FortiGate – Guía de resolución de problemas

Hubert Wisniewski

Título original:

FortiGate - Troubleshooting Guide Quick Reference

Traducción: Efren Teruel Dominguez

Sobre el autor

Hubert Z. Wisniewski es un *Principal Network Engineer* en AT&T. Lleva trabajando en el sector de las telecomunicaciones por más de 20 años, 10 de los últimos especializándose en redes y seguridad. Tiene las siguientes certificaciones: *Fortinet NSE4 NSE5 NSE7, CCNP R&S Sec, CCDP, CompTIA Network+ Security+*. Hubert, como Instructor Oficial *Fortinet*, imparte cursos oficiales. También ha desarrollado un *Fortigate Bootcamp*. Hubert es miembro del *Cisco Security Exam Advisory Group.*

Sobre los revisores técnicos (en orden alfabético)

Eduard Dulharu hace más de 6 años que desarrolla su labor como Arquitecto de Redes Sénior en AT&T. Tiene en total más de 10 años de experiencia en diseñar, resolver incidencias e implementar redes de gran tamaño con diferentes compañías y clientes.

Efren Teruel Dominguez trabaja como Ingeniero de Redes Sénior en AT&T, habiendo estado presente en todaslas fases de implementación de las redes, desde *deployment*, desarrollo o *troubleshooting*. En el pasado trabajó en Amazon, con base en Dublín y posteriormente en Seattle, enfocado más en labores de automatización. Es miembro del *Cisco Service Provider Advisory Group* como *SME*.

Giovanni Pagano Dritto trabaja como consultor de redes y programador. Lleva en la industria de IT más de 10 años y tiene una gran experiencia en multitud de distintas tecnologías y compañías. Es también programador en Python con trabajos que abarcan distintos sectores en el campo de las telecomunicaciones.

Lucian Lisov es un Ingeiero de Redes Senior en AT&T, con más de 10 años de experiencia. Especializado en infraestructuras de Data Centre a nivel *Enterprise*, dedica la mayoría de su tiempo a la nube (privada y pública) y a la automatización.

Advertencia, exención de responsabilidad, copyright, marca registrada, reconocimientos y errata

La información que se encuentra en este libro se ofrece como una guía de referencia que puede ayudar al lector en su día a día trabajando con equipos Fortigate, y como tal, debe ser considerado. El autor rechaza cualquier responsabilidad motivado por error u omisión, incluyendo sin límite los daños que puedan ser resultados por el uso de este libro.

Las opiniones expresadas en este libro pertenecen al autor y no son necesariamente las mismas que las de Fortinet.

Errata: http://myitmicroblog.blogspot.com/2020/04/errata.html

Nota del traductor

Como bien es sabido por toda la gente que nos dedicamos al sector de las telecomunicaciones, la mayoría de la documentación, comandos, bibliografía y demás, están siempre en inglés. La traducción de un libro técnico como este se puede hacer algo complejo, y, para que tenga sentido en español, hay que huir de la traducción literal (¡prometo no haber traducido ningún remove a remover!). Dicho esto, he decidico hacer uso de algunos (muchos..) vocablos en inglés, siempre haciendo referencia a su traducción en español en su primer uso. Siguiendo los consejos (designios) de la RAE, he puesto en cursiva los anglicismos. Nombres propios, como el de los protocolos, aparecen en mayúsculas. Hemos dedicado bastante tiempo a la elaboración, revisión y, después, traducción de este libro. Espero que os facilite vuestro día a día y os haga vuestra labor más sencilla (y amena).

Efren Teruel Dominguez

Sobre el libro

FortiGate – Guía de resolución de problemas presenta técnicas y métodos sencillos sobre la resolución de fallos en plataformas FortiGate. En el libro intento hacer uso de distintos ejemplos de comandos *debug* que sirven para explicar cómo leer y entender toda la información que nos pueda ofrecer. El objetivo de este libro no es enseñar o explicar cómo funciona en sí el dispositivo. No se explican ejemplos de configuración o diseño ya que se dan por entendido que son conocidos por el lector. Si no te sientes seguro sobre cómo actuar ante una incidencia, entonces, este libro es para ti.

Contenido

1 Prefacio

La resolución de fallas o *troubleshooting* es un elemento clave junto con el diseño de redes e implementación y una de las habilidades más solicitadas hoy en la industria IT. Una de las principales diferencias es el tiempo que lleva completar una tarea en cada uno de estos trabajos. No podemos comparar lo que nos llevaría diseñar una red entera con intentar solucionar una incidencia. Además, en este último, siempre trabajamos con presión, ya que, al ser un problema, todo el mundo quiere saber el porqué del fallo y, sobre todo, como resolverlo **ya**. En mis más de 20 años de experiencia, he participado en cientos de llamadas de *tickets* que afectaban a uno u otro servicio, y he aprendido los siguientes 'principios' que debes saber antes de empezar:

- Cuál es el problema
- Entender el diseño o el diagrama sobre la red
- Tener un entendimiento mínimo sobre los protocolos/aplicación
- Disponer de un enfoque bien estructurado sobre la resolución de fallas
- Conocimientos de los comandos necesarios

El primer punto puede parecer ciertamente obvio, pero creedme que, tras 10 horas en una llamada donde cada uno tiene una visión o punto de vista bastante parcelado sobre el problema, a veces se puede convertir en algo muy abstracto, con demasiadas ramificaciones y al final te puedes perder con toda esa información. Mi consejo es, no tengas miedo a preguntar, aunque con ello pueda parecerte que no dominas la materia o la situación. Es mucho peor, tras varias horas de llamada, que no sepas de qué va el problema. Las redes hoy día pueden ser muy complejas, se mezclan diversos elementos como virtualización, VRFs, contextos, etc. Y las aplicaciones también pueden serlo. Hay que tener en cuenta que nosotros, como Ingenieros de Redes o Seguridad, no somos especialistas en sistemas, o aplicaciones o base de datos. ¡Ciertamente no se puede saber de todo! Debemos saber racionalizar la información que nos dan y conocer los síntomas, que se puedan extrapolar en algo tan sencillo como: no hay acceso entre este origen y el destino", con lo que ello conlleva, direcciones IPs, puertos o aplicaciones que se usan, etc.

El Segundo punto de la lista trata sobre el diseño de redes. Con esto me refiero sobre todo a entender todos los componentes que puedan estar involucrados, esto es, que sean relevantes al flujo de información o *traffic Flow*, tipo *VRFs*, *VLAN*, los citados contextos, tablas de enrutamiento, etc. Lamentablemente, a veces no tenemos el privilegio de conocer toda esa información antes de que surja cualquier incidencia. Puede pasar que a veces nos tengamos que meter en una llamada para resolver un problema en un entorno que nos sea completamente desconocido. No pasa nada, simplemente necesitaremos algo más de tiempo para entenderlo.

Los primeros dos elementos de la lista los podemos aprender antes o durante la llamada. Evidentemente, es lógico que dediquemos tiempo al principio para recabar toda la información posible.

Sobre los otros tres puntos, los necesitaremos conocer o manejar antes de que el problema surja. Una vez conozcas toda la información mínima, como Ips, protocolos, etc., podremos empezar cuanto antes a realizar una investigación para intentar determinar dónde está la falla. Es necesario disponer de un conocimiento mínimo para no divagar demasiado entre uno u otro apartado. Si no entendemos con lo que estamos trabajando, vamos a acabar saltando de un elemento a otro, perdiendo tiempo y la paciencia de los clientes. Una vez tengamos la experiencia suficiente, todo será más sencillo, hasta a veces resultará obvio el método a utilizar. Por ejemplo, si entendemos las fases de *IPsec* y sus variaciones (fase 1, agresivo contra modo *main*), tiene sentido que verifiquemos la fase 1 como primera opción. Si esta falla, no se puede establecer la negociación IPsec SAs. Si la conexión de internet falla cada varios minutos, haciendo que la interfaz tunnel VPN caiga, no tiene mucho sentido que revisemos primero la aplicación, que corre sobre dicha VPN. Si algo no funciona, verifica que la tecnología subyacente está bien: conexión a internet -> VPN fase 1 -> VPN fase 2 -> Conectividad entre los equipos, etc.

Hay varios métodos de *troubleshooting* que puedes seguir (de abajo a arriba -*down level up*-, de arriba abajo –*up level down*-, sigue al *path* –*follow the path*-, etc.), pero, personalmente, prefiero elegir cuál de ellos tomar una vez tenga toda la información sobre el incidente en el que trabajo.

El último punto (Conocimientos de los comandos necesarios) se basa en saber los comandos y distintas herramientas de las que disponemos para realizar el *troubleshooting*. Si no estamos familiarizados con la plataforma, puede ser que pasemos más tiempo intentando averiguar cómo podemos acceder a información relevante que nos puedan pedir en la llamada, más que en resolver la incidencia en la que nos encontremos. Evidentemente, esto no es aceptable. Uno no puede memorizar todos los comandos, sobre todo si trabajamos con diferentes plataformas, pero sí es útil tenerlos a mano. Los conocidos *cheat list* son un buen ejemplo para tener la información a nuestra disponibilidad de manera esquemática. Hoy día, con la cantidad de sistemas, con una sintaxis diferente cada uno, ya ni intento recordar todos los comandos. Es por esto precisamente que me decidí a escribir este libro para ayudarte con esto. Espero que encuentres la información que he recapitulado y escrito en este libro útil durante tus sesiones de *troubleshooting*. ¡Buena Suerte!

2 Traffic flow

Traffic flow hace referencia al flujo de información o de comunicación entre dos equipos. El problema más común con el que nos vamos a enfrentar es la falta de conectividad entre dos equipos o *hosts*. La mayoría de las veces oiremos a nuestro cliente decir "la aplicación ha dejado de funcionar" o "ya no hay acceso a..". A veces, si no ha habido ningún cambio reciente, puede ser debido al fallo de un enlace o un nodo (relacionado con la capa 1 de OSI). Puede que un equipo, como el Firewall, este bloqueando ese tráfico de manera intencionada. De una manera u otra tenemos que averiguar por qué está sucediendo. Con la plataforma *Forgitate* podemos hacer uso de multitud de herramientas que nos pueden ayudar durante la investigación. Durante los siguientes episodios os mostraré cómo y cuándo usarlas.

2.1 Sesión de diagnóstico

Antes de empezar a meternos en el *troubleshooting*, vamos a echarle un vistazo a cómo verificar si el tráfico simplemente está permitido o denegado. Existen un par de métodos:

a) Verificar la tabla de sesiones –vista simplificada:

Ejemplo 2.1

```
forti (test) # get system session list
PROTO   EXPIRE  SOURCE                  SOURCE-NAT   DESTINATION         DESTINATION-NAT
tcp     3559    10.0.48.139:55506 -                  192.168.40.20:27017 -
udp     132     172.16.52.11:36094 -                 172.16.161.22:53 -
tcp     4       192.168.134.152:60655 -              10.2.3.45:135 -
udp     118     192.168.134.87:52408 -               10.2.3.20:53  -
tcp     9       172.16.105.252:3178 -                192.168.2.188:443 -
```

En la figura 2.1 podemos comprobar que, efectivamente, el tráfico está permitido. Vamos a analizar más en detenimiento la información que nos ofrece el comando:

- PROTO es el protocolo, y en dicho ejemplo podemos ver sesiones TCP y UDP.
- EXPIRE, es un temporizador TTL de cuenta atrás (*TTL session timer*). Su valor por defecto para TCP es 3600 segundos.
- SOURCE – En la tercera columna, es la IP y el puerto origen.
- SOURCE-NAT – Al no haber ningún valor en esta columna, significa que no hay source NAT configurado.
- DESTINATION – En la quinta columna, IP y puerto destino.
- DESTINATION-NAT – Tampoco hay NAT para el destino.

Más adelante explicaré *Source y Destination NAT*

En producción, la lista con las sesiones puede ser muy larga. No existe un subcomando para poder filtrarlo (*built-in*), pero grep nos valdrá para dicho propósito:

Ejemplo 2.2

```
forti (test) # get system session list | grep tcp
```

b) Verificar la tabla de sesiones – vista detallada

Existe una versión del comando ***session list*** con más opciones. En este caso he señalado los más importantes en el ejemplo de abajo.

Ejemplo 2.3

```
forti (test) # diagnose sys session list


session info: proto=6 proto_state=05 duration=2 expire=0 timeout=3600 flags=00000000
sockflag=00000000 sockport=0 av_idx=0 use=5

origin-shaper=
```