Gert Döring
SpaceNet
BGP Security Overview
BGP Security

* who?
* why?
* how?
* where?

Gert Döring
address and routing geek

SpaceNet AG
medium-sized ISP in Munich
doing proper BGP since 1995

we’re hiring!
Routing ***stability*** needs robust & reliable transport

- Ensure sessions between routers are not disturbed
- Ensure payload of TCP session is not modified

- Countermeasures are well-known
  - MD5 checksum on session = packet integrity
  - Anti-Spoofing (BCP38) = no TCP injection possible
  - proper ingress ACLs and rate limits = protect router CPU
  - see also RFC7454
More interesting problems involve announcing unauthorized prefixes or prefix sets

- honest mistakes / „fat fingers“
  - typo in router config („195.30.4.0/24“ instead of „195.39.4.0/24“)
  - forgetting a filter and announcing full table to decix peers
- fully intentionally and maliciously announcing address space that is not authorized for this AS
  - to send SPAM from „fresh“ addresses, complaints go elsewhere
  - to steal DNS traffic to redirect banking customers to own servers
  - null-route someone else‘s network for denial-of-service
Mitigation strategy 1: prefix and as-path filtering

towards your BGP customers:

- filter on positive prefix list („only this list of prefixes“)
- filter on AS-path according to customer cone
- filter source: internet routing registry (RIPE, NTTcom, ...)

pros and cons?

- stops all unauthorized announcements from customers
- but does not scale to 10.000+ customer prefixes
- not all regions have reliable and secure routing registry
Mitigation strategy 1: prefix and as-path filtering

- towards your BGP peers and transit providers:
  - filter out „bogons“ (negative list):
  - reject subnets from your own space
  - reject subnets for IXP peering LANs you’re connected to
  - reject tier1 ASes in paths received from peers/customers

- pros and cons
  - will stop attacks or mishaps involving your own addresses („steal customer traffic towards your recursive DNS servers“)
  - will stop problems when someone leaks a „DECIX more specific“
  - will not stop most other sorts of unauthorized announcement
prefix filtering: customer example

unix$ bgpq3 -l from-as-5539 -6 AS5539                # accepts AS number or AS-Macro
no ipv6 prefix-list from-as-5539
ipv6 prefix-list from-as-5539 permit 2001:608::/32
ipv6 prefix-list from-as-5539 permit 2001:67c:158c::/48
ipv6 prefix-list from-as-5539 permit 2001:4150::/32

... copy-paste to IOS router, then bind to customer BGP session:

# conf term
(config)# ipv6 prefix-list from-as-5539 ...
(config)# router BGP 65001
(config-router)# address-family ipv6 unicast
(config-router-af)# neighbour 2001:db8::2 prefix-list from-as-5539 in
(config-router-af)# ^Z

... done. If prefixes change frequently, this can be automated fairly easily.
Biggest drawbacks of „prefix filtering from IRR data“
- lack of good quality IRR data in some regions
- copying filters to router NVRAM does not scale to 10,000+ lines of config for larger peers or customers

RPKI to the rescue
- software running outside router calculates list of (prefix,origin-as) pairs
- sent to router via TCP or SSH session (rpki rtr protocol)
- incoming prefixes validated against list, drop if INVALID
- data source can be RPKI certificate hierarchy or IRR data
* regional registry (RIPE NCC) certifies holdership of an IP address block
* address holder certifies digital „Route Origin Authorisation“ (ROA)
* collector software gets certs and ROAs from „everywhere“
* builds list of (prefix,origin) pairs
* router compares prefixes to list and uses result in decision making
Resource usage (RAM, CPU) on routers is fairly low.

Can be used to filter peers and upstream providers:
- Even if a default route exists, it will stop "bad" prefixes from peers.
- Will stop many "fat finger" mistakes (but not "leaks").
- Will not stop intentional malicious announcements.
  - "If this is only authorized by AS X, then I just pretend to have AS X as one of my customers" = origin validation succeeds.
  - But prefix-length in ROA will prevent more-specific prefix hijacks.

No clear concept on "which AS may give transit to what AS" yet.

Prefix filter-lists based on IRR data and AS-sets ("customer cone") still recommended on "smallish" customer links.
unix$ routinator -b /var/rpki-cache rtrd -l 0.0.0.0:3323

XR# show run router bgp
router bgp 65039
rpki server 10.30.20.26
transport tcp port 3323
...

XR# sh bgp ipv4 u origin-as val inv
Network Path
*> 2.3.4.0/24  5539 2914 3356 3549 i
*> 2.59.118.0/24 5539 2914 9121 9121 43260 60721 i
*> 2.183.132.0/22 5539 1273 174 200612 49666 48159 58224 i
...
Processed 2814 prefixes, 2814 paths

XR#5#sh bgp ipv4 u 93.175.147.0/24
5539 2914 12859 12654
193.149.y.y from 193.149.y.y (193.149.y.z)
Origin IGP, valid, external
Origin-AS validity: invalid

XR# sh route ipv4 93.175.147.1
Routing entry for 0.0.0.0/0
Known via "static", distance 1, metric 0, candidate def...

XR(config)# router bgp 65039
XR(config-bgp)# add ipv4 u
XR(config-bgp-af)# bgp bestpath origin-as use validity
XR(config-bgp-af)# commit

XR# sh bgp ipv4 u 93.175.147.0/24
Paths: (1 available, no best path)
5539 2914 12859 12654
193.149.y.y from 193.149.y.y (193.149.y.z)
Origin IGP, valid, external
Origin-AS validity: invalid

XR# sh route ipv4 93.175.147.1
Routing entry for 0.0.0.0/0
Known via "static", distance 1, metric 0, candidate def...

XR(config)# route-policy ROA
XR(config-rpl)# if validation-state is ?
  invalid  Invalid State
  not-found  State not found
  valid  Valid State

Example: IOS XR

run routinator on unix box of choice
tell IOS XR box to talk to routinator

only informative(!), paths are still being used

tell IOS XR to look at origin-as validity

INVALID paths are no longer accepted

no VALID path exists (here), fall back to default route

more fine-grained filtering is possible via route-policy
number of prefixes announced between ASes differ by orders of magnitude – solution needs to scale

on customer links, use prefix-lists and possibly as-path filters, based on AS-Set objects in IRR data

on very large customers, peers and upstream links, use RPKI based filters (plus prefix-list filters for bogons and „max-prefix“ against full-table leaks)

if you only receive a default route, no filters needed
* http://bgpfilterguide.nlno.net/ (BGP filtering guide)
* https://github.com/denog/routing-bcp (community shared)
* http://rpki.readthedocs.io (RPKI docs+examples)
* https://blog.cloudflare.com/rpki-details/ (nice writeup)
* https://labs.ripe.net/Members/markd/routing-certification-beacons/
* https://www.slideshare.net/apnic/improving-the-peering-business-case-with-rpki
* https://www.de-cix.net/en/about-de-cix/academy/videos-and-webinars (BGP 07 on BGP security)

* gert@space.net – questions welcome