GTSM

GTSM

Section 5.2 of RFC 7454 discusses BGP TTL filtering, otherwise known as the Generalized TTL Security Mechanism, GTSM.

The concept uses the TTL value in TCP packets to provide a quite strong mechanism to prevent spoofed packets from being accepted. When a packet passes through the router the router decrements the TTL value by one. If your directly connected BGP neighbor explicitly sets the TTL on their outgoing BGP packets to 255 then you can check all packets from that neighbor to ensure they equal 255. If the TTL value is less than 255 then you know the packet is spoofed; it has passed through at least one router, which has decremented the TTL value.

Since the TTL field is 8-bits, and 255 is therefore the maximum, sessions using GTSM set their values to 255 and check to ensure that their adjacent neighbors BGP session packets have a value of 255, throwing out all others with a lower value from that neighbor. This is a very powerful mechanism that uses the default behaviour of the routers interacting with the TTL field to ensure spoofed packets can't reach your session from more than one hop away … at least for those neighbors that have it configured.

There are some things to watch out for …

First, GTSM needs to be configured on both sides of the BGP session, on a per BGP session basis. You need to set your TTL to 255 and you need to check to ensure that your neighbor s BGP packets have 255 TTL when you receive them. And your neighbor needs to ensure the same thing. At a minimum to you need to coordinate the setting of the TTL to 255 with them.

Second, a TTL of 255 is not always correct. If your neighbor is not directly connected/you're doing multi-hop BGP, then the TTL value will be something other than 255. You and your neighbor will need to coordinate on the appropriate value to check for. (Although, you WILL be setting your TTL to 255 regardless of how many hops away they are.) The drawback, of course, is that you are now more vulnerable to spoofing from that connection. As the RF notes "anyone inside the TTL diameter could spoof the TTL."

Juniper Example

Junipers implementation is a bit cumbersome, relying on setting up a filter with the "ttl-except" keyword and applying that filter on the appropriate interfaces.

```
filter ttl-security {
 term gtsm {
  from {
   source-address {
    10.1.2.3/32; }
   protocol tcp;
   ttl-except 255;
   port 179; }
  then {
   discard; }}
 term else {
  then {
   Accept;
```

There's more information on the ttl mechanism on the Juniper site at:

https://www.juniper.net/documentation/en_US/junos13.3/topics/reference/configuration-statement/ttl-edit-protocols-bgp-mulithop.html

Cisco Example

Cisco BGP configurations support the "ttl" option, meaning it's quite easy to configure GTSM. The following example uses groups, but you can also use the neighbor statement "ttl-security hops 1"

```
bgp {
 group toAS2 {
  type external;
  peer-as 2;
  ttl 255;
  neighbor 10.1.2.3;
```

There a decent set of document about this feature on Cisco's site at:

https://supportforums.cisco.com/document/86776/securing-ebgp-sessions-ttl-security-feature

Brocade Example

Brocade supports GTM with the “ebgp-btsh” bgp neighbor command. This works with a TTL for 255/254. If the BGP neighbor is configured for multihop then that value will be used to calculate the specific TTL.

```
neighbor 10.10.10.1 remote-as 2
neighbor 10.10.10.1 ebgp-btsh
```

There’s Brocade documentation to explain the command further at:

http://www.brocade.com/content/html/en/command-reference-guide/FI_08030_CMDREF/GUID-20AF9E52-5746-4D0D-8871-663F15EFE4F7.html