IPv6 Router Security

Objective: Using an existing dual stack topology, investigate securing the router and the routing infrastructure for the network.

Prerequisites: Knowledge of Cisco IOS CLI.

The following will be the common topology used for this supplement.

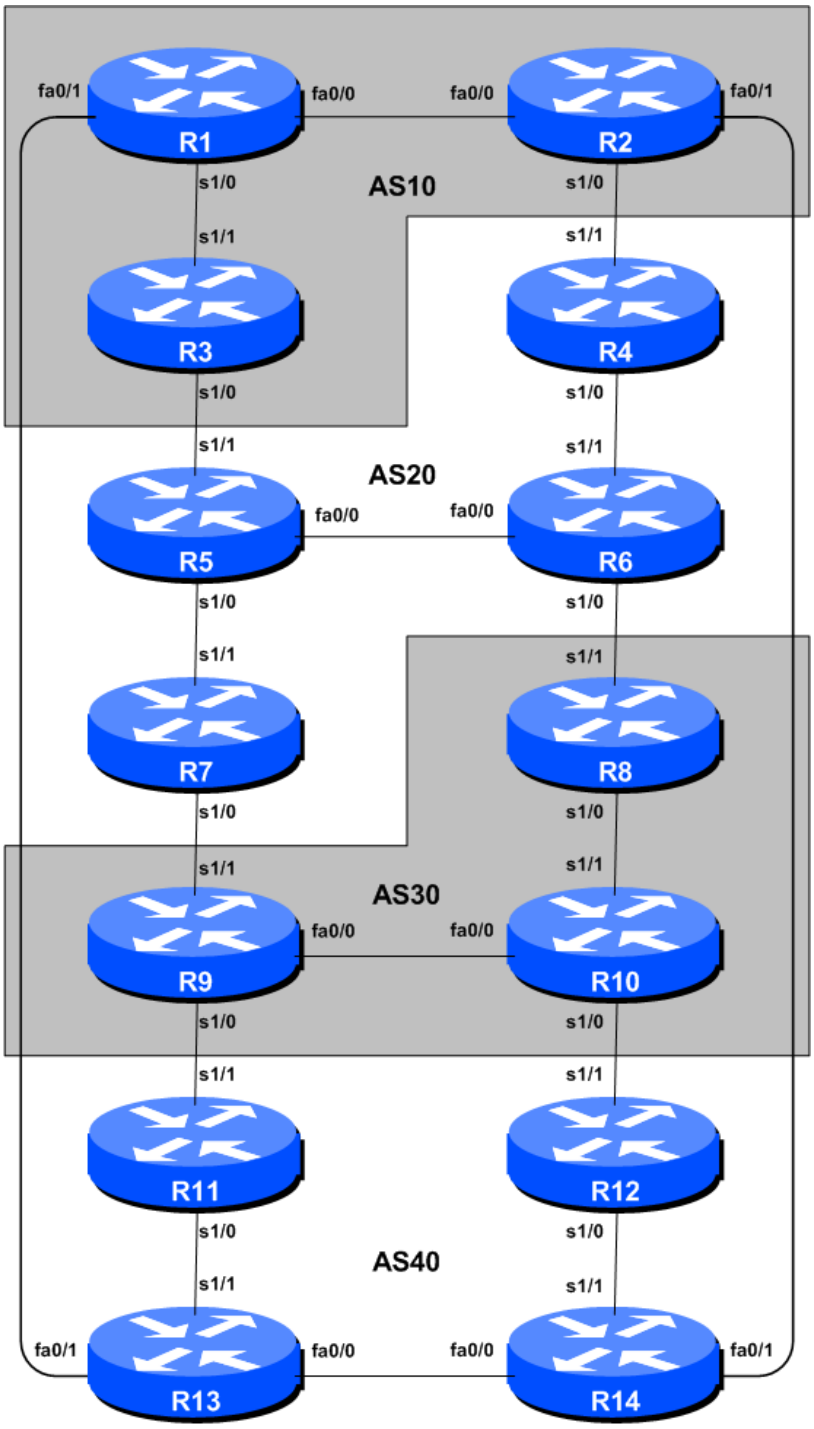

Figure 1 – Lab Basic Configuration

Lab Notes

This Module provides a guide to introducing network security. The configuration steps described below are aimed at securing the IPv6 portion of the infrastructure – the IPv4 portion can also be secured following a similar procedure.

The Module is split into two parts. Part 1 involves setting up the lab, ready to carry out the network security configurations in Part 2. The entire Module can be worked through from scratch – Part 2 is appropriate if this lab is part of a larger workshop, in which case participants can move directly to Part 2.

The routers used for this portion of the workshop must support IPv6. This is basically any IP Plus/Advanced IP Services image from 12.2T onwards.

Note: these labs assume that the routers used are using a minimum of IOS 12.4T mainline. Syntax predating IOS 12.4T will be similar, assuming the features being described are actually supported.

Part 1 – Setting up the lab

Lab Exercise One – Preliminary

1. **Introducing the lab.** This workshop uses Cisco IOS routers running IOS, but on the Dynamips systems – Dynamips translates the Cisco 7200 router MIPS processor instructions in IOS to those of the Intel based host system, allowing Cisco IOS images, and therefore network configurations, to be run on a host PC system (usual Linux or MacOS based).

   The lab will have been preconfigured by the instructors, allowing participants to enter the following exercises directly. Please read the following steps carefully.

2. **Accessing the lab.** The instructors will assign routers to each class group, and will indicate the method of access to the Dynamips server. This will usually be by wireless – if this is the case, make a note of the SSID and any password required. Also make a note of the IP address (IPv4, as Dynamips only supports IPv4 access) of the Dynamips server.

   Access to Dynamips will be by telnet, to a high port, which the instructor will specify. Each participant should ensure that their device has a suitable telnet client. Linux and MacOS system have access to a shell command prompt (or Terminal) programme, which allows telnet at the command line. Windows users can use the Windows "Command Prompt" with the telnet client there, but it's notoriously unreliable. Better to install software such as Putty, TeraTerm, HyperTerm or similar third party telnet client.

   Using the client, connect to the router you have been assigned; for example, to connect to the console port of Router 1:

   ```
   telnet 10.0.2.1  2001
   ```

   or to Router 12:

```
telnet 10.0.2.1  2012
```

Once connected, you will see the Dynamips response, followed by the login or command prompt of the router:

```
bash-3.2$ telnet 10.0.2.1 2001
Trying 10.0.2.1...
Connected to dynamips.
Escape character is '^]'.
Connected to Dynamips VM "r1" (ID 0, type c7200) - Console port

Router>
```

If the "Connected to Dynamips VM" won't appear, even after pressing the Return key several times, please request help from the workshop instructors.

3. **Router Hostname.** Each router will be named according to the table location, Router1, Router2, Router3, etc. Documentation and labs will also refer to *Router1* as R1. At the router prompt, first go into enable mode, then enter "config terminal", or simply "config" by itself:

```
Router> enable
Router# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# hostname Router1
Router1(config)#
```

4. **Configuring Cisco Routers – Best Practices.** The following set of configuration commands sets up Cisco routers for industry best practices – they are different from the Cisco configured defaults and are based on years of Network Operations experience.

```
no ip domain-lookup
!
line con 0
 transport preferred none
line vty 0 4
 transport preferred none
!
ipv6 unicast-routing
ipv6 cef
!
no ip source-route
no ipv6 source-route
!
username seclab secret lab-PW
enable secret lab-EN
service password-encryption
!
aaa new-model
aaa authentication login default local
aaa authentication enable default enable
!
no logging console
logging buffer 8192 debug
!
```

5. **Summary of the commands entered in the previous step.**
    a. **Turn Off Domain Name Lookups.** Cisco routers will always try to look up the DNS for any name or address specified in the command line.
    b. **Disable Command-line Name Resolution.** The router by default attempts to use the various transports it supports to resolve the commands entered into the command line during normal and configuration modes.
    c. **Enable IPv6.** Cisco routers with an IOS supporting IPv6 currently do not ship with IPv6 enabled by default.
    d. **Enable IPv6 CEF.** Unlike IPv4, CEFv6 is not enabled by default.
    e. **Disable IPv4 and IPv6 Source Routing.** Unless you really believe there is a need for it, source routing should be disabled. This option, enabled by default, allows the router to process packets with source routing header options. This feature is a well-known security risk.
    f. **Usernames and Passwords.** All router usernames should be ***seclab*** and password should be ***lab-PW***. Configuration mode password should be ***lab-EN***.
    g. ***service password-encryption*** directive tells the router to encrypt all passwords stored in the router's configuration (apart from *enable secret* which is already encrypted).
    h. **Enabling login access for other teams.** This is done using the ***aaa new-model*** directive in IOS.
    i. **Configure system logging.** A vital part of any Internet operational system is to record logs. We disables console logs and instead record all logs in a 8192byte buffer set aside on the router.

6. **Save the Configuration.** With the basic configuration in place, save the configuration. To do this, exit from enable mode by typing "end" or "<ctrl> Z", and at the command prompt enter "write memory".

```
Router1(config)#^Z
Router1# write memory
Building configuration...
[OK]
Router1#
```

It is highly recommended that the configuration be saved quite frequently to NVRAM, especially in the workshop environment where it is possible for power cables to become dislodged. If the configuration is not saved to NVRAM, any changes made to the running configuration will be lost after a power cycle.

Log off the router by typing exit, and then log back in again. Notice how the login sequence has changed, prompting for a "username" and "password" from the user. Note that at each checkpoint in the workshop, you should save the configuration to memory – remember that powering the router off will result in it reverting to the last saved configuration in NVRAM.

Checkpoint #1: *call the lab assistant to verify that you are properly logged in and can use the router.*

Exercise Two – Setting up Addressing

7. **IPv4 & IPv6 Addressing.** We now need to come up with a good scalable addressing plan for each AS in this network. Each AS gets their own IPv4 address block, a /20. This address block should be assigned to links and loopbacks on the routers making up each ASN. The allocations are as follows:

AS10	**10.10.0.0/20**	**AS30**	**10.30.0.0/20**
AS20	**10.20.0.0/20**	**AS40**	**10.40.0.0/20**

We need to do the same for IPv6. Each AS again gets their own address block, a /32 (typical minimum allocation for a starter Network Operator). This address block should be assigned to links and loopbacks on the routers making up each ASN. The allocations are as follows:

AS10	**2001:db8::/32**	**AS30**	**2001:dba::/32**
AS20	**2001:db9::/32**	**AS40**	**2001:dbb::/32**

We need to divide up each address block so that we have customer address space, network infrastructure address space, and some space for loopbacks. Figure 2 below reminds how this could be done for IPv4:

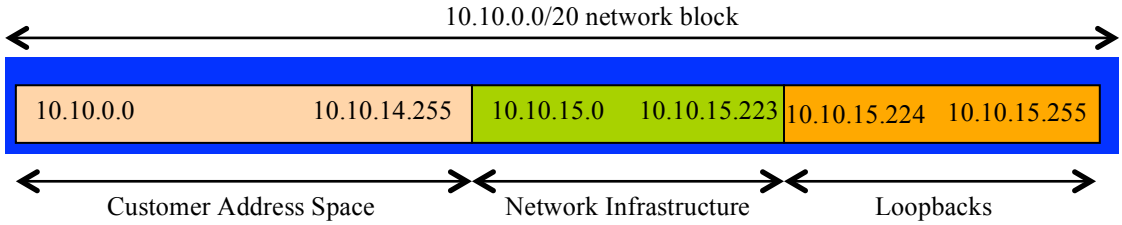

Figure 2 – Dividing allocated block of /20 into Customer, Infrastructure and Loopbacks

And similarly for IPv6 as Figure 3 shows:

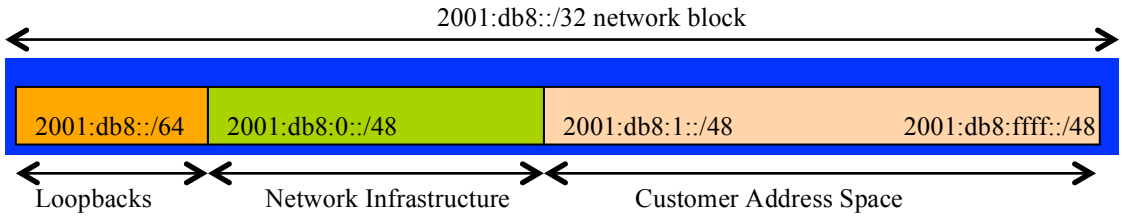

Figure 3 – Dividing allocated block of /20 into Customer, Infrastructure and Loopbacks

Please refer to the accompanying documents describing the detailed address plan which should be used in this lab – they are is entitled "Addressing Plan – Modules 6 to 9". Configure the addresses on each interface which will be used for this module, and check basic IP connectivity with your immediately adjacent neighbours.

8. **Serial Connections.** Consult the address plan mentioned in the previous step and assign addresses to the serial interfaces on your router. Here is an example for Router2:

```
Router2(config)# interface serial 1/0
Router2(config-if)# ip address 10.10.15.9 255.255.255.252
Router2(config-if)# ipv6 address 2001:db8:0:3::/127
Router2(config-if)# description 2 Mbps Link to Router4 via Serial
Router2(config-if)# bandwidth 2000
Router2(config-if)# no shutdown
```

Q: What network mask should be used for a point-to-point link address for IPv4 and for IPv6?

A: On serial interfaces, the network mask should be for IPv4 /30 (or 255.255.255.252 in dotted quad format). There is no point in using any other size of mask as there are only two hosts on such a link. For IPv6 the network mask should be /127. This is the subnet size used for all point-to-point links as recommended in RFC6164. We still reserve the entire /64 for this point-to-point link though, allowing simpler operational scalability should future changes be required.

9. **Ethernet Connections.** As with the serial connections, consult the address plan to assign IPv4 and IPv6 addresses to your router's Ethernet interfaces. Don't make the mistake of assigning a /24 mask to the interface's IPv4 address – there are only two hosts on the Ethernet connecting the two routers, so a /30 mask should be entirely sufficient. For IPv6, note that the link is point-to-point, so we only need a /127 for the netmask for the IPv6 addresses, exactly as we did for the Serial connections.

10. **Router Loopback Interface Addressing for IPv4.** We have set aside a /27 for loopbacks even though each AS has either 3 or 4 routers in it – this leaves more than sufficient room for future expansion. For example, if a Network Operator had 20 routers, they would need a /27 (or 32 host addresses) to provide a loopback address for each router. We have 14 routers in our lab – to be prudent and allow for growth, we will set aside a /27 (allows us 32 loopbacks) but only use 3 or 4 of them depending on which AS. The loopback address assignments which will be used for this module are below:

Router1	**10.10.15.224/32**	**Router8**	**10.30.15.224/32**
Router2	**10.10.15.225/32**	**Router9**	**10.30.15.225/32**
Router3	**10.10.15.226/32**	**Router10**	**10.30.15.226/32**
Router4	**10.20.15.224/32**	**Router11**	**10.40.15.224/32**
Router5	**10.20.15.225/32**	**Router12**	**10.40.15.225/32**
Router6	**10.20.15.226/32**	**Router13**	**10.40.15.226/32**
Router7	**10.20.15.227/32**	**Router14**	**10.40.15.227/32**

For example, Router Team 1 would assign the following address and mask to the loopback on Router 1:

```
Router1(config)#interface loopback 0
Router1(config-if)#ip address 10.10.15.224 255.255.255.255
```

Q: Why do we use /32 masks for the loopback interface address?

A: There is no physical network attached to the loopback so there can only be one device there. So we only need to assign a /32 mask – it is a waste of address space to use anything else.

11. **Router Loopback Interface Addressing for IPv6.** As the minimum subnet size possible for IPv6 is a /64, we will assign the first /64 out of our /48 infrastructure block to be used for loopbacks

even though each AS has either 3 or 4 routers in it. The loopback address assignments which will be used for this module are below:

Router1	**2001:db8::1**	**Router8**	**2001:dba::1**
Router2	**2001:db8::2**	**Router9**	**2001:dba::2**
Router3	**2001:db8::3**	**Router10**	**2001:dba::3**
Router4	**2001:db9::1**	**Router11**	**2001:dbb::1**
Router5	**2001:db9::2**	**Router12**	**2001:dbb::2**
Router6	**2001:db9::3**	**Router13**	**2001:dbb::3**
Router7	**2001:db9::4**	**Router14**	**2001:dbb::4**

For example, Router Team 1 would assign the following address and mask to the loopback on Router 1:

```
Router1(config)#interface loopback 0
Router1(config-if)#ipv6 address 2001:db8::1/128
```

Q: Why do we use /128 masks for the loopback interface address?

A: There is no physical network attached to the loopback so there can only be one device there. So we only need to assign a /128 mask – it is a waste of address space to use anything else.

Checkpoint #2: *call the lab assistant to verify that the addressing is correctly configured.*

Exercise Three – Setting up OSPF

12. Configure OSPF on the routers within each AS. In each AS configure OSPF routing. As we are running a dual stack network, we need to run OSPFv2 for IPv4 routes and OSPFv3 for IPv6 routes. This means setting up the OSPF process, marking **internal** interfaces as non-passive, and then configuring each **internal** interface and the loopback with the OSPF process. All the routers in an AS will be in the same OSPF *area 0* and use the same OSPF process ID.

OSPF should be configured on internal interfaces **only**. You do not want to set up adjacencies with devices outside your AS. Make sure that there are no *ip[v6] ospf* commands on external interfaces. A side effect of this is that external link addresses will not appear in the IGP (see the next section discussion iBGP deployment).

For example, Router 1, with two interfaces connecting to other routers in their AS, would have the following:

```
Router1 (config)# router ospf 10
Router1 (config-router)# passive-interface default
Router1 (config-router)# no passive-interface fastethernet 0/0
Router1 (config-router)# no passive-interface serial 1/0
Router1 (config-router)# log-adjacency-changes
!
Router1 (config)# ipv6 router ospf 10
Router1 (config-router)# passive-interface default
Router1 (config-router)# no passive-interface fastethernet 0/0
Router1 (config-router)# no passive-interface serial 1/0
Router1 (config-router)# log-adjacency-changes
!
Router1 (config)# interface fastethernet 0/0
```

```
Router1 (config-interface)# ip ospf 10 area 0
Router1 (config-interface)# ipv6 ospf 10 area 0
!
Router1 (config-interface)# interface serial 1/0
Router1 (config-interface)# ip ospf 10 area 0
Router1 (config-interface)# ipv6 ospf 10 area 0
!
Router1 (config-interface)# interface loopback 0
Router1 (config-interface)# ip ospf 10 area 0
Router1 (config-interface)# ipv6 ospf 10 area 0
!
```

Notes:

- *Passive-interface default* makes sure that OSPF does not attempt to set up adjacencies on any interfaces apart from those specified by the *no passive-interface* commands.
- The number following "*[ipv6] router ospf*" is a process ID and is used inside the router only (so it can be any number). But for this lab we recommend the OSPF process ID be the same as the AS number (which is the convention used by a number of Network Operators).

Hint: look at your OSPFv2 configuration and compare it with the OSPFv3 configuration. It should be identical (apart from the IPv6 commands). The topology is the same so the configuration of both OSPFs should be the same too.

13. **OSPF on Point-to-Point Ethernet Links.** We need to modify OSPF's behaviour on point-to-point broadcast media links, such as Ethernet, when there are only two devices on that media. If we declare such a situation to be point to point, then OSPF does not try and determine a designated or backup designated router; furthermore, there will be an improvement/simplification in SPF calculations and memory requirements on the router.

    Those router teams which have OSPF configured over an Ethernet interface should now convert OSPF to point-to-point mode, for example:

    ```
    Router1 (config)# interface fastethernet 0/0
    Router1 (config-interface)# ip ospf network point-to-point
    Router1 (config-interface)# ipv6 ospf network point-to-point
    ```

    The result will be that the DR or BDR entry in the status column of *"show ip[v6]ospf neighbor"* will disappear, to be replaced with a FULL. The link is now treated like a point-to-point serial connection.

14. **Ping Test.** Check the routes via OSPF. Make sure you can see all the networks within your AS. Ping all loopback interfaces within your AS. Use the "*show ip[v6] ospf neighbor"* and *"show ip[v6] route"* commands. If you cannot see the other routers in your AS, you will not be able to bring up BGP in the next steps.

15. **Telnet source address.** Most Network Operators use the router Loopback address for administrative purposes as well as the anchor point for their network's iBGP sessions. In this step we will configure telnet so that it uses the loopback interface as the source address for all telnet packets (IPv4 and IPv6) originated by the router.

    ```
    ip telnet source-interface loopback 0
    ```

To check that this has worked, telnet from your router to a neighbouring router and then enter the "who" command. You will see that you are logged in, and the source address will be displayed. For example, using telnet from Router1 to Router3 gives:

```
Router3>who
    Line      User       Host(s)                 Idle Location
*  2 vty 0    isplab     idle                 00:00:00 10.10.15.224
```

16. Save the configuration. Don't forget to save the configuration to NVRAM!

Checkpoint #3: *call the lab assistant to verify the connectivity within your AS – you should be able to ping the loopbacks of the other routers in your AS, and see the loopback interface addresses in the OSPFv2 and OSPFv3 RIBs.*

Exercise Four – Setting up BGP

17. Configure IPv4 iBGP peering between routers within an AS. Use the loopback address for the iBGP peerings. Also, configure the *network* command to add the address block assigned to each AS for advertisement in BGP. Each router team should announce this address block from their routers.

```
Router1 (config)# router bgp 10
Router1 (config-router)# bgp deterministic-med
Router1 (config-router)# no bgp default ipv4-unicast
Router1 (config-router)# address-family ipv4
Router1 (config-router-af)# distance bgp 200 200 200
Router1 (config-router-af)# network 10.10.0.0 mask 255.255.240.0
Router1 (config-router-af)# neighbor 10.10.15.225 remote-as 10
Router1 (config-router-af)# neighbor 10.10.15.225 update-source loopback 0
Router1 (config-router-af)# neighbor 10.10.15.225 next-hop-self
Router1 (config-router-af)# neighbor 10.10.15.225 send-community
Router1 (config-router-af)# neighbor 10.10.15.225 description iBGP Link to R2
Router1 (config-router-af)# neighbor 10.10.15.226 remote-as 10
Router1 (config-router-af)# neighbor 10.10.15.226 update-source loopback 0
Router1 (config-router-af)# neighbor 10.10.15.226 next-hop-self
Router1 (config-router-af)# neighbor 10.10.15.226 send-community
Router1 (config-router-af)# neighbor 10.10.15.226 description iBGP Link to R3
Router1 (config-router-af)# exit
Router1 (config)# ip route 10.10.0.0 255.255.240.0 Null0
```

18. Configure IPv6 iBGP peering between routers within an AS. Use the loopback address for the iBGP peerings. Also, configure the *network* command to add the address block assigned to each Router Team for advertisement in BGP.

```
Router1 (config)# router bgp 10
Router1 (config-router)# bgp log-neighbor-changes
Router1 (config-router)# address-family ipv6
Router1 (config-router-af)# distance bgp 200 200 200
Router1 (config-router-af)# network 2001:db8::/32
Router1 (config-router-af)# neighbor 2001:db8::2 remote-as 10
Router1 (config-router-af)# neighbor 2001:db8::2 update-source loopback 0
Router1 (config-router-af)# neighbor 2001:db8::2 next-hop-self
Router1 (config-router-af)# neighbor 2001:db8::2 send-community
Router1 (config-router-af)# neighbor 2001:db8::2 description iBGP Link to R2
Router1 (config-router-af)# neighbor 2001:db8::3 remote-as 10
```

```
Router1 (config-router-af)# neighbor 2001:db8::3 update-source loopback 0
Router1 (config-router-af)# neighbor 2001:db8::3 next-hop-self
Router1 (config-router-af)# neighbor 2001:db8::3 send-community
Router1 (config-router-af)# neighbor 2001:db8::3 description iBGP Link to R3
Router1 (config-router-af)# exit
Router1 (config)# ipv6 route 2001:db8::/32 Null0
```

19. **Next-hop-self configuration.** The previous step introduced the next-hop-self configuration command. The next-hop-self configuration makes the iBGP speaking router use the iBGP source address (in this case the loopback) rather than the external next-hop address (as per the BGP specification). This is industry best practice and means that Network Operators do not need to carry external next-hops in their IGP.

20. **Configure eBGP peering.** Use Figure 1 to determine the links between the AS's. Addressing for eBGP links between 2 AS's will use the point-to-point interface addresses, ***NOT*** the loopback addresses (review the BGP presentation if you don't understand why).

```
Router1 (config)# router bgp 10
Router1 (config-router)# address-family ipv4
Router1 (config-router-af)# neighbor 10.10.15.14 remote-as 40
Router1 (config-router-af)# neighbor 10.10.15.14 descr eBGP to Router13
Router1 (config-router-af)# address-family ipv6
Router1 (config-router-af)# neighbor 2001:db8:0:4::1 remote-as 40
Router1 (config-router-af)# neighbor 2001:db8:0:4::1 descr eBGP to Router13
```

Use the BGP Show commands to ensure you are sending and receiving the BGP advertisements from your eBGP neighbours.

21. **Adding a IPv4 "customer" route into BGP.** We are now going to add an IPv4 "customer" route into BGP on each router. We don't have any "customers" as such connected to our routers in the lab, so we are going to simulate the connectivity by simply using a Null0 interface. The "customer" address space that each router team will introduce into the iBGP is listed below – again we will each use a /26, for simplicity's sake.

R1	**10.10.0.0/26**	**R8**	**10.30.0.0/26**
R2	**10.10.0.64/26**	**R9**	**10.30.0.64/26**
R3	**10.10.0.128/26**	**R10**	**10.30.0.128/26**
R4	**10.20.0.0/26**	**R11**	**10.40.0.0/26**
R5	**10.20.0.64/26**	**R12**	**10.40.0.64/26**
R6	**10.20.0.128/26**	**R13**	**10.40.0.128/26**
R7	**10.20.0.192/26**	**R14**	**10.40.0.192/26**

Each team should now set up a static route pointing to the **NULL0** interface for the /26 that they are to originate. Once the static is set up, the team should then add an entry into the BGP table. Here is an example for Router8:

```
Router8 (config)# ip route 10.30.0.0 255.255.255.192 Null0
Router8 (config)# router bgp 30
Router8 (config-router)# address-family ipv4
Router8 (config-router-af)# network 10.30.0.0 mask 255.255.255.192
```

22. **Adding an IPv6 "customer" route into BGP.** As we did for IPv4 in the previous step, we are now going to add a "customer" route into BGP on each router. We don't have any "customers" as

such connected to our routers in the lab, so we are going to simulate the connectivity by simply using a Null0 interface. The "customer" address space that each router team will introduce into the iBGP is listed below – again we will each use a /48, for simplicity's sake.

R1	**2001:db8:1::/48**	**R8**	**2001:dba:1::/48**
R2	**2001:db8:2::/48**	**R9**	**2001:dba:2::/48**
R3	**2001:db8:3::/48**	**R10**	**2001:dba:3::/48**
R4	**2001:db9:1::/48**	**R11**	**2001:dbb:1::/48**
R5	**2001:db9:2::/48**	**R12**	**2001:dbb:2::/48**
R6	**2001:db9:3::/48**	**R13**	**2001:dbb:3::/48**
R7	**2001:db9:4::/48**	**R14**	**2001:dbb:4::/48**

Each team should now set up a static route pointing to the **NULL0** interface for the /48 that they are to originate. Once the static is set up, the team should then add an entry into the BGP table. Here is an example for Router8:

```
Router8 (config)# ipv6 route 2001:dba:1::/48 Null0
Router8 (config)# router bgp 30
Router8 (config-router)# address-family ipv6
Router8 (config-router-af)# network 2001:dba:1::/48
```

23. Check the BGP table. Are there routes seen via ***show ip bgp*** and ***show bgp ipv6 unicast***? If not, why not? Once every team in the class has done their configuration, each team should see the aggregate from each AS as well as the fourteen customer prefixes introduced in the previous steps. If this is not happening, work with your neighbours to fix the problem.

Checkpoint #4: *call the lab assistant to verify that you are properly logged in and can use the router. Demonstrate various show commands, for example, "show ip bgp summary" to demonstrate the BGP sessions which are running, and "show ip bgp" to show the BGP routing table.*

Part 2 – IPv6 Network Security

Lab Notes

This Module provides a guide to introducing network security. The configuration steps described below are aimed at securing the IPv6 portion of the infrastructure – the IPv4 portion can also be secured following a similar procedure.

The routers used for this portion of the workshop must support IPv6. This is basically any IP Plus image from 12.2T onwards (IP Plus was renamed to Advanced IP Services for most platforms as from 12.3 mainline). As always, it is best to check the Cisco Feature Navigator www.cisco.com/go/fn to be absolutely sure which images set and platform supports IPv6. Unfortunately IPv6 is not part of the basic IP only or Service Provider IOS images used by most Network Operators.

Note: these labs assume that the routers used are using a minimum of IOS 12.4T mainline. Syntax predating IOS 12.4T will be similar and is discussed in the optional sections throughout the workshop.

Exercise Five

24. Getting used to IPv6 show commands. Try the following IPv6 show commands:

```
show ipv6 interface
show ipv6 neighbors
show ipv6 route
show ipv6 routers
show ipv6 ospf neighbors
show ipv6 ospf rib
show ipv6 traffic
show bgp ipv6 unicast summary
show ipv6 ?
```

The last show command will display all the possible IPv6 status commands on the router. What you see in the list will depend on the IOS in use.

Also, make a note of what each of the above show commands do – the lab instructors may ask later in the exercise.

25. Set the time zone on the router. The router can be set with a time zone offset from GMT. The timezone command takes a string of characters – obviously set it to the local timezone. Note that only the first seven characters are used in any time display. The following sets the time zone for IndoChina with a GMT offset of +7.

```
clock timezone ICT 7
```

or

```
clock timezone GMT+7 7
```

26. Set time stamps for all logs on the router. By default the router will apply only time stamps on log messages from when the router was last powered on. This is not entirely useful in an operational environment, and is a potential security problem, especially when trying to cross

compare logs. So we will set time stamps according to the real date and time, and include resolution down to the millisecond:

```
service timestamps debug datetime localtime show-timezone msec
service timestamps log datetime localtime show-timezone msec
```

27. **Login Banner.** IOS by default has a simple welcome message when a new administrative connection to the router is opened. Most Network Operators tend to customise this banner to be appropriate to their business. We will now set up a login banner for the routers in the workshop lab. Use an appropriate greeting – one that doesn't give information away, and makes it very clear that access to the device is restricted to those with permission to do so. If you use an inappropriate greeting, expect the lab instructors to ask you to change it. Use the following example:

```
login banner ^
ITU/APNIC IPv6 Security Workshop Lab
^
```

28. **Logging.** Routers by default capture syslog data produce locally by various features in the IOS. However, the default logging set up is probably not optimal for Network Operators. Each router team should configure logging defaults on their router to be as follows:

```
no logging console
logging source-interface Loopback 0
logging trap debugging
logging buffered 16384 debugging
logging facility local4
logging 192.168.1.4
```

This command set will set the log source interface to the Loopback 0 interface, trap level to debug (i.e. most detailed), create a 16K buffer on the router and store the most detailed logs there, and any logs sent to the 192.168.1.4 loghost should be sent using syslog facility local4.

It is highly desirable (if not best practice) to disable logging to the router console. If you still haven't done this then the command to do so is `no logging console`. Console logging is on by default in IOS.

NOTE: For Internet Network Operations, it is strongly recommended to DISABLE console logging – router consoles are usually connected to terminal servers as we have just seen, and if the router is under some stress due to events taking place on it, or on the network, it will waste considerable CPU cycles by sending log messages to the 9600baud console port, thereby further slowing it down. Virtually all Network Operators disable console logging, and alternatively have the syslog messages sent to the local syslog host, as per the configuration above.

Exercise Six – Administrative Access

29. **Configuring Telnet VTY access for IPv6.** Configure a filter to allow only the trusted hosts to have Telnet access. Note that all attempts are logged by the router system log process, so that there is an audit trail of all access to the router. Part of the AAA suite in Cisco IOS allows these authentication logs to be exported to a syslog server where further access tracking can be undertaken.

```
router1(config)# ipv6 access-list v6-vty-filter
router1(config-ipv6-acl)# permit host ipv6-address any
```

Replace "ipv6-address" with the IPv6 address of the host you would like to have access. Test this with a physically adjacent router in the class. For example, Router3 could choose either Router1 or Router5. Take the IPv6 address of the physical interface of the adjacent router connecting to your router, and add that into the access-list you have configured.

30. **Applying the filter to the VTY ports**. Once the filter is set up, apply it to the vty ports on the router, as in the following example for Router1:

```
router1(config)# line vty 0 4
router1(config-line)# ipv6 access-class v6-vty-filter
```

Test to make sure that only telnet from the configured host can have access to the router. To do this, telnet to the adjacent router, and try and telnet back in to yours. You should have ready access. Now telnet to another router in the network and check again – do you get access? If you do, check your filters! Use the debug command to see if you can capture the telnet packets and see the clear-text passwords (be careful with debug!).

Checkpoint 5: *call the lab assistant and demonstrate the function of your telnet VTY filter.*

STOP AND WAIT HERE

Exercise Seven – Remote Access

31. **Configuring the VTYs for SSH access.** Now that we have configured and tested basic IPv6 filtering on the router, we are going to configure access to the router to be somewhat more secure. Everything is sent in the clear through telnet, and it's use is considered historical and deprecated by most network operators today.

First of all, we need to enable SSH on the routers – SSH is not enabled by default on Cisco IOS. To do this, we first create a domain name for the network:

```
router1(config)# ip domain-name workshop.net
```

and then we need to generate the SSH key pairs for the router:

```
router1(config)# crypto key generate rsa
```

The router will respond, asking what key modulus is required. Choose a modulus of at least 768 (the minimum for SSH version 2), preferably 1024. For example:

```
The name for the keys will be: router1.workshop.net
Choose the size of the key modulus in the range of 360 to 4096 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take a few
minutes.

How many bits in the modulus [512]:  1024
```

```
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)

router1(config)#
```

Finally set the SSH version to be version 2:

```
router1(config)# ip ssh version 2
```

SSHv2 is now configured on the router, and ready to use. To verify that SSH is working, try and SSH to the router itself:

```
router1# ssh 2001:db8::1
Password:

router1>
```

You should get the password prompt, and following entry of the correct password, the standard command prompt.

32. **Modifying VTY access for SSH.** IOS by default allows all transports to connect to the VTY ports. We want to change this behaviour to enhance the security of the router access.

```
router1(config)# line vty 0 4
router1(config-line)# ipv6 access-class v6-vty-filter in
router1(config-line)# exec-timeout 15 0
router1(config-line)# transport input ssh
```

This sequence of commands applies the previous configured VTY filter to VTYs 0 through 4, changes the exec-timeout (how long the vty session can remain idle before disconnection) to 15 minutes, and disabling all connecting transports apart from SSH.

Test to make sure that SSH from the permitted host can get access to the router.

Try using telnet from the permitted host as well. Do you get access?

Use the debug command to see if you can capture the SSH packets and see the encrypted passwords.

33. **SSH source address.** Most Network Operators use the router Loopback address for administrative purposes as well as the anchor point for their network's iBGP sessions. In this step we will configure SSH so that it uses the loopback interface as the source address for all SSH packets originated by the router.

```
ip ssh source-interface loopback 0
```

To check that this has worked, SSH from your router to a neighbouring router and then enter the "who" command. You will see that you are logged in, and the source address will be displayed.

34. **Disabling TELNET access.** Telnet is a well-known security risk and most network operators disable its use on their network infrastructure equipment. (Equipment which doesn't support SSH should probably be retired and replaced with equipment which supports secure administrator access.) We will now also disable telnet access from the router, as the following example shows:

```
router1(config)# line vty 0 4
router1(config-line)# transport output ssh
router1(config-line)# line con 0
router1(config-line)# transport output ssh
```

Try using telnet now for a connection to another router – does it work? What happens?

Checkpoint #6: *call the lab assistant and demonstrate the function of your SSH VTY filter. Also demonstrate that you can no longer use telnet to access your router.*

STOP AND WAIT HERE

Exercise Eight – IPv6 Traffic Filters

35. Configuring IPv6 Traffic Filters. We now configure a traffic filter to only allow traffic from just your address block out of your network (this is BCP38 requirement). For this lab, the address block being used by each AS is in the following table:

AS10	**2001:db8::/32**	**AS30**	**2001:dba::/32**
AS20	**2001:db9::/32**	**AS40**	**2001:dbb::/32**

Each team should set up a traffic filter to only allow traffic from this subnet to exit each router interface. Here is an example for Router1:

```
router1(config)# ipv6 access-list ipv6-packetfilter
router1(config-ipv6-acl)# permit ipv6 2001:db8::/32 any
router1(config-ipv6-acl)# permit icmp any any
router1(config-ipv6-acl)# deny ipv6 any any log
router1(config-ipv6-acl)# exit
router1(config)# interface <interface>
router1(config-if)# ipv6 traffic-filter ipv6-packetfilter out
```

Do this for each physical interface on the router which is configured and active. Some routers have two physical interfaces (Serial 1/0 and Serial 1/1), others have three physical interfaces (either two serial and one Ethernet, or one serial and two Ethernet).

What happens? Look in the router's logs. Can you explain what you see? Why do your external BGP sessions go down?

36. Inbound packet filtering for IPv6 testing. We will now create an access-list which can be used for initial IPv6 testing. It shows had to trap and test for various traffic types running on a router's interface.

```
ipv6 access-list v6starter
  permit icmp any 2001:db8::/32 echo-reply log-input
  permit icmp any 2001:db8::/32 echo-request log-input
  permit icmp any 2001:db8::/32 time-exceeded log-input
  permit icmp any 2001:db8::/32 packet-too-big log-input
  permit icmp any 2001:db8::/32 parameter-problem log-input
```

```
  permit ipv6 any host <specific host> log-input
  deny ipv6 any any log-input
!
interface <interface>
 ipv6 traffic-filter v6starter in
!
```

Note that the 'log-input' has been included to check what ipv6 traffic is coming in from the outside. Send some ipv6 pings and see if you can see traffic from a 'show log'.

Note: 'log' simply displays the source and destination addresses in the log messages. 'log-input' includes the input interface as well.

37. **Disabling Router Advertisement on interfaces.** Interfaces on core infrastructure routers are generally manually configured. And the devices connected to these interfaces also are manually configured. We want to now disable support for auto-configuration of IPv6 addresses on our router's ethernet interfaces and disable the announcement of a default route – this will prevent anyone connecting a device to that ethernet and it automatically getting an IPv6 configuration and this IPv6 connectivity. The following command shows what needs to be done for a FastEthernet interface.

```
interface fastethernet 0/0
 ipv6 nd prefix default no-advertise
 ipv6 nd ra suppress all
!
```

Each group should do this for all active Ethernet interfaces on their router.

Checkpoint #7: *call the lab assistant and demonstrate the function of your IPv6 starter interface filter. Show the log messages you see from the various testing you have been doing.*

STOP AND WAIT HERE

Exercise Nine – OSPF neighbour authentication

38. **Remove the packet filters from the previous examples.** We need OSPFv3 and BGP to work for the remainder of the lab, so we will remove the packet filters we installed in the previous exercise. Simply go to the interfaces where you defined the "traffic-filter" commands and remove those. Also delete the "ipv6 access-list" configurations for these two traffic filters.

39. **Configuring Neighbour Security for OSPFv2 (for IPv4).** 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.

    OSPF 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 OSPFv2. This first step will enable the area to support authentication using the *area N authentication message-digest* command. As soon as this is done, all adjacencies will be expected to use neighbour authentication.

An example configuration for Router6 might be:

```
Router6(config)#router ospf 20
Router6(config-router)# area 0 authentication message-digest
```

Note that this does not affect the actual adjacencies on the routers – it only tells the router that the area mentioned will use authentication, if it is configured.

40. **Intra Area Authentication – Part 2**. Now that support for authentication in each area has been configured, the second step is to actually set the authentication password to be used, and the interface it has to be used on. The password that should be used for all areas in this example is *cisco*. MD5 encryption should be used rather than the standard simple encryption – to do this, use the *message-digest-key* sub-interface command.

    An example configuration for Router6 might be:

    ```
    interface fastethernet 0/0
      ip ospf message-digest-key 1 md5 cisco
    !
    interface serial 1/1
      ip ospf message-digest-key 1 md5 cisco
    ```

    Notice now that the OSPF adjacencies do not come up unless the neighbouring router has also entered the same configuration and key. Notice also how the OSPF adjacencies were reset as the configuration was entered – security is being introduced, so the adjacencies are reset.

    Note: the *message-digest-key* allows up to 255 keys to be set 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.

41. **Check OSPFv2 operation.** Use the various "*show ip ospf*" commands to see the OSPF 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.

42. **Configuring Neighbour Security for OSPFv3 (for IPv6).** 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.

    Each team will turn on authentication for area 0 using the *area N authentication* command. An example configuration for Router4 might be:

    ```
    ipv6 router ospf 20
     area 0 authentication ipsec spi 256 md5 0123456789ABCDEF0123456789ABCDEF
    !
    ```

Notice now that the OSPFv3 adjacencies do not come up unless the neighbouring router has also entered the same configuration and key. Notice also how the OSPFv3 adjacencies timed out after the configuration was entered – the adjacent router still had not set up OSPFv3 neighbour authentication. Once they also have done this, the adjacency is re-established.

Also, if we want to configure neighbour authentication per area but disable it on a per link basis, we need to use the `authentication null` option. The following example shows how to disable neighbour authentication on one specific interface, hssi 5/0.

```
interface hssi 5/0
 ipv6 ospf authentication null
 ipv6 ospf 1 area 10
```

43. **Check OSPFv3 operation.** Use the various "*show ipv6 ospf*" commands to see the OSPFv3 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.

44. **Aside: Neighbour authentication in ISIS.** This particular lab is not using ISIS as the IGP but the ISIS configuration is included for the sake of completeness. As with most things in ISIS, the neighbour authentication support is much simpler to configure.

    While an attack on ISIS is harder as it runs on the link layer alongside IP rather than on top of IP like OSPF, some Network Operators are still prudent and implement neighbour authentication. Notice that for ISIS, the neighbour authentication is applied per neighbour, not per protocol address family, as it would be in OSPF.

    The first step for ISIS is to set up the keychain to be used – it has been called "lab-key". The key "cisco" is used in the example below:

```
key chain lab-key
  key 1
  key-string cisco
```

    Once the keychain has been defined, the router interfaces can be individually activated to support authentication. The first step is to enable MD5 for level-2 IS's:

```
interface fastethernet 0/0
  isis authentication mode md5 level-2
```

    And then associate the key-chain we defined earlier with the configured authentication:

```
interface fastethernet 0/0
  isis authentication key-chain lab-key level-2
```

    The ISIS adjacencies do not come up unless the neighbouring router has also entered the same configuration and key.

Checkpoint #8: *call the lab assistant and demonstrate your working authenticated OSPFv2 and OSPFv3 adjacencies. Show the log messages you see from the various testing you have been doing.*

STOP AND WAIT HERE

Exercise Ten – BGP Neighbour Authentication

45. Configure passwords on the iBGP sessions. Passwords should now be configured on the iBGP sessions. Review the presentation why this is necessary. Agree amongst all your team members in your AS what the password should be on the iBGP session, and then apply it to all the iBGP peerings on your router. For example, on Router2's peering with Router3, with "cisco" used as the password:

```
Router2 (config)# router bgp 10
Router2 (config-router)# neighbor 10.10.15.226 password cisco
```

IOS currently resets the iBGP session between you and your neighbouring router whenever an MD5 password is added. So when passwords are added to BGP sessions on live operational networks, this work should be done during a maintenance period when customers know to expect disruptions to service. In the workshop lab, it doesn't matter so much. (Future IOS releases will avoid having this rather serious service disruption.)

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
```

46. Configure passwords on the eBGP session. Passwords should now be configured on the eBGP sessions between your and your neighbouring ASes. Agree between you and your neighbouring AS what the password should be on the eBGP session, and then apply it to the eBGP peering. For example, on Router2's peering with Router4, with "cisco" used as the password:

```
Router2 (config)# router bgp 10
Router2 (config-router)# neighbor 10.10.15.10 password cisco
```

As previously for the iBGP session, watch the logs for password mismatches, or missing passwords. As with the iBGP sessions previously, you will find that the router will reset the eBGP session as soon as the password is applied.

47. Configure passwords on the IPv6 iBGP sessions. Passwords should now be configured on the IPv6 iBGP sessions. Again agree amongst all your team members in your AS what the password

should be on the iBGP session, and then apply it to all the iBGP peerings on your router. For example, on Router2's peering with Router3, with "cisco" used as the password:

```
router bgp 10
 address-family ipv6
  neighbor 2001:db8::3 password cisco
```

Watch the router logs – with the BGP session neighbour changes being logged, any mismatch in the password should be easy to spot.

48. Configure passwords on the IPv6 eBGP session. Passwords should now be configured on the eBGP sessions between your and your neighbouring ASes. Agree between you and your neighbouring AS what the password should be on the eBGP session, and then apply it to the eBGP peering. For example, on Router2's peering with Router4, with "cisco" used as the password:

```
router bgp 10
 address-family ipv6
  neighbor 2001:db8:0:3::2 password cisco
```

As previously for the iBGP session, watch the logs for password mismatches, or missing passwords. As with the iBGP sessions previously, you will find that the router will reset the eBGP session as soon as the password is applied.

Checkpoint #9: *call the lab assistant and demonstrate the passwords applied to the IPv4 and IPv6 BGP sessions.*

Exercise Eleven – Bogon Filtering

49. Configuring 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 10.10.0.0/20
ip prefix-list bogon-filter permit 10.20.0.0/20
ip prefix-list bogon-filter permit 10.30.0.0/20
ip prefix-list bogon-filter permit 10.40.0.0/20
! Drop all the Bogons
ip prefix-list bogon-filter deny 0.0.0.0/0
```

```
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.

50. Apply the prefix-filter to the IPv4 eBGP sessions. We now apply this prefix filter to our external BGP sessions. For example for Router1:

```
router bgp 10
 address-family ipv4 unicast
  neighbor 10.10.15.14 remote-as 40
  neighbor 10.10.15.14 version 4
  neighbor 10.10.15.14 prefix-list bogon-filter in
  neighbor 10.10.15.14 prefix-list bogon-filter out
!
```

Once you have entered the above configuration, refresh the BGP session by entering the following command (example again for Router1). This refresh command applies the newly added BGP configuration to the BGP session. The following example is again for the eBGP session between Router1 and Router13.

```
clear ip bgp 40 out
clear ip bgp 40 in
```

Note that it is common for smaller Network Operators to have an upstream send only a default route which is enforced via an inbound prefix list. It is also common to create 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 DEFAULT-IN permit 0.0.0.0/0
ip prefix-list BGP-OUTBOUND permit <specific aggregate>
!
router bgp 64510
 neighbor 1.2.3.4 remote-as 64509
 neighbor 1.2.3.4 prefix-list DEFAULT-IN in
 neighbor 1.2.3.4 prefix-list BGP-OUTBOUND out
!
```

51. Repeat with IPv6. Each Router Team 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::/32
ipv6 prefix-list v6bogon-filter permit 2001:db9::/32
ipv6 prefix-list v6bogon-filter permit 2001:dba::/32
ipv6 prefix-list v6bogon-filter permit 2001:dbb::/32
! 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::/32 le 128
ipv6 prefix-list v6bogon-filter deny 2001:db8::/32 le 128
ipv6 prefix-list v6bogon-filter permit 2002::/16
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.

52. 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. Here is the Router1 to Router13 eBGP session example:

```
router bgp 10
 address-family ipv6 unicast
  neighbor 2001:DB8:0:4::1 remote-as 40
  neighbor 2001:DB8:0:4::1 version 4
  neighbor 2001:DB8:0:4::1 prefix-list v6bogon-filter in
  neighbor 2001:DB8:0:4::1 prefix-list v6bogon-filter out
!
```

Don't forget to refresh the BGP session with AS40 after applying the filter. Again using the previous Router1 to Router13 eBGP session as an example:

```
clear bgp ipv6 uni 40 in
clear bgp ipv6 uni 40 out
```

Checkpoint #10: *call the lab assistant and demonstrate the IPv4 and IPv6 BGP filters as applied to your eBGP sessions.*

STOP AND WAIT HERE

Exercise Twelve – Netflow

53. Exploring Netflow. Netflow identifies anomalous and security-related network activity by tracking network flows. NetFlow data can be viewed and analysed via the command line interface (CLI), or the data can be exported to a commercial or freeware NetFlow collector for aggregation and analysis. NetFlow collectors, through long-term trending, can provide network behaviour and usage analysis. NetFlow functions by performing analysis on specific attributes within IP packets and creating flows. Version 5 is the most commonly used version of NetFlow, however, version 9 is more extensible and is required to support IPv6. NetFlow flows can be created using sampled traffic data in high-volume environments. Cisco Express Forwarding (CEF) is a prerequisite to enabling NetFlow.

NetFlow can be configured on routers and switches. In older releases of Cisco IOS software, the command to enable NetFlow on an interface was:

```
ip route-cache flow
```

In newer releases of Cisco IOS (12.4 onwards), the command has been replaced by:

```
ip flow {ingress | egress}
```

The following configuration illustrates the basic configuration of this feature.

```
ip flow-export destination <ip-address> <udp-port>
ip flow-export version <version>
!
interface fastethernet 0/0
 ip flow ingress
 ip flow egress
!
```

The following is an example of NetFlow output from the router command line interface. The SrcIf attribute can aid in traceback.

```
router#show ip cache flow
IP packet size distribution (26662860 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .741 .124 .047 .006 .005 .005 .002 .008 .000 .000 .003 .000 .001 .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .001 .007   .039   .000   .000  .000  .000  .000   .000

IP Flow Switching Cache, 4456704 bytes
  55 active, 65481 inactive, 1014683 added
  41000680 ager polls, 0 flow alloc failures
  Active flows timeout in 2 minutes
  Inactive flows timeout in 60 seconds
IP Sub Flow Cache, 336520 bytes
  110 active, 16274 inactive, 2029366 added, 1014683 added to flow
  0 alloc failures, 0 force free
  1 chunk, 15 chunks added
  last clearing of statistics never

Protocol        Total   Flows     Packets  Bytes  Packets Active(Sec) Idle(Sec)
--------        Flows   /Sec       /Flow   /Pkt     /Sec     /Flow     /Flow
TCP-Telnet      11512    0.0        15      42      0.2      33.8      44.8
TCP-FTP          5606    0.0         3      45      0.0      59.5      47.1
TCP-FTPD         1075    0.0      13      52      0.0       1.2      61.1
TCP-WWW         77155    0.0      11     530      1.0      13.9      31.5
```

```
TCP-SMTP         8913    0.0       2      43      0.0      74.2      44.4
TCP-X             351    0.0       2      40      0.0       0.0      60.8
TCP-BGP           114    0.0         1      40      0.0       0.0      62.4
TCP-NNTP          120    0.0       1      42      0.0       0.7      61.4
TCP-other      556070    0.6         8     318      6.0       8.2      38.3
UDP-DNS        130909    0.1         2      55      0.3      24.0      53.1
UDP-NTP        116213    0.1         1      75      0.1       5.0      58.6
UDP-TFTP          169    0.0         3      51      0.0      15.3      64.2
UDP-Frag            1    0.0         1    1405      0.0       0.0      86.8
UDP-other       86247    0.1       226      29     24.0      31.4      54.3
ICMP            19989    0.0        37      33      0.9      26.0      53.9
IP-other          193    0.0         1      22      0.0       3.0      78.2
Total: 1014637    1.2      26      99     32.8      13.8      43.9

SrcIf    SrcIPaddress      DstIf  DstIPaddress     Pr    SrcP     DstP   Pkts
Gi0/1    192.168.128.21     Local 192.168.128.20   11    CB2B     07A       3
Gi0/1    192.168.150.60     Gi0/0 10.89.17.146     06    0016     101F     55
Gi0/0    10.89.17.146       Gi0/1 192.168.150.60   06    101F     0016      9
Gi0/1    192.168.150.60     Local 192.168.206.20   01    0000     0303     11
Gi0/0    10.89.17.146       Gi0/1 192.168.150.60   06    07F1     0016      1
```

54. Netflow for IPv4. To get some practice, we will first turn on Netflow for IPv4. The IPv4 command set uses Cisco's original Netflow configuration. For IPv6 flow information, we can only use Flexible Netflow, and we will try that out in the next section.

Each team member should turn on Netflow on the border interfaces of their routers (the border interfaces are those which connect to another Autonomous System). To do this, simply go to the border interface and do something similar to (in the case of Router1):

```
interface fastethernet 0/1
 ip flow ingress
 ip flow egress
!
```

Once this has been running for a few minutes, commands like "show ip cache flow" will display output similar to that from the introduction above. To create traffic for Netflow to see, try some ICMPs, traceroutes, and even telnet or ssh to other routers in the lab. This will generate traffic, and the info will persist in Netflow's cache for a few minutes.

55. Top Talkers in Netflow. Each team should also configure a set of top-talkers, to see what the busiest source and destinations are. Try this configuration:

```
ip flow-top-talkers
 top 20
 sort-by bytes
```

This displays the top 20 talkers, sorting them in descending order of bytes transferred.

Try some of the other CLI options available under the ip flow-top-talkers configurations. There are many match options:

```
gw(config-flow-top-talkers)# match ?
  byte-range        Match a range of bytes
  class-map         Match a class
  destination       Match destination criteria
  direction         Match direction
  flow-sampler      Match a flow sampler
  input-interface   Match input interface
```

```
  nexthop-address   Match next hop
  output-interface  Match output interface
  packet-range      Match a range of packets
  protocol          Match protocol
  source            Match source criteria
  tos               Match TOS
```

Try some of these and see what happens to the output.

56. Netflow for IPv6. Cisco IOS used to support IPv6 with standard Netflow. But this was only briefly the case in IOS 12.3 and 12.4. From 12.4T onwards, IPv6 support in Netflow was replaced by Flexible Netflow for IPv6 (it is also available for IPv4).

The configuration syntax for Flexible Netflow is somewhat different and a lot more sophisticated. First off we need to create Flow Monitors for our incoming and outgoing Netflow captures. Here is an example:

```
flow monitor FLOW-MONITOR-V6-IN
 cache timeout active 300
 record netflow ipv6 original-input
!
flow monitor FLOW-MONITOR-V6-OUT
 cache timeout active 300
 record netflow ipv6 original-output
!
```

And then we apply these flow monitors to the interface we want to monitor:

```
interface FastEthernet0/0
 ipv6 flow monitor FLOW-MONITOR-V6-IN input
 ipv6 flow monitor FLOW-MONITOR-V6-OUT output
!
```

57. Top Talkers in Flexible Netflow. The top talkers in the Flexible Netflow configuration is somewhat different – there is no need to create a specific stanza to set up the top talkers as the router can simply display the top talkers from the command line. Here is an example:

```
show flow monitor FLOW-MONITOR-V6-OUT cache aggregate \
ipv6 source address ipv6 destination address sort counter \
bytes top 20
```

This is all one command line and displays the top 20 talkers for outbound traffic, sorting them in descending order of bytes transferred. The command above can be modified to look at the inbound traffic also, by using the inbound flow monitor.

Try some of the other CLI options available under the “show flow monitor” command.

58. Summary. While this exercise has shown how to set up Netflow for both IPv4 and IPv6, it has a more serious aspect. It is possible for a network operator to very simply see what traffic is traversing their network. If it very easy to spot malicious activity, scanning, etc, simply by looking at the flow data and searching for particular signatures (tcp or udp ports, addresses, etc). This makes Netflow a valuable security tool for all network operators, whether they are running an IPv4-only network, or are dual stack IPv4 and IPv6.

Checkpoint #12: *call the lab assistant and demonstrate what you have been able to catch by using Netflow on your router. Show both the IPv4 and IPv6 netflow output.*

Exercise Thirteen – RTBH Filtering

59. Remotely Triggered Black Hole Filtering. RTBH, as it is known as, is a commonly used technique to assist with the mitigation of Distributed Denial of Service Attacks. Remotely triggered blackhole filtering is a technique that provides the ability to drop undesirable traffic at the ingress into the network. RTBH provides a method for quickly dropping undesirable traffic at the edge of the network, based on either source addresses or destination addresses by forwarding it to a null0 interface. A typical deployment scenario for RTBH would require running internal Border Gateway Protocol (iBGP) at the access and aggregation points and configuring a separate device in the network operations centre (NOC) to act as a trigger. For destination-based drops, the triggering device sends iBGP updates to the edge that sets the next-hop of the victim's IP address to the null0 interface. Source-based drops are similar but it relies on the pre-existing deployment of uRPF which drops a packet if its source is "invalid"; invalid includes routes to Null0. Using the same mechanism for destination-based drops, a BGP update is sent, and this update sets the next hop for a source to Null0. Now all traffic entering an interface with uRPF enabled drops traffic from that source.

60. Demonstrating the use of RTBH. This exercise will demonstrate how to configure RTBH, and then will demonstrate its use in dealing with undesired traffic targeted at a particular destination.

To make this work, the team running AS10 will "attack" a target in AS20. The team running AS20 will "attack" a target in AS30. The team running AS30 will "attack" a target in AS40. And the team running AS40 will "attack" a target in AS10.

61. Configuring a Null Route. On all routers in the autonomous system configure a static route for the host address 0100::1 pointing to the null0 interface. (0100::/64 is listed in the IANA registry as a Discard Prefix – it is not routed on the Internet.)

```
ipv6 route 100::1/128 null0
```

62. Trigger Router – Step 1. We will now select one router in your autonomous system to be the trigger router. The following table recommends which router should be the trigger router in each AS:

AS10	**R3**	**AS30**	**R10**
AS20	**R7**	**AS40**	**R14**

The trigger router will announce the address which we want the whole AS to block traffic to. On this router we will configure a route-map called v6blackhole-trigger which will set policy to announce a specifically identified prefix with next-hop pointing to a null interface.

Here is an example for Router7 in AS20. Note that we set community to be no export (which means the prefix will not be announced to any other AS) and we set community to <ASN>:666 to identify this route for our internal policy as being a blackhole route (standard industry practice):

```
route-map v6blackhole-trigger permit 10
 description Look for blackholed routes
 match tag 66
 set ipv6 next-hop 100::1
 set local-preference 200
 set origin igp
 set community no-export
 set community 20:666
!
route-map v6blackhole-trigger deny 20
 description deny everything else – default
!
```

63. **Trigger Router – Step 2.** With the route-map now configured on the trigger router, we now add the route-map into the BGP process on the trigger router so that destinations we want to block will be distributed around our AS by iBGP. For example, for Router 10 in AS30:

```
router bgp 30
 address-family ipv6
  redistribute static route-map v6blackhole-trigger
```

Which will redistribute all static routes configured on the router through the route-map black-hole-trigger. Any static routes matching the conditions in the route-map will have their next-hops set to 100::1 address.

Verify that the iBGP and eBGP sessions in the AS all have "send-community" configuration enabled – Cisco IOS does not send communities by BGP by default so it will have to be activated if not already configured. For example:

```
router bgp 30
 address-family ipv6
  neighbor 2001:dba::1 remote-as 30
  neighbor 2001:dba::1 send-community
  neighbor 2001:dba::1 ...
```

64. **Trigger Router – Step 3.** If the trigger router uses iBGP with neighbour routers, it cannot be configured with "next-hop-self". This is because Cisco IOS will replace the next-hop installed by the redistribute statement's route-map with the IPv6 address of the local router. Which is no good – we want the next hop to be specifically 100::1 for the destination we want to blackhole. This is normally not a problem, as the Trigger Router usually resides in the NOC and won't introduce any prefixes into the network apart from the ones which have to be blackholed. However, it is a problem here, as the routers we are using will all learn routes from their eBGP neighbours.

This means for the Trigger Router we have to remove the *next-hop-self* statements we added to the iBGP at the start of the workshop, and replace them with a route-map which will do the same task, but not change the next-hop of the blackholed routes. Here is an example for Router14:

```
router bgp 40
 address-family ipv6
  no neigh 2001:dbb::1 next-hop-self
  ...etc...
```

Now we need to create a route-map to replace the *next-hop-self* statement, and then apply it to the iBGP neighbours only. The route-map will test for blackhole routes (they have community <ASN>:666 set), and not do anything with them. It will then set the next-hop for all the remaining routes to the loopback address of the local router. Here is an example of what needs to be done for Router10:

```
ip community-list 66 permit 30:666
!
route-map v6nhs permit 10
 match community 66
!
route-map v6nhs permit 20
 set ipv6 next-hop 2001:DBA::3
!
router bgp 30
 address-family ipv6
  neigh 2001:dba::1 route-map v6nhs out
  neigh 2001:dba::2 route-map v6nhs out
!
```

Once the route-map has been applied to the iBGP neighbours, the iBGP sessions should be refreshed outbound – example for AS30:

```
clear bgp ipv6 unicast 30 out
```

And then you should see prefixes learned from the trigger router with appropriate next-hops set. The blackhole routes will have 100::1 as their next-hop, as you will see in the next step.

65. **Testing the RTBH set up – Part 1.** The groundwork for the RTBH is now in place. This will let us use the trigger router as a remote triggering device for dropping packets based on destination address in your network. All we need to do to have our AS null route traffic to a particular destination is to create a static route to this destination with a tag of 66 – this tag is matched by the route-map resulting in the prefix having its next hop address set to 100::1.

    Now that the groundwork is in place, let us test our RTBH set up. To do this we enlist the support of one of our neighbour AS teams. As mentioned earlier, AS20 will ask AS10 to attack their network; AS30 will ask AS20 to attack their network; and so on. Here are suggested target addresses to be used within each AS:

AS10	**2001:DB8::FF**	**AS30**	**2001:DBA::FF**
AS20	**2001:DB9::FF**	**AS40**	**2001:DBB::FF**

    On the respective trigger routers, create a static route to null for the above host addresses, tagging the null route with the number 66. Here is an example for Router 14:

```
ipv6 route 2001:DBB::FF/128 null0 tag 66
```

    Once the null route is in place, verify the entry in the router's routing table. You should see the route to null for the selected trigger address.

66. **Testing the RTBH set up – Part 2.** Now the trigger router has been prepared, check the other routers in the same AS to see what the BGP table and Routing table entries are for the route we want to test RTBH for. Verify that the next hop address for selected trigger address on the router is

100::1 and that you have a routing table entry for 100::1 pointing to the Null interface. If not, please check with your team members in your AS.

67. **Testing the RTBH set up – Part 3.** Now ask your "attacker" AS to send a stream of ping packets to the destination address you are using as the target. The simplest way to do it is if each router in the neighbour AS sets up an extended ping, for example as below (where R1 in AS10 is pinging the target address in AS20):

```
R1#ping ipv6
Target IPv6 address: 2001:DB9::FF
Repeat count [5]: 10000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands? [no]:
Sweep range of sizes? [no]:
Type escape sequence to abort.
Sending 10000, 100-byte ICMP Echos to 2001:DB9::FF, timeout is 2 seconds:
```

Check on the external interfaces to your neighbouring AS (the one which is "attacking" your target address). Do you see the interface counters going up showing there is traffic? Also check the Netflow you configured on your external facing interfaces for the previous exercise – do you see the ICMPv6 packets heading in to the "attacked" destination?

Now check on the trigger router? Do you see any traffic inbound to it from any of the iBGP neighbours?

68. **Conclusion.** If the previous step demonstrates success in "defeating" the "attack", then you will have successfully configured RTBH for your AS. This is a common tool used by many network operators globally, and is considered mandatory by many end-users when searching for connectivity options from their upstream providers.

Checkpoint #13: *call the lab assistant and demonstrate the function RTBH set up.*