IPv6 Security Lab - Routing Protocol Security

IS-IS neighbour authentication

Configuring Neighbour Security for IS-IS

Network operators consider it more and more important to turn on neighbour authentication inside their networks as attacks on infrastructure increase and operators seek to use all available tools to secure their networks.

IS-IS supports neighbour authentication. This is quite important inside discrete networks to prevent the introduction of improperly configured or unintended equipment.

Each router team will turn on authentication for IS-IS. This first step will create the key-chain to be used for the authentication.

An example configuration might be:

```
key chain as10X-key
 key 1
  key-string cisco
```

We will just use *cisco* as the key-string - clearly we do not do this on a public operational network, but instead choose something more obscure.

Note: the *key-chain* allows up to 255 keys to be set, potential a different one per interface. It is generally not recommended to set more than one per interface, as the router will try and communicate with its neighbours using all keys. If a key needs to be upgraded, common practice then is to set a second key, allowing a graceful changeover without compromising the functioning of the network. Once all the routers on the network are using the new key, the old one should be removed.

Neighbour Authentication - Part 2

Now that the key chain has been set up, the second step is to actually enable neighbour authentication for IS-IS.

An example configuration for the Core Router might be:

```
router isis as10X
 authentication mode md5 level-2
 authentication key-chain as10X-key level-2
```

Notice now that the IS-IS adjacencies do not come up unless the neighbouring router has also entered the same configuration and key. Notice also how the IS-IS adjacencies were reset soon after the configuration was entered.

Also notice that the neighbour authentication for IS-IS is independent of which protocol/topology is

being used.

Check IS-IS operation

Use the various “*show isis*” commands to see the IS-IS status of the lab network now. Check the routing and the routing table. If you are missing any adjacencies, work with your neighbouring routers in your AS to work out why, and what might have gone wrong with the neighbour authentication.

Aside: OSPF neighbour authentication

Neighbour authentication for OSPF is a little more complex than it is for IS-IS. First off, OSPFv2 only supports IPv4 adjacencies and prefixes, while OSPFv3 is used for IPv6.

We are not using OSPF in this lab, so the following is provided for your information only. Please do not configure the examples given below!

OSPFv2

If we were using OSPF for this lab, the configuration would look something like this on the core router:

```
router ospf 10X
 area 0 authentication message-digest
!
interface fastethernet 0/0
 ip ospf message-digest-key 1 md5 cisco
!
interface gigabit 1/0
 ip ospf message-digest-key 1 md5 cisco
!
interface gigabit 2/0
 ip ospf message-digest-key 1 md5 cisco
```

Notice that this sets up the area to use neighbour authentication, and then applies the authentication key to each interface in turn.

OSPFv3

Neighbour authentication for OSPFv3 is no longer built in as it is for OSPFv2, but relies on the IPSEC authentication header support built into IPv6.

The configuration for the core router would look something like this:

```
ipv6 router ospf 10X
 area 0 authentication ipsec spi 256 md5 0123456789ABCDEF0123456789ABCDEF
!
```

Note also that we don't need to apply any authentication per interface, as the key has been applied to the entire area. Each interface in area 0 needs to have the correct key to set up the adjacency.

BGP neighbour authentication

Configure passwords on the iBGP sessions

Passwords should now be configured on the iBGP sessions. Go back to the peer-groups which were configured earlier in this workshop and now add passwords to them. For example, here is the peer-group on the Core router:

```
router bgp 101
 address-family ipv4
  neighbor ibgp-partial password cisco
  neighbor ibgp-full password cisco
!
 address-family ipv6
  neighbor ibgp-v6partial password cisco
  neighbor ibgp-v6full password cisco
!
```

Do the same for the peer-groups on the border, access and peering routers.

Once the passwords have been added to the IPv4 and IPv6 peer-groups, reset the BGP sessions. Cisco IOS does not automatically reset peerings once passwords have been added to the configuration.

Watch the router logs – with the BGP session neighbour changes being logged, any mismatch in the password should be easy to spot. A missing password on one side of the BGP session will result in the neighbouring router producing these errors:

```
%TCP-6-BADAUTH: No MD5 digest from 3.3.3.3:179 to 2.2.2.2:11272
%TCP-6-BADAUTH: No MD5 digest from 3.3.3.3:179 to 2.2.2.2:11272
%TCP-6-BADAUTH: No MD5 digest from 3.3.3.3:179 to 2.2.2.2:11272
```

whereas a mismatch in the configured passwords will result in these messages:

```
%TCP-6-BADAUTH: Invalid MD5 digest from 3.3.3.3:11024 to 2.2.2.2:179
%TCP-6-BADAUTH: Invalid MD5 digest from 3.3.3.3:11024 to 2.2.2.2:179
%TCP-6-BADAUTH: Invalid MD5 digest from 3.3.3.3:11024 to 2.2.2.2:179
```

Configure password on the eBGP session

Passwords should now be configured on the eBGP sessions between your and your upstream. Just use “cisco” as the password on the eBGP session. Here is an example for AS101.

```
router bgp 101
 address-family ipv4
```

```
  neighbor 100.121.1.1 password cisco
!
 address-family ipv6
  neighbor 2001:18:0:10:: password cisco
!
```

Bogon Filtering

Configuring Inbound IPv4 BGP Prefix Filtering

Prefix lists allow a network administrator to permit or deny specific prefixes that are sent or received via BGP. Prefix lists should be used where possible to ensure network traffic is sent over the intended paths. Prefix lists should be applied to each eBGP peer in both the inbound and outbound directions.

Configured prefix lists limit the prefixes that are sent or received to those specifically permitted by the routing policy of a network. If this is not feasible due to the large number of prefixes received, a prefix list should be configured to specifically block known bad prefixes. These known bad prefixes include unallocated IP address space and networks that are reserved for internal or testing purposes by RFC 6890/BCP153. Outbound prefix lists should be configured to specifically permit only the prefixes that an organization intends to advertise.

This configuration example uses prefix lists to ensure that no bogon routes are learned or advertised. Create the IPv4 prefix filter named ‘bogon-filter’ (we will do the same for IPv6 shortly):

```
ip prefix-list bogon-filter description == IPv4 Bogons ==
! Allow our workshop prefixes
ip prefix-list bogon-filter permit 100.68.10.0/24
ip prefix-list bogon-filter permit 100.68.20.0/24
ip prefix-list bogon-filter permit 100.68.30.0/24
ip prefix-list bogon-filter permit 100.68.40.0/24
ip prefix-list bogon-filter permit 100.68.50.0/24
ip prefix-list bogon-filter permit 100.68.60.0/24
ip prefix-list bogon-filter permit 100.121.0.0/16
ip prefix-list bogon-filter permit 100.122.0.0/16
! All default route so we can propagate in IGP
ip prefix-list bogon-filter permit 0.0.0.0/0
! Drop all the Bogons
ip prefix-list bogon-filter deny 0.0.0.0/8 le 32
ip prefix-list bogon-filter deny 10.0.0.0/8 le 32
ip prefix-list bogon-filter deny 100.64.0.0/10 le 32
ip prefix-list bogon-filter deny 127.0.0.0/8 le 32
ip prefix-list bogon-filter deny 169.254.0.0/16 le 32
ip prefix-list bogon-filter deny 172.16.0.0/12 le 32
ip prefix-list bogon-filter deny 192.0.0.0/24 le 32
ip prefix-list bogon-filter deny 192.0.2.0/24 le 32
ip prefix-list bogon-filter deny 192.168.0.0/16 le 32
ip prefix-list bogon-filter deny 198.18.0.0/15 le 32
ip prefix-list bogon-filter deny 198.51.100.0/24 le 32
ip prefix-list bogon-filter deny 203.0.113.0/24 le 32
```

```
ip prefix-list bogon-filter deny 224.0.0.0/3 le 32
ip prefix-list bogon-filter deny 0.0.0.0/0 ge 25
! Allow everything else
ip prefix-list bogon-filter permit 0.0.0.0/0 le 32
```

Note the last line – it has a permit statement, allowing the remaining addresses in the BGP session. Cisco IOS has a default deny for its prefix-list filter. Also remember that we need to allow the address space used in our workshop here too – those are the first 4 permit lines of the prefix-list.

Apply the prefix-filter to the IPv4 eBGP sessions

We now apply this prefix filter incoming to our external BGP sessions. For example for AS101:

```
router bgp 101
 address-family ipv4
  neighbor 100.121.1.1 prefix-list bogon-filter in
!
```

Once you have entered the above configuration, refresh the BGP session by entering the following command (example again for AS101). This refresh command applies the newly added BGP configuration to the BGP session.

```
clear ip bgp 121 in
```

Configuring Outbound IPv4 BGP Prefix Filtering

Note that it is also important to create an outbound prefix lists to announce only the aggregate allocated prefix outbound to the upstream Network Operator. The configuration would look something like the following:

```
ip prefix-list upstream permit 100.68.10.0/24
!
router bgp 101
 address-family ipv4
  neighbor 100.121.1.1 prefix-list upstream out
!
```

Again remember to refresh the BGP session outbound.

```
clear ip bgp 121 out
```

Configuring Inbound IPv6 BGP Prefix Filtering

Each Group should now repeat the previous steps above using IPv6 instead. First of all, we will create the IPv6 bogon prefix-list:

```
ipv6 prefix-list v6bogon-filter description == IPv6 Bogons ==
! Allow our workshop prefixes
ipv6 prefix-list v6bogon-filter permit 2001:db8:10::/48
ipv6 prefix-list v6bogon-filter permit 2001:db8:20::/48
ipv6 prefix-list v6bogon-filter permit 2001:db8:30::/48
ipv6 prefix-list v6bogon-filter permit 2001:db8:40::/48
ipv6 prefix-list v6bogon-filter permit 2001:db8:50::/48
ipv6 prefix-list v6bogon-filter permit 2001:db8:60::/48
ipv6 prefix-list v6bogon-filter permit 2001:18::/32
ipv6 prefix-list v6bogon-filter permit 2001:19::/32
! Allow default route so we can propagate in IGP
ipv6 prefix-list v6bogon-filter permit ::/0
! Drop all the Bogons
ipv6 prefix-list v6bogon-filter permit 64:ff9b::/96
ipv6 prefix-list v6bogon-filter permit 2001::/32
ipv6 prefix-list v6bogon-filter deny 2001::/23 le 128
ipv6 prefix-list v6bogon-filter deny 2001:2::/48 le 128
ipv6 prefix-list v6bogon-filter deny 2001:10::/28 le 128
ipv6 prefix-list v6bogon-filter deny 2001:db8::/32 le 128
ipv6 prefix-list v6bogon-filter deny 2002::/16 le 128
ipv6 prefix-list v6bogon-filter deny 3ffe::/16 le 128
! Allow rest of Global Unicast space
ipv6 prefix-list v6bogon-filter permit 2000::/3 le 48
ipv6 prefix-list v6bogon-filter deny ::/0 le 128
```

Note that the logic here is reversed from the IPv4 filter – basically the only routable IPv6 address space is the 2000::/3 Global Unicast address block, so that is what is permitted through our filters. The exceptions to this last permit line are listed in the previous entries of the prefix filter.

Apply the prefix-filter to the IPv6 eBGP sessions

We now apply the prefix filter to our IPv6 eBGP session (or sessions) with our neighbours, in the same style as we did for IPv4.

```
router bgp 101
 address-family ipv6
  neighbor 2001:18:0:10:: prefix-list v6bogon-filter in
!
```

Once you have entered the above configuration, refresh the BGP session by entering the following command (example again for AS101). This refresh command applies the newly added BGP configuration to the BGP session.

```
clear bgp ipv6 unicast 121 in
```

Configuring Outbound IPv6 BGP Prefix Filtering

Note that it is also important to create an outbound prefix lists to announce only the aggregate

allocated prefix outbound to the upstream Network Operator. The configuration would look something like the following:

```
ipv6 prefix-list upstreamv6 permit 2001:db8:10::/48
!
router bgp 101
 address-family ipv6
  neighbor 2001:18:0:10:: prefix-list upstreamv6 out
!
```

Again remember to refresh the BGP session outbound.

```
clear bgp ipv6 unicast 121 out
```

Summary

We have now applied inbound bogon filters for both IPv4 and IPv6 on the eBGP session on our border routers. And we have configured outbound filters on our eBGP sessions to only allow the address block originated by our AS to get to our transit provider.

Note: in cases where the upstream provider only supplies the default route, we would not need to do the bogon filtering, but instead replace it with a filter only allowing the default route inbound.