CHAPTER 3

Routing Infrastructure

Routing is one of the most important parts of the infrastructure that keeps a network running, and as such, it is absolutely critical to take the necessary measures to secure it. There are different ways routing can be compromised, from the injection of illegitimate updates to DoS specially designed to disrupt routing. Attacks may target the router devices, the peering sessions, and/or the routing information. Fortunately, protocols like BGP, IS-IS, OSPF, EIGRP and RIPv2 provide a set of tools that help secure the routing infrastructure. This section provides the guidelines for using such tools.

The router's primary functions are to learn and propagate route information, and ultimately to forward packets via the most appropriate paths. Successful attacks against routers are those able affect or disrupt one or more of those primary functions by compromising the router itself, its peering sessions, and/or the routing information.

Routers are subject to the same sort of attacks designed to compromise hosts and servers, such as password cracking, privilege escalation, buffer overflows, and even social engineering. Most of the best practices in this document help mitigate and even prevent some of those threats.

Peering relationships are also target of attacks. For most routing protocols routers cannot exchange route information unless they establish a peering relationship, also called neighbor adjacency. Some attacks attempt to break established sessions by sending the router malformed packets, resetting TCP connections, consuming the router resources, etc. Attacks may also prevent neighbor adjacencies from being formed by saturating queues, memory, CPU and other router resources. This section of the document presents a series of best practices to protect neighbor adjacencies from those threats.

Finally, routing can also be compromised by the injection of false route information, and by the modification or removal of legitimate route information. Route information can be injected or altered by many means, ranging from the insertion of individual false route updates to the installation of bogus routers into the routing infrastructure. Potential denial of service conditions may result from intentional loops or black-holes for particular destinations. Attackers may also attempt to redirect traffic along insecure paths to intercept and modify user's data, or simply to circumvent security controls. This section also includes a collection of best practices designed to prevent the compromising of routing information.

CSF Methodology Assessment

The results of applying the CSF methodology are shown inTable 3-1 and Table 3-2. The tables highlight the technologies and features identified for securing the routing plane and which are integrated in Network Secure Baseline.

Total Visibility

Table 3-1 Routing Infrastructure—Total Visibility

Identify	Monitor	Correlate
• Neighbor Authentication • Routing Peer Definition • Route Redistribution Filtering	• Neighbor adjacency logging • Logging – Syslog – SNMP	

Complete Control

Table 3-2 Routing Infrastructure—Complete Control

Harden	Isolate	Enforce
• Default Passive Interface • BGP TTL Security Check • Standby interfaces • Standby devices • Element redundancy • Topological Redundancy		• Prefix Filtering • Maximum Prefix Filtering • Route Redistribution Filtering • Stub Routing • iACL • rACL • Control Plane Protection • Control Plane Policing

Restricted Routing Protocol Membership

Many dynamic routing protocols, particularly interior gateway protocols, implement automatic peer discovery mechanisms that facilitate the deployment and setup of routers. By default, these mechanisms operate under the assumption that all peers are to be trusted, making it possible to establish peering sessions from bogus routers and to inject false routing data. Fortunately, Cisco IOS provides a series of recommended features designed to restrict routing sessions to trusted peers and that help validate the origin and integrity of routing updates:

- Neighbor authentication
- Routing peer definition
- Default passive interface
- BGP TTL Security Check
- iACLs
- rACLs
- Control Plane Policing
- Control Plane Protection

Neighbor Authentication

Neighbor authentication is a feature available on most routing protocols, and that ensures a router only receives reliable routing information and from trusted neighbors. That is achieved by certifying the authenticity of each neighbor and the integrity of its routing updates. Technically, each router is initially configured with a shared secret key that is used to validate each routing update. Before sending a routing update, each router is required to sign it with the predefined secret key; and include the resulting signature as part of the update message. Finally, the update is verified by the receiving neighbor to prove its authenticity and integrity. Neighbor authentication is supported for BGP, IS-IS, OSPF, RIPv2 and EIGRP.

Neighbor authentication helps protect peering sessions from attacks such as session reset attempts and insertion of unauthorized routing peers. Neighbor authentication also helps secure routing data from the injection of false routes, and the removal or modification of legitimate routing information from unauthorized routing peers. It should be noted however that neighbor authentication does not prevent incorrect routing data from being injected by a valid router that has been compromised trusted router. Fortunately, such attack scenarios can be mitigated by route filtering, as explained later in this section.

Most routing protocols support two types of neighbor authentication, plain text and Message Digest Algorithm Version 5 (MD5) authentication. Plain text authentication consists on sending the secret key in the clear inside each routing update message, which does not provide much security since keys can be intercepted while in transit. MD5 authentication works by processing each routing update with a MD5 hash function; and by including the resulting signature (digest) as part of the routing update message. This method is more secure because the actual shared secret key is never sent over the network. For this reason, MD5 authentication should be preferred over clear text.

Enabling neighbor authentication is a recommended practice for all routers, but especially for those more exposed to threats such the routers facing the Internet or other external networks. Ideally, secret keys should be unique to each peering relationship or interface, in the case of broadcast media like Ethernet. However, having all unique passwords may pose an operational challenge in large networks; hence, it is up to the administrators to find the right balance between security and the easy of operation.

The following example shows the configuration of OSPF MD5 neighbor authentication on an IOS router:

```
! OSPF MD5 authentication
interface Ethernet1
  ip address 10.139.20.1 255.255.255.0
  ip ospf message-digest-key 10 md5 oursharedsecret
!
router ospf 20
  network 10.139.20.0 0.0.0.255 area 0
  area 0 authentication message-digest
```

In EIGRP MD5, authentication is enabled at the interface or subinterface level as shown in the following example. Note that once EIGRP MD5 authentication is enabled on an interface or subinterface, the router stops processing routing messages received from that interface or subinterface until the peers are also configured for message authentication. This does interrupt routing communications on your network.

```
! EIGRP authentication
interface Ethernet 1
 ip authentication mode eigrp 10 md5
 ip authentication key-chain eigrp 10 mychain
!
router eigrp 10
 network 10.0.0.0
!
key chain mychain
 key 1
  key-string oursharedsecret
!
```

Similar to EIGRP, in RIP version 2 authentication is enabled at the interface or subinterface level as shown in the following example. Note that once RIPv2 MD5 authentication is enabled on an interface or subinterface, the router stops processing routing messages received from that interface or subinterface until the peers are also configured for message authentication. This does interrupt routing communications on your network.

```
interface ethernet 0
 ip rip authentication key-chain mychain
 ip rip authentication mode md5
!
router rip
 network 10.0.0.0
 version 2
!
key chain mychain
 key 1
 key-string oursharedsecret
!
```

In BGP MD5 neighbor authentication is configured within the routing process and as a neighbor attribute, as shown in the example below. Note that once BGP MD5 authentication is enabled for a peer, no peering session will be established until the peer is also configured for message authentication. This does interrupt routing communications on your network.

```
router bgp 10
  no synchronization
  bgp log-neighbor-changes
  network 64.104.0.0
  neighbor 198.133.219.10 remote-as 10
  neighbor 198.133.219.10 password 7 05080F1C22431F5B4A
!
```

Refer to the *Cisco IOS IP Routing Protocols Configuration Guide* fat the following URL for your IOS Software Release to learn more on how to configure neighbor authentication in your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Routing Peer Definition

The same dynamic peer discovery mechanisms that facilitate the deployment and setup of routers can be used potentially to insert bogus routers into the routing infrastructure. This problem may be prevented by disabling such mechanisms by statically configuring a list of trusted neighbors with known IP addresses. This can be used in conjunction with other routing security features such neighbor authentication and route filtering.

In the case of EIGRP, by default, most hello, update, and query messages are sent in multicast packets on broadcast interfaces like Ethernet. This default behavior can be changed by configuring a static neighbor, after which all routing messages are sent in unicast packets. In addition, after the first static neighbor is configured, the router only accepts EIGRP packets from peers that are explicitly configured with a neighbor statement. Consequently, any messages coming from routers without a corresponding neighbor statement are discarded.

The fact the router discards any messages coming from any routers not expressly configured as neighbors helps prevent the insertion of unauthorized routing peers. At the same time, the change of routing messages from multicast to unicast makes unauthorized interception of routing data more difficult, preventing a potential attacker from learning topological information that could be used to carry out future attacks. It should be noted however that, since static neighbors are recognized by their IP

addresses, there is a possibility of IP address spoofing on these neighbors. With the adequate knowledge, an attacker may spoof the IP address of a valid static neighbor. Neighbor authentication provides a second layer of protection to help mitigate IP spoofing. With neighbor authentication, sessions and updates are only accepted from neighbors that use the proper secret keys. As a result, a spoofing attempt will not succeed as long as the secret keys are unknown to the attacker.

It should be also noted that static neighbors do not prevent incorrect routing data from being injected by a compromised trusted router. Fortunately, such attack scenarios can be mitigated by route filtering, as later explained in this section.

The definition of static neighbors is a recommended practice for all routers, but especially for those at the Internet edge or facing other external networks.

As an example, EIGRP static peers can be defined by using the **neighbor** command. Dynamic peer discovery is disabled as soon as the first static neighbor is configured. The following example configures EIGRP peering sessions with the 192.168.1.1 and 192.168.2.2 neighbors:

```
router eigrp 100
 network 10.0.0.0
 neighbor 10.139.20.1 FastEthernet0/0
```

Note In OSPF the configuration of static neighbors does not prevent other routers in the network from establishing an adjacency. For this reason, the technique here described does not apply for OSPF.

Refer to the *Cisco IOS IP Routing Protocols Configuration Guide* fat the following URL for your IOS Software Release to learn how to configure static peers with your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Default Passive Interface

In large service provider and enterprise networks some routers often have a large number of interfaces, for example, at the WAN edge. A common practice to facilitate the configuration of a routing protocol on such routers is to enable the routing processes on a network range matching all the interfaces. While this technique facilitates the configuration of the routing protocol, enabling routing indiscriminately on all interfaces may increase the chances for the insertion of unauthorized routing peers. To prevent those problems, one can either manually set the **passive-interface** command on the interfaces where router adjacencies are not expected, or set all interfaces as passive by default with the router **passive-interface default** command. The **passive-interface default** command changes the configuration logic to a default passive, therefore interfaces where router adjacencies are expected need to be configured with the **no passive-interface** command. This feature works for all routing protocols that support the **passive-interface** command, and has been tested with all supported Cisco routing protocols.

It should be noted that the effects of the **passive-interface** vary depending on the routing protocol. In RIP and IGRP, the **passive-interface** command stops the router from sending updates on the selected interface, but the router continues listening and processing updates received from neighbors on that interface. In EIGRP and OSPF, the **passive-interface** command prevents neighbor sessions to be established on the selected interface. This stops not only routing updates from being advertised, but it also suppresses incoming routing updates.

In terms of security, using the **passive-interface** command in RIP and IGRP helps control the propagation of routing updates, but it does neither prevent the insertion of unauthorized peers nor the manipulation of incoming routing updates. On the other hand, configuring an interface as passive in EIGRP and OSPF prevents the establishment of peering relationships on that interface, therefore

preventing unauthorized peering sessions as well as the insertion, modification, and/or removal of routing information. At the same time, passive interfaces protect the EIGRP and OSPF routers from routing–based DoS attacks that may arrive from the passive interfaces.

To summarize, using the **passive-interface default** command is a safe practice when enabling EIGRP or OSPF on a range matching multiple interfaces. In the case of RIP or IGRP, the **passive-interface default** command does not protect the router from unauthorized peers or the manipulation of incoming routing updates. For this reason, it is not recommended to enable these protocols on network ranges matching interfaces that are to be passive.

In the following example, all interfaces running OSPF are set to passive, while only interface serial 0 is enabled. This means that neighbor adjacencies may only be formed with peers reachable through interface serial 0:

```
router ospf 100
 passive-interface default
 no passive-interface Serial0
 network 10.139.5.0 0.0.0.255 area 0
 network 10.139.20.0 0.0.0.255 area 4
```

In this example, all interfaces running EIGRP are configured as passive, while the serial 0 interface is enabled:

```
router eigrp 10
 passive-interface default
 no passive-interface Serial0
 network 10.0.0.0
```

In this example, RIP is configured to not send any routing information on all interfaces except via interface serial 0. With this configuration, however, any incoming routing updates from any interface will continue to be received and processed:

```
router rip
 passive-interface default
 no passive-interface Serial0
 network 10.0.0.0
 version 2
```

More information on the **no passive-interface default** command can be found on the following URL:

http://www.cisco.com/en/US/products/sw/iosswrel/ps1830/products_feature_guide09186a008008784e.html

BGP TTL Security Check

TTL Security Check is a security feature that protects BGP peers from multi-hop attacks. This feature is based on the Generalized TTL Security Mechanism (GTSM, RFC 3682), and is currently available for BGP. Work is currently in progress to implement this feature for other routing protocols such as OSPF and EIGRP.

TTL Security Check allows the configuration of a minimum acceptable TTL value for the packets exchanged between two eBGP peers. When enabled, both peering routers transmit all their traffic to each other with a TTL of 255. In addition, routers establish a peering session only if the other eBGP peer sends packets with a TTL equal to or greater than the TTL value configured for the peering session. All packets received with TTL values less than the predefined value are silently discarded.

TTL Security Check prevents routing-based DoS attacks, unauthorized peering and session reset attacks launched from systems not directly connected to the same subnet as the victim routers. Though it should be noted that TTL Security Check does not provide integrity or authentication between BGP peers, and it does neither stop attacks launched from already compromised routers.

Because TTL Security Check provides protection against multi-hop attacks, it is recommended to enable it on all BGP routers, but especially on those in contact with external peers.

In Cisco IOS software, the TTL security check can be enabled per peer with the **neighbor ttl-security** command:

```
Router(config)# router bgp as-number
Router(config-router)# neighbor ip-address ttl-security hops hop-count
```

In this example, TTL security check is enabled for the 10.1.1.1 eBGP neighbor, and which resides two hops away:

```
Router(config)# router bgp 10
Router(config-router)# neighbor 198.133.219.10 ttl-security hops 2
```

For more information about TTL Security Check, refer to the following URL:

http://www.cisco.com/en/US/products/ps6350/products_configuration_guide_chapter09186a008045562 1.html

iACLs

Infrastructure protection access control lists (iACLs), are extended ACLs designed to protect the routing and switching infrastructure by only permitting legitimate routing and management traffic that originates from authorized devices. iACLs may be deployed as the first line of defense against external threats, therefore they may be configured at network ingress points such as the enterprise Internet edge, or the SP peering points. iACLs may be also implemented as a barrier to protect inner network segments from internal threats. In terms of routing, by only allowing routing sessions and traffic from valid peers, iACLs help protect routers from unauthorized access and DoS attacks based on unauthorized protocols and sources. In addition, they protect routing sessions by preventing the establishment of unauthorized sessions, and by reducing the chances for session reset attacks. iACLs however, are not effective mitigating attacks sourced from trusted routers and based on trusted protocols. iACLs are explained in more detail in Chapter 4, “Device Resiliency and Survivability.”

rACLs

Receive ACL (rACL) is a feature available on some Cisco distributed router platforms that allows to filter traffic destined to the central Route Processor (RP) at the line card level, and before it reaches the RP. This protects the router from attacks designed to overwhelm the capacity of the RP. A rACL consists of a standard or an extended ACL that is first created on the RP, and that is then pushed to the line card CPUs. When a packet destined to the RP enters a line card, the line card CPU processes the packet against the rACL, and it forwards the packet to the RP only if permitted by the rACL. rACL is currently supported on Cisco 1200 Series Routers, Cisco 7500 Series Routers and Cisco 10720 Internet Routers.

By enforcing an ACL at the line cards and by blocking unwanted traffic before it reaches the RP, rACLs help mitigate a wide range of DoS attacks based on control plane traffic, as well as unauthorized access attempts. In addition, rACLs protect routing sessions by preventing the establishment of unauthorized sessions and by reducing the chances for session reset attacks. rACLs however, are not effective

mitigating attacks sourced from trusted routers and based on trusted protocols. It should also be noted that rACLs apply to traffic destined to the RP, and does not affect transit traffic. rACLs are explained in more detail in Chapter 4, "Device Resiliency and Survivability."

Control Plane Policing and Protection

Control Plane Policing (CoPP) and Control Plane Protection are security infrastructure features that allow the configuration of QoS policies that rate limit the traffic sent to the RP in Cisco IOS software-based devices. Control Plane Policing (CoPP) works best on modular platforms with distributed processing power such as Cisco 12000 Series Routers and the Catalyst 6500 Series Switches, while Control Plane Protection is best suited for non-distributed, software-only platforms such as Cisco ISR Routers. Both features help protect routers from unauthorized access and DoS attacks, even when they originate from valid sources and for valid protocols. Both features also help protect routing sessions by preventing the establishment of unauthorized sessions, and by reducing the chances for session reset attacks. These features are explained in more detail in Chapter 4, "Device Resiliency and Survivability."

Route Filtering

Route filtering is another import tool to secure the routing infrastructure. Most routing protocols allow the configuration of route filters that prevent specific routes from being propagated throughout the network. In terms of security, these filters are useful because they help ensure that only legitimate networks are advertised; and that networks that are not supposed to be propagated are never advertised, i.e. networks falling within the private address space (RFC 1918) should not be advertised out to the Internet.

Route filtering can be divided in two forms, filtering of routing information exchanged between routing peers, and filtering of the routing information exchanged between routing processes in the same router as a result of redistribution. Both forms of route filtering are covered in this document

Cisco IOS provides a series of features used to control the propagation of routing data:

- Route Maps
- Prefix List
- Distribute Lists
- Peer Prefix Filtering
- Maximum Prefix Filtering
- EIGRP Stub Routing
- Route Redistribution Filtering

Route Maps

Similarly to access control lists (ACLs), route maps are used for a variety of purposes including, but not limited to, packet classification, policy routing, address translation, and route filtering. Like ACLs, route maps allow the definition of criteria matching packets or routes along with enforceable actions. But unlike ACLs that only enforce permit and deny actions, route maps allow the reconfiguration of a wide range of parameters in packets and routes, such as next-hop, community, metric, etc. Overall, route maps

provide greater policy flexibility and granularity than ACLs. Since the interest of this document is on route filtering, this section will focus on the use of route maps for peer route and route distribution filtering.

Route maps are configured with a tag-name used for identification. Route maps may consist of several entries organized in sequence. Each route map entry contains a list of **match** and **set** statements. The **match** statements specify the match criteria, which are the conditions under which a route will be matched for the given entry. The **set** statements specify the set actions, which are the particular filtering actions to be performed if the criteria specified by the **match** statements are met. Similar to ACLs, route map entries are processed sequentially until a match occurs. When a match occurs the **set** actions for that entry are applied. Any route that does not match at least one match clause relating to a route-map command is simply ignored; that is, the route will not be advertised or redistributed.

The following example shows a two-entry route-map configured to control the redistribution of routes from OSPF into EIGRP. As the route-map is inspected sequentially, external type-2 routes will match the first entry. Since this entry has a **deny** action associated it will prevent those routes from being injected into EIGRP. OSPF routes other than external type-2 will match the second entry of the route-map with a **permit** statement, allowing the redistribution of these routes into EIGRP. In addition, routes matching the second statement will be reconfigured with the metric parameters specified in the **set** statement.

```
route-map ospf-to-eigrp deny 10
 match route-type external type-2
route-map ospf-to-eigrp permit 20
 set metric 40000 1000 255 1 1500
!
router eigrp 1
 redistribute ospf 1 route-map ospf-to-eigrp
 default-metric 20000 2000 255 1 1500
```

As shown in the previous example, route maps are used in conjunction with the **redistribute** command to control the redistribution of routing information between routing processes. At the same time, route maps may be used with the **distribute-list** command to filter routing updates between peers. Both uses are explained with detail later in this document.

Prefix List

An **ip prefix-list** is list of prefixes used in BGP to control the propagation of inbound and outbound updates between routing peers. Similar to ACLs, each entry in the prefix-list contains an IP address and a bit mask, and it is associated with a **permit** or **deny** keyword to either permit or deny the prefix based on the matching condition. The IP address defined in each prefix-list entry can be a classful network, a subnet, or a single host route; while the bit mask is entered as a number from 1 to 32. Similarly to ACLs, prefix-lists are processed sequentially until a match is made; likewise an implicit **deny** is applied to routes that do match any prefix-list entry. Prefix lists can be configured to match an exact prefix length or a prefix range. Prefix ranges can be defined by using the **ge** (greater than or equal) and the **le** (less than or equal) keywords.

A prefix list may be applied to filter inbound or outbound updates for a specific peer or for all peers. Updates from or to specific peers can be controlled by using an **ip prefix-list** in conjunction with the **neighbor prefix-list** command. Updates to or from all peers can be filtered by using an **ip prefix-list** in conjunction with the **distribute-list** command. Both cases are covered in detail later in this document.

The following example applies the prefix list named CustomerA to outgoing advertisements to neighbor 198.133.219.10. The prefix-list CustomerA only allows the advertisement of network 192.133.0.0 to that neighbor:

```
ip prefix-list CustomerA permit 64.104.0.0/16
```

```
ip prefix-list CustomerA deny 0.0.0.0/0 le 32
!
router bgp 10
  network 64.104.0.0
  neighbor 198.133.219.10 prefix-list CustomerA out
```

In the following example, a prefix list and distribute list are defined to configure the BGP routing process to advertise only network 64.104.0.0 to all peers.

```
ip prefix-list AllCustomers permit 64.104.0.0/16
ip prefix-list AllCustomers deny 0.0.0.0/0 le 32
!
router bgp 10
  distribute-list prefix AllCustomers out
!
```

Distribute List

A **distribute-list** is a command configured within a routing process that controls what routes are accepted or advertised based on a criteria set by a standard ACL. A **distribute-list** affects all routes being received or advertised by the routing process, including updates between peers as well as routes derived from redistribution. Distribute-lists are unidirectional; the **distribute-list out** command filters outgoing routing updates, while the **distribute-list in** command controls routes received in updates.

In BGP, a **distribute-list** can be associated with not only a standard ACL but also with extended ACL or a **prefix-list**. In addition, BGP distribute-lists can be defined per neighbor with the **neighbor distribute-list** command. Neighbor distribute-lists allow the definition of per-neighbor route policies.

The following example illustrates the use of a distribute-list in EIGRP to control reception of routing updates from all neighbors. In this example, EIGRP process 100 is configured to accept two networks only from any routing neighbor, networks 0.0.0.0 and 10.0.0.0.

```
access-list 1 permit 0.0.0.0
access-list 1 permit 10.0.0.0
!
router eigrp 100
 network 10.0.0.0
 distribute-list 1 in
```

This example illustrates the use of a distribute-list to control the redistribution of routes. In this example, access list 11 allows OSPF to redistribute information learned from RIP only for network 10.139.0.0:

```
router ospf 100
 redistribute rip subnet
 distribute-list 11 out rip
!
access-list 11 permit 10.139.0.0 0.0.255.255
```

Peer Prefix Filtering

Prefix filtering is a practice that helps prevent the injection and propagation of invalid route information from routing peers. False routing information may be injected intentionally or by mistake causing unnecessary traffic redirection, and even triggering DoS conditions on the network infrastructure. Prefix filtering consists in the enforcement of route filters that block the advertisement and reception of unwanted routing updates. This practice also helps reduce the amounts of resources required for generating and processing routing updates.

Prefix filtering is deployed at the edge of routing domains, for instance in an enterprise at the internet edge; and it provides better results when applied to both reception and propagation of routing updates (inbound and outbound updates). In both cases, filtering may be designed to either block unwanted route updates explicitly, or to permit only known valid route prefixes. For example, it is a good practice for SPs to deploy ingress filters at the customer edge to only accept the prefixes assigned or allocated to downstream customers. At the same time, SPs should deploy ingress filters at the peering links to explicitly block common known bogus address spaces and special-use IPv4 addresses.

IGP Prefix Filtering

In Interior Gateway Protocols (IGPs) like EIGRP, IGRP, RIP and OSPF, the exchange of routing information between peers can be controlled with distribute lists. Distribute lists are defined within the routing process and are unidirectional, therefore for a given process two distribute lists can be defined, one for inbound routing updates, and another one for outbound routing updates. This allows for greater policy granularity. In these IGPs, the criteria used by distribute lists to block or permit specific routes is defined via standard ACLs.

Inbound routing updates are filtered by using the **distribute-list in** router configuration mode command:

```
Router(config-router)# distribute-list [[access-list-number | name] | [route-map map-tag]]
in [interface-type | interface-number]
```

- *access list numbe*r or *name*—Number of name of the standard IP access list that defines which networks are to be received and which are to be suppressed in routing updates.
- *map tag*—(OSPF Only) Name of the route map that defines which networks are to be installed in the routing table and which are to be filtered from the routing table. This argument is supported by OSPF only.
- *in*—Applies the access list to inbound routing updates.
- *interface type and number*—Interface type and number on which the access list should be applied to inbound updates. If no interface is specified, the access list will be applied to all inbound updates. The interface type and number arguments can apply if you specify an access list, not a route map.

For a given routing process, it is possible to define one inbound interface-specific distribute-list per interface, and one globally-defined distribute-list. For example, the following combination is possible:

```
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 10.122.139.0 0.0.0.255
router rip
 distribute-list 2 in ethernet 0
 distribute-list 1 in
```

In the example above, the router checks the interface on which the update comes in. If it is Ethernet 0, access-list 2 is applied before putting it in the routing table. If, based on this check, the network is denied, no further checking is done for this network. However, if distribute-list 2 allows the network, then the globally-defined distribute-list 1 is also checked. If both distribute-lists allow the network, it is put in the table.

Outbound routing updates are filtered by using the **distribute-list out** router configuration mode command:

```
distribute-list {access-list-number | access-list-name} out [interface-name |
routing-process | as-number]
```

- *access list number or name*—The list defines which networks are to be sent and which are to be suppressed in routing updates.

- *interface name*— Interface type and number on which the access list should be applied to outbound updates. If no interface is specified, the access list will be applied to all outgoing updates.
- *routing process and autonomous system number*—This applies to redistribution. It indicates the routing process and AS number to which routes are distributed. Route redistribution is controlled by the access list.

In the following example, the router only sends routes pertaining to the 10.122.139.0 subnet out of Ethernet 0, and any updates about networks in the 10.0.0.0 are flooded out to the remaining interfaces, including the 10.122.139.0 subnet.

```
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 10.122.139.0 0.0.0.255
router rip
 distribute-list 2 out ethernet 0
 distribute-list 1 out
```

BGP Prefix Filtering

Similarly to IGPs, distribute lists can be used in BGP to filter the exchange of routing information with all peers. Distribute lists are defined within the BGP routing process and are unidirectional, therefore for a given process two distribute lists can be defined, one for inbound routing updates, and another one for outbound routing updates. Unlike IGPs, in addition to standard ACLs, distribute lists in BGP support extended ACLs and prefix-lists.

Inbound routing updates may be filtered by using the **distribute-list in** router configuration mode command:

distribute-list {acl-number | prefix list-name} **in**

- *access list number*—Standard or extended IP access list that defines which networks are to be received and which are to be suppressed in routing updates.
- *prefix list-name*— The prefix list defines which networks are to be received and which are to be suppressed in routing updates, based upon matching prefixes. IP prefix lists are used to filter based on the bit length of the prefix. An entire network, subnet, supernet, or single host route can be specified.

In the following example, a prefix list and distribute list are defined to configure the BGP routing process to accept traffic from only network 198.133.219.0 and network 64.104.0.0.

```
ip prefix-list RED deny 0.0.0.0/0 le 32
ip prefix-list RED permit 64.104.0.0/16
ip prefix-list RED permit 198.133.219.0/24
router bgp 10
  network 64.104.0.0
  distribute-list prefix RED in
```

Outbound routing updates may be filtered by using the **distribute-list out** router configuration mode command:

distribute-list {acl-number | prefix list-name} **out** [**protocol** process-number | **connected** | **static**]

- *access list number*—Standard or extended IP access defining which networks are to be received and which are to be suppressed in routing updates.
- *prefix list-name*—The prefix list defines which networks are to be received and which are to be suppressed in routing updates, based upon matching prefixes in the prefix list.

- **protocol** *process-number*—This applies in case of redistribution. It specifies the routing protocol to apply the distribution list. BGP, EIGRP, OSPF, and RIP are supported. The process number is entered for all routing protocols, except RIP. The process number is a value from 1 to 65535.
- **connected**—Specifies peers and networks learned through connected routes.
- **static**—Specifies peers and networks learned through static routes.

In the following example, a prefix list and distribute list are defined to configure the BGP routing process to advertise only network 192.133.219.0.

```
ip prefix-list BLUE deny 0.0.0.0/0 le 32
ip prefix-list BLUE permit 192.133.219.0/24
router bgp 10
  distribute-list prefix BLUE out
```

BGP routing updates can also be controlled on a per neighbor basis. BGP provides four mechanisms to do that: AS-path filters (not covered in this document), **neighbor distribute-list**, **neighbor prefix-list** and **neighbor route-map**.

The **neighbor distribute-list** command allows you to control inbound and outbound routing updates on a per neighbor basis. The filtering criteria can be set by using a standard, an extended ACL, or a prefix list. Standard access lists may be used to filter routing updates. However, in the case of route filtering when using classless interdomain routing (CIDR), standard access lists do not provide the level of granularity that is necessary to configure advanced filtering of network addresses and masks, so extended access lists should be used instead. Neighbor distribute-lists are unidirectional, therefore for a given neighbor only two distribute lists can be applied, one per direction. Another characteristic of neighbor distribute-lists is that they can be applied to individual neighbors or on peer groups. When a peer group-name is used, all the members of the peer group will inherit the characteristic configured with this command. Specifying the command for a neighbor overrides the inbound policy that is inherited from the peer group.

```
neighbor {ip-address | peer-group-name} distribute-list {access-list-number |
expanded-list-number | access-list-name| prefix-list-name} {in | out}
```

- *ip-address*—IP address of the neighbor.
- *peer-group-name*—Name of a BGP peer group.
- *access list number* or *name*—Number of a standard, number or name of the extended access list that defines which networks are to be received and which are to be suppressed in routing updates.
- *prefix list name*—Name of a BGP prefix list.
- *in*—Access list is applied to inbound advertisements to that neighbor.
- *out*—Access list is applied to outbound advertisements to that neighbor.

The following router configuration mode example applies list 39 to inbound advertisements from neighbor 198.133.219.10. List 39 permits the advertisement of network 64.104.0.0.

```
access-list 39 permit 64.104.0.0 0.0.255.255
!
router bgp 10
 network 64.104.0.0
 neighbor 198.133.219.10 distribute-list 39 in
```

The **neighbor prefix-list** command allows you to control inbound and outbound routing updates on a per neighbor basis based on the bit length of the prefixes. An entire network, subnet, supernet, or single host route can be specified. Neighbor prefix-lists are unidirectional, therefore for a given neighbor only two distribute lists can be applied, one per direction. Neighbor prefix-lists can also be applied to

individual neighbors or on peer groups. When a peer group-name is used, all the members of the peer group will inherit the characteristic configured with this command. Specifying the command for a neighbor overrides the inbound policy that is inherited from the peer group.

```
neighbor {ip-address | peer-group-name} prefix-list prefix-list-name {in | out}
```

- *ip-address*—IP address of neighbor.
- *peer-group-name*—Name of a BGP peer group.
- *prefix-list-name*—Name of prefix list defining which networks are to be received and which are to be suppressed in routing updates.
- *in*—Filter list is applied to inbound advertisements from that neighbor.
- *out*—Filter list is applied to outbound advertisements to that neighbor.

In the following example, an SP deploys a customer edge prefix filter to only allow the prefix assigned to one customer (64.104.0.0/16).

```
router bgp 101
     neighbor 198.133.219.6  remote-as 10
     neighbor 198.133.219.6 prefix-list customer in
!
ip prefix-list customer permit 64.104.0.0/16
ip prefix-list customer deny 0.0.0.0/0 le 32
```

In BGP it is also possible to control the exchange of routing updates to or from particular neighbors by using route maps. Route maps are a powerful tool that can also be used to modify BGP attributes such as next-hop, communities, local-preference, etc; though such application is beyond the scope of this document.

To control routing updates on a per neighbor basis with route maps, use the **neighbor route-map** command. As the other neighbor filtering mechanisms, neighbor route-maps can be applied in each direction, one to control the reception of routing updates, and another one to filter the advertisement of routing updates. Neighbor route-maps can also be applied to individual neighbors or on peer groups. When a peer group-name is used, all the members of the peer group will inherit the characteristic configured with this command. Specifying the command for a neighbor overrides the inbound policy that is inherited from the peer group.

```
neighbor {ip-address | peer-group-name } route-map map-name {in | out}
```

- *ip-address*—IP address of the neighbor.
- *peer-group-name*—Name of a BGP or multiprotocol BGP peer group.
- *map-name*—Name of a route map that defines which networks are to be received and which are to be suppressed in routing updates. When a route map is specified, only routes that match at least one section of the route map are received or advertised.
- *in*—Applies route map to inbound routes.
- *out*—Applies route map to outbound routes.

In the following, the **route-map localonly** command allows only the locally generated routes to be advertised to neighbor 198.133.219.10. This prevents the risk of the autonomous system becoming a transit AS for Internet traffic.

```
ip as-path access-list 10 permit ^$
route-map localonly permit 10
  match as-path 10
!
router bgp 10
 network 64.104.0.0
 neighbor 198.133.219.10 remote-as 100
```

```
neighbor 198.133.219.10 route-map localonly out
```

See the *Cisco IOS IP Routing Protocols Configuration Guide* at the following URL for your IOS Software Release to learn how to configure prefix filtering with your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Maximum Prefix Filtering

Some routing protocols allow the definition of the maximum number of routes to be accepted from a routing peer. When this feature is enabled on a router and once the limit has been exceeded, by default the router tears down the peering session, clears all routes learnt from the peer, and then places the peer in a penalty state for a time period. After the penalty time period expires, normal peering is reestablished. This capability helps protect the router from attacks based on the injection of large volumes of routes and unintentional configuration mistakes leading to Denial of Service conditions. Setting a Maximum Prefix limit is particularly useful on routers at the border of routing domains.

In BGP and EIGRP, maximum prefix filtering is configured with the **neighbor maximum-prefix** command. OSPF has a similar feature that limits the number of nonself-generated link-state advertisements an OSPF routing process can keep in the OSPF link-state database. In OSPF, this feature is configured with the **max-lsa** command.

While most routing protocols by default respond by tearing down a session when the limit is exceeded, it is highly recommended to initially configure the routers to alarm only. In that case the router, instead of tearing down the session, only sends a log message, and it continues peering with the sender.

To illustrate the concept, the following EIGRP configures the maximum prefix limit for a single peer. The maximum limit is set to 1000 prefixes, and the warning threshold is set to 80 percent. When the maximum prefix limit is exceeded, the session with this peer will be torn down, all routes learned from this peer will be removed from the topology and routing tables, and this peer will be placed in a penalty state for 5 minutes (default penalty value).

```
Router(config)# router eigrp 100
Router(config-router)# neighbor 10.0.0.1 maximum-prefix 1000 80
```

See the *Cisco IOS IP Routing Protocols Configuration Guide* at the following URL for your IOS Software Release to learn how to configure maximum prefix filtering with your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

EIGRP Stub Routing

EIGRP allows the configuration of stub routers in hub and spoke topologies where the spoke routers direct all traffic to a hub distribution router. The stub configuration helps control the propagation of routing information from the spoke router, preventing both the distribution of false routing data and the manipulation of valid routes. The stub router can be configured to permit the propagation of static, connected, and summary routes independently. In the case a bogus router is installed in the stub network, and assuming the bogus router successfully establishes a peering session, the stub router will learn any routing data originated from the bogus router but it will not pass it onto other peers, shielding the rest of the routing infrastructure from the potential injection of faulty routing data.

Configuring EIGRP Stub Routing is a recommendable practice for stub routers that are part of a hub and spoke topology. In practice, EIGRP Stub Routing provides a layer of control on route propagation that helps reinforce the route filtering techniques described earlier in this document. EIGRP Stub Routing is complementary to those techniques and not a replacement.

Note Even though OSPF implements the concept of stub routing, its implementation differs from EIGRP Stub Routing. Configuring an OSPF area as stub does not prevent a router connected to the stub area from accidentally or intentionally injecting invalid routing information.

Stub routing can be configured in EIGRP with the **eigrp stub** command. In the following example, the **eigrp stub** command is issued with the **connected** and **static** keywords to configure the router as a stub that advertises connected and static routes (sending summary routes will not be permitted):

```
router eigrp 100
 network 10.0.0.0
 eigrp stub connected static
```

See the *Cisco IOS IP Routing Protocols Configuration Guide* at the following URL of your IOS Software Release for more information on stub routing.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Route Redistribution Filtering

In networks where route redistribution is required, it is a good practice to strictly control which routes are advertised as well as to limit the maximum number of routes learned from another routing domain. Invalid routes intentionally or unintentionally introduced into a routing domain may be propagated throughout redistribution leading to the bypass of security controls and even creating denial of service conditions. Implementing route redistribution filters helps contain the effects of such conditions. In addition, from a pure routing standpoint, redistribution filters may help prevent a routing process from sending routing updates back to the process from where they have been originated, preventing loops and maintaining network stability.

Route redistribution is configured with the **redistribute** command. Filters can be enforced by either associating a **route-map** to the redistribute command, or by using a **distribute list** in conjunction with the redistribute command.

To illustrate, the following is the syntax of the redistribute command. Note that the redistribute command support more parameters than the ones shown here and that were omitted for simplicity.

```
redistribute protocol [process-id][as-number][route-map map-tag]
```

- *Protocol*—Source protocol from which routes are being redistributed. It can be one of the following keywords: **bgp**, **connected**, **eigrp**, **isis**, **mobile**, **ospf**, **static** [ip], or **rip**.
- Process ID—For the **bgp** or **eigrp** keyword, this is an autonomous system number, which is a 16-bit decimal number.
- *AS-number*: Autonomous system number for the redistributed route.
- *Route-map and map-tag*—Route map that should be interrogated to filter the importation of routes from this source routing protocol to the current routing protocol. If not specified, all routes are redistributed. If this keyword is specified, but no route map tags are listed, no routes will be imported.

When used with the **redistribute** command, a **route-map** serves as an inbound filter, controlling only the importation of routes from the source routing protocol into the current protocol. To control the redistribution of routing updates in both directions either configure a **route-map** in each routing process, or define an outbound filter in the current routing protocol with the **distribute-list out** command.

The following example illustrates the use of **route-map** with the **redistribute** command. In this example, routes are being redistribute between EIGRP and RIP. Route map **rip**-to-**eigrp** prevents the import of network 10.0.0.0/8 into EIGRP. Likewise, route map **eigrp**-to-**rip** prevents the import of network 20.0.0.0/8 into RIP.

```
route-map rip-to-eigrp deny 10
 match ip address 1
route-map rip-to-eigrp permit 20
!
route-map eigrp-to-rip deny 10
 match ip address 2
route-map eigrp-to-rip permit 20
!
router eigrp 100
 network 10.0.0.0
 redistribute rip route-map rip-to-eigrp
!
router rip
 network 20.0.0.0
 redistribute eigrp 1 route-map eigrp-to-rip
!
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 20.0.0.0 0.255.255.255
```

Route redistribution can also be filtered by using **distribute lists** in the same way as explained forPeer Prefix Filtering. Route updates can be filtered simultaneously in both directions by configuring a **distribute-list in** and a **distribute-list out** in the same routing process. The **distribute-list in** controls the import of network routes from all sources into the current routing process, while the **distribute-list out** filters the export of network routes from the current routing process. A **distribute-list out** can be applied to all route advertisements or to those of a particular routing process.

As discussed in Peer Prefix Filtering, page 3-10, in interior gateway protocols the criteria can only be defined with a standard ACL. OSPF however supports route-maps in addition to standard ACLs. BGP distribute lists support extended ACLs and prefix-lists in addition to standard ACLs.

This example shows the same redistribution scenario as the previous example but this time using distribute lists. In this case, the EIGRP process is configured with a distribute list preventing the redistribution of network 10.0.0.0/8 into RIP. Likewise, RIP is configured with a distribute-list that prevents the advertisement of network 20.0.0.0/8 into EIGRP.

```
router eigrp 100
 network 10.0.0.0
 redistribute rip
 distribute-list 10 out rip
!
router rip
 network 20.0.0.0
 redistribute eigrp 1
 distribute-list 20 out eigrp 1
!
access-list 10 deny 10.0.0.0 0.255.255.255
access-list 10 permit 0.0.0.0 255.255.255.255
access-list 20 deny 20.0.0.0 0.255.255.255
access-list 20 permit 0.0.0.0 255.255.255.255
```

Another good practice is to limit the number of prefixes redistributed into a routing process. In OSPF and EIGRP a maximum limit of prefixes can be set by using the **redistribute maximum-prefix** command. When the redistributed maximum-prefix command is configured, and if the number of redistributed routes reaches the maximum value configured, no more routes are redistributed (unless the router is configured to generate a warning message).

In the following OSPF example, a maximum number of 1200 prefixes can be redistributed into OSPF process 1. If the number of prefixes redistributed reaches 80 percent of 1200 (960 prefixes), a warning message is logged. Another warning is logged if the limit is reached, and no more routes are redistributed.

```
router ospf 1
 network 10.0.0.0 0.0.0.255 area 0
 redistribute eigrp 10 subnets
 redistribute maximum-prefix 1200 80
```

See the *Cisco IOS IP Routing Protocols Configuration Guide* at the following URL to learn how to configure route redistribution filtering with your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Logging

Frequent neighbor status changes (up or down) and resets are common symptoms of network connectivity and network stability problems that should be investigated. These symptoms may also indicate ongoing attacks against the routing infrastructure. Logging the status changes of neighbor sessions is a good practice that helps identify such problems and that facilitates troubleshooting. In most routing protocols, status change message logging is enabled by default. When enabled, every time a router session goes down, up, or experiences a reset, the router generates a log message. If syslog is enabled, the message is forwarded to the syslog server; otherwise is kept in the router's internal buffer.

To enable status change message logging in BGP use the **bgp log-neighbor-changes** command, in EIGRP use the **eigrp log-neighbor-changes** command, and in OSPF use the **log-adjacency-changes** command.

The following example logs neighbor changes for BGP in router configuration mode:

```
Router(config)# router bgp 10
Router(config-router)# bgp log-neighbor-changes
```

See the *Cisco IOS IP Routing Protocols Configuration Guide* at the following URL to learn how to configure route redistribution filtering with your routing protocol.

http://www.cisco.com/web/psa/products/tsd_products_support_configure.html

Secure Routing Plane Summary

Table 3-3 summarizes the best practices explained in this document and lists the threat they help mitigate.

Table 3-3 Secure Routing Plane Best Practices Summary

Restricted Routing Protocol Membership	**Router**	**Peering Session**	**Routing Data**
Neighbor authentication		session resets, unauthorized peering	injection of false routes, removal or modification of routes
Routing peer definition		unauthorized peering	interception of routing information
Default passive interface	Denial-of-Service (DoS)	unauthorized peering	
BGP TTL Security Check	DoS	unauthorized peering, session resets	
iACLs	unauthorized access, invalid source/protocol DoS	session resets, unauthorized peering	
rACLs	unauthorized access, invalid source/protocol DoS	session resets, unauthorized peering	
Control Plane Policing	unauthorized access, invalid/valid DoS, saturate router queues	session resets, unauthorized peering	
Control Plane Protection	unauthorized access, invalid/valid DoS, saturate router queues	session resets, unauthorized peering	
Route Filtering	**Router**	**Peering Session**	**Routing Data**
Peer Route Filtering	DoS-based on routing info		injection of false routes
Max Prefix Filtering	DoS-based on routing info		injection of false routes
EIGRP Stub Routing	DoS-based on routing info		injection of false routes, removal or modification of routes
Route Redistribution	DoS-based on routing info		injection of false routes, removal or modification of routes