Material de descarga - Módulo 4
Anuncios de rutas y decisiones administrativas en BGP

Introducción:

Bienvenidos al cuarto módulo del curso “Fundamentos de BGP e introducción a RPKI” de LACNIC. En esta oportunidad se podrá aprender cómo se incorporan ciertos tipos de prefijos en la tabla de BGP, además de, cómo controlar sus anuncios a través de los filtros.

Al finalizar estos temas se podrá ejercitar sobre filtrado de prefijos entrantes y salientes.

TEMA 1 - Cómo incorporar prefijos en BGP

Se ha visto en el módulo 2 de este curso que para incorporar un prefijo en la tabla de BGP podía hacerse de 3 maneras diferentes: a través de otras sesiones BGP, con el comando **redistribute**, o con el comando **network**. En este último caso quedó claro que para que una ruta pueda ser anunciada, no sólo debe estar en la tabla de BGP, sino que una ruta coincidente deberá estar, previamente, en la tabla de ruteo.
Un ejemplo de esta implementación podría ser la siguiente:

```
router bgp 64496
     network 203.0.113.128 mask 255.255.255.128

ip route 203.0.113.128 255.255.255.128 serial0
```

donde la ruta que se quiere anunciar a través de BGP, también existe en la tabla de ruteo gracias a la implementación de una ruta estática que se ha configurado en el *router*.

De alguna manera, lo que se está haciendo con esta ruta estática, es forzar al atributo “origin” para que se muestre como “i”, o sea, a ser “IGP”.

Ahora, qué pasa si no existe tal ruta en la tabla de ruteo, bajo ninguna

de sus formas? o sea, si no hay ruta estática ni ruta aprendida por un IGP, que coincida con la ruta que se quiere anunciar con el comando **network**? En esos casos, habrá que acudir a algún método que permita generar una ruta para agregar a la tabla de ruteo.

La forma de resolver esta situación es mediante la incorporación de la siguiente sentencia basada en el ejemplo ya mostrado:

```
ip route 203.0.113.128 255.255.255.128 null0

router bgp 64496
 network 203.0.113.128 mask 255.255.255.128
```

Se trata de incorporar en la tabla de ruteo una ruta estática ~~a~~ "null 0", es decir, una ruta que no apunta a un next-hop sino que descarta los paquetes. Este tipo de ruta se denomina "pull up", y sólo será utilizada si no existe una ruta más específica para la tabla de ruteo.

Por lo general, lo que se hará es insertar este tipo de rutas para prefijos sumarizados: de esa forma, si existen prefijos más específicos en la tabla de ruteo, serán utilizados. Si no existe ningún prefijo específico para alguna porción de la ruta sumarizada, se utilizará la ruta a "null 0" y los paquetes serán descartados. Por ejemplo, si se cuenta con un rango /20, en BGP se debería publicar hacia los *peers* el rango /20 sumarizado y, por lo tanto, la ruta "pull up" a null 0 será el /20, mientras que habrá rutas más específicas para los /24 o prefijos más largos dentro del sistema autónomo.

Siguiendo con el tema de cómo insertar prefijos en la tabla de BGP, hay un caso que es muy específico y que tiene que ver con la redistribución de la ruta default.

Después de lo que hemos visto, resulta natural que el modo de anunciar la ruta default sea a través del comando "network 0.0.0.0". Esto claramente puede hacerse y es correcto, pero habrá que tener en cuenta dos cosas. Una es que se estará anunciando el default a todos los *neighbor*, y la otra es que deberá existir la misma ruta, sí o sí, en la tabla de ruteo para que pueda ser anunciada.

Otra opción es usar el comando "**default-information originate**". Al igual que en el caso anterior, la ruta default será anunciada a todos los *neighbors*, pero este comando tiene la particularidad que anunciará la ruta **default** sin necesidad de que la ruta esté en la tabla de ruteo.

Finalmente, otra opción posible es utilizar el comando: **neighbor <IP_Neighbor> default-originate.** En este caso, la ruta **default** solo se anuncia a un *neighbor*, el indicado en el comando, y al igual que en el

comando “default-information originate”, tampoco es necesario que la ruta esté previamente en la tabla de ruteo para que pueda ser anunciada.

Como se ve, el anuncio de rutas, y en particular el de las rutas por default, merece especial atención, pues lo que se anuncie a los vecinos será tomado en cuenta para que estos redirijan su tráfico.

Por lo dicho anteriormente, es muy aconsejado el uso de filtros en las sesiones BGP para controlar los anuncios y las rutas que se aprenden en nuestro AS.

TEMA 2 - Filtros por IP, AS_PATH y route-maps

Tal como se ha visto en el tema anterior sobre la incorporación de rutas a la tabla de BGP y los efectos de ciertos anuncios, es muy importante que el anuncio de las rutas se realice en forma responsable. De forma análoga, aprender rutas debe ser un proceso que esté bajo nuestro control.

Hay varias razones para querer controlar el tráfico entrante y saliente del AS, y algunas podrían ser no tan obvias. Por ejemplo, técnicamente es importante controlar la cantidad de rutas que los *routers* manejan, pues todo incide en la capacidad de procesamiento, uso de la memoria, entre otros. Además de esto, un ISP por ejemplo, estará interesado en que las rutas que envía y recibe de un cliente sean las acordadas, por cuestiones económicas y contractuales. De la misma manera, existen costos asociados a utilizar un sistema autónomo como AS de tránsito. Todo esto sin mencionar que la seguridad que se debe brindar a los clientes con las rutas que les anunciamos es crucial para dar un buen servicio.

En fin, son muchas las razones para querer tener procesos de anuncio y aprendizaje de rutas controlados. Para ello, una de las formas de contribuir al control de las rutas es a través de los filtros de BGP.

Se verá a continuación qué tipos de filtros hay y cómo se implementan.

Existen dos grandes tipos de filtros en BGP: los filtros basados en direcciones IP y los filtros basados en el *path*.

Cuando se refiere a los filtros basados en direcciones IP, hay dos subgrupos. Aquellos denominados “**distribute-list**”, los cuales se implementan con **access-list**, las mismas que los lectores seguramente ya conocen y han aplicado a las interfaces de los routers. Y por otro lado, están aquellos denominados “**prefix-lists**”, cuya implementación es más reciente que la de **distribute-list**. Por esta razón, este curso se abocará sólo a los prefix-list cuando explique los filtros basados en direcciones IP.

Dentro de los filtros basados en el *path*, están los denominados "**filter-list**", los cuales analizan y aplican los filtros de acuerdo a la información que contiene el atributo **AS_Path**.

Importante: se debe tener en cuenta que los filtros pueden ser aplicados a la entrada o a la salida de una sesión BGP. Si se aplican a la entrada, entonces estarán afectando lo que se aprende del *neighbor*. En cambio si se aplican a la salida, los prefijos afectados serán los que se anuncian al vecino.

A continuación se mostrará cómo implementar el filtrado de rutas con **prefix-list**.

El primer paso es crear el **prefix-list** que se aplicará luego al vecino.

Para ello, y aunque no se verán filtros implementados con **access-list**, será útil repasarlas, y ver las diferencias con el **prefix-list**.

Como recordarán, las **access-list** tiene esta forma:

```
access-list <nro_access-list> permit|deny ip <prefijo>
<máscara de wildcard>
```

(El comando deberá escribirse en una sola línea)

Las listas de acceso se construyen con múltiples sentencias como la indicada, donde cada sentencia especifica qué se quiere filtrar o dejar pasar. Por ejemplo, en la lista de acceso 101 que se muestra a continuación, se especifican dos sentencias, las cuales están indicando que se permitirá el anuncio o el aprendizaje de los prefijos 10.0.0.0/8, 203.0.113.0/24 y 192.0.2.0/24, dependiendo de si la ACL se aplica a la entrada o a la salida de una sesión BGP:

Access-list (ACL)	**Prefix-list**
access-list <nro_access-list> permit|deny ip <prefijo> <máscara de wildcard>	ip prefix-list <nombre_prefix-list> <nro_seq> permit|deny <red/prefijo>
access-list 101 deny ip 10.0.0.0 0.255.255.255 access-list 101 deny ip 203.0.113.0 0.0.0.255 access-list 101 permit ip 192.0.2.0 0.0.0.255	ip prefix-list Entrada seq 5 deny 10.0.0.0/8 le 32 ip prefix-list Entrada seq 10 deny 203.0.113.0/24 le 32 ip prefix-list Entrada seq 15 permit 0.0.0.0/0 le 32

Estas sentencias se van ejecutando en forma secuencial, tal como fue creada la ACL, con el inconveniente de que si se quiere agregar o quitar alguna de ellas, se debe borrar y volver a crear toda la ACL con la modificación. Los **prefix-list** en cambio tienen la siguiente forma:

```
ip prefix-list <nombre_prefix-list> <nro_seq> permit|deny
<red/prefijo>
```

(El comando deberá escribirse en una sola línea)

Las ventajas son varias: entre ellas, que pueden ser representados por un nombre, lo cual los vuelve más legibles e intuitivos. Además, la posibilidad de identificar a cada sentencia con un número de secuencia permite que realizar una modificación en el **prefix-list** sea más fácil pues sólo se remueve la sentencia a modificar, identificada con su número de secuencia. Pero la ventaja más significativa es que las rutas que se quieren permitir o denegar, se expresan con el formato de red y prefijo de subred, lo cual los hace más amigables en su lectura.

Un detalle que no es menor es que estas implementaciones permiten que, a través de unos modificadores llamados “ge”, los cuales significan “mayor o igual”, y “le”, que significan “menor o igual”, las longitudes de los prefijos sean variables.

Un ejemplo de **prefix-list** es el que se ve en la columna de la derecha del cuadro, donde el nombre elegido para identificar al prefix-list es “Entrada”, pues se especificará en ese **prefix-list** todo lo que no se quiere permitir que entre a nuestro AS.

En este caso se estarán denegando todos los prefijos de entrada de la red 10.0.0.0 de longitud entre 8 y 32, es decir, denegamos toda la red privada 10.0.0.0; También se filtran los prefijos de la red 203.0.113.0/24 menores o iguales a 32. Por último, se permite todo el resto de los prefijos. Esta última línea es necesaria ya que de lo contrario existe un "**deny**" implícito al final del **prefix-list**.

Se han visto las diferencias en la construcción de una ACL y de un **prefix-list**, y se ha llegado a la conclusión de que los **prefix-list** suelen ser más flexibles, razón por la cual se verá con estos últimos cómo es que se lleva a cabo el filtrado de rutas en BGP.

Para aplicar un filtro de rutas en BGP basado en prefix-list deberá configurarse de la siguiente manera:

```
neighbor <ip-address|peer-group> prefix-list
<nombre_prefix-list> in|out
```

(El comando deberá escribirse en una sola línea)

Simplemente se indicará a quién se aplica el filtro, cuál es el **prefix-list** que se aplica, y si se aplicará sobre las rutas que vienen como información de ese *neighbor*, en cuyo caso será **in**, o en lo que se le enseña al *neighbor*, que será **out**.

Importante: Notar también que se puede aplicar el filtro a un determinado *neighbor* o a un grupo de *neighbors* que fueron declarados previamente en un "**peer-group**", de la siguiente manera:

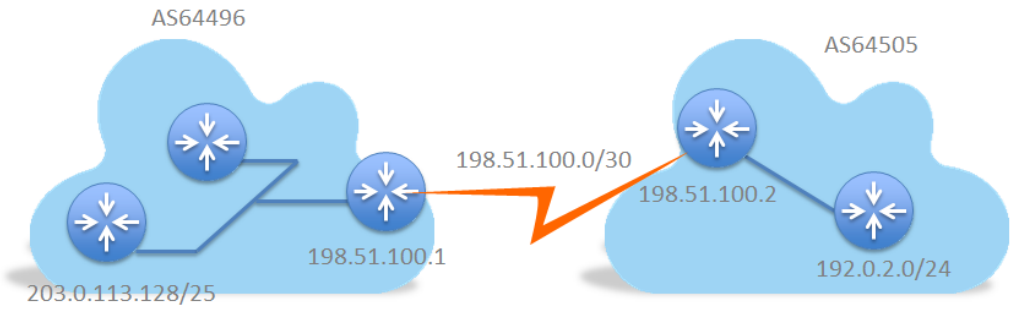
```
neighbor <nombre_peer-group> peer-group
```

Y luego:

```
neighbor <IP_neighbor> peer-group <nombre_peer-group>
```

Repitiendo esta última sentencia por cada *neighbor* que se desee hacer pertenecer al **peer-group**.

A continuación se mostrará un ejemplo de lo explicado sobre los filtros con **prefix-list**, donde teniendo en cuenta esta topología que ya se ha visto, los filtros podrían aplicarse de la siguiente manera:



EJEMPLO

```
router bgp 64496
 neighbor 198.51.100.2 remote-as 64505
 neighbor 198.51.100.2 prefix-list VECINO-IN in
 neighbor 198.51.100.2 prefix-list VECINO-OUT out
!
ip prefix-list VECINO-IN deny 192.0.2.0/24
ip prefix-list VECINO-IN permit 0.0.0.0/0 le 32
ip prefix-list VECINO-OUT permit 203.0.113.128/25
```

Lo que significa que no se querrá recibir el prefijo 192.0.2.0/24 del *neighbor* 198.51.100.2, pero sí se le anunciará la subred 203.0.113.128/25.

A continuación se mostrará cómo implementar filtros basados en **AS_PATH**.

Hasta ahora los filtros que se vieron filtraban rutas indicando los prefijos o las direcciones IP. En este nuevo caso los filtros actuarán según información del **AS_Path**, o dicho de otra forma, del camino que realizan los prefijos.

Esto es así porque muchas veces es más simple filtrar por la procedencia de las rutas, que filtrar ruta a ruta. Pero no es necesario adelantarse, se verá cómo son estos filtros.

Para implementar filtros basados en AS_path, se considerarán dos pasos bien diferenciados.

- **Dos pasos:**

  1. Crear sentencia con expresión regular.

  ip as-path access-list **<nro_filtro>** permit | deny <regexp>

  2. Aplicar el filtro

  neighbor **<IP_neighbor>** filter-list **<nro_filtro>** in | out

Por un lado (paso 1), se deberá crear una expresión regular que denote un AS_Path. Notar que hay un número de filtro que deberá mantenerse para indicar que se trata de sentencias dentro del mismo filtro. Al final del comando se deberá indicar la expresión regular que denota lo que queremos controlar con el filtro.

El paso siguiente (paso 2), es aplicar el filtro con la expresión regular, a la entrada o salida de una sesión BGP, donde se debe especificar el *neighbor* al cual se le aplica el filtro, el número de filtro que se le está aplicando (según lo que se creó previamente), y si se aplica sobre las rutas que se aprenden de un *neighbor*, o las que se anuncian a ese *neighbor*, en cuyo caso habrá que especificar con la palabra "**out**" en vez de "**in**" como en el primer caso.

Se verá con un ejemplo que ayudará a comprender mejor lo explicado:

EJEMPLO

```
neighbor 198.51.100.1 filter-list 10 in
 neighbor 198.51.100.1 filter-list 11 out
...
ip as-path access-list 10 permit 64400$
ip as-path access-list 11 deny 64496$
ip as-path access-list 11 deny ^645
ip as-path access-list 11 permit  _64497_64498_
...
```

Lo que quiere decir este ejemplo es que se recibirán del *neighbor* 198.51.100.1 todos los prefijos originados en el AS 64400, ya que la expresión 64400$ que se define en el filtro 10, implica un AS_Path con el 64400 al final.

En cuanto a lo que lo que se va a anunciar, que está especificado en el filtro 11 con 3 sentencias, se dice: que no se anunciará al *neighbor* 198.51.100.1 ninguna ruta cuyo AS_Path termine con 64496, o lo que es lo mismo, ninguna ruta originada en el AS 64496.

Además de esto, tampoco se enseñará al *neighbor* 198.51.100.1 ninguna ruta proveniente del ningún AS que comience con 645, tal como indica la segunda sentencia del filtro, pero sí se anunciarán los prefijos cuyo **AS_Path** contengan en alguna parte del camino los ASs 64497 y 64498, en ese orden.

Para poder leer con facilidad qué quiere decir cada filtro de este tipo, hay que comprender las expresiones regulares. Algunas de ellas, las más comunes son las que se muestran en la siguiente tabla. No obstante, la interpretación de las expresiones regulares requiere un entrenamiento aparte, por lo que se sugiere consultar documentación al respecto.

Caracter	Función
^	empieza con
$	termina con
.	cualquier caracter
_	cualquier delimitador (espacio, comienzo, fin, coma)
[0-9]	rango del 0 al 9
[123]	1, 2 ó 3
( )	asocia
|	ó
*	cero o más veces
?	cero o una vez
+	una o más veces
\#	llama a la expresión ubicada en la posición # del regexp

La construcción de filtros que sigue es el que se implementa a través de **route-maps**.

Los **route-maps** son sentencias que se construyen en forma similar a un lenguaje de programación. Cada sentencia se identifica con un número de secuencia, lo cual facilita sus modificaciones, pero lo más importante es que el número de secuencia implica el orden en el que cada sentencia será ejecutada, en forma ascendente.

La ejecución del **route-map** avanza hasta que una de las sentencias da **verdadero**, y se ejecuta.

Se verá además que existe una palabra clave dentro del route-map, la palabra "**match**", la cual será el disparador para permitir o denegar una acción.

Si la palabra "**match**", por su naturaleza de condicionante, no está dentro de ninguna sentencia del **route-map**, entonces todas las rutas resultan con criterio verdadero y el comando "**set**" se aplicará a todas ellas.

Lo mismo ocurre si tampoco existe en él una lista de acceso para la sentencia "**match**".

Como consideración final, si existen en el **route-map** varias sentencias "**match**", todas deberán ser verdaderas para que el mismo resulte verdadero y se pueda tomar una acción.

Al igual que ocurre con las listas de acceso, una denegación está implícitamente incluida al final del **route-map**.

Para poder comprender mejor el funcionamiento se verán algunos ejemplos:

```
router bgp 64496
 neighbor 198.51.100.2 route-map filter-on-as-path in
!
route-map filter-on-as-path permit 10
 match as-path 1
 set local-preference 80
 set weight 200
 set metric 127
 set next-hop 192.0.2.10
!
route-map filter-on-as-path permit 20
 match as-path 2
 set local-preference 200
 set weight 500
 set metric 327
 set next-hop 192.0.2.100
!
route-map filter-on-as-path permit 30
!
ip as-path access-list 1 permit _64505$
ip as-path access-list 2 permit _64510_
```

En este primer ejemplo puede verse que se aplica un filtro filter-on-as-path creado con un **route-map**, a la entrada de la sesión BGP con el *neighbor* 198.51.100.2.

La primer sentencia del filtro, definida en el **route-map** bajo el número de secuencia 10, implica que si se trata de rutas que se originaron en el AS64505, entonces que ingrese al AS, y se les configuren todos los atributos

del listado. Aquí se ve claramente un ejemplo de configuración de local_preference, weight, med y next-hop.

Si esto ocurriera, ahí terminaría la ejecución del **route-map**. Si en lugar de eso, el **match** resultara falso, se seguiría con la secuencia 20. Entonces se fijará si en alguna parte del AS_Path está el AS64510, tal como se indica en las sentencias recuadradas.

Si esta sentencia 20 resulta verdadera, se configurará todo lo indicado con el comando **set**. Si no, el **route-map** continuaría con la cláusula 30, permitiendo que pasen todos los prefijos sin modificar nada.

Notar que este ejemplo utiliza filtros basados en **as-path**, pero lo mismo podría hacer bajo filtros basados en **prefix-list** si se quiere filtrar según las rutas que se aprenden.

A continuación puede verse un ejemplo de ello:

```
router bgp 64496
 neighbor 198.51.100.2 route-map infilter in
!
route-map infilter permit 10
 match ip address prefix-list HIGH-PREF
 set local-preference 120
!
route-map infilter permit 20
 match ip address prefix-list LOW-PREF
 set local-preference 80
!
route-map infilter permit 30
!
ip prefix-list HIGH-PREF permit 192.0.2.0/25
ip prefix-list LOW-PREF permit 192.0.2.128/25
```

En este caso, el filtro intenta discriminar entre los prefijos 192.0.2.0/25 y 192.0.2.128/25 para tomar la acción.

Como se ve, el uso de **route-map** permite un mayor grado de administración de los atributos de BGP y una forma más avanzada de configuración. Los **route-maps** pueden ser más complejos, pero otorgan una

mayor flexibilidad a la hora de implementar políticas de ruteo en nuestra organización. De hecho el equivalente a **route-maps**, de otros proveedores de equipamiento, se llama "**policy**".

Los **route-maps** se pueden utilizar también para filtrar o tomar acciones en función de otros atributos que no sean sólo las direcciones IP o el AS_Path, como se puede ver en la sentencia **match**. Sin embargo, para los alcances de este curso será suficiente con lo que se ha mostrado en los ejemplos.

Hasta ahora se ha visto cómo se pueden implementar filtros en BGP, sus tipos y alcances. A partir de ahora estará en condiciones de realizar la práctica que corresponde a este módulo.