BGP Policy Control

Applying Policy with BGP

- Policy-based on AS path, community or the prefix
- Rejecting/accepting selected routes
- Set attributes to influence path selection
- Tools:
  - Prefix-list (filters prefixes)
  - Filter-list (filters ASes)
  - Route-maps and communities

Policy Control – Prefix List

- Per neighbour prefix filter
  - incremental configuration
- Inbound or Outbound
- Based upon network numbers (using familiar IPv4 address/mask format)
- Using access-lists in Cisco IOS for filtering prefixes was deprecated long ago
  - **Strongly discouraged!**

Prefix-list Command Syntax

- Syntax:

  `[no] ip prefix-list list-name [seq seq-value] permit|deny network/len [ge ge-value] [le le-value]`

  `network/len:`   The prefix and its length

  `ge ge-value:`   "greater than or equal to"

  `le le-value:`   "less than or equal to"

- Both "ge" and "le" are optional
  - Used to specify the range of the prefix length to be matched for prefixes that are more specific than network/len

- Sequence number is also optional
  - `no ip prefix-list sequence-number` to disable display of sequence numbers

Prefix Lists – Examples

- Deny default route

  `ip prefix-list EG deny 0.0.0.0/0`

- Permit the prefix 35.0.0.0/8

  `ip prefix-list EG permit 35.0.0.0/8`

- Deny the prefix 172.16.0.0/12

  `ip prefix-list EG deny 172.16.0.0/12`

- In 192/8 allow up to /24

  `ip prefix-list EG permit 192.0.0.0/8 le 24`

  – This allows all prefix sizes in the 192.0.0.0/8 address block, apart from /25, /26, /27, /28, /29, /30, /31 and /32.

Prefix Lists – Examples

- In 192/8 deny /25 and above

  `ip prefix-list EG deny 192.0.0.0/8 ge 25`

  – This denies all prefix sizes /25, /26, /27, /28, /29, /30, /31 and /32 in the address block 192.0.0.0/8.

  – It has the same effect as the previous example

- In 193/8 permit prefixes between /12 and /20

  `ip prefix-list EG permit 193.0.0.0/8 ge 12 le 20`

  – This denies all prefix sizes /8, /9, /10, /11, /21, /22, … and higher in the address block 193.0.0.0/8.

- Permit all prefixes

  `ip prefix-list EG permit 0.0.0.0/0 le 32`

  – 0.0.0.0 matches all possible addresses, "0 le 32" matches all possible prefix lengths

Policy Control – Prefix List

- Example Configuration

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 remote-as 110
  neighbor 102.10.1.1 prefix-list AS110-IN in
  neighbor 102.10.1.1 prefix-list AS110-OUT out
!
ip prefix-list AS110-IN deny 218.10.0.0/16
ip prefix-list AS110-IN permit 0.0.0.0/0 le 32
ip prefix-list AS110-OUT permit 105.7.0.0/16
ip prefix-list AS110-OUT deny 0.0.0.0/0 le 32
```

Policy Control – Filter List

- Filter routes based on AS path
  - Inbound or Outbound
- Example Configuration:

```
router bgp 100
 network 105.7.0.0 mask 255.255.0.0
 neighbor 102.10.1.1 filter-list 5 out
 neighbor 102.10.1.1 filter-list 6 in
!
ip as-path access-list 5 permit ^200$
ip as-path access-list 6 permit ^150$
```

Policy Control – Regular Expressions

- Like Unix regular expressions
    - .   Match one character
    - *   Match any number of preceding expression
    - \+   Match at least one of preceding expression
    - ^   Beginning of line
    - $ End of line
    - \   Escape a regular expression character
    - _   Beginning, end, white-space, brace
    - | Or
    - ()   brackets to contain expression
    - []   brackets to contain number ranges

Policy Control – Regular Expressions

- Simple Examples

    .*   match anything

    .+   match at least one character

    ^$ match routes local to this AS

    _1800$ originated by AS1800

    ^1800_   received from AS1800

    _1800_   via AS1800

    _790_1800_ via AS1800 and AS790

    _(1800_)+   multiple AS1800 in sequence   (used to match
        AS-PATH prepends)

    _\(65530\)_ via AS65530 (confederations)

Policy Control – Regular Expressions

- Not so simple Examples

  ^[0-9]+$ Match AS_PATH length of one

  ^[0-9]+_[0-9]+$ Match AS_PATH length of two

  ^[0-9]*_[0-9]+$ Match AS_PATH length of one or two

  ^[0-9]*_[0-9]*$ Match AS_PATH length of one or two
      (will also match zero)

  ^[0-9]+_[0-9]+_[0-9]+$ Match AS_PATH length of three

  _(701|1800)_   Match anything which has gone

                          through AS701 or AS1800

  _1849(_.+_)12163$ Match anything of origin AS12163

                          and passed through AS1849

Policy Control – Route Maps

- A route-map is like a "programme" for IOS
- Has "line" numbers, like programmes
- Each line is a separate condition/action
- Concept is basically:

  if *match* then do *expression* and exit

  else

  if *match* then do *expression* and exit

  else etc

- Route-map "continue" lets ISPs apply multiple conditions and actions in one route-map

Route Maps – Caveats

- Lines can have multiple set statements
- Lines can have multiple match statements
- Line with only a match statement
  - Only prefixes matching go through, the rest are dropped
- Line with only a set statement
  - All prefixes are matched and set
  - Any following lines are ignored
- Line with a match/set statement and no following lines
  - Only prefixes matching are set, the rest are dropped

Route Maps – Caveats

- Example
  - Omitting the third line below means that prefixes not matching list-one or list-two are dropped

```
route-map sample permit 10
 match ip address prefix-list list-one
 set local-preference 120
!
route-map sample permit 20
 match ip address prefix-list list-two
 set local-preference 80
!
route-map sample permit 30 ! Don't forget this
```

Route Maps – Matching prefixes

- Example Configuration

```
router bgp 100
 neighbor 1.1.1.1 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
!
ip prefix-list HIGH-PREF permit 10.0.0.0/8
ip prefix-list LOW-PREF permit 20.0.0.0/8
```

Route Maps – AS-PATH filtering

- Example Configuration

```
router bgp 100
 neighbor 102.10.1.2 remote-as 200
 neighbor 102.10.1.2 route-map filter-on-as-path in
!
route-map filter-on-as-path permit 10
 match as-path 1
 set local-preference 80
!
route-map filter-on-as-path permit 20
 match as-path 2
 set local-preference 200
!
ip as-path access-list 1 permit _150$
ip as-path access-list 2 permit _210_
```

Route Maps – AS-PATH prepends

- Example configuration of AS-PATH prepend

```
router bgp 300
  network 105.7.0.0 mask 255.255.0.0
  neighbor 2.2.2.2 remote-as 100
  neighbor 2.2.2.2 route-map SETPATH out
!
route-map SETPATH permit 10
  set as-path prepend 300 300
```

- Use your own AS number when prepending
  - Otherwise BGP loop detection may cause disconnects

Route Maps – Matching Communities

- Example Configuration

```
router bgp 100
 neighbor 102.10.1.2 remote-as 200
 neighbor 102.10.1.2 route-map filter-on-community in
!
route-map filter-on-community permit 10
 match community 1
 set local-preference 50
!
route-map filter-on-community permit 20
 match community 2 exact-match
 set local-preference 200
!
ip community-list 1 permit 150:3 200:5
ip community-list 2 permit 88:6
```

Community-List Processing

- Note:
  - When multiple values are configured in the same community list statement, a logical AND condition is created. All community values must match to satisfy an AND condition
    ```
    ip community-list 1 permit 150:3 200:5
    ```

  - When multiple values are configured in separate community list statements, a logical OR condition is created. The first list that matches a condition is processed
    ```
    ip community-list 1 permit 150:3
    ```

Route Maps – Setting Communities

- Example Configuration

```
router bgp 100
  network 105.7.0.0 mask 255.255.0.0
  neighbor 102.10.1.1 remote-as 200
  neighbor 102.10.1.1 send-community
  neighbor 102.10.1.1 route-map set-community out
!
route-map set-community permit 10
 match ip address prefix-list NO-ANNOUNCE
 set community no-export
!
route-map set-community permit 20
 match ip address prefix-list AGGREGATE
!
ip prefix-list NO-ANNOUNCE permit 105.7.0.0/16 ge 17
ip prefix-list AGGREGATE permit 105.7.0.0/16
```

Route Map Continue

- Handling multiple conditions and actions in one route-map (for BGP neighbour relationships only)

```
route-map peer-filter permit 10
 match ip address prefix-list group-one
 continue 30
 set metric 2000
!
route-map peer-filter permit 20
 match ip address prefix-list group-two
 set community no-export
!
route-map peer-filter permit 30
 match ip address prefix-list group-three
 set as-path prepend 100 100
!
```

Managing Policy Changes

- New policies only apply to the updates going through the router AFTER the policy has been introduced or changed
- To facilitate policy changes on the entire BGP table the router handles the BGP peerings need to be "refreshed"
  - This is done by clearing the BGP session either in or out, for example:

  `clear ip bgp <neighbour-addr> in|out`

- Do NOT forget in or out — doing so results in a hard reset of the BGP session

Managing Policy Changes

- Ability to clear the BGP sessions of groups of neighbours configured according to several criteria

- **`clear ip bgp <addr> [in|out]`**

  `<addr>` may be any of the following

  `x.x.x.x`   IP address of a peer

  `*`   all peers

  ASN   all peers in an AS

  `external`   all external peers

  `peer-group <name>`   all peers in a peer-group

Acknowledgement and Attribution

This presentation contains content and information originally developed and maintained by the following organisation(s)/individual(s) and provided for the African Union AXIS Project

Cisco ISP/IXP Workshops

Philip Smith: - pfsinoz@gmail.com

BGP Policy Control

End