BGP—Origin AS Validation

The BGP—Origin AS Validation feature helps prevent network administrators from inadvertently advertising routes to networks they do not control. This feature uses a Resource Public Key Infrastructure (RPKI) server to authenticate that certain BGP prefixes originated from an expected autonomous system before the prefixes are allowed to be advertised.

Information About BGP Origin AS Validation

Benefit of BGP—Origin AS Validation

Occasionally network administrators have unintentionally advertised routes to networks that they do not control. This security issue can be avoided by configuring the BGP—Origin AS Validation feature. This feature uses an RPKI server to authenticate certain BGP prefixes as having originated from an expected autonomous system before prefixes are accepted.

How BGP—Origin AS Validation Works

The network administrator must set up a Resource Public Key Infrastructure (RPKI) server, using third-party software. The RPKI server handles the actual authentication of public key certificates. The server is set up so that certain prefixes or prefix ranges are allowed to originate from certain autonomous systems.

The administrator then configures the router to establish a TCP connection to the RPKI server. This is done by configuring the **bgp rpki server** command. Upon such configuration or booting the router, the router opens a TCP connection to the indicated IP address and port number. The router downloads a list of prefixes and permitted origin AS numbers from one or more router/RPKI servers using the RPKI-Router protocol (RTR). Thus, the router obtains information from the server about which autonomous systems are permitted to advertise which routes, that is, from which AS a route may originate.

If the TCP connection attempt fails, the router retries the connection once per minute. In the meantime, BGP will behave without performing origin validation.

After the TCP session between the router and the server is established, the server will normally send to the router incremental updates with new prefixes that have been added to the RPKI database. The router might also query the server every refresh interval. The router will not send a serial query message or reset query message during the interval between when it sends a serial query or reset query and when it receives an End of Data (EOD) message. Serial queries in this interval are stripped, and reset queries in this interval are sent upon receipt of the EOD message.

A prefix or prefix range and the origin-AS corresponding to it are considered an SOVC record. Overlapping prefix ranges are allowed. An SOVC table containing three records might look like this:

10.0.1.0/20-25 AS 3

10.0.1.0/19-24 AS 4

10.0.1.0/23-27 AS 5

When a prefix (network) is received from an external BGP (eBGP) peer, the prefix is initially placed in the Not Found state. It is then examined and marked as Valid, Invalid, or Not Found:

- Valid—Indicates the prefix and AS pair are found in the SOVC table.
- Invalid—Indicates the prefix meets either of the following two conditions: 1. It matches one or more Route Origin Authorizations (ROAs), but there is no matching ROA where the origin AS matches the origin AS on the AS-PATH. 2. It matches the one or more ROAs at the minimum-length specified in the ROA, but for all ROAs where it matches the minimum length, it is longer than the specified maximum length. Origin AS does not matter for condition #2.
- Not Found—Indicates the prefix is not among the valid or invalid prefixes.

By default, a prefix that is marked Invalid is not advertised to any peer, will be withdrawn from the BGP routing table if it was already advertised, and will not be flagged as a bestpath or considered as a candidate for multipath (unless a BGP bestpath command indicates otherwise). Unless a BGP bestpath command is configured indicating otherwise, the bestpath computation prefers Valid prefixes over Not Found prefixes, and both types of prefixes are advertised.

A prefix marked as Valid is installed in the BGP routing table.

By default, a prefix marked as Not Found is installed in the BGP routing table and will only be flagged as a bestpath or considered as a candidate for multipath if there is no Valid alternative (independently of other BGP attributes such as Local Preference or ASPATH).

If more than one RPKI server is configured, the router will connect to all configured servers and download prefix information from all of them. The SOVC table will be made of the union of all the records received from the different servers.

Once the **bgp rpki server** command (or the **neighbor announce rpki state** command) is configured for an address family, the router starts doing RPKI validation for every path in that address family.

Option to Announce RPKI Validation State to Neighbors

You may optionally announce (and receive) the validation state of a prefix to (and from) internal BGP (iBGP) neighbors by using an extended community attribute. This option might be more convenient for some routers than configuring the **bgp rpki server** command, because it saves that router from having to connect to an RPKI server.

The **neighbor announce rpki state** command causes the router to send the RPKI status with the route to its iBGP neighbors in the BGP extended community attribute. The router also receives RPKI status with the

route from its iBGP neighbor. The announcement works in both directions. The extended community attribute announced is:

0x4300 0x0000 (4 bytes indicating state)

The four bytes indicating state are treated as a 32-bit unsigned integer having one of the following values:

- 0—Valid
- 1—Not Found
- 2—Invalid

If the **neighbor announce rpki state** command is configured, upon receiving a route with this extended community attribute attached from an iBGP peer, the router assigns the route the corresponding validation state. If the **neighbor announce rpki state** command is not configured, all prefixes received from an iBGP peer will be marked as Valid, including the prefixes that must have marked as Not Found.

Note This extended community attribute is not sent to eBGP neighbors, even if they are configured to allow sending of this attribute.

The RPKI state extended community follows these additional behaviors:

- The configuration of the **neighbor announce rpki state** command is possible only if the router is configured to send extended communities to that neighbor on that address family.
- The **neighbor announce rpki state** command is completely independent of whether RPKI is configured for the address family.
- Once the **neighbor announce rpki state** command or the **bgp rpki server** command is configured for an address family, the router starts doing RPKI validation for every path in that address family.
- The enabling and disabling of the **neighbor announce rpki state** command causes neighbors to be split into their own update groups based on whether this portion of their configuration is identical.
- If the **neighbor announce rpki state** command is not configured, the router will save the RPKI state received from other routers, but will use it only if at least one other neighbor in the address family is configured with the **neighbor announce rpki state** command or if the topology is otherwise enabled for the use of RPKI.
- If the **neighbor send-community extended** or **neighbor send-community both** command is removed from the configuration, the **neighbor announce rpki state** configuration is also removed.
- When configuring a route reflector (RR), if the RR server receives a network that includes an RPKI state extended community from a client for whom the **neighbor announce rpki state** command is not configured, the RR will reflect the extended community to all its clients that are capable of receiving it.
- If a network has an RPKI state extended community and is received by an RR from a neighbor for which the **neighbor announce rpki state** command is configured, then it will be reflected to all RR clients that are configured to accept extended communities, regardless of whether the **neighbor announce rpki state** command is configured for those other RR clients.
- A **neighbor announce rpki state** command can be used in a peer policy template, and it is inherited.
- If a **neighbor announce rpki state** command is used in a peer policy template, it must be in the same template as the **send-community extended** command. The **neighbor announce rpki state** command

and the **send-community extended** command must come from the same template or be configured for the same neighbor.

Use of the Validation State in BGP Best Path Determination

There are two ways you can modify the default BGP best path selection process when using RPKI validation states:

- You can completely disable the validation of prefixes by the RPKI server and the storage of that validation information. This is done by configuring the **bgp bestpath prefix-validate disable** command. You might want to do this for configuration testing. The router will still connect to the RPKI server and download the validation information, but will not use the information.
- You can allow an invalid prefix to be used as the BGP best path, even if valid prefixes are available. This is the default behavior. The command to allow a BGP best path to be an invalid prefix, as determined by the BGP Origin AS Validation feature, is the **bgp bestpath prefix-validate allow-invalid** command. The prefix validation state will still be assigned to paths, and will still be communicated to iBGP neighbors that have been configured to receive RPKI state information. You can use a route map to set a local preference, metric, or other property based on the validation state.

During BGP best path selection, the default behavior, if neither of the above options is configured, is that the system will prefer prefixes in the following order:

- Those with a validation state of valid.
- Those with a validation state of not found.
- Those with a validation state of invalid (which, by default, will not be installed in the routing table).

These preferences override metric, local preference, and other choices made during the bestpath computation. The standard bestpath decision tree applies only if the validation state of the two paths is the same.

If both commands are configured, the **bgp bestpath prefix-validate disable** command will prevent the validation state from being assigned to paths, so the **bgp bestpath prefix-validate allow-invalid** command will have no effect.

These configurations can be in either router configuration mode or in address family configuration mode for the IPv4 unicast or IPv6 unicast address families.

Use of a Route Map to Customize Treatment of Valid and Invalid Prefixes

You can create a route map to match on any of the RPKI states, and thereby create a custom policy for handling valid or invalid prefixes.

By default, the router overrides all other preferences to reject routes that are in an invalid state. You must explicitly configure the **bgp bestpath prefix-validate allow-invalid** command if you want to use a route map to do something such as permit such prefixes, but with a nondefault local preference.

How to Configure BGP Origin AS Validation

Enabling BGP—Origin AS Validation

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp rpki server tcp** {*ipv4-address* | *ipv6-address*} **port** *port-number* **refresh** *seconds*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**bgp rpki server tcp** {*ipv4-address* | *ipv6-address*} **port** *port-number* **refresh** *seconds* **Example:** `Device(config-router)# bgp rpki server tcp 192.168.2.2 port 1029 refresh 600`	Configures the router to connect to the specified RPKI server and download prefix information at intervals specified by the **refresh** *seconds* keyword and argument.

Announcing the RPKI State to iBGP Neighbors

Perform this task to cause the router to announce the RPKI state with routes to its iBGP neighbors in the BGP extended community attribute and to also receive the RPKI state with routes from iBGP neighbors. This task might be more convenient than configuring the BGP—Origin AS Validation feature on the router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *ipv6-address*} **send-community extended**
5. **neighbor** {*ip-address* | *ipv6-address*} **announce rpki state**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *ipv6-address*} **send-community extended** **Example:** `Device(config-router)# neighbor 192.168.1.2 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 5	**neighbor** {*ip-address* | *ipv6-address*} **announce rpki state** **Example:** `Device(config-router)# neighbor 192.168.1.2 announce rpki state`	Causes the router to send and receive the RPKI state to and from its iBGP neighbor in the BGP extended community attribute.

Disabling the Validation of BGP Prefixes, But Still Downloading RPKI Information

Perform this task if the BGP—Origin AS Validation feature is enabled, but you want to disable the validation of prefixes based on origin AS and disable the storage of validation information. The router will still connect

to the RPKI server and still download the validation information, but the information will not be used in any way. This task is useful for configuration testing.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family** {**ipv4** | **ipv6**} **unicast**
5. **bgp bestpath prefix-validate disable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family** {**ipv4**	**ipv6**} **unicast** **Example:** `Device(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations.
Step 5	**bgp bestpath prefix-validate disable** **Example:** `Device(config-router-af)# bgp bestpath prefix-validate disable`	Disables the validation of prefixes and the storage of validation information.

Allowing Invalid Prefixes as the Best Path

Perform this task if the BGP—Origin AS Validation feature is enabled, and you want to allow invalid prefixes to be used as the best path, even if valid prefixes are available. Thus, you have control over announcing invalid

networks, but preferring them less than valid and not-found prefixes. Also, the downstream peer can modify path attributes based on a route map that matches invalid prefixes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family** {**ipv4** | **ipv6**} **unicast**
5. **bgp bestpath prefix-validate allow-invalid**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family** {**ipv4**	**ipv6**} **unicast** **Example:** `Device(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations.
Step 5	**bgp bestpath prefix-validate allow-invalid** **Example:** `Device(config-router-af)# bgp bestpath prefix-validate allow-invalid`	Allows invalid prefixes to be used as the best path, even if valid prefixes are available.

Configuring a Route Map Based on RPKI States

Perform this task to create a route map based on RPKI states. The route map in this particular task sets a policy for all three RPKI states based on local preference, but other **set** commands can be used to set a policy. This task does not include a command that makes use of this route map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family** {**ipv4** | **ipv6**} **unicast**
5. **bgp bestpath prefix-validate allow-invalid**
6. **exit**
7. **exit**
8. **route-map** *map-tag* {**permit** | **deny**} [*sequence-number*]
9. **match rpki** {**not-found** | **invalid** | **valid**}
10. **set local-preference** *number*
11. **exit**
12. **route-map** *map-tag* {**permit** | **deny**} [*sequence-number*]
13. **match rpki** {**not-found** | **invalid** | **valid**}
14. **set local-preference** *number*
15. **exit**
16. **route-map** *map-tag* {**permit** | **deny**} [*sequence-number*]
17. **match rpki** {**not-found** | **invalid** | **valid**}
18. **set local-preference** *number*
19. **exit**
20. **route-map** *map-tag* {**permit** | **deny**} [*sequence-number*]
21. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 4	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `Device(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations.
Step 5	**bgp bestpath prefix-validate allow-invalid** **Example:** `Device(config-router-af)# bgp bestpath prefix-validate allow-invalid`	Allows invalid prefixes to be used as the best path, even if valid prefixes are available. • This command is necessary to allow invalid prefixes, which are part of the example route map in Step 16.
Step 6	**exit** **Example:** `Device(config-router-af)# exit`	Exits a configuration mode to the next highest mode in the CLI mode hierarchy.
Step 7	**exit** **Example:** `Device(config-router)# exit`	Exits a configuration mode to the next highest mode in the CLI mode hierarchy.
Step 8	**route-map** *map-tag* {**permit** | **deny**} [*sequence-number*] **Example:** `Device(config)# route-map ROUTE-MAP-NAME-1 permit 10`	Enters route map configuration mode and creates a route map that will permit routes that are allowed by the match clauses that follow.
Step 9	**match rpki** {**not-found** | **invalid** | **valid**} **Example:** `Device(config-route-map)# match rpki valid`	Creates a match clause to permit prefixes with the specified RPKI state. • This example matches on the RPKI state of valid.
Step 10	**set local-preference** *number* **Example:** `Device(config-route-map)# set local-preference 200`	Creates a set clause to set matched prefixes to a local preference of 200.
Step 11	**exit** **Example:** `Device(config-route-map)# exit`	Exits a configuration mode to the next highest mode in the CLI mode hierarchy.
Step 12	**route-map** *map-tag* {**permit** | **deny**} [*sequence-number*] **Example:** `Device(config)# route-map ROUTE-MAP-NAME-1 permit 20`	Continues in the same route map, but a later sequence number, and enters route map configuration mode.

	Command or Action	Purpose
Step 13	**match rpki** {**not-found** | **invalid** | **valid**} **Example:** `Device(config-route-map)# match rpki not-found`	Creates a match clause to permit prefixes with the specified RPKI state. • This example matches on the RPKI state of not found.
Step 14	**set local-preference** *number* **Example:** `Device(config-route-map)# set local-preference 100`	Sets the local preference of prefixes with the RPKI state of not found to 100.
Step 15	**exit** **Example:** `Device(config-route-map)# exit`	Exits a configuration mode to the next highest mode in the CLI mode hierarchy.
Step 16	**route-map** *map-tag* {**permit** | **deny**} [*sequence-number*] **Example:** `Device(config)# route-map ROUTE-MAP-NAME-1 permit 30`	Continues in the same route map, but a later sequence number, and enters route map configuration mode.
Step 17	**match rpki** {**not-found** | **invalid** | **valid**} **Example:** `Device(config-route-map)# match rpki invalid`	Creates a match clause to permit prefixes with the specified RPKI state. • This example matches on the RPKI state of invalid.
Step 18	**set local-preference** *number* **Example:** `Device(config-route-map)# set local-preference 50`	Sets the local preference of prefixes with the RPKI state of invalid to 50.
Step 19	**exit** **Example:** `Device(config-route-map)# exit`	Exits a configuration mode to the next highest mode in the CLI mode hierarchy.
Step 20	**route-map** *map-tag* {**permit** | **deny**} [*sequence-number*] **Example:** `Device(config)# route-map ROUTE-MAP-NAME-1 permit 40`	Continues in the same route map, but a later sequence number, and enters route map configuration mode. • This example permits other routes rather than deny all other routes.
Step 21	**end** **Example:** `Device(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.

Configuration Examples for BGP Origin AS Validation

Example: Configuring BGP to Validate Prefixes Based on Origin AS

In the following example, the router is configured to connect to two RPKI servers, from which it will receive SOVC records of BGP prefixes and AS numbers.

```
router bgp 65000
 no bgp log-neighbor changes
 bgp rpki server tcp 10.0.96.254 port 32001 refresh 600
 bgp rpki server tcp FEC0::1002 port 32002 refresh 600
```

Example: Announcing RPKI State to Neighbors

```
router bgp 65000
 neighbor 10.10.10.10 remote-as 65000
 address-family ipv4 unicast
 neighbor 10.10.10.10 send-community extended
 neighbor 10.10.10.10 announce rpki state
```

Example: Disabling the Checking of Prefixes

The following example, for the IPv4 address family, disables the checking of prefixes to ensure they are valid. It also disables the storage of validation information. However, the router will still connect to the RPKI server and download the validation information. This example is useful for configuration testing.

```
router bgp 65000
 bgp rpki server tcp 10.0.96.254 port 32001 refresh 600
 address-family ipv4 unicast
 bgp bestpath prefix-validate disable
```

Example: Allowing Invalid Prefixes as Best Path

In the following example, for the IPv6 address family, invalid prefixes are allowed to be used as the best path, even if valid prefixes are available.

```
 router bgp 65000
  bgp rpki server tcp FEC0::1002 port 32002 refresh 600
  address-family ipv6 unicast
  bgp bestpath prefix-validate allow-invalid
```

Example: Using a Route Map Based on RPKI State

In the following example, a route map named rtmap-PEX1-3 sets a local preference of 50 for invalid prefix/AS pairs, 100 for not-found prefix/AS pairs, and 200 for valid prefix/AS pairs. The local preference values are set for incoming routes from the neighbor at 10.0.102.1. The neighbor at 10.0.102.1 is an eBGP peer. Note that the **bgp bestpath prefix-validate allow-invalid** command is required in order to permit invalid prefixes.

```
router bgp 65000
 address-family ipv4 unicast
 neighbor 10.0.102.1 route-map rtmap-PEX1-3 in
 bgp bestpath prefix-validate allow-invalid
!
route-map rtmap-PEX1-3 permit 10
 match rpki invalid
 set local-preference 50
!
route-map rtmap-PEX1-3 permit 20
 match rpki not-found
 set local-preference 100
!
route-map rtmap-PEX1-3 permit 30
 match rpki valid
 set local-preference 200
!
route-map rtmap-PEX1-3 permit 40
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands	Cisco IOS IP Routing: BGP Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for eiBGP Multipath for Non-VRF Interfaces (IPv4/IPv6)

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for eiBGP Multipath for Non-VRF Interfaces (IPv4/IPv6)

Feature Name	Releases	Feature Information
eiBGP Multipath for Non-VRF Interfaces (IPv4/IPv6)		The eiBGP Multipath for Non-VRF Interfaces (IPv4/IPv6) feature allows you to configure multipath load sharing among native IPv4 and IPv6 external Border Gateway Protocol (eBGP) and internal BGP (iBGP) paths for improved load balancing in deployments. The following command was modified: **maximum-paths eibgp**.