NATIONAL SECURITY AGENCY
CYBERSECURITY REPORT

A GUIDE TO BORDER GATEWAY PROTOCOL (BGP) BEST PRACTICES

A TECHNICAL REPORT FROM NETWORK SYSTEMS ANALYSIS BRANCH

U/OO/202911-18

PP-18-0645

10 September 2018

DOCUMENT CHANGE HISTORY

DATE	VERSION	DESCRIPTION
07/24/18	01	Initial Release
08/24/18	02	Re-templated

DISCLAIMER OF WARRANTIES AND ENDORSEMENT

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government, and shall not be used for advertising or product endorsement purposes.

A Guide to Border Gateway Protocol (BGP) Best Practices

CONTACT INFORMATION

Client Requirements and Inquiries or General Cybersecurity Inquiries

CYBERSECURITY REQUIREMENTS CENTER (CRC)

410-854-4200

Cybersecurity_Requests@nsa.gov

Executive Summary

The dominant routing protocol on the Internet is the Border Gateway Protocol (BGP). BGP has been deployed since the commercialization of the Internet and version 4 of BGP is over a decade old. BGP works well in practice, and its simplicity and resilience enabled it to play a fundamental role within the global Internet. However, BGP inherently provides few performance or security protections.

With BGP being the primary protocol driving the Internet, the security of devices dedicated to running the protocol is vital. Unfortunately, there are many vulnerabilities that can be exploited if proper mitigations are not configured. This error seems to be far more common than it should. For that reason this guidance paper is provided.

Table of Contents

1. BGP Threats

BGP depends on the Transmission Control Protocol (TCP) as its transport protocol and is therefore vulnerable to any TCP-based attacks. There are four major threats against BGP:

- **Denial of Service (DoS)**: A malicious host sends unexpected or unwanted BGP traffic to a BGP neighbor in an attempt to saturate control plane resources, not leaving enough resources to process legitimate BGP traffic on the BGP neighbor.
- **Route Manipulation**: A malicious host modifies the contents of a BGP routing table, diverting sender traffic and preventing it—without the sender's knowledge--from reaching its intended destination. An example of this type of attack would be when Pakistani Internet Service Providers (ISPs), instructed by the government, injected a BGP route for YouTube®[1] that routed the users' traffic to nowhere. Neighboring ISPs propagated the manipulated route across the Internet, rendering YouTube inaccessible until the route was removed. (Alaettinoglu, 2015).
- **Route Hijacking**: A rogue BGP neighbor maliciously advertises a victim's networks to redirect some or all of victim's traffic to itself. In 2014, Turkish Internet Service Providers hijacked Google®[2]'s DNS servers, blocking social media sites from Turkish citizens (Alaettinoglu, 2015).
- **Misconfiguration** (non-malicious): An unintentionally misconfigured BGP router could affect the Internet's BGP routing table, possibly leading to network outages and, worse, unauthorized access to the network traffic.

2. Securing BGP

Many configurable mitigations exist to secure BGP connections between BGP neighbors. Note that none of these mitigations alone will protect the BGP service. In order to mitigate the common threats mentioned above, most, if not all mitigations should be implemented. These mitigation methods include using access control lists (ACL) to only accept traffic from legitimate or known BGP neighbors, rate-limiting the flow of traffic to the router control plane to prevent the router resources from being overwhelmed by DoS attacks, validation and filtering of exchanged routing information, authentication amongst BGP neighbors to ensure the neighbors are authentic, and enabling logging to monitor BGP neighbor activities such as unauthorized changes in the event of an attack to the router.

For more details on each mitigation, please see the sub sections below.

2.1. ENABLING ACCESS CONTROL LISTS (ACL)

ACLs on routers prevent undesired or malicious traffic from reaching the router in the first place, thereby blocking unwanted traffic and minimizing the risk to a router. Properly administered ACLs specifically permit only legitimate BGP traffic from authorized BGP neighbors. Once created, the ACLs must be applied to the interface that connected to the trusted BGP neighbors for inbound traffic filtering

[1] YouTube is a registered trademark of Google, Inc.

[2] Google is a registered trademark of Google, Inc.

.

Below are the Cisco® IOS®3 command syntax to create an access control list on a router that only permits BGP traffic from a known BGP neighbor:

```
Router(config)# access-list access-list-number permit tcp host {trusted-BGP-neighbor-addr} host {local-bgp-addr}
name eq bgp
Router(config)# access-list access-list-number permit tcp host {trusted-BGP-neighbor-addr} eq bgp host {local-
bgp-addr} name
Router(config)# access-list access-list-number deny tcp any any eq bgp
Router(config)# access-list access-list-number deny tcp any eq bgp any
```

When creating an ACL to prevent unauthorized BGP traffic it can be broken into two logical sections. The first section will be a list of rules pertaining to permitting traffic from known BGP neighbors, and the second section denies all other BGP traffic. Since a BGP session can be initiated by either neighbor, each permitted neighbor will require two lines in the ACL. The first line in the example above allows a trusted neighbor to initiate a BGP session. The second line allows for the return traffic from a BGP session established by the host. After permitting each trusted host, a deny statement for all other BGP traffic must be applied. These rules can be inserted before or after any existing ACL rules for the given interface.

Below is an example of how one would create and apply these ACLs to the host 10.1.1.1 with remote neighbor 10.1.1.2. In this example the existing security policy allows all traffic through interface FastEthernet 0/0 and the goal is to filter unknown BGP connections:

```
Router(config)# access-list 101 permit tcp host 10.1.1.2 host 10.1.1.1 eq bgp
Router(config)# access-list 101 permit tcp host 10.1.1.2 eq bgp host 10.1.1.1
Router(config)# access-list 101 deny tcp any any eq bgp
Router(config)# access-list 101 deny tcp any eq bgp any
Router(config)# access-list 101 permit any any
Router(config)# interface FastEthernet 0/0
Router(if-config)# ip access-group 101 in
```

In Juniper an ACL is also known as a filter policy. Each policy is made up of a series of rules that contain matching criteria and the desired action. Most filter policies will consist of several rules. Similar to the Cisco example, the first term in the example below allows a trusted neighbor to initiate a BGP session. The second term allows for the return traffic from a BGP session established by the host. After permitting each trusted host, a deny statement for all other BGP traffic is needed. The last term allows for other non-BGP traffic to pass. The ACL is then applied to an interface. Below is an example of a rule that goes into a filter policy:

```
user@Router# edit firewall family inet filter [FilterName]
user@Router# set term [TermName1] from source-address [local-bgp-addr]
user@Router# set term [TermName1] from destination-address [trusted-BGP-neighbor-addr]
user@Router# set term [TermName1] from destination-port [port/protocol]
user@Router# set term [TermName1] then accept

user@Router# set term [TermName2] from source-address [local-bgp-addr]
user@Router# set term [TermName2] from destination-address [trusted-BGP-neighbor-addr]
user@Router# set term [TermName2] from source-port [port/protocol]
```

[3] Cisco IOS is a registered trademark of Cisco Systems, Inc.

```
user@Router# set term [TermName2] then accept

user@Router# set term [TermName3] from destination-port [port/protocol]
user@Router# set term [TermName3] then reject

user@Router# set term [TermName4] then accept

user@Router# edit interface fe-0/1/0 unit 0 inet
user@Router# set filter input/output [FilterName]
```

Below is an example of how to configure a Juniper device with a filter policy that permits BGP traffic between the host 10.1.1.1 and neighbor 10.1.1.2 and denies all other BGP traffic.

```
user@Router# edit firewall family inet filter bgp_acl
user@Router# set term 1 from source-address 10.1.1.2
user@Router# set term 1 from destination-address 10.1.1.1
user@Router# set term 1 from destination-port bgp
user@Router# set term 1 then accept

user@Router# set term 2 from source-address 10.1.1.2
user@Router# set term 2 from destination-address 10.1.1.1
user@Router# set term 2 from source-port bgp
user@Router# set term 2 then accept

user@Router# set term 3 from destination-port bgp
user@Router# set term 3 then reject

user@Router# set term 4 then accept

user@Router# edit interface fe-0/1/0 unit 0 inet
user@Router# set filter input bgp_acl
```

2.2. ENABLING CONTROL PLANE POLICING (COPP)

Control Plane Policing (CoPP) is a security feature that protects the router's control plane from attacks such as DoS. CoPP allows the administrator to manage and rate-limit the flow of traffic to the control plane. CoPP helps prevent malicious or unnecessary traffic from overwhelming the route processor, which would impact the overall performance. CoPP protects the router processor by treating the route processor resources as distinct objects with their own input interface. This enables administrators to create and apply CoPP Quality of Service (QoS) Policies to traffic destined for the control plane, permitting desired traffic and dropping traffic that is unwanted.

Below is the Cisco IOS syntax to configure CoPP to protect against various BGP attacks like DoS. There are four basic steps to configuring CoPP. The first step is to create an ACL that will match on the desired traffic. Next is the class-map, which will utilize the ACL to tag the traffic with a specific QoS label, or class. When building a CoPP policy it is possible to have several of these class-maps, depending on the level of granularity desired. Once all class-maps are created the next step is to apply policy to each class, which is known as the policy-map. There are several options on the actions or policy that can be applied to each class, but for the purposes of this paper only police and drop will be shown. Last of all is to apply the policy map to the control plane.

```
Router(config)# access-list access-list-number permit protocol {tcp|udp} {any | host {source-addr|name} eq
port number {any|host {source-addr} name}} eq port number
```

```
Router(config)# class-map {match-all|match-any} class-map-name
Router(config-cmap)# match access-group access-list-index
Router(config-cmap)# exit
Router(config)# policy-map policy-map-name
Router(config-cmap-)# class class-map-name
Router(config-cmap-c)# police pps [burst-normal] [burst-max] conform-action action exceed-action action
[violate-action action]
Router(config-cmap-c-police)# exit
Router(config-cmap-)# exit
Router(config)# control plane
Router(config-cp)# service-policy {input|output} policy-map-name
```

The following example shows the steps required to rate limit all BGP traffic from the trusted neighbor 10.1.1.3, and drop all other BGP traffic destined for the control plane. The ACL entries that match the BGP packets to and from 10.1.1.3 with the permit action will result in a match to the “BGP-Policy” and hence the packets will be policed (rate-limited) and allowed through by the CoPP as long as the number of packets fall below the defined threshold (4000 bps in this case). Note: the rate limiting parameters are used as an example only and depending on user requirements or hardware constraints should be adjusted to each device appropriately.

```
Router(config)# access-list 101 permit tcp host 10.1.1.3 any eq bgp
Router(config)# access-list 101 permit tcp host 10.1.1.3 eq bgp any

Router(config)# access-list 102 deny tcp host 10.1.1.3 any eq bgp
Router(config)# access-list 102 deny tcp host 10.1.1.3 eq bgp any
Router(config)# access-list 102 permit tcp any any eq bgp
Router(config)# access-list 102 permit tcp any eq bgp any

Router(config)# class-map match-all COPP-BGP
Router(config-cmap)# match access-group 101
Router(config-cmap)# exit
Router(config)# class-map match-all DROP-BGP
Router(config-cmap)# match access-group 102
Router(config-cmap)# exit
Router(config)# policy-map BGP-Policy
Router(config-cmap-)# class COPP-BGP
```

```
Router(config-cmap-c)# police 4000 1500 conform-action transmit exceed-action drop
Router(config-cmap-c-police)# exit
Router(config-cmap-)# class DROP-BGP
Router(config-cmap-c)# drop
Router(config-cmap-)# exit
Router(config)# control plane
Router(config-cp)# service-policy input BGP-Policy
```

Below is the syntax for Juniper routers. One major difference between Cisco and Juniper is that Juniper utilizes the Loopback 0 interface as the control plane interface. Configuring CoPP in Juniper requires three basic steps. The first step is to create a firewall policer policy that defines the actions to be taken on a packet that is sent to the policer. The second step is to create a firewall filter. The filter consists of multiple term statements that determine which packets will be sent to the policer and which policer policy will be used. The last step is to apply the firewall filter policy to Loopback 0.

```
user@Router# set firewall policer [policy-name] filter-specific
user@Router# set firewall policer [policy-name] if-exceeding bandwidth-limit [bits-per-second-bandwidth-value] burst-side-limit [bust-value]
user@Router# set firewall policer [policy-name] then discard
user@Router# set firewall filter [filter-name] term [rule-name] from protocol [protocol-name] destination-port [Port/Protocol]
user@Router# set firewall filter [filter-name] term [rule-name] from source-address [ip-address]
user@Router# set firewall filter [filter-name] term [rule-name] then policer [policy-name]
user@Router# set interfaces [interface-name] unit [interface-unit-number] family inet filter input [filter-name]
```

```
user@Router# set firewall policer BGP-Policy filter-specific
user@Router# set firewall policer BGP-Policy if-exceeding bandwidth-limit 40000 burst-side-limit 1500
user@Router# set firewall policer BGP-Policy then discard
user@Router# set firewall filter CoPP-Filter term BGP from protocol tcp destination-port bgp
user@Router# set firewall filter CoPP-Filter term BGP from protocol tcp source-port bgp
user@Router# set firewall filter CoPP-Filter term BGP from source-address 10.1.1.3/32
user@Router# set firewall filter CoPP-Filter term BGP then policer BGP-Policy
user@Router# set interface lo0 unit 0 family inet filter input CoPP-Filter
```

2.3. ENABLING THE MAXIMUM BGP PREFIX

The BGP maximum prefix feature allows administrators to control the number of prefixes received from a neighbor. Limiting the number of BGP prefixes learned from a specific neighbor will prevent a single neighbor from exhausting router resources. When configured, this feature terminates a neighbor relationship when the number of received prefixes from the neighbor exceeds the configured maximum prefix limit.

The neighbor maximum-prefix command specifies the maximum number of prefixes that a router accepts from its BGP neighbors before it terminates the BGP session.

Below is the Cisco IOS syntax for configuring the maximum prefixes from specific neighbors, or a group of neighbors:

```
Router(config-router)# neighbor {ip-address | neighbor-group-name} maximum-prefix maximum [threshold]
[restart restart-interval] [warning-only]
```

The following example sets the maximum prefixes for the neighbor at 10.1.1.3 to 10 prefixes. The threshold is set to 80 which indicates 80% of the maximum. This causes a warning message to be generated once 8 prefixes are received (80%). On the 11th prefix, the BGP session terminates and restarts 60 minutes later. Again, these parameters are used as an example only and not recommended values.

```
Router(config-router)# neighbor 10.1.1.3 maximum-prefix 10 80 restart 60
```

Below is the Juniper syntax for configuring the maximum prefixes from specific neighbors, or a group of neighbors and an example:

```
user@Router# edit routing-options
user@Router# set maximum-prefixes [limit] threshold [percent-limit-warning]

user@Router# edit routing-options
user@Router# set maximum-prefixes 10 threshold 80
```

2.4. ENABLING BGP PREFIXES FILTERING WITH PREFIX LISTS

Prefix filtering allows a network administrator to permit or deny specific prefixes for each BGP neighbor, preventing BGP from inadvertently adding unwanted or illegitimate routes to the routing table. This configuration ensures that only the correct routes are accepted (inbound) or advertised (outbound). Similar to white listing, prefix filtering can be configured to only permit known or legitimate prefixes and deny others (as defined by the network policy) that are sent to or received from each BGP neighbor. The configured prefix lists should be applied to each BGP neighbor in both inbound and outbound directions.

Prefix lists can specifically allow only those prefixes that are permitted by the routing policy of a network, which is an example of white list-based filtering. If this configuration is not feasible due to the large number of prefixes, a prefix list can specifically block known undesirable prefixes (a technique known as black list filtering).

Below is the syntax for Cisco IOS to configure a prefix-list and apply it to a neighbor. To add additional lines to the prefix-list simply put in a new sequence number. Similar to ACLs, the order of prefix-lists is very important. Since prefix-lists are processed top to bottom and stop processing a route once it matches a single line, white list prefix-list administrators should always put more specific prefixes near the beginning and less specific near the end.

```
Router(config)# ip prefix-list {list-name [seq number] {deny|permit} network/length}
Router(config)# router bgp {AS-number}
Router(config-router)# neighbor {ip-address | neighbor-group-name} prefix-list {prefix-list name} [in|out]
```

The following examples show how a white listing or black listing approach can be implemented for the BGP neighbor at 11.1.1.1. In the white listing example only routes from 1.1.1.0/24 and 1.2.2.0/24 will be accepted. However in the blacklisting example the 1.1.1.0/24 and 1.2.2.0/24 routes will be rejected and all other routes accepted. It should also be noted that in this example the prefix-list is applied in the inbound direction which limits the routes learned from a neighbor, however if the "out" keyword would have been used then these prefix-lists would limit the routes advertised by the router.

White List Filtering:

```
Router(config)# ip prefix-list Ingress-White seq 5 permit 1.1.1.0/24
Router(config)# ip prefix-list Ingress-White seq 10 permit 1.2.2.0/24
Router(config)# ip prefix-list Ingress-White seq 15 deny 0.0.0.0/0 le 32
Router(config)# router bgp 65001
Router(config-router)# neighbor 11.1.1.1 prefix-list Ingress-White in
```

Black List Filtering:

```
Router(config)# ip prefix-list Ingress-Black seq 5 deny 1.1.1.0/24
Router(config)# ip prefix-list Ingress-Black seq 10 deny 1.2.2.0/24
Router(config)# ip prefix-list Ingress-Black seq 15 permit 0.0.0.0/0 le 32
Router(config)# router bgp 65001
Router(config-router)# neighbor 11.1.2.1 prefix-list Ingress-Black in
```

Configuring prefix filtering on Juniper routers is slightly different because Juniper separates the list of prefixes from the action to be taken on the prefix. This is implemented by having a prefix-list object that contains a list of all prefixes to be grouped together. Then a policy-statement is created that can contain multiple term statements. Each term statement will be the permit or deny for the prefix-list objects. After creating the policy-statement it is applied to the desired BGP group in an inbound or outbound direction.

```
user@Router# edit policy-options prefix-list [list-name]
user@Router# set [prefix-address]

user@Router# edit policy-options policy-statement [policy-name] term [term-name]]
user@Router# set from prefix-list [list-name]
user@Router# set then accept/reject

user@Router# edit protocols bgp group [group-name]
user@Router# set type [internal/external]
user@Router# set local-address [local-ip-address]
user@Router# set [import|export] [import/export-policy]
user@Router# set neighbor [neighbor-ip-address]
```

The following two examples perform the same functionality as the example above with Cisco IOS.

White List Filtering:

- Configure prefix list:

```
user@Router# edit policy-options prefix-list Ingress-White
user@Router# set 1.1.1.0/24
```

```
user@Router# set 1.2.2.0/24

user@Router# edit policy-options policy-statement Ingress-White-Policy term 1
user@Router# set from prefix-list Ingress-White
user@Router# set then accept

user@Router# edit policy-options policy-statement Ingress-White-Policy term others
user@Router# set then reject

user@Router# edit protocols bgp group BGP-Neighbors
user@Router# set type external
user@Router# set local-address 10.1.1.1
user@Router# set import Ingress-White-Policy
user@Router# set neighbor 11.1.1.1
```

Black List Filtering:

- Configure prefix list:

```
user@Router# edit policy-options prefix-list Ingress-Black
user@Router# set 1.1.1.0/24
user@Router# set 1.2.2.0/24

user@Router# edit policy-options policy-statement Ingress-Black-Policy term 1
user@Router# set from prefix-list Ingress-Black
user@Router# set then reject

user@Router# edit policy-options policy-statement Ingress-Black-Policy term others
user@Router# set then accept

user@Router# edit protocols bgp group BGP-Neighbors
user@Router# set type external
user@Router# set local-address 10.1.1.1
user@Router# set import Ingress-Black-Policy
user@Router# set neighbor 11.1.2.1
```

2.5. ENABLING BGP PREFIX FILTERING WITH AUTONOMOUS SYSTEM (AS) PATH ACCESS LISTS

BGP Prefix Filtering with Autonomous System (AS) path access lists can also filter illegitimate prefixes inserted to the BGP routing table. AS path access lists allow network administrators to filter inbound and outbound prefixes based on the AS attribute. Used together, a combination of AS Path Access Lists to ensure traffic is taking the desired route through the internet and Prefix Lists to prevent unwanted routes from being learn creates a strong set of BGP route filters.

The first step to configuring an as-path filter on Cisco routers is to create an as-path ACL. This list will use a regular expression to perform as-path matching. After generating the as-path ACL it is linked to the desired BGP neighbor. The syntax for this configuration is as follows:

```
Router(config)# ip as-path access-list number {permit|deny} regexp
Router(config-router)# neighbor ip-address filter-list number {in/out}
```

The following example creates two as-path ACLs and applies them in the inbound and outbound directions to the BGP neighbor 11.1.1.1. The as-path ACL 1 is configured to only accept BGP routes where the AS path is only 1000. The second AS path is designed to be applied in the outbound direction and will only allow routes that are locally originated to be advertised.

Note: regular expression "^" means the start of string", while "$" means end of string, and "^$" together means locally originated routes.

```
Router(config)# ip as-path access-list 1 permit ^1000$
Router(config)# ip as-path access-list 2 permit ^$
Router(config-router)# neighbor 11.1.1.1 filter-list 1 in
Router(config-router)# neighbor 11.1.1.1 filter-list 2 out
```

Juniper routers require the AS path regular expressions be defined separately from the policy. Once the as-path objects are created with the desired regular expressions they are used in the creation of a policy. Each policy can have several terms to accept or reject different AS paths. Finally the policies are linked to a BGP group in either the inbound or outbound direction. Below is the command syntax to perform these steps:

```
user@Router# edit policy-options
user@Router# set as-path [as-path-name] [regular-expression]

user@Router# edit policy-options policy-statement [policy-name] term [term-name]
user@Router# set from as-path [as-path-name]
user@Router# set then accept/reject

user@Router# edit protocols bgp
user@Router# set group [group-name] import [import-policy]
user@Router# set group [group-name] export [export-policy]
```

The following example creates two as-path objects and two policies. The first policy "Match-AS-1000" limits all imported routes to routes with an AS path of only "1000". The second policy "Match-AS-Local" limits all exported routes to routes that are locally originated from the router, where the router's local AS number is 1005:

```
user@Router# edit policy-options
user@Router# set as-path From-AS-1000 "^1000$"
user@Router# set as-path From-AS-Local "1005"

user@Router# edit policy-options policy-statement Match-AS-1000 term 1
user@Router# set from as-path From-AS-1000
user@Router# set then accept

user@Router# edit policy-options policy-statement Match-AS-Local term 1
user@Router# set from as-path From-AS-Local
user@Router# set then accept

user@Router# edit policy-options policy-statement Match-AS-Local term 2
user@Router# set then reject

user@Router# edit protocols bgp
user@Router# set group AS-1000 import Match-AS-1000
user@Router# set group AS-1000 export Match-AS-Local
```

2.6. ENABLING BGP NEIGHBORS AUTHENTICATION

Successful authentication between BGP neighbors proves that the neighbors are legitimate and trusted, verifies communications between those neighbors, and ensures that only routes learned from legitimate neighbors is added to the routing table. Without BGP authentication a malicious user can spoof a neighbor's IP address and establish BGP connections allowing them to alter BGP routing tables.

The authentication password must be enabled on both sides of the peering session. Passwords should conform to the local password policy in terms of length and complexity. Make sure to share this password out-of-band for best security.

Cisco currently only supports MD5 authentication, and therefore only requires a single line of configuration to enable neighbor authentication. When choosing the password string it is important to use a long complex password that is not easily guessable. The syntax for the neighbor authentication command is as follows:

```
Router(config-router)# neighbor ip-address password string
```

Below is a simple configuration example to enable MD5 authentication with the neighbor at 10.1.1.3. After applying this to one side of the connection the BGP session will terminate and fail to establish until both neighbors have the same password configured.

```
Router(config-router)# neighbor 10.1.1.3 password w9jZ<_D2.%YJ]M5N
```

Juniper provides a slightly more advanced configuration for BGP neighbor authentication. Some of the major differences include the ability to select both the authentication algorithm and a key start time. The key start time allows an opportunity for configuration on both sides of a BGP session prior to a switch to new keys at a predefined time. This reduces down time and allows for a session to stay active while configuring BGP authentication. The first step to configuring BGP authentication in Juniper is to define a key-chain object and enter a complex password that will be used to authenticate neighbors. The second step is to link the key-chain to a BGP group, effectively activating BGP authentication. This syntax is as follows:

```
user@Router# edit security authentication-key-chains key-chain [key-chain-name]
user@Router# set key [key-identifier] secret [password]
user@Router# set key [key-identifier] start-time [YYYY-MM-DD.HH:MM)

user@Router# edit protocols bgp group [group-name]
user@Router# set authentication-key-chain [key-chain-name]
user@Router# set authentication-algorithm [md5|hmac-sha-1-96|aes-128-cmac-96]
```

The example below activate BGP authentication for the BGP group Ext.

```
user@Router# edit security authentication-key-chains key-chain BGP-auth
user@Router# set key 0 secret w9jZ<_D2.%YJ]M5N
user@Router# set key 0 start-time 2018-03-14.0000-2400

user@Router# edit protocols bgp group Ext
user@Router# set authentication-key-chain BGP-auth
user@Router# set authentication-algorithm aes-128-cmac-96
```

NOTE: Though MD5 has known security weaknesses, it is the only available authentication mechanism in wide use. Other authentication technologies exist, such as Resource Public Key Infrastructure (RPKI) and Border Gateway Protocol Security (BGPSec), though they are not widely adopted currently by industry. RPKI and BGPSec should be considered when deploying an internal BGP neighbors. When establishing neighbor relationships with external partners or ISPs requesting RPKI and BGPSec is recommended, as they offer stronger authentication.

2.7. ENABLING TIME TO LIVE SECURITY CHECK

The BGP Time to Live (TTL) security check protects the BGP process from DoS attacks and route manipulation attempts. When the security check is enabled, the expected final TTL value is set for incoming packets received from each external BGP neighbors. As an IP packet makes its way to its final destination, its TTL value decrements once for each router hop along its path. The BGP TTL security check then compares the TTL value in the received packet to the expected TTL value for the BGP session, and drops the packet if its TTL value is less than the expected TTL value. If the packet's TTL value is equal to or greater than the expected value, it passes the check and the session establishes.

Cisco Syntax:

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

The next example sets the expected incoming TTL value for a directly connected external BGP neighbor. The hop-count is set at 1 which configures BGP to only accept IP packets with a TTL count in the IP header that is equal or greater to 254 (255 - 1 hop-count = 254). If the external BGP neighbor is more than 1 hop away (which means the TTL is less than the expected TTL value of 254) the router drops those packets.

```
Router(config)# router bgp 65001
Router(config-router)# neighbor 11.1.1.1 ttl-security hops 1
```

Juniper Syntax:

```
user@Router# edit protocols bgp group [group-name]
user@Router# set neighbor [neighbor-ip-address]
user@Router# set multihop ttl [ttl-value]
```

When configuring TTL security on Juniper devices the TTL value of 64 is the set by default. However, value of 1 indicates that the BGP neighbor is 1 hop away. This means that for a directly connected neighbor the value should be configured with a TTL of 1.

```
user@Router# edit protocols bgp group test
user@Router# set neighbor 11.1.1.1
user@Router# set multihop ttl 1
```

2.8. ENABLING LOGGING

As a best practice ensure that logging of BGP neighbor activities is included and enabled on the router. In the event of an attacker targeting the router, logging provides the ability to account for any unauthorized change activities.

For Cisco devices the bgp log-neighbor-changes command enables logging of BGP neighbor status changes (up or down) and resets for troubleshooting network connectivity problems and measuring network stability. Unexpected neighbor resets might indicate high error rates or high packet loss in the network and should be investigated. Use the show logging command to display the log for the BGP neighbor changes. Best practices recommend sending all logging to a syslog server. The following example only provides the command syntax to enable logging stored locally. To configure logging to a syslog server, refer to vendor guidance.

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

On Juniper devices the procedure is very similar to Cisco. The only command that is required is **set log-updown** within the **protocol bgp** configuration.

Juniper example:

```
user@Router# edit protocol bgp
user@Router# set log-updown
```

3. CONCLUSION

BGP inherently does not provide any built-in security mechanisms as it was designed to operate on a trust relationship between the BGP neighbors. Trust that includes the router is properly secured and only sends/advertises legitimate data. However, misconfigurations can occur and with today's threats against BGP such as Denial of Service (DoS), Route Manipulation, and Route Hijacking, protecting the integrity of BGP is essential. Implementing the above configurations is critical to the integrity and reliability of the network.

Finally, though these best practices are widely available and adopted by industry, recent router security technologies such as Resource Public Key Infrastructure (RPKI) and Border Gateway Protocol (BGPSec) should be considered as they become more prevalent and offer stronger authentication.

4. REFERENCES

• Cengiz Alaettinoglu (2015, February) BGP Security: No Quick Fix