Bgpq - автоматизация построения prefix-list

Опубликовано muff в Ср, 2013-05-01 01:00

Являясь сетевым администратором, повседневно работаю с **BGP**. Стоит отметить, что BGP-соединения всегда можно разделить на соединения с апстримами, с клиентами и пиринговые (т.е. паритетные). При настройке паритетных соединений и соединений с клиентами, желательно накладывать **prefix-list** на принимаемые анонсы. Для автоматизации постройки **prefix-list** согласно **as-set**, уже довольно давно пользуюсь утилитой **bgpq**. Утилита автоматически строит префикс-листы, беря за исходные данные номер автономной системы или **as-set**. Утилита становится незаменимой довольно быстро.

Для начала, выполним установку утилиты из системы портов:

```
# cd /usr/ports/net-mgmt/bgpq && make install clean && rehash
```

По завершению установки необходимо ознакомиться с ее возможностями. Для подробной информации стоит обратиться к страницам man-руководства, однако если вызвать утилиту без ключей запуска, то в ответ будет выдан короткий хелп:

```
# bgpq
  Usage: bgpq [-l name] [-h host] [-p port] [-d] [-Pijosge]|[-f number]|[-F format]|[-G number] [-aq]
[-S src] [-H] [-c] [-R masklen] [-bv] objects [EXCEPT objects]
        bgpq -r filename
        -A - try to aggregate routes
        -a - print all routes uncommented
        -b - show unresolved ASes
        -c - include commandline in output
        -d - debugging
        -e - generate standard access-list
        -F - generate output in given format
        -f - generate as-path access-list (number used as starting)
        -G - generate outgoing as-path access-list (number used as starting)
        -g - generate GateD network filter
        -H - do not print headers
        -h - host running irrd (can be set with IRRD_HOST also)
        -i - generate input packet filter
        -I - invert logic of access lists
        -j - generate Juniper filter/as-path groups
        -J - generate Juniper 'load merge' filter/as-path groups
        -l - name of generated access/prefixlist
        -L - generate last entry as logging one
        -o - generate output packet filter
        -P - generate prefixlist (access-list extended by default)
        -p - irrd port (43 by default)
        -q - print only aggregated routes
        -Q - be quiet about non-aggregated specifics
        -R - allow more specific routes up to masklen
        -r - recreate file
        -s - print statistics about
        -S - use only specified source (can be set with IRRD_SOURCE)
             defaults now are - RADB,RIPE,APNIC
        -v - be more verbose on operations

"objects" mean networks (in prefix aka a.a.a.a/b format,
autonomous systems (in ASnnn format), as-macros (AS-xxxx format)
```

Для ознакомления с возможностями bgpq, в качестве обьекта возьмем **as-set** последней найстраиваемой мной сети - **AS-KLNK** (kyivlink.com [1]). Дополнительно вывод утилиты будем перенаправлять в утилиту **grep**, чтобы отсеять вставку информации, о том, что вывод сгенерирован именно утилитой **bgpq**.

Передадим в **bgpq** только обьект, без указания ключей:

```
# bgpq AS-KLNK | grep -v generated


no ip access-list extended UNKNOWNip access-list extended UNKNOWN permit ip host 91.
211.120.0 host 255.255.252.0 permit ip host 194.28.100.0 host 255.255.252.0! overlay
ed by 194.28.100.0/22 in the same as50956! -  permit ip host 194.28.100.0 host 255.2
55.254.0! overlayed by 194.28.100.0/22 in the same as50956! -  permit ip host 194.28
.102.0 host 255.255.254.0! overlayed by 194.28.100.0/22 in the same as50956! -  perm
it ip host 194.28.102.0 host 255.255.255.0! overlayed by 194.28.100.0/22 in the same
 as50956! -  permit ip host 194.28.103.0 host 255.255.255.0 permit ip host 195.12.56
.0 host 255.255.252.0 deny ip any any
```

В результате имеем сгенерированный **access-list** с названием UNKNOWN, согласно которого разрешены сети 91.211.120.0/22, 194.28.100.0/22 и 195.12.56.0/22. Также, в виде комментариев, подана информация о том, что сеть 194.28.100.0/22 разбита на подсети (самостоятельно создавал эти route objects в RIPE database, поэтому и взял именно этот **as-set** для примера) , однако по факту в **access-list** пошла вся сеть 194.28.100.0/22 целиком, не разрешая разбиение на подсети.

Попробуем исправить ситуацию и сделать вывод утилиты более "дружественным", а вместе с тем и ознакомимся с некоторыми возможностями bgpq .

Для начала укажем, что нам необходимо генерировать **prefix-list**. Для этого необходимо воспользоваться ключем **-P**:

```
# bgpq -P AS-KLNK | grep -v generated


no ip prefix-list UNKNOWNip prefix-list UNKNOWN permit 91.211.120.0/22ip prefix-list
 UNKNOWN permit 194.28.100.0/22! overlayed by 194.28.100.0/22 in the same as50956! -
 ip prefix-list UNKNOWN permit 194.28.100.0/23! overlayed by 194.28.100.0/22 in the
same as50956! - ip prefix-list UNKNOWN permit 194.28.102.0/23! overlayed by 194.28.1
00.0/22 in the same as50956! - ip prefix-list UNKNOWN permit 194.28.102.0/24! overla
yed by 194.28.100.0/22 in the same as50956! - ip prefix-list UNKNOWN permit 194.28.1
03.0/24ip prefix-list UNKNOWN permit 195.12.56.0/22
```

Отлично... Теперь необходимо присвоить сгенерированному префикс-листу определенное имя. Для этого воспользуемся ключем **-l** и в качестве аргумента передадим необходимое имя:

```
# bgpq -P -l AS-KLNK-IN AS-KLNK | grep -v generated


no ip prefix-list AS-KLNK-INip prefix-list AS-KLNK-IN permit 91.211.120.0/22ip prefi
x-list AS-KLNK-IN permit 194.28.100.0/22! overlayed by 194.28.100.0/22 in the same a
s50956! - ip prefix-list AS-KLNK-IN permit 194.28.100.0/23! overlayed by 194.28.100.
0/22 in the same as50956! - ip prefix-list AS-KLNK-IN permit 194.28.102.0/23! overla
yed by 194.28.100.0/22 in the same as50956! - ip prefix-list AS-KLNK-IN permit 194.2
8.102.0/24! overlayed by 194.28.100.0/22 in the same as50956! - ip prefix-list AS-KL
NK-IN permit 194.28.103.0/24ip prefix-list AS-KLNK-IN permit 195.12.56.0/22
```

Теперь попробуем "избавиться" от лишних комментариев - при дальнейшем использовании утилиты в скриптах, толку от них немного, поэтому необходимо предвидеть такие ситуации. Для начала воспользуемся ключем **-q**, получив на выходе только агрегированные маршруты:

```
# bgpq -P -l AS-KLNK-IN -q AS-KLNK | grep -v generated


no ip prefix-list AS-KLNK-INip prefix-list AS-KLNK-IN permit 91.211.120.0/22ip prefi
x-list AS-KLNK-IN permit 194.28.100.0/22ip prefix-list AS-KLNK-IN permit 195.12.56.0
/22
```

Однако довольно часто сети "разбиты" на подсети, в таком случае агрегированные маршруты нас не устраивают. Поэтому желательно пользоваться ключем **-Q**, при использовании которого будут перечислены и все специфические маршруты:

```
# bgpq -P -l AS-KLNK-IN -Q AS-KLNK | grep -v generated


no ip prefix-list AS-KLNK-INip prefix-list AS-KLNK-IN permit 91.211.120.0/22ip prefi
x-list AS-KLNK-IN permit 194.28.100.0/22ip prefix-list AS-KLNK-IN permit 194.28.100.
0/23ip prefix-list AS-KLNK-IN permit 194.28.102.0/23ip prefix-list AS-KLNK-IN permit
 194.28.102.0/24ip prefix-list AS-KLNK-IN permit 194.28.103.0/24ip prefix-list AS-KL
NK-IN permit 195.12.56.0/22
```

Также можно пользоваться ключем **-A**, чтобы наложить на агрегированный маршрут минимально и максимально допустимые маски:

```
# bgpq -P -l AS-KLNK-IN -A AS-KLNK | grep -v generated


no ip prefix-list AS-KLNK-INip prefix-list AS-KLNK-IN permit 91.211.120.0/22ip prefi
x-list AS-KLNK-IN permit 194.28.100.0/22 le 23ip prefix-list AS-KLNK-IN permit 194.2
8.102.0/23 ge 24 le 24ip prefix-list AS-KLNK-IN permit 195.12.56.0/22
```

При первом знакомстве можно также использовать ключи **-v** и **-d**, для расширенного вывода работы утилиты.

А после знакомства с утилитой можно использовать ее в скриптах для автоматического формирования access и prefix-list'ов, для дальнейшей передачи их на маршрутизаторы, либо же для сравнения и уведомления на почту о изменениях...

Источник (получено *2026-11-10 23:02*):
http://muff.kiev.ua/content/bgpq-avtomatizatsiya-postroeniya-prefix-list

Ссылки:
[1] http://kyivlink.com