sin perfiles vs SSH tunnel

4.4.2.2 Perfil APP block-high-risk, IPS y SSH inspection por defecto

Un caso de política habitual es aprovechar los perfiles de seguridad que ofrecen los fabricantes, pero con una configuración ineficaz, ya sea utilizando perfiles por defecto o creando estos sin la suficiente contundencia.

Se ha reutilizado la política de seguridad del punto anterior pero esta vez aplicando varios perfiles de seguridad. El primero, el de control de aplicaciones *block-high-risk*, que bloquea aplicaciones con categoría "proxy". Estamos utilizando SSH como proxy, así que quizá el perfil detecte este uso y bloquee. También añadimos todas las firmas de IPS que tiene descargadas y el perfil de inspección SSL/SSH *custom-deep-inspection*. Este perfil no es el menos restrictivo que viene instalado, ya que hace inspeccion SSH profunda mientras que *certificate-inspection* no.

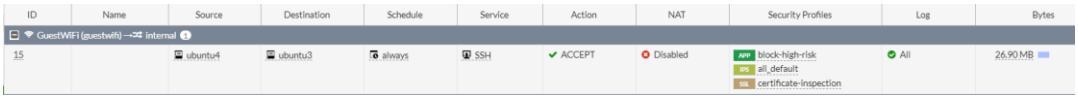


Ilustración 50 – regla Fortigate con perfiles de seguridad

Con este perfil sigue siendo posible levantar el túnel y enrutar tráfico por él mediante la técnica de Port Forwarding. Los perfiles de control de aplicación e IPS no están sirviendo para bloquear esta técnica de evasión.

Herramienta	Acción Fortigate
SSH+port forwarding	✅

Tabla 4 – Fortigate sin con perfiles vs SSH tunnel

4.4.2.3 SSH inspector bloqueando port forwarding

Después de la investigación de las capacidades de este perfil de seguridad se descubre de que es posible bloquear Port Forwarding mediante la creación de un perfil de filtrado a través de línea de comandos. No es posible hacer esto mediante interfaz gráfica lo que

hace pensar que la mayoría de las configuraciones de Fortigates no tienen aplicado un filtrado de este tipo, además configurarlo correctamente resultó complicado, ya que hubo que modificar el modo de inspección del equipo (basado en flujo a modo proxy) e ir cambiando opciones hasta que se consiguió que bloqueará únicamente lo requerido y no todas las sesiones SSH.

Cuando lanzamos el comando, la sesión SSH levanta, pero cuando hacemos una redirección a través del proxy SOCKS configurado en el navegador se nos muestra el mensaje de la ilustración 51.

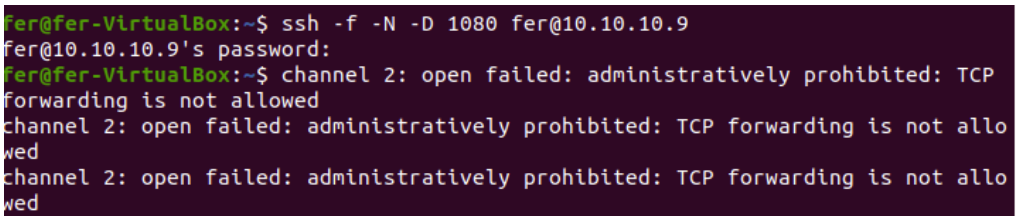


Ilustración 51 – mensaje contra Port Forwarding

Parece que el Fortigate es capaz de bloquear Port Forwarding y evitar los túneles SSH pero la correcta configuración no resulta sencilla.

Herramienta	Acción Fortigate
SSH+port forwarding	✗

Tabla 5 – Fortigate con filtrado de acciones SSH vs SSH tunnel

4.4.3 DNS tunnel sobre Fortigate

En este punto se va a probar la técnica utilizada en el punto 4.2 sustituyendo el firewall iptables por el Fortigate FWF40F.

Para ello se van a estudiar distintas configuraciones junto con las distintas herramientas de tunelización vistas en el punto 2.3.

1. Sin perfiles de seguridad.
2. Con perfil de aplicación por defecto (inspección de capa 7) y firmas de IPS basadas en DNS tunneling.
3. Con perfil de aplicación bloqueando aplicaciones de tipo Proxy.

4.4.2.1 DNS sin perfiles de seguridad

En este caso el Fortigate se comporta como un firewall de capa 4 similar a iptables. Las herramientas funcionan y el túnel DNS es levantado y con capacidades de cursar tráfico.

Herramienta	Acción Fortigate
Dns2tcp	✓
Iodine	✓
Dnscat2	✓

Tabla 6 – Fortigate sin perfiles de seguridad vs herramientas DNS tunnel

4.4.2.2 Perfil APP por defecto e IPS DNS Tunnel

Se crea un perfil de IPS llamado “dns tunnel” con las únicas tres firmas que ofrece Fortinet: *Generic.DNS:Tunnel.Detection.Variant.A*, *Dnscat2DNS.Tunnel* y *Thunder.DNS.Tunnel*.

El perfil de aplicaciones se deja por defecto, de modo que no bloquea nada, únicamente monitoriza la categoría del tráfico cursado por el firewall de las aplicaciones que se ejecutan en los dispositivos protegidos por él.

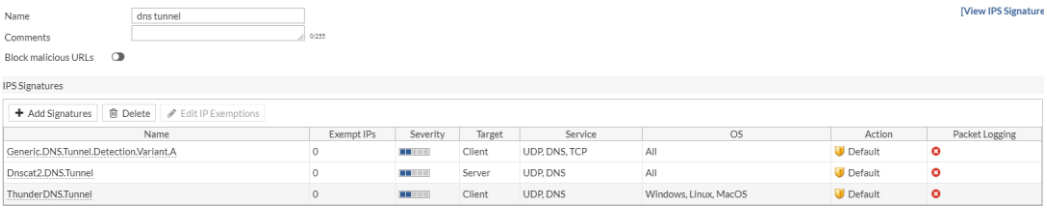


Ilustración 52 - perfil IPS contra DNS tunneling

Se aplican estos perfiles a una regla de firewall, con origen la máquina ubuntu3 y destino todo Internet. Como servicio, sólo se dejará DNS, esto es puerto 53 TCP o UDP.

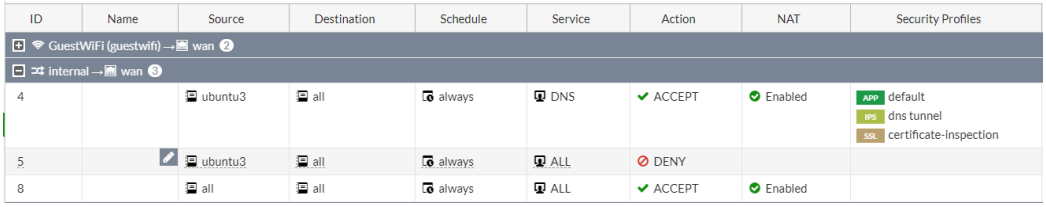


Ilustración 53 - reglas Fortigate permitiendo DNS y aplicando perfiles de seguridad

Se comprueba el funcionamiento de la política de seguridad. La máquina es incapaz de realizar comunicaciones ICMP contra los DNS de Google, sin embargo puede utilizar estos para resolver google.com.

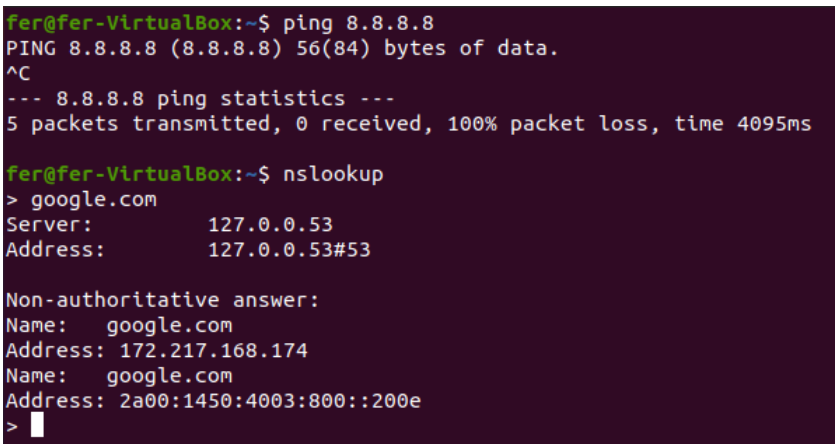
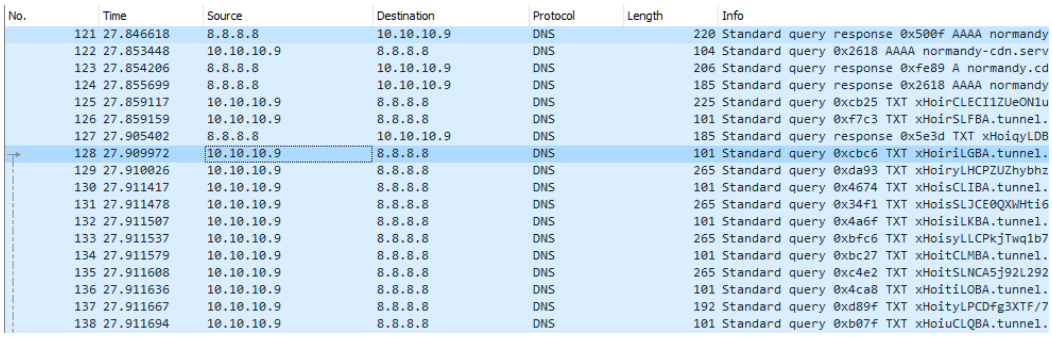


Ilustración 54 - ICMP bloqueado y DNS permitido

Se procede a levantar el túnel DNS mediante la herramienta utilizada en el punto 4.2, dns2tcp. El túnel levanta y es capaz de cursar tráfico.

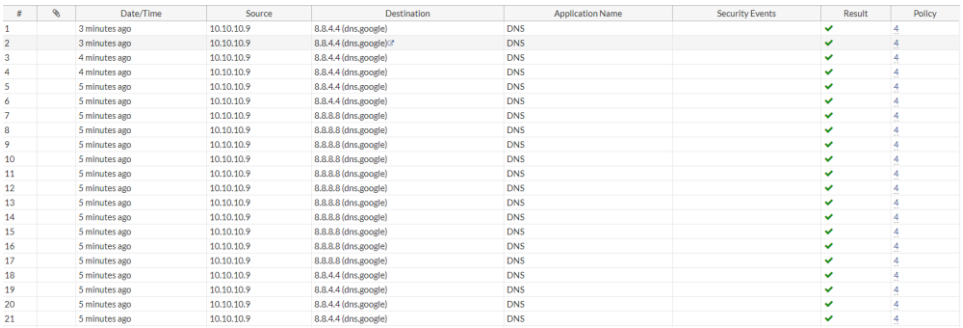
Paralelamente se han capturado paquetes en el Fortigate. El firewall solo está viendo tráfico DNS a pesar de que se está utilizando para levantar un túnel SSH y utilizar el otro extremo como proxy SOCKS para navegación.



No.	Time	Source	Destination	Protocol	Length	Info
121	27.846618	8.8.8.8	10.10.10.9	DNS	220	Standard query response 0x500f AAAA normandy
122	27.853448	10.10.10.9	8.8.8.8	DNS	104	Standard query 0x2618 AAAA normandy-cdn.serv
123	27.854206	8.8.8.8	10.10.10.9	DNS	206	Standard query response 0xfe89 A normandy.cd
124	27.855699	8.8.8.8	10.10.10.9	DNS	185	Standard query response 0x2618 AAAA normandy
125	27.859117	10.10.10.9	8.8.8.8	DNS	225	Standard query 0xcb25 TXT xHoirCLECI1ZUeON1u
126	27.859159	10.10.10.9	8.8.8.8	DNS	101	Standard query 0xf7c3 TXT xHoirSLFBA.tunnel.
127	27.905402	8.8.8.8	10.10.10.9	DNS	185	Standard query response 0x5e3d TXT xHoiqyLDB
128	27.909972	10.10.10.9	8.8.8.8	DNS	101	Standard query 0xcbc6 TXT xHoiriLGBA.tunnel.
129	27.910026	10.10.10.9	8.8.8.8	DNS	265	Standard query 0xda93 TXT xHoiryLHCPZUZhybhz
130	27.911417	10.10.10.9	8.8.8.8	DNS	101	Standard query 0x4674 TXT xHoisCLIBA.tunnel.
131	27.911478	10.10.10.9	8.8.8.8	DNS	265	Standard query 0x34f1 TXT xHoisSLJCE0QXWHti6
132	27.911507	10.10.10.9	8.8.8.8	DNS	101	Standard query 0x4a6f TXT xHoisiLKBA.tunnel.
133	27.911537	10.10.10.9	8.8.8.8	DNS	265	Standard query 0xbfc6 TXT xHoisyLLCPkjTwq1b7
134	27.911579	10.10.10.9	8.8.8.8	DNS	101	Standard query 0xbc27 TXT xHoitCLMBA.tunnel.
135	27.911608	10.10.10.9	8.8.8.8	DNS	265	Standard query 0xc4e2 TXT xHoitSLNCA5j92L292
136	27.911636	10.10.10.9	8.8.8.8	DNS	101	Standard query 0x4ca8 TXT xHoitiLOBA.tunnel.
137	27.911667	10.10.10.9	8.8.8.8	DNS	192	Standard query 0xd89f TXT xHoityLPCDfg3XTF/7
138	27.911694	10.10.10.9	8.8.8.8	DNS	101	Standard query 0xb07f TXT xHoiuCLQBA.tunnel.

Ilustración 55 - captura paquetes dns2tcp en Fortigate

El log del Fortigate arroja la misma información: multitud de sesiones DNS con origen la IP de la máquina ubuntu3. Este tráfico está permitido por la regla anterior, pero debería ser denegado por las firmas de IPS, en concreto *Generic.DNS.Tunnel.Detection.Variant.A* no está detectando esta herramienta.

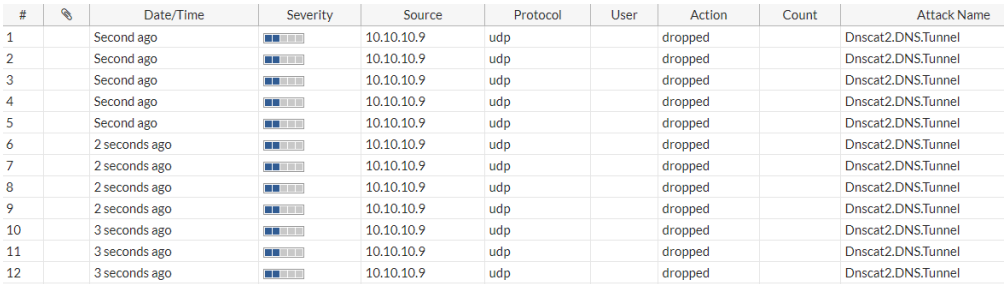


#	Date/Time	Source	Destination	Application Name	Security Events	Result	Policy
1	3 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
2	3 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
3	4 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
4	4 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
5	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
6	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
7	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
8	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
9	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
10	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
11	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
12	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
13	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
14	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
15	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
16	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
17	5 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	DNS		✔	4
18	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
19	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
20	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4
21	5 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	DNS		✔	4

Ilustración 56 - log Fortigate como tráfico DNS legítimo

Se prueba también con la herramienta Iodine y de igual modo, el túnel levanta y cursa tráfico. Las firmas de IPS no detectan estas dos herramientas.

Se va a probar ahora con dnscat2, una herramienta con una firma de IPS dedicada. En esta ocasión el motor de IPS hace su trabajo e impide el levantamiento del túnel.



#	Date/Time	Severity	Source	Protocol	User	Action	Count	Attack Name
1	Second ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
2	Second ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
3	Second ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
4	Second ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
5	Second ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
6	2 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
7	2 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
8	2 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
9	2 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
10	3 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
11	3 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel
12	3 seconds ago		10.10.10.9	udp		dropped		Dnscat2.DNS.Tunnel

Ilustración 57 - IPS drop Dnscat2

Los logs aportan información adicional sobre el incidente resuelto, mostrando la política que ha hecho saltar la detección, la acción que se ha llevado a cabo (drop), el perfil creado que lo ha detectado, la firma que ha detectado el ataque, un enlace con información sobre el ataque y las interfaces de origen y destino del firewall que han manejado el paquete.

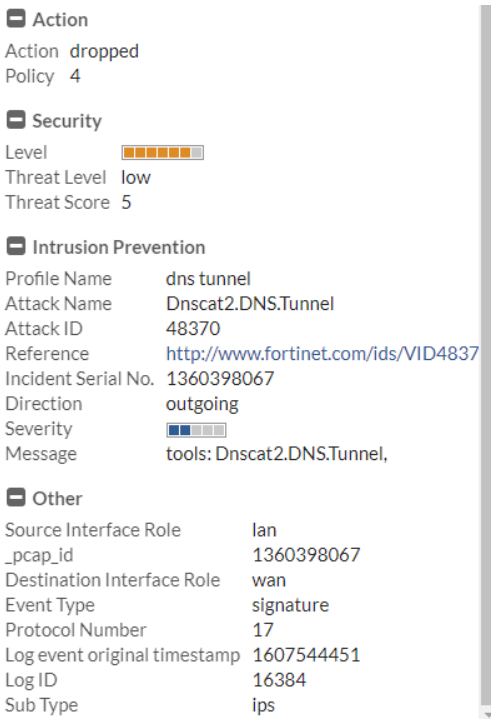


Ilustración 58 - detalle log Dnscat2

Herramienta	Acción Fortigate
Dns2tcp	✅
Iodine	✅
Dnscat2	❌

Tabla 7 – Fortigate con firmas de IPS vs herramientas DNS tunnel

4.4.2.3 Perfil APP block-high-risk

Para este punto se han modificado los perfiles de seguridad de la regla, el perfil de aplicaciones que estaba por defecto (solo monitorizaba) se ha cambiado por un perfil llamado *block-high-risk* que básicamente bloquea aplicaciones que Fortinet califica como de tipo proxy y monitoriza el resto.

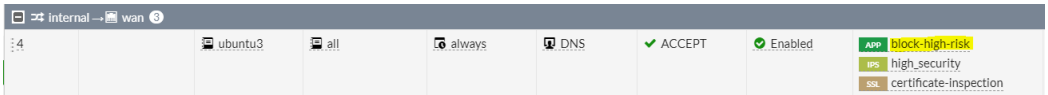


Ilustración 59 - regla Fortigate con inspección capa7

La caida del túnel DNS es inmediata cuanto se aplican los citados cambios.

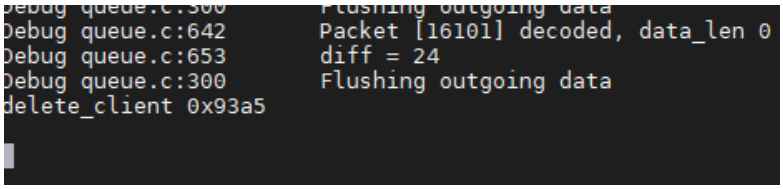


Ilustración 60 - DNS tunnel cae de inmediato

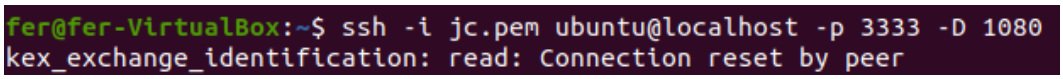


Ilustración 61 - la sesión SSH cae debido a que se corta el tráfico tunelizado

En este caso es el perfil de aplicaciones el que está detectando la herramienta dns2tcp. Esta vez la acción tomada por el equipo es block en vez de drop. La diferencia entre estas es que block envía un mensaje al dispositivo de origen confirmando el bloqueo y drop lo corta, pero no avisa a la fuente.

#		Date/Time	Source	Destination	Application Name	Action
1		4 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
2		9 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
3		15 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
4		20 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
5		25 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
6		29 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
7		30 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
8		35 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
9		35 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
10		40 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
11		41 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
12		45 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block
13		46 seconds ago	10.10.10.9	8.8.4.4 (dns.google)	TCP.Over.DNS	block
14		50 seconds ago	10.10.10.9	8.8.8.8 (dns.google)	TCP.Over.DNS	block

Ilustración 62 - logs Fortigate con aplicacion TCP.Over.DNS

La misma suerte corre Iodine si intenta levantar un túnel DNS a través del Fortigate con este perfil de aplicaciones aplicado. Será detectado y bloqueado.

#		Date/Time	Source	Destination	Application Name	Action
1		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
2		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
3		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
4		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
5		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
6		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
7		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
8		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
9		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
10		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
11		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block
12		30 minutes ago	10.10.10.9	8.8.8.8 (dns.google)	Iodine	block
13		30 minutes ago	10.10.10.9	8.8.4.4 (dns.google)	Iodine	block

Ilustración 63 - detección aplicacion Iodine

Los logs muestran más información, cabe destacar que a estos ataques el fabricante parece darles un umbral de riesgo más alto que a dnscat2 a pesar de que en esencia son los mismos, pero son detectados por dos motores diferentes.

Si además del perfil de aplicaciones en modo bloqueo a aplicaciones de tipo proxy le unimos un perfil de IPS que incluya la firma de dnscat2 seremos capaces de bloquear las tres herramientas.

Herramienta	Acción Fortigate
Dns2tcp	✗
Iodine	✗
Dnscat2	✗

Tabla 8 – Fortigate con inspección a capa de aplicación+IPS vs herramientas DNS tunnel

4.4.3 ICMP tunnel sobre Fortigate

Para probar los túneles ICMP en el Fortigate se van a utilizar la herramienta *hanstunnel* que se utilizó para burlar el iptables y la herramienta *icmptunnel,* vista en el punto 2.4, a modo de control.

Como en el anterior punto se observó que si no se aplican perfiles de seguridad el Fortigate actúa a nivel 4 y dado que no tiene capacidad para detectar y parar las técnicas de evasión se va a obviar esta parte. Se van a probar las herramientas con un perfil de control de aplicación bloqueando proxy y el perfil de IPS *high_security*, que aplica la totalidad de firmas descargadas en el Fortigate.

Las maquinas con las que se va a probar son las llamadas ubuntu3 y ubuntu4 del esquema presentado en el punto 3.2.

Se configuran dos reglas en el Fortigate para permitir únicamente el tráfico ICMP entre las subredes de las máquinas, con los perfiles mencionados.

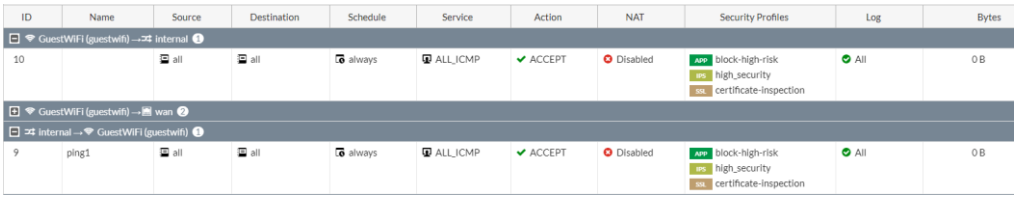


Ilustración 64 - reglas Fortigate ICMP

En primer lugar, probaremos el uso de hanstunnel. El túnel levanta correctamente y se prueba a lanzar una sesión SSH entre los extremos del túnel. Funciona correctamente, el Fortigate no ha podido detectarlo.

Mediante una captura de paquetes en el firewall podemos ver lo mismo que vimos en el punto 4.3.3, gran cantidad de sesiones ICMP con tamaños de paquete variable, algunos bastante pesados de hasta 1514 bytes.

No.	Time	Source	Destination	Protocol	Length	Info
406	53.568795	10.10.10.9	10.10.20.4	ICMP	99	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
407	53.568805	10.10.10.9	10.10.20.4	ICMP	60	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
408	53.570989	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
409	53.571006	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
410	53.571018	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
411	53.571029	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
412	53.571040	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
413	53.571051	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
414	53.571063	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
415	53.571073	10.10.10.9	10.10.20.4	ICMP	1514	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
416	53.571083	10.10.10.9	10.10.20.4	ICMP	60	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
417	53.571093	10.10.10.9	10.10.20.4	ICMP	99	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64
418	53.571103	10.10.10.9	10.10.20.4	ICMP	60	Echo (ping) reply id=0x5065, seq=58146/8931, ttl=64

Ilustración 65 - captura paquetes hanstunnel

Llama la atención que los datos de los paquetes comienzan por los caracteres *hans* y *hanc* (¿hans -s para el servidor y hans -c para el cliente?). Parece relativamente sencillo detectar el patrón y crear una firma para esta aplicación y categorizarla como proxy para el fabricante.

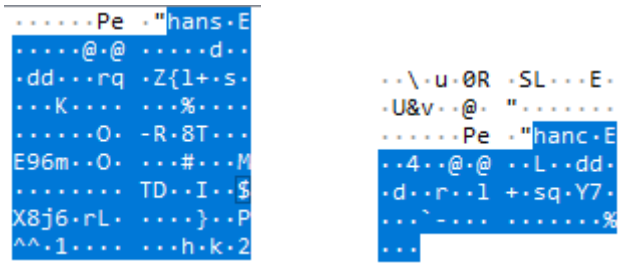


Ilustración 66 - patrones hanstunnel

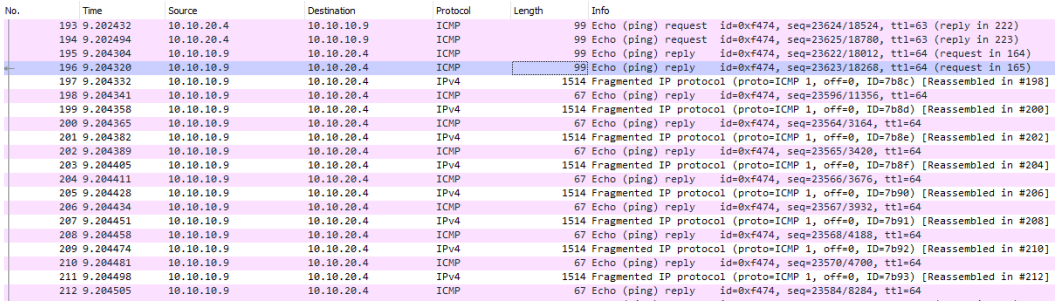
Respecto a la herramienta icmptunnel, el funcionamiento es similar a hans, ya que crea una interfaz de túnel en cada extremo, pero en esta hay que configurar manualmente está dando a cada interfaz una IP dentro del mismo rango.

```
fer@fer-VirtualBox:~/icmptunnel$ sudo ./icmptunnel 10.10.10.9
opened tunnel device: tun0
connection established.
^Z
[1]+  Stopped                 sudo ./icmptunnel 10.10.10.9
fer@fer-VirtualBox:~/icmptunnel$ bg
[1]+ sudo ./icmptunnel 10.10.10.9 &
fer@fer-VirtualBox:~/icmptunnel$ sudo ifconfig tun0 10.10.100.100 netmask 255.2
55.255.0
fer@fer-VirtualBox:~/icmptunnel$ ssh fer@10.10.100.1
fer@10.10.100.1's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

Ilustración 67 - detalle icmptunnel levantando sesión SSH

Al igual que con hanstunnel, el túnel levanta correctamente con esta herramienta y el firewall no es capaz de detectarlo. La diferencia al capturar paquetes es que icmptunnel utiliza paquetes más grandes que el MTU. Para ello hace uso de fragmentación de paquetes, seguramente para que sea más complicado detectar la técnica para el equipo cortafuegos.

Como con hans, también es fácil detectar patrones en el campo de datos de los paquetes ICMP, en este caso la cadena TUNL es observada como inicio en la mayoría de estos.



No.	Time	Source	Destination	Protocol	Length	Info
193	9.202432	10.10.20.4	10.10.10.9	ICMP	99	Echo (ping) request id=0xf474, seq=23624/18524, ttl=63 (reply in 222)
194	9.202494	10.10.20.4	10.10.10.9	ICMP	99	Echo (ping) request id=0xf474, seq=23625/18780, ttl=63 (reply in 223)
195	9.204304	10.10.10.9	10.10.20.4	ICMP	99	Echo (ping) reply id=0xf474, seq=23622/18012, ttl=64 (request in 164)
196	9.204320	10.10.10.9	10.10.20.4	ICMP	99	Echo (ping) reply id=0xf474, seq=23623/18268, ttl=64 (request in 165)
197	9.204332	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b8c) [Reassembled in #198]
198	9.204341	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23596/11356, ttl=64
199	9.204358	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b8d) [Reassembled in #200]
200	9.204365	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23564/3164, ttl=64
201	9.204382	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b8e) [Reassembled in #202]
202	9.204389	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23565/3420, ttl=64
203	9.204405	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b8f) [Reassembled in #204]
204	9.204411	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23566/3676, ttl=64
205	9.204428	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b90) [Reassembled in #206]
206	9.204434	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23567/3932, ttl=64
207	9.204451	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b91) [Reassembled in #208]
208	9.204458	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23568/4188, ttl=64
209	9.204474	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b92) [Reassembled in #210]
210	9.204481	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23570/4700, ttl=64
211	9.204498	10.10.10.9	10.10.20.4	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7b93) [Reassembled in #212]
212	9.204505	10.10.10.9	10.10.20.4	ICMP	67	Echo (ping) reply id=0xf474, seq=23584/8284, ttl=64

Ilustración 68 - captura paquetes icmptunnel

Como conclusión a este apartado, el firewall Fortigate no ha sido capaz de detectar la técnica de ICMP tunnel con ninguna de las dos herramientas probadas. Las protecciones del perfil de capa de aplicación no funcionan contra un protocolo como ICMP, que funciona en capa 3.

Herramienta	Acción Fortigate
hanstunnel	✅
icmptunnel	✅

Tabla 9 – Fortigate con inspección a capa de aplicación+IPS vs herramientas ICMP tunnel

4.5 Tunelización sobre la solución Cloud

Ahora vamos a probar las técnicas de evasión en un entorno en la nube, concretamente en Azure, evadiendo un Azure Firewall.

Como configuración compartida para el resto de puntos, se crean dos reglas de NAT para tener acceso a la gestión de las máquinas traduciendo los puertos 10022 y 10023 de la IP pública del Azure Firewall a los puertos 22 de las IP privadas de las máquinas azureubuntu1 y azureubuntu2.

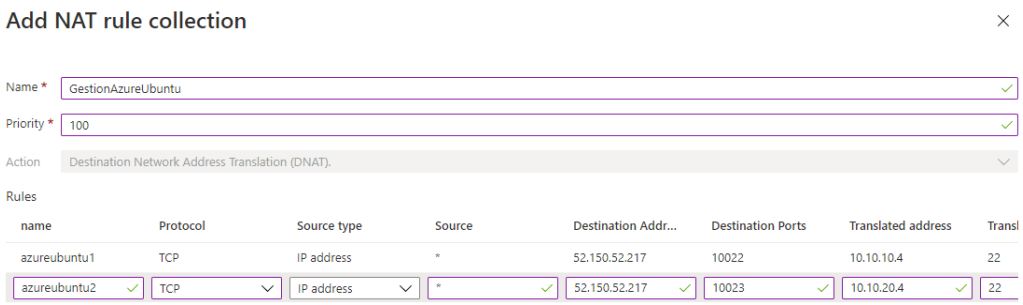


Ilustración 69 - regla de NAT para gestión en Azure

4.5.1 SSH tunnel sobre Azure FW

Lo primero que vamos a hacer en este caso es crear una regla para permitir conexiones SSH desde la máquina azureubuntu2 en la subred isolated hacia la azureubuntu1 en la red protected.

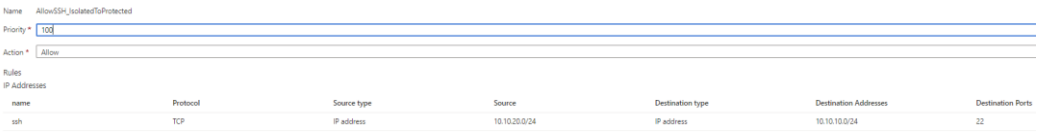


Ilustración 70 – regla Azure FW permitiendo SSH entre subredes

Además se copia la clave .pem mediante scp en azureubuntu2 para autenticarnos contra azureubuntu1.

También se permite a la máquina azureubuntu1 navegar por Internet a través de los puertos 80, 443 y 53 TCP. A la máquina azureubuntu2 se le deniega ese tipo de tráfico.

Desde la máquina azureubuntu2 se levanta la conexión SSH haciendo Port Forwarding del puerto 1080 TCP mediante el comando visto en el punto 4.1.2. Esta vez se le pasa el .pem como autenticación: **ssh -f -N -D 1080 -i .pem azureubuntu@10.10.10.4**

Debido a que en este entorno no se dispone de entorno gráfico las pruebas tendrán que hacerse desde la terminal. Haremos uso de la herramienta *proxychains* para salir a Internet desde la terminal utilizando un proxy SOCKS5 en el puerto 1080. La configuración de la herramienta se hace desde un archivo de texto llamado *proxychains.conf* en la ruta */etc/.*

```
azureubuntu@azureubuntu2:/etc$ curl https://www.google.com
curl: (35) OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to www.google.com:443
azureubuntu@azureubuntu2:/etc$ proxychains curl https://www.google.com
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-request| www.google.com
|S-chain|-<>-127.0.0.1:1080-<><>-4.2.2.2:53-<><>-OK
|DNS-response| www.google.com is 142.250.64.68
|S-chain|-<>-127.0.0.1:1080-<><>-142.250.64.68:443-<><>-OK
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><met
ures to help you find exactly what you're looking for." name="description"><meta content="n
/2020/december-holidays-days-2-30-6753651837108830.3-law.gif" itemprop="image"><meta conten
r:description"><meta content="December Holidays 2020 #GoogleDoodle" property="og:descriptio
"><meta content="https://www.google.com/logos/doodles/2020/december-holidays-days-2-30-6753
olidays-days-2-30-6753651837108830.3-2xa.gif" property="og:image"><meta content="1100" prop
dles/2020/december-holidays-days-2-30-6753651837108830.3-2xa.gif" property="og:url"><meta c
ndow.google={kEI:'9IHoX5b8BM_M_Aam8LyIAQ',kEXPI:'0,18168,1341241,730,224,5104,207,3204,10,1
6549,501,6,328978,13677,4855,32692,16114,19397,9287,9188,8384,4859,1361,283,9007,3026,4742,
840,518,1465,56,4258,312,1135,1,3,2063,606,2023,1777,143,377,1946,2230,93,329,1283,2943,224
20,1,4623,149,5990,4050,2325,1610,4,1528,2304,1238,269,874,405,1860,2393,74,1717,266,2163,1
130,1173,2550,140,2439,444,211,990,52,3284,2215,1593,712,638,37,631,826,130,427,48,2,1484,2
173,180,48,308,171,12,670,472,112,2,1148,58,778,88,256,19,20,16,354,105,915,804,925,5,524,8
55,488,384,611,4,2,7,143,147,324,423,114,718,2,5,2,8,603,181,1738,1,706,480,513,55,97,245,9
80,1,900,896,1,9,2,2551,1,748,200,736,4,559,1,4265,1,1,2,1017,9,305,3299,248,595,1,2301,19,
(function(){
google.lc=[];google.li=0;google.getEI=function(a){for(var b;a&&(!a.getAttribute||!(b=a.getA
||!(b=a.getAttribute("leid")));)a=a.parentNode;return b};google.ml=function(){return null};
```

Ilustración 71 - navegando con proxychains sobre SSH tunnel

Los resultados muestran que somos capaces de navegar a través del proxy cursando el tráfico por una sesión SSH a pesar de que la navegación directa no está permitida. Hemos evadido la política de seguridad configurada en el Azure Firewall.

Herramienta	Acción Azure FW
SSH+port forwarding	✅

Tabla 10 – Azure FW vs SSH tunnel

4.5.2 DNS tunnel sobre Azure FW

Pasamos ahora al establecimiento de un túnel DNS a través del dominio utilizado en el punto 4.2.1. Las reglas del Azure Firewall ahora permiten la salida a Internet de azureubuntu1 únicamente a través del puerto 53 TCP con lo que puede resolver DNS contra servidores públicos.

En primer lugar vamos a probar con la herramienta dns2tcp. El servidor ya está instalado en la máquina en AWS por lo que solo tendremos que levantarlo para que escuche peticiones en el puerto 53 TCP e instalar el cliente en azureubuntu1.

Levantamos el servidor y nos conectamos desde el cliente. En cuanto empezamos a cursar tráfico a través del túnel vemos en el servidor que entran paquetes, pero la sesión SSH con el Port Forwarding no llega a establecerse. Como el túnel SSH sobre DNS no llega a levantar, proxychains obtiene timeout en las peticiones que adelanta a través del túnel.

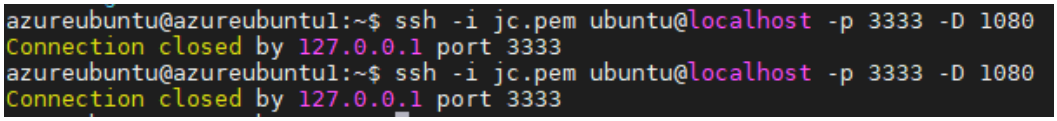


Ilustración 72 - Azure vs dns2tcp

Parece que la insfraestructura de Azure está bloqueando correctamente los túneles DNS. Para asegurarnos de esto, vamos a probar otra herramienta conocida para levantar túneles DNS, Iodine.

Con Iodine el túnel se establece y somos capaces de cursar tráfico correctamente. La fragmentación que utiliza esta herramienta está resultando diferencial para la evasión en Azure.

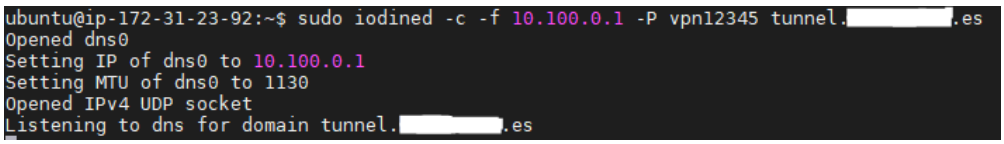


Ilustración 73 - servidor Iodine para DNS tunnel

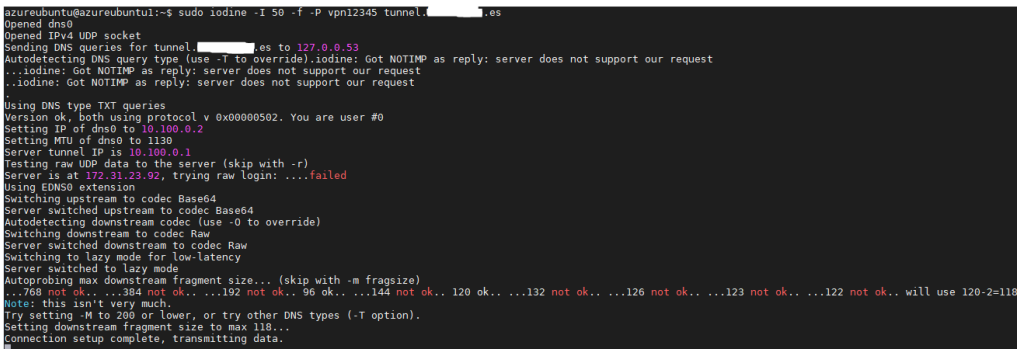


Ilustración 74 - cliente Iodine cursando tráfico por el túnel

Con dnscat2 el tunel levanta correctamente por lo que la infrastructura de Azure solo bloquea túneles DNS levantados con la herramienta dns2tcp.

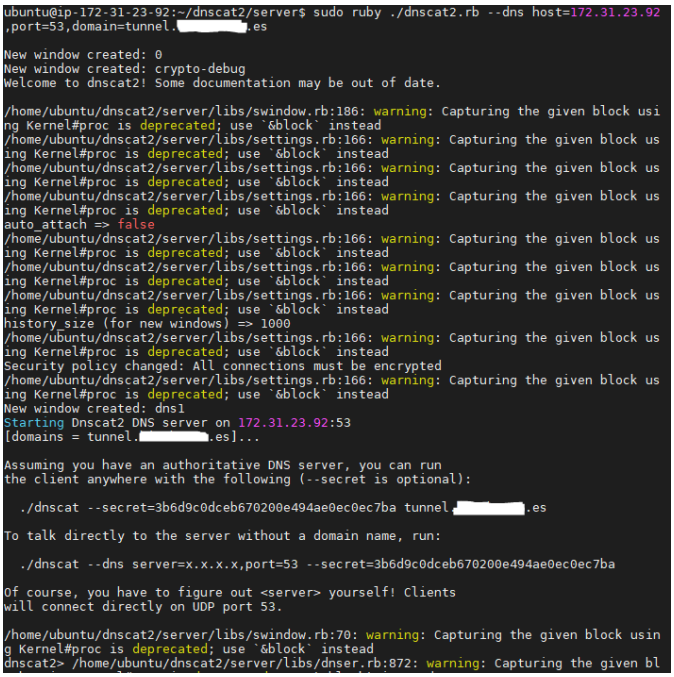


Ilustración 75 - servidor dnscat2

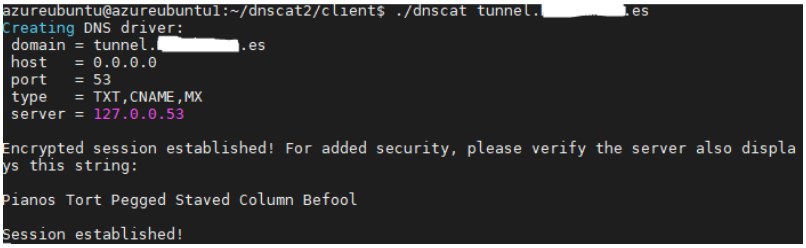


Ilustración 76 - cliente dnscat2 estableciendo sesión

Herramienta	Acción Azure FW
Dns2tcp	✗
Iodine	✓
Dnscat2	✓

Tabla 11 – Azure FW vs herramientas DNS tunnel

4.5.3 ICMP tunnel sobre Azure FW

Vamos a crear una regla que permita la comunicación ICMP entre las dos subredes, también crearemos otra para denegar el tráfico SSH.

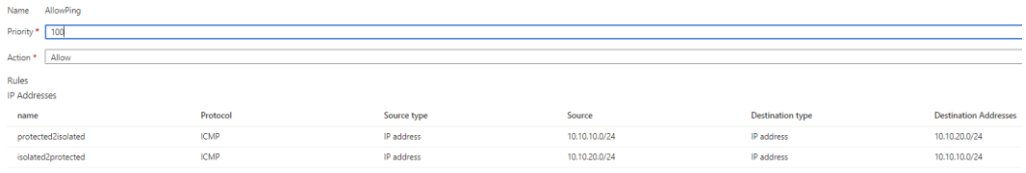


Ilustración 77 - Azure FW reglas ICMP

El túnel ICMP creado con las herramientas hanstunnel e icmptunnel funcionó correctamente y fuimos capaces de navegar mediante proxychains una vez adelantado el puerto 1080 a través de una sesión SSH, utilizando la interfaz de túnel creada mediante las herramientas.

Herramienta	Acción Azure FW
hanstunnel	✅
icmptunnel	✅

Tabla 12 – Azure FW vs herramientas ICMP tunnel

La infraestructura de Azure es vulnerable contra técnicas de evasión de seguridad de red a través de ICMP tunneling.

4.6 Domain Fronting

Ahora intentaremos evadir el firewall mediante la técnica de ofuscación Domain Fronting. Como dijimos en el apartado 2.5, esta técnica tiene su motivación en la lucha contra la censura en Internet.

Aunque esta técnica es efectiva contra bloqueos de ISP contra dominios específicos, vamos a explotar esta técnica a modo local para evadir un firewall con filtrado Web por categorías.

Para realizar el Domain Fronting se va a reutilizar el nombre de dominio referido en el punto 4.2, dns2tcp.*******.es. El dominio dado de alta en GoDaddy va a ser administrado mediante la CDN Cloudflare. Para ello, se crea una cuenta en la red de distribución y se configura el dominio. En GoDaddy actualizamos los servidores autoritativos de nombres con los de Cloudflare. De este modo los servidores DNS de la CDN resolverán las peticiones con sus IP propias, aprovechando las ventajas de la caché: la primera petición accederá al servidor desplegado en AWS, Cloudflare guardará la respuesta y las siguientes peticiones no tendrán que acceder al recurso, sino que serán respondidas con la respuesta almacenada.

Cuando la actualización se propague, una consulta whois sobre el dominio devolverá que sus NS son los de Cloudflare y podremos empezar a trabajar el Domain Fronting contra la CDN.

Para realizar esta técnica nos vamos a apoyar en la herramienta Noctilucent, comentada en el punto 2.6. Esta herramienta escrita en Go proporcionará un cliente capaz de establecer conexiones TLS 1.3 manipulando los valores de ESNI y SNI.

4.6.1 Domain Fronting sobre Fortigate

Se va a configurar el Fortigate para denegar el acceso a dns2tcp.******.es y permitirlo para eleyo.com, una plataforma educativa alojada en Cloudflare. Este sitio será el dominio con el que enmascararemos la petición DNS.

Para comprobar el correcto funcionamiento de la técnica, se va a desplegar un servidor web en la máquina en AWS a la que apunta dns2tcp.******.es.

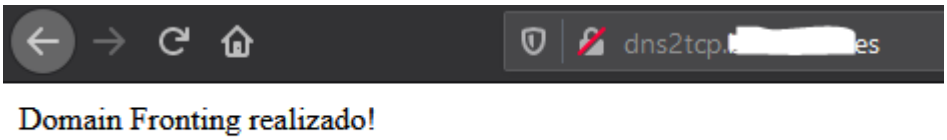


Ilustración 78 - detalle HTML para comprobar funcionamiento

Se comprueba que los contadores de bytes de las reglas del firewall están a 0 debido a que todavía no han cursado tráfico.

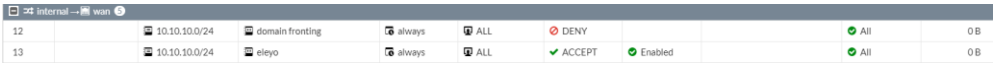


Ilustración 79 - reglas Fortigate para Domain Fronting

El valor del objeto “domain fronting” de la ilustración 79 corresponde con dns2tcp.******.es.

Una vez instalada la herramienta se lanza la conexión contra el dominio enmascarada, indicando el ESNI y el SNI.

```
fer@fer-VirtualBox:~/Noctilucent/client/build$ ./noctilucent-client-linux -TLSH
ost www.eleyo.com -esni -ESNIServerName dns2tcp.[redacted].es -HostHeader dns2tc
p.[redacted].es -serverName www.eleyo.com
[+] Using resolver: https://mozilla.cloudflare-dns.com/dns-query
[+] Successfully queried _esni TXT record for host: www.eleyo.com
[=] TLS 1.3 with TLS_CHACHA20_POLY1305_SHA256
[=] ESNI host set to: dns2tcp.[redacted].es
[=] SNI host has been unset
[+] Connecting to https://www.eleyo.com:443
[+] TLS handshake complete
[+] Sending GET request: GET / HTTP/1.1
Host: dns2tcp.[redacted].es
User-Agent: ESNI_FRONT_TEST
Accept: */*
Connection: close
```

Ilustración 80 - Domain Fronting en Cloudflare con Noctilucent

En la ilustración 80 se puede observar que se resuelve el registro TXT para eleyo.com. A continuación comienza el handshake en TLS 1.3 inyectando el ESNI del dominio enmascarado. Después se obtiene una respuesta HTTP 200 OK y la página de test desplegada en el servidor web de AWS.

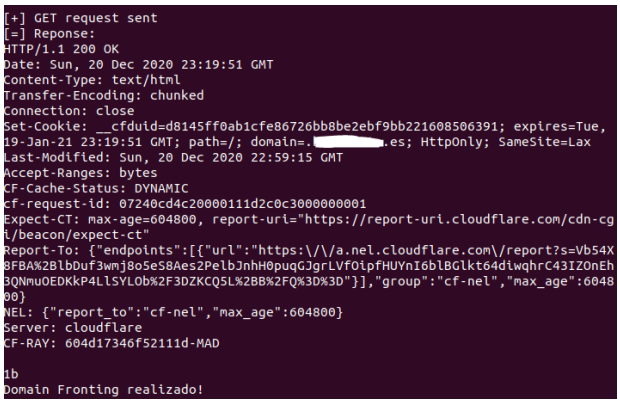


Ilustración 81 - Domain Fronting realizado

En la ilustración 82 se puede comprobar que el contador de la regla del firewall de eleyo ha aumentado. El firewall ha sido evadido.

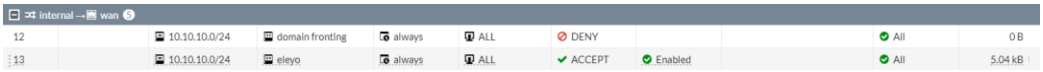


Ilustración 82 - comprobación regla matcheada

Los firewalls de nueva generación incorporan perfiles de seguridad de bloqueo de navegación web basado en categorías. Los fabricantes han categorizando multitud de dominios basándose en una serie de categorías definidas por ellos mismos. Las categorías no son fijas y están siendo observadas: si una página de noticias comienza a ser utilizado como phishing los fabricantes no tardarán demasiado en recategorizarla. A la hora de manejar una petición de navegación por parte de un usuario, los equipos cortafuegos consultan a una API que acepta peticiones de nombres de dominio devolviendo las categorías a la que pertenece. Si la categoría no está permitida se denegará la petición.

Un caso de uso más ajustado a la realidad es, mediante la técnica de Domain Fronting, evadir reglas de firewalls con un perfil de Web Control. Una entidad que proteja la navegación mediante un firewall de nueva generación podría crear un perfil que bloqueara páginas conocidas como maliciosas. Mediante Domain Fronting el firewall creerá que nos dirigimos a una página con la categoría del SNI.

Por ejemplo, eleyo.com es categorizada como *Information Technology* por Fortiguard, la API de categorización de Fortinet. El dominio utilizado durante este proyecto, sin embargo, es categorizado como *Newly Observed Domain*.

Se crea un perfil de Web Control que permita la primera categoría y bloquee la segunda. Se crea una nueva regla aplicando este perfil. Mediante Domain Fronting se está evadiendo esta protección.

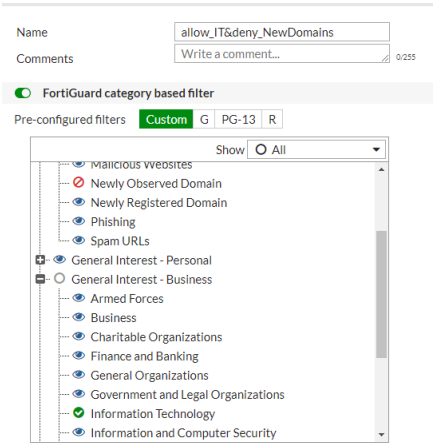


Ilustración 83 - perfil de filtrado Web basado en categorias

Para visualizar el comportamiento de Domain Fronting se han realizado una captura de paquetes en el firewall. Se puede apreciar la consulta DNS hacia eleyo.com que es respondida por los NS de Cloudflare con la IP de la caché de estos.

```
Transaction ID: 0xea15
> Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 5
Authority RRs: 0
Additional RRs: 1
v Queries
  v www.eleyo.com: type A, class IN
      Name: www.eleyo.com
      [Name Length: 13]
      [Label Count: 3]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
v Answers
  > www.eleyo.com: type CNAME, class IN, cname 2171647g47.secure0010.hubspot.net
  > 2171647g47.secure0010.hubspot.net: type CNAME, class IN, cname 2171647.group47.sites.hubspot.net
  > 2171647.group47.sites.hubspot.net: type CNAME, class IN, cname group47.sites.hscoscdn40.net
  > group47.sites.hscoscdn40.net: type A, class IN, addr 199.60.103.228
  > group47.sites.hscoscdn40.net: type A, class IN, addr 199.60.103.28
v Additional records
  > <Root>: type OPT
[Request In: 221]
[Time: 0.344390000 seconds]
```

Ilustración 84 - answer DNS

Si se inspecciona el TLS contra la IP devuelta por la consulta se puede ver el ESNI. Si pudiéramos desencriptarlo con la clave privada de Cloudflare en él aparecería el dominio dns2tcp.*****.es

```
v Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 570
    Version: TLS 1.2 (0x0303)
    Random: 905e32296cc8329708d0c0b3f51641b6b1d7df9829bec7f1…
    Session ID Length: 16
    Session ID: edd30b8ccdfd6c0ae4c5496a00bb8547
    Cipher Suites Length: 6
  > Cipher Suites (3 suites)
    Compression Methods Length: 1
  > Compression Methods (1 method)
    Extensions Length: 507
  v Extension: encrypted_server_name (len=366)
      Type: encrypted_server_name (65486)
      Length: 366
      Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    > Key Share Entry: Group: x25519, Key Exchange length: 32
      Record Digest Length: 32
      Record Digest: eb6ac531920ccdd9769da159db2c6c7adb35d44c6aceda0a…
      Encrypted SNI Length: 292
      Encrypted SNI: 385b9add91dc30e049f2481dd17b6567ee86df469359de80…
  > Extension: status_request (len=5)
  > Extension: supported_groups (len=10)
  > Extension: ec_point_formats (len=2)
```

Ilustración 85 - encrypted SNI

4.6.2 Domain Fronting sobre Azure

Vamos a replicar el escenario en Azure. Lo primero que haremos es cambiar las reglas del firewall para permitir el acceso a eleyo.com y bloquearlo a dns2tcp.*****.es.

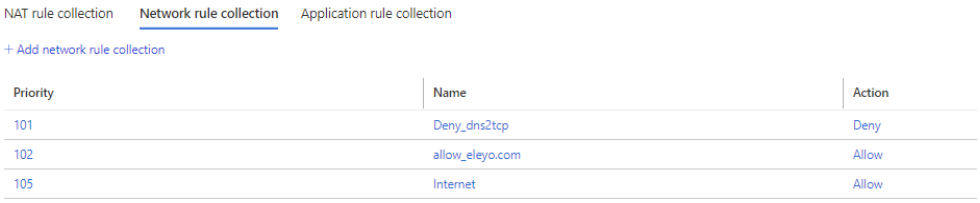


Ilustración 86 - reglas de Azure basadas en FQDN

Una vez configurado, lanzamos el mismo comando desde la máquina azureubuntu1. Como se puede ver en la ilustración 87, obtenemos una respuesta 403 (Forbiden).

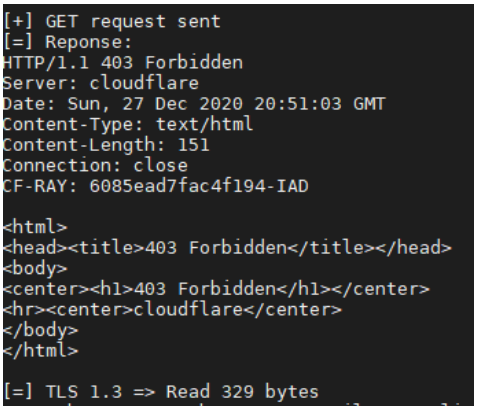


Ilustración 87 - respuesta 403 desde Azure

Desde el Fortigate, con el mismo comando obtuvimos un 200 OK. Parece que la infraestructura Azure no es vulnerable a la técnica de ofuscación Domain Fronting.

4.7 Comparación de velocidades de transferencia tunelizadas

Se ha realizado un estudio de la velocidad de transferencia del tráfico tunelizado por cada una de las herramientas vistas en los puntos anteriores. Para ello, se ha generado tráfico a través del túnel creado y se han capturado paquetes con la herramienta Wireshark durante un tiempo arbitrario. Mediante las herramientas estadísticas de esta se han obtenido las velocidades de transferencia. Se ha realizado la captura utilizando el entorno del firewall iptables para proteger los resultados de problemas de la red inalámbrica o de la latencia ocasionada debido a que los recursos de Azure se encuentran físicamente en EE.UU. y el servidor de AWS en Europa.

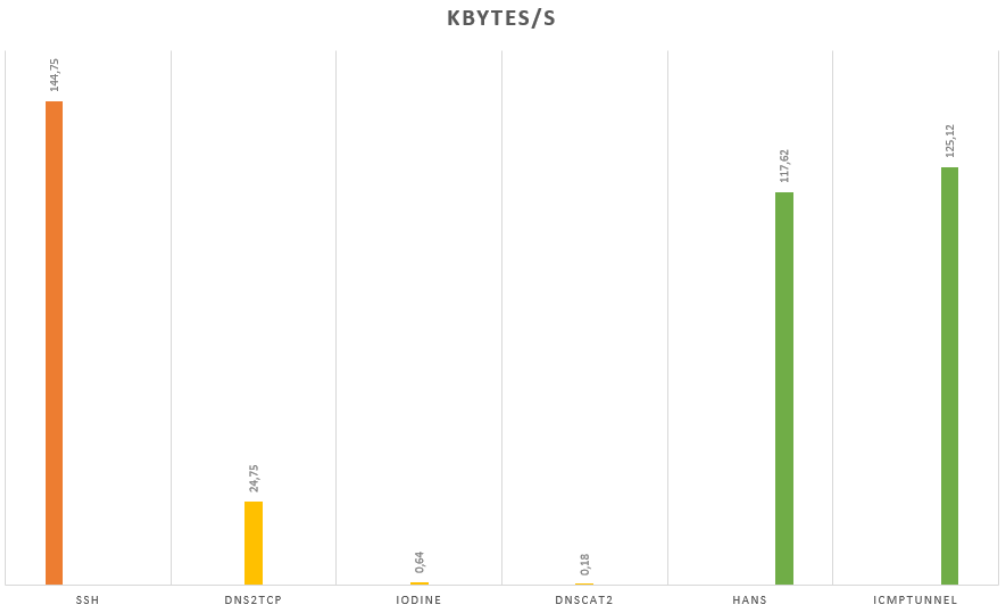


Ilustración 88 - velocidades de transmisión entre herramientas de tunelización

El tráfico generado ha sido de navegación hacia Internet. Se puede apreciar que el túnel SSH es el más rápido, a poca diferencia con las herramientas de tunelización ICMP. La tunelización DNS es la más lenta, es cierto que es el único caso en el que los dos extremos del túnel están separados por Internet, es decir, en los casos de SSH e ICMP, se tuneliza a través del iptables entre las subredes protected e isolated. En DNS se tuneliza entre protected y la red del servidor en AWS, atravesando Internet. Dentro de las herramientas DNS, dns2tcp destaca entre las demás porque es posible navegar, aunque sea de forma lenta. Las otras dos herramientas están pensadas para transferir pequeñas cantidades de bytes (instrucciones de C&C, claves de encriptación…).

4.8 Mitigación

En esta sección se va a tratar la mitigación de las técnicas para evitar que puedan ser aprovechadas por atacantes que ya han penetrado dentro de la entidad. Hay que recordar que todas estas técnicas de evasión y ofuscación son útiles para salir a través de un firewall, para ponerlas en práctica primero se debe haber conseguido acceso a la red interna.

Para evitar los túneles SSH, en primer lugar sería deseable tener segmentada la red, con subredes de servicio y subredes de gestión. Únicamente se debería permitir conexiones SSH con origen la red de gestión. Sobre el destino de las conexiones, cada caso necesitaría su estudio apropiado, pero lo que parece seguro es que la red de usuarios no debería permitir conexiones SSH entrantes.

Parece evidente que si las sesiones SSH están controladas será más complicado verse afectados por un incidente que incluya túneles SSH. Los firewalls de nueva generación pueden bloquear acciones específicas

dentro de SSH, pero podría darse el caso que los administradores necesitaran realizar estas acciones. En ese caso, aun siguiendo las buenas prácticas de segmentación y filtrado, todavía somos vulnerables a los usuarios administradores con privilegios para realizar estas conexiones. Los motivos para tunelizar conexiones hacia Internet de un administrador pueden ser múltiples, incluyendo la exfiltración confidencial de la empresa si un sysadmin quiere “poner a prueba” el sistema DLP de la compañía.

En el caso de túneles DNS la solución parece clara. Primero pasa por tener un servidor DNS interno y segundo, parece que los firewalls de nueva generación tienen capacidad para detectar este tipo de túneles fácilmente.

El túnel ICMP ha resultado indetectable para el firewall de nueva generación. Parece que tenemos dos opciones, o denegar el protocolo ICMP o disminuir el tamaño máximo de los paquetes TCP/IP.

Denegar el protocolo ICMP puede resultar en quejas de los administradores: se les están quitando herramientas necesarias para realizar su trabajo. La diferencia con las conexiones SSH es que probablemente utilicen el ping desde ordenadores de usuarios que han abierto un ticket porque no les funciona algún servicio, así que no va a ser efectivo el filtrado por origen/destino desde el firewall. Una alternativa serían las reglas de firewall por usuario, integrando este con el Directorio Activo de la entidad. Estas reglas aportan un campo más a inspeccionar en los flujos, el usuario que genera el tráfico. Se pueden crear reglas que permitan el tráfico ICMP para el grupo *Administradores* de AD. Puede parecer que se ha resuelto la papeleta solo a través de una solución de firewall, pero los administradores tendrían que iniciar sesión con su usuario en los equipos de los usuarios con lo que esto supone: además de ser incómodo, se estarían esparciendo contraseñas con privilegios de administración en forma de hash por todos los dispositivos de la compañía.

Sobre la segunda opción para forzar el tamaño máximo de los paquetes (MTU) para una interfaz, en los equipos Windows se puede realizar mediante la modificación del registro MTU en la ruta **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\<ID_INTERFAZ>**. Esta acción va a limitar el tamaño con lo que la herramienta hanstunnel no va a funcionar correctamente. Sin embargo cuando observamos el comportamiento de icmptunnel, vimos que utilizaba fragmentación y reensamblado de paquetes para burlar el MTU, con lo cual habría que buscar una forma de evitar esto. Limitar el MTU va a ocasionar problemas en el rendimiento de la red y si además se evita la fragmentación van a existir aplicaciones, además de icmptunnel, que no van a funcionar correctamente. Tampoco parece una opción válida.

Para la evasión de censura de sitios web utilizando Domain Fronting se tiene que romper el túnel SSL para ver la URL a la que se está intentando acceder y no basarse en la consulta DNS. De todas formas, las CDN parece que van solucionando las vulnerabilidades que permiten esta técnica de ofuscación.

5. Conclusiones

En este apartado se van a exponer las conclusiones en base al trabajo realizado en los anteriores puntos.

Sobre el firewall iptables podemos decir que hace tiempo ha quedado relegado a funciones educativas: funciona bien para aprender los conceptos de firewall, control del tráfico a nivel 4 y reglas de NAT, pero a día de hoy es vulnerable a este tipo de técnicas. Sin inspección de tráfico a capa de aplicación las soluciones de firewall no son efectivas, sistemas de este tipo no deberían estar en producción en este momento, ya que resulta relativamente sencillo burlarlos.

Sobre el firewall Fortigate, es una solución robusta que consiguió detener la mayoría de técnicas, no sin una complicada configuración para evitarlas. Es un firewall con una interfaz sencilla y agradable al usuario, pero no es útil para sacarle el máximo rendimiento. Será necesario repasar la documentación oficial, investigar en foros y realizar pruebas de configuración para endurecer la seguridad del equipo. Los perfiles por defecto no solucionan prácticamente ninguna de las técnicas de evasión que hemos visto. Le queda pendiente detectar y bloquear túneles ICMP, ya que no fue capaz de hacerlo. Sobre Domain Fronting, tampoco fue capaz de detectarlo, dejando en entredicho las capacidades de su perfil de filtrado Web. Por parte del autor, queda pendiente investigar si desencriptando SSL el Fortigate hubiera sido capaz de responder a la ofuscación. También hay que entender que un desencriptado SSL para los usuarios de la red implica una pérdida de la privacidad en la navegación y cuyo uso debe estar debidamente justificado y configurado para no vulnerar leyes como GDPR.

Las conclusiones sobre el Azure Firewall, es una solución más barata comparada con soluciones IaaS de fabricantes de seguridad reconocidos, pero no parece eficaz contra ataques en capa de aplicación. Ahora mismo es la solución de seguridad que ofrece Microsoft para los clientes que migren hacia su nube, pero no parece cumplir los requisitos de seguridad deseables: con una configuración basada en este equipo va a ser posible, entre otras cosas, la exfiltración de documentos confidenciales almacenados en un servidor virtual. Que fuera capaz de bloquear una herramienta de tunelización DNS (pero no otras 2) u otra basada en Domain Fronting hace pensar que la infraestructura cloud va a ir incrementando sus protecciones contra este tipo de técnicas.

Los túneles ICMP resultaron indetectables incluso para el Fortigate y no encontramos formas de mitigarlos desde el firewall o los equipos. Tiene sentido que las protecciones de capa de aplicación no sean efectivas contra un protocolo que funciona a nivel de capa de red. Una revisión manual detectaría anomalías en el tráfico ICMP, pero seguramente las consecuencias ya estarían latentes. De una forma más profesional esta

técnica podría detectarse mediante la creación de reglas de SIEM que dispararan una alarma cuando se detectaran estas anomalías. Las herramientas que se utilizaron requirieron de permisos de administrador en la ejecución, hace pensar en que para utilizar esta técnica primero debemos estar dentro del sistema y haber pivotado lo suficiente como para haber extraído credenciales de administrador. Una vez obtenidas se puede exfiltrar información o conectarse con un servidor de C&C por un protocolo del que no se esperan este tipo de comportamientos. Una vez hecho esto se podrían descargar claves de cifrado para ejecutar un ransomware, incluir los dispositivos a una botnet, etc.

La solución de Fortinet fue capaz de detectar y mitigar los túneles DNS, incluso la plataforma de Azure bloqueó una de las herramientas, lo que hace pensar en que son los túneles menos efectivos. También porque la mayoría de las entidades posee un servidor DNS interno lo que resulta más fácil detectar este tipo de técnica.

Sobre los túneles SSH, han resultado ser los más sencillos de ejecutar y combinar con el resto. No requieren de permisos de administrador a la hora de lanzarlos, aunque sí para configurar el servidor SSH para que admita Port Forwarding. Han podido ser detectados y bloqueados mediante el Fortigate, pero con una administración atendida del equipo, es decir, se ha probado la protección en un laboratorio y no en un entorno real de producción. Habría que probar con diferentes clientes SSH, valorar si se están bloqueando falsos positivos, permitir este tipo de sesiones para ciertos usuarios, etc.

Sobre Domain Fronting, extraemos la conclusión de que hoy solo es posible realizarlo en algunas CDN. A mediados del año pasado redes de distribución de contenidos como CloudFront o Cloud CDN desarrollaron protecciones contra esta técnica con lo cual, a día de hoy, se considera una técnica de ofuscación residual y que poco a poco acabará desapareciendo. A pesar de eso, a día de hoy es capaz de burlar un Fortigate sin desencriptado SSL y a la CDN de Cloudflare.

Como conclusión final, a día de hoy es factible evadir un firewall desde una red interna. La mera adquisición de un equipo de nueva generación más una configuración profesional sin tener en cuenta este tipo de técnicas hace que un incidente de seguridad pueda llevarse a cabo a pesar de las protecciones. La seguridad perimetral tiene que ser complementada con antivirus, EDR, sondas IPS y SIEM para ser efectiva contra todo tipo de incidentes de seguridad.

6. Glosario

Botnet: red de equipos infectados por malware destinados a ser controlados remotamente desde servidores de C&C con objetivos maliciosos, por ejemplo, ataques de DDoS.

C&C: *Command and Control*. Servidor o servidores desde los que se controla una botnet.

DDoS: *Distributed Denial of Service*. Tipo de ataque en el que se busca impedir la disponibilidad de un sistema a través grandes números de conexiones desde muchas localizaciones.

EDR: *Endpoint Detection & Response*. Tecnología de monitorización de dispositivos que centraliza la información acerca de los procesos que ha ejecutado el equipo, conexiones de red, ficheros que almacena, etc.

Exploit: software que explota una vulnerabilidad en un sistema informático para causar un error o un comportamiento incontrolado e inesperado de modo que obtiene el control del sistema, una elevación de privilegios o un ataque de denegación de servicio.

Firewall: sistema que controla y monitoriza tráfico de red entrante y saliente basándose en reglas de seguridad.

GDPR: *General Data Protection Regulation*. Reglamento para el tratamiento de datos de carácter personal de los ciudadanos y empresas afincados en la Unión Europea.

Hotspot: punto de acceso por el cual se puede lograr acceso a Internet a través de tecnología wireless.

IaaS: *Infrastructure as a Service*. Tipo de servicio cloud en el que el proveedor ofrece la parte física (servidores, almacenamiento, redes…) y el cliente controla el Sistema Operativo y los datos y programas que se ejecutan en él.

IPS: *Intrusion Prevention System*. Dispositivo de seguridad de red que analiza el tráfico en una red en busca de patrones de ataques conocidos. Es capaz de bloquear el tráfico que ha identificado con una firma de ataque.

Iptables: programa que permite controlar reglas de seguridad en el sistema operativo Linux.

ISP: *Internet Service Provider*. Empresa que ofrece servicios de conexión a Internet a entidades públicas, empresas y particulares.

Malware: software especialmente diseñado para mediante distintos métodos causar daños a equipos informáticos o para robar información privada de estos.

Man-In-The-Middle: Tipo de ataque en red en el que el atacante se sitúa entre dos participantes de una comunicación obteniendo la información de la sesión sin que estos lo noten.

Modelo OSI: *Open System Interconexion model*. Modelo para protocolos de red, basado en 7 capas. Es el estándar de funcionalidad para la comunicación entre sistemas.

Modelo OSI - Capa 4 (filtrado): capa de transporte. Trabaja con puertos lógicos. Un filtrado en esta capa controla los puertos origen y destino de las comunicaciones. Una regla de filtrado a capa 4 permitiría una sesión SSH configurada sobre un puerto diferente del 22. Bloquea el puerto pero no la aplicación.

Modelo OSI - Capa 7(filtrado): capa de aplicación. Un filtrado en esta capa permite controlar que el tráfico dirigido a un puerto es originado por una aplicación válida. Una regla de filtrado a capa 7 denegaría una sesión SSH aunque corriera sobre un puerto diferente al de por defecto.

MTU: *Maximun Transmision Unit*. Tamaño máximo que se puede transmitir en una única transmision de red.

NAC: *Network Access Control*. Dispositivo de seguridad que implementa una política de seguridad para proteger el acceso de dispositivos a la red local.

NAT: *Network Address Translation*. Método para mapear dos IP de diferentes redes. Como motivo principal, se utiliza para traducir una IP privada en una IP pública para que un dispositivo dentro de una red privada pueda salir a navegar a Internet.

On-premises: referente a la infraestructura local, cuyo entorno está dentro de las instalaciones de la organización y que, por tanto, no es cloud.

Port Forwarding: técnica que permite la redirección de un puerto de un sistema dentro de una red hacia otra red a la que esté conectado.

Sandbox: mecanismo de seguridad que consiste en la ejecución de programas no confiables dentro de máquinas virtuales para determinar si se trata de un programa malicioso basándose en los cambios observados en el entorno una vez ejecutados.

SIEM: *Security Information and Event Management*. Dispositivo de seguridad de red que almacena y correla logs de los diferentes equipos

conectados permitiendo análisis en tiempo real y alertas en caso de incidente.

SOCKS: protocolo de Internet de comunicación a través de proxy, que enruta el trafico dirigido a él reenviándolo hacia otra red.

Spoofing: ataque en el que el agresor suplanta a otro elemento para obtener una ventaja.

SSID: *Service Set Identifier*. Nombre por el que se identifica a una red WiFi.

Switch: Conmutador, dispositivo de red cuya función es la de interconectar equipos por medio de redes.

Ransomware: tipo de malware que encripta ficheros de los dispositivos a los que afecta haciendo imposible el uso por parte de los usuarios, pidiendo un rescate a cambio de la desencriptación.

Tunneling: técnica que consiste en encapsular un protocolo de red dentro de otro.

7. Bibliografía

Source Port Manipulation
"Dibyendu". (19/07/2019). How to Bypass Firewall using NMAP. *Node4tech*. https://www.note4tech.com/post/how-to-bypass-firewall-using-nmap

IP fragmentation
Dawid Czagan. (20/04/2015). Bypassing Packet Filters with IP Fragmentation Overlapping. *InfoSec Institute*. https://resources.infosecinstitute.com/bypassing-packet-filters-with-ip-fragmentation-overlapping/#gref

Stateful Inspection
Check Point Software Technologies, Ltd. (02/08/2005). Stateful Inspection Technology. *Check Point*. https://www.checkpoint.com/downloads/product-related/whitepapers/stateful-inspection-technology.pdf

Ataques MITRE
Strom, Applebaum, Miller, Nickels, Pennington, Thomas. (01/07/2018). MITRE ATT&CK: Design and Philosophy. *MITRE*. https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

Iptables
Korbin Brown. (27/08/2020). The Beginner's Guide to iptables, the Linux Firewall. *How to geek*. https://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/

"Linode". (07/10/2020). Controlling Network Traffic with iptables. *Linode*. https://www.linode.com/docs/guides/control-network-traffic-with-iptables/

Protocolo DNS y túneles
Cloudflare Inc. (01/01/2020). What is encrypted DNS?. *Cloudflare*. https://www.cloudflare.com/es-es/learning/dns/what-is-dns/

Aamir Lakhani. (01/07/2020). Into the Rabbit Hole – Offensive DNS Tunneling Rootkits. *Fortinet*. https://www.fortinet.com/blog/threat-research/into-the-rabbit-hole-offensive-dns-tunneling-rootkits

Roger Galobardes. (30/10/2018). Learn how easy is to bypass firewalls using DNS tunneling (and also how to block it) . *Medium*. https://medium.com/@galolbardes/learn-how-easy-is-to-bypass-firewalls-using-dns-tunneling-and-also-how-to-block-it-3ed652f4a000

Protocolo ICMP y túneles
Raj Chandel. (28/07/2019). Command and Control & Tunnelling via ICMP. *Hacking Articles*. https://www.hackingarticles.in/command-and-control-tunnelling-via-icmp/

Daniel Stødle. (26/05/2005). Ping Tunnel. *Mit.edu*.
https://www.mit.edu/afs.new/sipb/user/golem/tmp/ptunnel-0.61.orig/web/

Nir Chako. (17/12/2008). Ping Power — ICMP Tunnel. *Medium*.
https://medium.com/bugbountywriteup/ping-power-icmp-tunnel-31e2abb2aaea

Jeff Parker. (15/01/2020). What is ICMP? The Protocol, Port Number and PING!. *Pc&Networks Downloads*.
https://www.pcwdld.com/what-is-icmp-and-port

Domain Fronting

Cloudflare Inc. (01/01/2020). What is encrypted SNI (ESNI)?. *Cloudflare*.
https://www.cloudflare.com/es-es/learning/ssl/what-is-encrypted-sni/

David Bueno. (11/09/2019). Túneles en la sombra 3: Domain Fronting. *Security-Garage*. https://security-garage.com/index.php/herramientas/tuneles-en-la-sombra-3-domainfronting-3

Artem Rukavytsia. (08/07/2019). Domain Fronting in a nutshell. *Hackernoon*.
https://hackernoon.com/domain-fronting-in-a-nutshell-159e21bf23a4

James Sanders. (02/05/2018).
As Google and AWS kill domain fronting, users must find a new way to fight censorship. *Tech Republic*. https://www.techrepublic.com/article/as-google-and-aws-kill-domain-fronting-users-must-find-a-new-way-to-fight-censorship/

“Digi-Ninja”. (11/02/2019). A 101 on Domain Fronting. *Digi-Ninja.*
https://digi.ninja/blog/domain_fronting.php

Censura en Internet

Catalin Cimpanu. (18/07/2019). Kazakhstan government is now intercepting all HTTPS traffic. *Zero Day.* https://www.zdnet.com/article/kazakhstan-government-is-now-intercepting-all-https-traffic/

Modificando MTU en Windows

Microsoft. (21/04/2018). Configuración TCP/IP recomendada para los vínculos WAN con un tamaño de MTU inferior a 576. *Microsoft.*
https://support.microsoft.com/es-es/help/900926/recommended-tcp-ip-settings-for-wan-links-with-a-mtu-size-of-less-than

Azure

Taylor, Rabeler, Dwivedi, Mabee (03/12/2020). What is Azure Virtual Network? *Microsoft.* https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

David Coulter. (08/10/2020). Azure Firewall features. *Microsoft.*
https://docs.microsoft.com/en-us/azure/firewall/features

8. Anexos

Configuración iptables para túnel SSH

sudo iptables -F
sudo iptables -X
sudo iptables -P FORWARD DROP
iptables -A FORWARD -p tpc --dport 22 -s 10.10.20.0/24 -d 10.10.10.0/24 -j ACCEPT
iptables -A FORWARD -p tpc --sport 22 -s 10.10.10.0/24 -d 10.10.20.0/24 -j ACCEPT
iptables -A FORWARD -p tcp --match multiport --dports 53,80,443 -o enp0s9 -s 10.10.10.0/24 -j ACCEPT
iptables -A FORWARD -p tcp --match multiport --sport 53,80,443 -i enp0s9 -d 10.10.10.0/24 -j ACCEPT
iptables -t nat -A POSTROUTING -s 10.10.10.0/24 -o enp0s9 -j SNAT --to 192.168.1.150

Configuración iptables para túnel DNS

iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -t nat -A POSTROUTING -s 10.10.20.0/24 -o enp0s9 -j SNAT --to 192.168.1.150
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -j ACCEPT

Configuración iptables para túnel ICMP

systemctl status firewalld
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -A FORWARD -p icmp -s 10.10.20.0/24 -d 10.10.10.0/24 -j ACCEPT
iptables -A FORWARD -p icmp -s 10.10.10.0/24 -d 10.10.20.0/24 -j ACCEPT