Juniper Networking Technologies

DAY ONE: DEPLOYING BGP FLOWSPEC

Protect your network against DDoS attacks! Get into the lab with this hands-on book and learn how the Junos OS supports BGP FlowSpec.

By Justin Ryburn

DAY ONE: DEPLOYING BGP FLOWSPEC

DDoS attacks are becoming increasingly prevalent on public IP networks. They have huge economic impact to the victim as well as the customers who share infrastructure with the victim. This book gives the reader the tools they need to quickly respond and mitigate DDoS attacks using BGP FlowSpec technology.

The Junos OS supports BGP FlowSpec for both IPv4 and VPNv4 route types. These routes are automatically converted into firewall filters to block attacks quickly. And that's just the start of the book's customization of the Junos OS to defend and protect your network's assets from attack.

Be proactive and defend your network before something happens. Get in the lab and learn how to deploy BGP FlowSpec using the Junos OS with this practical and well-written book.

"Day One: Deploying BGP FlowSpec *is a concise primer on identifying DDOS attacks and using BGP FlowSpec on the MX Series to defeat them."*

Alex Latzko, Lead Network Engineer, Server Central

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Understand how DDoS attacks affect their victims.
- Learn legacy approaches to mitigating DDoS attacks.
- Understand how BGP FlowSpec works.
- Design a dynamic DDoS mitigation solution.
- Understand and configure advanced BGP FlowSpec features including policy best practices.
- Verify your configuration using basic troubleshooting commands.

ISBN 978-1941441251

51600

9 781941 441251

Published by Juniper Networks Books

JUNIPER NETWORKS

Junos® OS Fundamentals Series

Day One: Deploying BGP FlowSpec

By Justin Ryburn

JUNIPer NETWORKS®

Published by Juniper Networks Books
Authors: Justin Ryburn
Technical Reviewers: Eddie Parra, Pat Barnes, and Jeff Haas
Editor in Chief: Patrick Ames
Copyeditor and Proofer: Nancy Koerbel
Illustrator: Karen Joice
J-Net Community Manager: Julie Wider

ISBN: 978-1-941441-25-1 (print)
Printed in the USA by Vervante Corporation.
ISBN: 978-1-941441-26-8 (ebook)

Version History: v1, November 2015
2 3 4 5 6 7 8 9 10

About the Author
Justin Ryburn is a Senior Systems Engineer at Juniper Networks. He has 15 years of experience in various operations, engineering, and sales engineering positions with Service Providers and vendors. Justin contributed content to Cyber Forensics (Auerbach Publishing, 2007). He holds an MBA and a MS in IT Management from Webster University in St. Louis, MO as well as numerous industry certifications.

Author's Acknowledgments:
I would like to thank Eddie Parra, Pat Barnes, and Jeff Haas for taking the time to do technical review of this book. Their advice, feedback, and insight helped shape the ideas in my head into something useful for the reader. There are countless others who allowed me to bounce ideas off of them, offered encouragement, and coached me along the way. These contributions were invaluable. I would like to thank Vincent Celindro and Alan Ball for giving me air cover so I could take the time to complete the writing. I would like to thank Editor in Chief, Patrick Ames for many rounds of editing as well as pushing me to complete the project. I would also like to thank copyeditor Nancy Koerbel, and illustrator Karen Joice for their guidance and assistance with the development of this book. Last but certainly not least, I would like to thank my wife, Kati, my daughter Allexa, and my son Jacob. Their encouragement and patience while I took time away from them allowed me to successfully complete this book.

This book is available in a variety of formats at: http://www.juniper.net/dayone.

Welcome to Day One

This book is part of a growing library of *Day One* books, produced and published by Juniper Networks Books.

Day One books were conceived to help you get just the information that you need on day one. The series covers Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

The *Day One* library also includes a slightly larger and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a weeklong seminar.

You can obtain either series in multiple formats:

- Download a free PDF edition at http://www.juniper.net/dayone.
- Get the ebook edition for iPhones and iPads from the iTunes Store. Search for Juniper Networks Books.
- Get the ebook edition for any device that runs the Kindle app (Android, Kindle, iPad, PC, or Mac) by opening your device's Kindle app and going to the Kindle Store. Search for Juniper Networks Books.
- Purchase the paper edition at either Vervante Corporation (www.vervante.com) for between $12-$28, depending on page length.

Audience

This book is intended for large office and Enterprise network administrators and provides field-tested device and server configurations for common network deployment scenarios, as well as brief background information needed to understand and deploy these solutions in your own environment.

What You Need to Know Before Reading This Book

Before reading this book, you should be familiar with the basic administrative functions of the Junos OS, including the ability to work with operational commands and to read, understand, and change configurations. There are several books in the *Day One* library on learning Junos, at www.juniper.net/dayone.

This book makes a few assumptions about you, the reader, that you have intermediate level knowledge of:

- The Junos OS and its command line interface (CLI).
- General BGP protocol usage in medium to large networks.
- General troubleshooting techniques for medium to large networks running the Junos OS.
- The configuration of basic BGP connectivity in the Junos OS, including configuring neighbors and routing policy.
- Basic Junos OS network and system operation.

By Reading This Booklet You'll Be Able To...

- Understand how DDoS attacks affect their victims.
- Learn the legacy approaches to mitigating DDoS attacks.
- Understand how BGP FlowSpec operates.
- Design a dynamic DDoS mitigation solution.
- Understand and configure advanced BGP FlowSpec features including policy best practices.
- Verify your configuration using basic troubleshooting commands.

Chapter 1

Introduction to DDoS

This chapter introduces the reader to the basics of BGP FlowSpec technology. In order to provide a thorough explanation, it explores exactly what a DDoS network attack is and how such an attack can affect its victim. This chapter also reviews the tools network operators have used in the past to mitigate or block these types of attacks, and analyzes the pros and cons of each of those approaches. Finally, this chapter also provides a quick introduction to network DDoS security so the reader can better understand BGP FlowSpec technology.

Let's start by briefly exploring BGP FlowSpec, and then move on to the different types of DDoS attacks you might encounter in your network today.

BGP FlowSpec

BGP FlowSpec is a DDoS mitigation solution that is specified in RFC 5575. The first BGP FlowSpec draft was submitted to IETF on August 14th, 2007, and it was later published in 2009 as RFC 5575 (http://tools.ietf.org/html/rfc5575). The idea behind FlowSpec is to use BGP to advertise detailed information about the attack vector. Using BGP to disseminate this information allows the network operator to reuse the BGP session and route policy they already have in place between the two networks. Figure 1.1 illustrates how this book's main solution works.

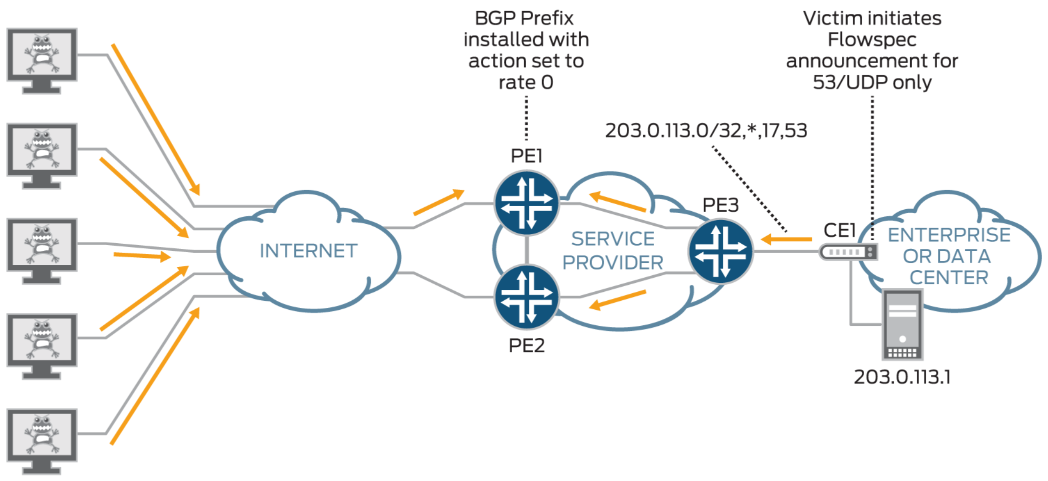
BGP Prefix installed with action set to rate 0
Victim initiates Flowspec announcement for 53/UDP only
203.0.113.0/32,*,17,53
PE1
PE3
PE2
CE1
INTERNET
SERVICE PROVIDER
ENTERPRISE OR DATA CENTER
203.0.113.1

Figure 1.1 DDoS Mitigation Using BGP FlowSpec

In Figure 1.1, the BGP session between the service provider (SP) and the enterprise customer needs to be configured for the BGP FlowSpec address family *before* an attack takes place. When an attack occurs, the enterprise customer advertises the details of the attack to the service provider via a BGP FlowSpec advertisement. This advertisement is automatically converted into a firewall filter on the edge routers in the service provider's network, giving the solution the same granularity as all firewall filters, but with the automation of both Destination Remote Triggered Blackhole (D/RTBH) and Source Remote Triggered Blackhole (S/RTBH). We'll take a look at each of these approaches later in the chapter.

That's the overview of how BGP FlowSpec got to be here. Chapter 2 gets into the details of the technology and its implementation inside the Junos OS. But first, let's review how you might be trying to solve DDoS attacks right now, using other methods and other network designs.

Defining a (D)DoS Attack

The term *Denial of Service* (DoS) is used to describe any type of attack launched to temporarily or permanently deny the legitimate use of services or hosts connected to a network. The term *Distributed* Denial of Service (DDoS) is defined as a type of DoS attack originating from two or more sources targeting a single victim.

DDoS attacks are usually carried out by the use of botnets (a term derived from *robot)* via a command and control server, in which the attacker (the entity who is in command of the command and control server, known as the *botmaster*) takes control of a system through the use of malware. Once the systems have been compromised, the attacker can use them as a *bot,* and to automatically generate attack traffic towards a designated victim. By creating a large number of bots, an attacker is able to gain access to larger pools of computer resources and bandwidth than they could with just a single system.

The motivations for an attacker carrying out a DDoS attack commonly include competition, extortion, activism, political protest, cyberware, support for cybercrime, state supported cyber-warefare, and advanced persistent threats (APTs). Whatever the reasons for an attack may be, the results are usually severe and can include a loss of revenue and brand equity, a loss of productivity, a loss of flexibility and business agility, and additional costs for equipment and staffing to protect against future attacks.

There are a number of ways in which an attacker can carry out an attack. Historically, attackers have used volumetric attacks to flood the network resources with more traffic than that network is designed to handle. However, more recent attacks, called *low and slow,* are attacking the applications being used on the network. These attacks do not require large volumes of traffic so they are more likely to go undetected until the attack uses up so many resources that the application is not able to respond to legitimate requests.

NOTE The focus of this book will be on network attacks not application attacks. However, if a network operator has a DDoS detection tool capable of generating BGP FlowSpec routes for application layer attacks, the same techniques can be used to mitigate both.

MORE? If you are interested in a solution that addresses application attacks, consider Accumuli Security's DDoS Secure Product. Accumuli's DDoS Secure integrates with BGP FlowSpec, allowing for the integration of application attack protection alongside existing volumetic attack protections. For more information on Accumuli's DDoS Secure, check out their datasheet at: https://www.accumuli.com/sites/default/files/resource_files/accumuli_-_ddos_datasheet_v2.1.pdf.

MORE? Need more information on what DDoS attacks are? See Chapter 11 in *The SRX Series*, by Brad Woodberg and Rob Cameron, O'Reilly Media, 2013, at http://www.juniper.net/books.

Figure 1.2 is this *Day One* book's network topology and is used throughout all its chapters.

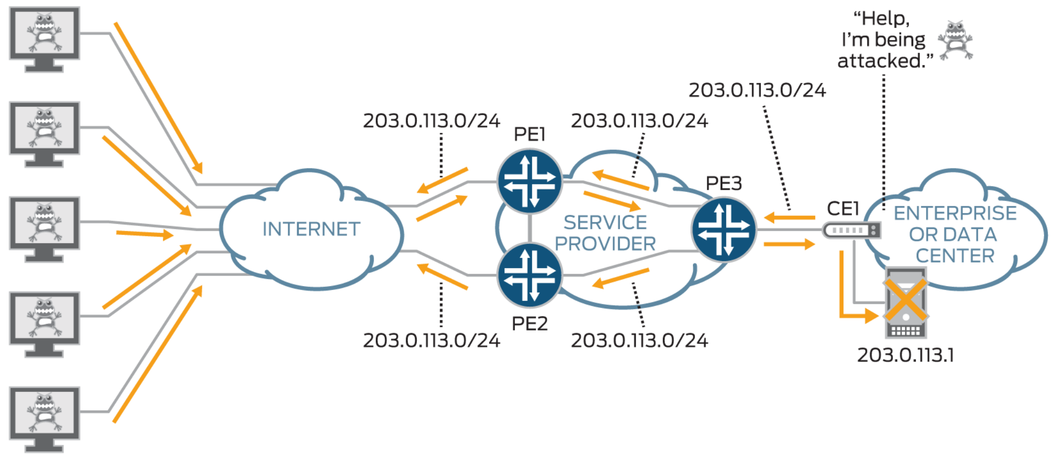

Figure 1.2 This Book's Basic Network Setup

On the right side of Figure 1.2, you can see a server with an IP address of 203.0.113.1 residing in a data center operated by an enterprise company. This company has a single connection to their service provider over which they advertise 203.0.113.0/24 via BGP. The service provider in turn advertises this route out to the remainder of the Internet via BGP.

On the left side of Figure 1.2, you can see the various bots attacking the server. Large service providers have multiple peering points to other service providers and a truly distributed attack would arrive at multiple points, pass through the service provider's core network, and move onto the enterprise company's data center. To keep things simple, let's show a service provider with only two peering points.

The attack is keeping the server from being usable by legitimate customer traffic. What are you going to do?

Legacy Approaches to Mitigating DDoS

In the past, the tools that operators had at their disposal were limited. For those readers who did not experience the ways network operators mitigated DDoS attacks before the BGP FlowSpec technology existed, a short review as to why certain things are done a certain way might be advantageous. Let's quickly review.

Static Discard Routes

One of the easiest and most straightforward ways to deal with a DDoS attack is to create a *discard route* to the destination of the attack. Figure 1.3 illustrates how this is done.

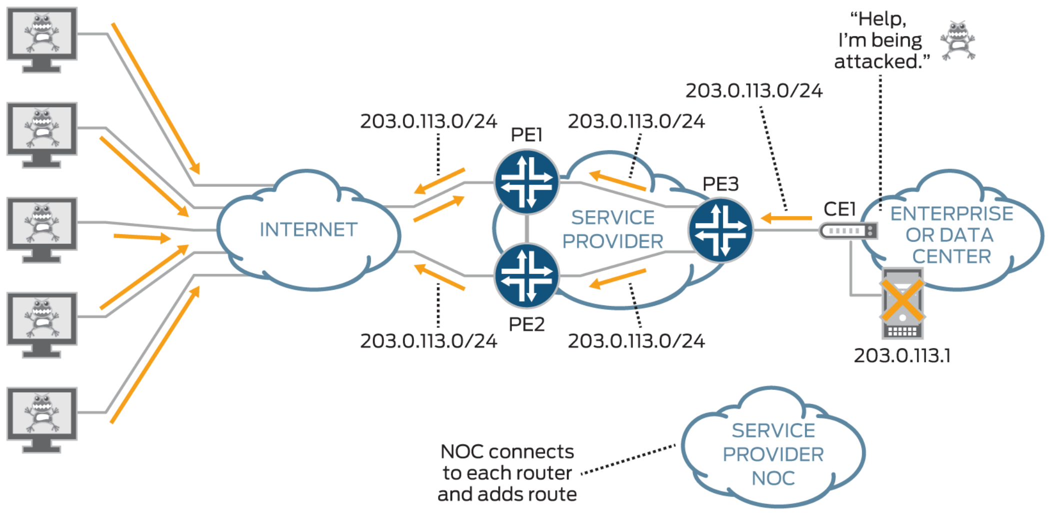

Figure 1.3 DDoS Mitigation Using Static Discard Routes

Once the server is under attack, the enterprise customer has to contact their service provider and request the attack be blocked. The service provider's network operations center (NOC) then installs a static discard route of 203.0.113.1/32 in all of their edge routers:

```
routing-options {
    static {
        route 203.0.113.1/32 discard;
    }
}
```

NOTE Juniper recommends the use of the `discard` Junos OS option over the `reject` option for these routes. The use of `discard` will silently discard the packet while `reject` will cause an ICMP unreachable message to be sent to the originator, creating additional unwanted traffic.

Creating static discard routes requires manual intervention on the part of *both* the enterprise customer and the service provider. The other disadvantage is that it completes the attack because the victim that is under attack is no longer reachable at that destination address. It also requires the service provider to have some procedure for removing these routes once the attack stops or risk the routes being forgotten. However, it does minimize any collateral damage to other customers sharing the same network resources.

NOTE The service provider will have to put the route in each of their edge routers or the attack traffic will be carried through their network until it reaches the edge router containing the discard route. If there are multiple attacks requiring multiple routes, this solution becomes difficult to implement at scale.

Firewall Filters

Another straightforward way to mitigate DDoS attacks is through the use of firewall filters, as illustrated in Figure 1.4.

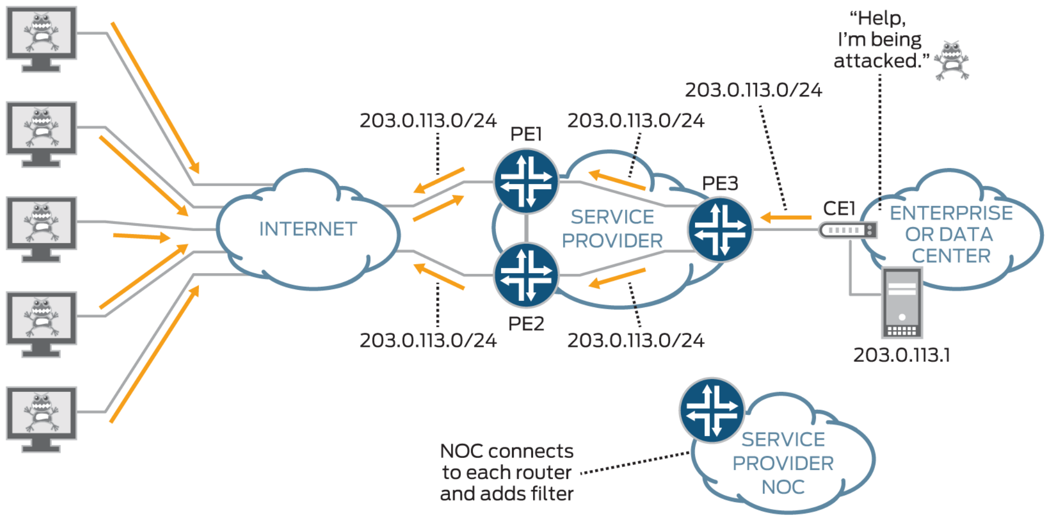

Figure 1.4 DDoS Mitigation Using Firewall Filters

Here again, the enterprise customer must contact the service provider to request the attack be blocked. The service provider NOC then logs in to each edge router and applies the firewall filter to block the attack, (one reason why most of the staff employed by a service provider are very adept at creating firewall filters and applying them). Once again, this solution becomes difficult for the service provider to manage once it starts to scale to multiple attacks across their multiple PE devices.

The use of firewall filters does provide more granularity than one can get with a discard route. For example, if the attack traffic is determined to be UDP Port 53, with packets of 99-200 bytes in size, a filter can be created for just those packets. All other UDP Port 53 packets, as well as the other protocols, are allowed to pass through untouched.

Destination Remote Triggered Blackhole

Destination Remote Triggered Blackhole (D/RTBH) is a DDoS mitigation solution that is specified in RFC 3882 published in 2004: http://tools.ietf.org/html/rfc3882. Figure 1.5 illustrates how this solution works with our standard topology.

In Figure 1.5, the service provider prepares their network by putting a static discard route in each of their routers before an attack takes place. Once an attack is launched, the enterprise customer can advertise a route for the prefix under attack with a BGP community that rewrites the next hop to a pre-staged discard route, allowing D/RTBH to be a more dynamic way to block traffic than the static discard route allowed. However, it has the same effect of completing the attack for the victim, and in addition, it obliges the enterprise customer to be aware of the BGP community required by their service provider.

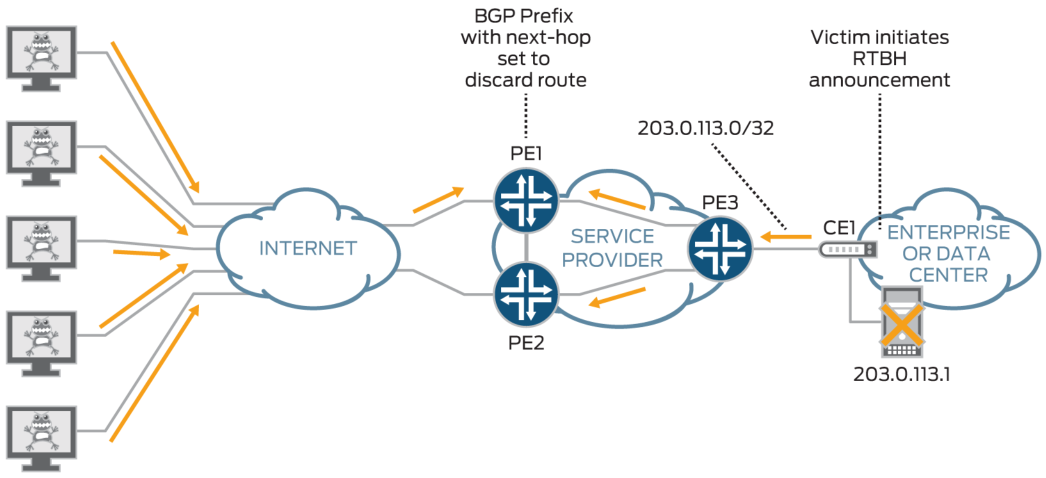
BGP Prefix with next-hop set to discard route
Victim initiates RTBH announcement
203.0.113.0/32
PE1
PE3
CE1
INTERNET
SERVICE PROVIDER
ENTERPRISE OR DATA CENTER
PE2
203.0.113.1

Figure 1.5 DDoS Mitigation Using D/RTBH

Source Remote Triggered Blackhole

The Source Remote Triggered Blackhole (S/RTBH) solution is a DDoS mitigation that was specified in RFC 5635 and published in 2009: http://tools.ietf.org/html/rfc5635. Figure 1.6 illustrates how it works.

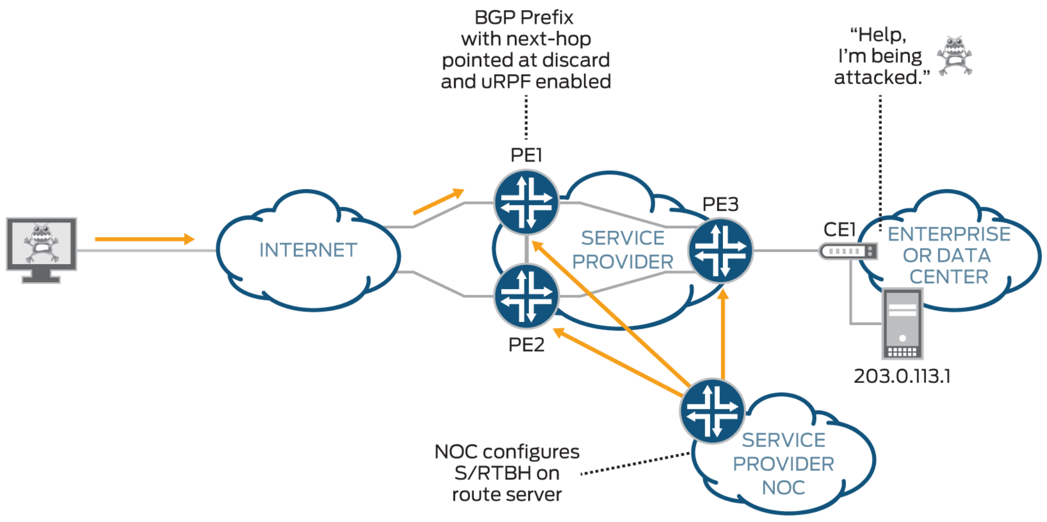
BGP Prefix with next-hop pointed at discard and uRPF enabled
"Help, I'm being attacked."
PE1
PE3
CE1
INTERNET
SERVICE PROVIDER
ENTERPRISE OR DATA CENTER
PE2
203.0.113.1
SERVICE PROVIDER NOC
NOC configures S/RTBH on route server

Figure 1.6 DDoS Mitigation Using S/RTBH

In Figure 1.6, the service provider prepares their network by putting a static discard route in each of their routers as well as unicast Remote Path Forwarding (uRPF) on each of their interfaces before an attack takes place. Once an attack is launched, the enterprise customer contacts the service provider and asks them to block the attack. The service provider NOC then creates a discard route for the source of the attack. This route is advertised from the service provider's router to all of their edge routers and rewrites the next hop to the pre-staged discard route. This allows the attack victim to still be reachable from all sources except the one that is being blocked. However, this method only works for a small number of sources and is not fit for a truly distributed attack.

Summary

That's the short version about DDoS. Did you recognize any of these as your only defense? Do you have your current BGP FlowSpec configured correctly? These and other matters lie ahead of you. Be sure to get your lab ready using this book's sample topology and roll up your sleeves. It's day one and you have a DDoS attack to thwart. ;)

Chapter 2

BGP FlowSpec Details

The IETF defined a procedure to encode traffic flows, for example 5-tuple and Layer 4 traffic properties, into the Network Layer Reachability Information (NLRI) within the existing BGP protocol. The idea is to leverage the considerable power of the BGP protocol and its policy capabilities to disseminate flow specification information (a flow specification is a definition of the parameters of the DDoS attack that needs to be filtered). This chapter explores the details of these new NLRIs as defined in RFC 5575, the BGP FlowSpec standard: http://tools.ietf.org/html/rfc5575.

NLRIs Defined by RFC 5575

The BGP protocol works by exchanging NLRI between BGP-speaking devices, such as routers.

An address family indicator (AFI) and subsequent address family indicator (SAFI) identify the NLRI type within the BGP.

MORE? Need more information on BGP AFI and SAFI? See the IETF's RFC 2858: https://tools.ietf.org/html/rfc2858.

The IETF allocated an AFI/SAFI of 1/133 for Unicast BGP FlowSpec, and an AFI/SAFI of 1/134 for VPN BGP FlowSpec.

The NLRI field in the BGP update message also contains a field type so that different types of information can be passed. Table 2.1 lists the different BGP FlowSpec NLRI component types as defined by the IETF in RFC5575.

Table 2.1 BGP FlowSpec NLRI Types

Type	Information
1	Destination Prefix
2	Source Prefix
3	IP Protocol
4	Source or Destination Port
5	Destination Port
6	Source Port
7	ICMP Type
8	ICMP Code
9	TCP flags
10	Packet length
11	DSCP
12	Fragment Encoding

MORE? A current list of all IETF defined FlowSpec NLRI component types is available at: https://www.iana.org/assignments/flow-spec/flow-spec.xhtml.

Not all the NLRI component types are required in every NLRI advertisement, but the types that are present must be presented in numerical order. A network operator can use any combination of these NLRI types to specify the attack vector they wish to act upon.

Traffic Filtering Actions

The IETF also defined various actions, beyond simple filtering, that can be taken on the traffic that matches the FlowSpec route. This information is carried as an *extended community* added to the route and RFC 5575 defined the actions that can be taken on the traffic, listed in Table 2.2.

Table 2.2 Traffic Filtering Actions

BGP Community	Action
0x8006	traffic-rate
0x8007	traffic-action
0x8008	redirect
0x8009	traffic-marking

The *traffic-rate community* tells the receiving router what rate limiter, in bytes per second, to apply to the traffic. If the desire is to drop/discard all the traffic, a rate of 0 should be used.

The *traffic-action community* is used to sample traffic. This could be useful if the operator wants to sample the attack traffic to get better information about the attack itself.

The *redirect community* is used to redirect the specified traffic into a virtual routing and forwarding (VRF) instance. It uses the same community that would be used for identifying VPNv4 routes. Thus, the router can import the routes in the same manner for both applications. RFC 7674 clarified how the redirect community is used: http://tools.ietf.org/html/rfc7674.

The *traffic-marking community* is used to change the Differentiated Services (DiffServ) code point (DSCP) value in the IP header to the specified value. This could be used by the operator to set the traffic to highest discard probability available in their network. That way they do not drop the traffic until congestion is taking place.

Traffic Filter Rule Ordering

In a large network, it is possible that a particular flow could match more than one BGP FlowSpec route. Thus, it is critical to determine the order the filters should be applied by the devices receiving the BGP FlowSpec advertisements. Ordering them by the sequence in which they were received could cause inconsistent filtering if different devices receive them in different orders. A *deterministic algorithm* to order the filter rules is required. This order is documented in section 5.1 of RFC 5575 and is briefly summarized here.

NOTE The reader should note that by default the Junos OS does not follow the RFC-defined rule ordering. Later in the book we will cover the Junos OS configuration needed to follow the rule ordering presented here.

The order of FlowSpec filter rules is determined by comparing their components. The algorithm starts by comparing the left-most component of each NLRI:

- If the types differ, the lowest type (by numeric value) is used. If they are the same, then the values within that component are compared.
- For IP prefix values (types 1, 2, and 3), the lowest IP is chosen. If the IP addresses are the same, the most specific prefix is used.
- For all other types, the binary string of the contents is compared to determine the order.

Validation Procedure

An important concern with BGP FlowSpec is validating that a BGP-speaking device has the authority to advertise that particular route. If not done, an attacker advertising a BGP FlowSpec route for a prefix that is owned by someone else could carry out a new type of DoS attack.

Just like a unicast BGP route, the first step is to validate that the route is feasible – meaning the receiver of the route checks if it can resolve the next hop. Otherwise, the route is not considered as part of the selection process. The same process is applied for BGP FlowSpec routes.

In addition, the IETF has extended the validation process for BGP FlowSpec. Citing RFC 5575, a flow specification route is only considered feasible if:

> *1. The originator of the flow specification matches the originator of the best-match unicast route for the destination prefix embedded in the flow specification.*
>
> *2. There are no more specific unicast routes, when compared with the flow destination prefix, that have been received from a different neighboring AS than the best-match unicast route, which has been determined in step 1.*

In order to validate the originator, the BGP path attribute is used if it is present for the route. If it isn't present, the transport address of the BGP neighbor is used.

The goal is to verify that the originator of the BGP FlowSpec route is the same as the originator of the BGP unicast route. If that is the case, then the FlowSpec route can be accepted without concern that it would cause a DoS in itself.

MORE? For more exact information on the BGP FlowSpec standard, see: https://tools.ietf.org/html/rfc5575.

Junos OS Implementation of BGP FlowSpec

The Junos OS supports BGP FlowSpec for both IPv4 unicast and VPNv4 unicast route types. For IPv4, routes are placed in a table named *inetflow.0*. Once a route is added to this table, a corresponding filter is installed into the kernel. For VPN routes, the table is named *instance-name.inetflow.0*. Routes from this table are applied as a forwarding table filter against the routing instance's forwarding table.

NOTE The Junos OS has planned support for IPv6. Check with your Juniper account team for actual release dates.

While the current Junos OS implementation supports the current RFC, it was built with the ability to extend into the future in a way that is similar to the way in which the IETF designed BGP FlowSpec to be extensible by the addition of new types.

Juniper has been a leader in the adoption of BGP FlowSpec. In fact, the Junos OS contained support for BGP FlowSpec before the RFC was completely ratified by the IETF. Due to this, the default filter rule ordering in JUNOS does not follow the RFC since it did not exist when the initial implementation was done. Juniper added the following syntax once the RFC was published so that a user can order the terms as required by the RFC:

```
routing-options {
          flow{
            term-order standard;
          }
}
```

IMPORTANT Juniper Networks recommends always using the RFC defined term ordering!

By default, the Junos OS validates the BGP FlowSpec against the unicast routing table as defined by the RFC. However, the user can disable this validation via a `no validate` configuration and apply their own Junos policy to validate that the FlowSpec routes are indeed owned by the BGP peer. Details on how this is accomplished are presented in the upcoming chapters of this book.

MORE? For more information on how to configure BGP FlowSpec in the Junos OS go to the Juniper TechLibrary: https://www.juniper.net/documentation/en_US/junos15.1/topics/example/routing-bgp-flow-specification-routes.html.

Chapter 3

DDoS Detection

Before you can begin to filter a DDoS attack, you must first detect that you are *under* attack and determine what the attack vector looks like. This chapter explores various methods that can be used to detect an attack as well as to determine what the attack packets look like. The goal is for you to use this information to create the Flow-Spec routes to automate the filtering of this traffic.

SNMP Statistics

All modern routers and switches provide the operator with counters for compiling the number of packets per second and bits per second that are flowing through their interfaces. One method for detecting a DDoS attack is to monitor these counters for sudden increases in traffic. If the traffic level spikes, then there is a high probability, although not a guarantee, that an attack is underway. Either way, it alerts the operator to take a look at what caused the increase.

Monitoring these counters manually by logging in to the devices would not be feasible for a network of any size. Therefore, most operators deploy tools that gather these statistics via the Simple Network Management Protocol (SNMP). There are many open source tools that allow you to graph network utilization. Using these graphs, an operator can easily see when an increase has taken place.

MORE? Need an easy-to-use open source SNMP graphing tool? Check out MRTG at http://oss.oetiker.ch/mrtg/ or RRDTool at http://oss.oetiker.ch/rrdtool/.

Analyzing these graphs is a useful way to determine where the attack is coming from. If all of the interfaces are being graphed, you can look for spikes in the incoming packet rate to determine which interface is the source of the traffic.

Most network operators want a more proactive way of detecting an attack. Ideally, they would like to be proactively notified when they have a spike in traffic in their network, so most SNMP collection tools allow you to set thresholds on the counters. Once that threshold is crossed, the program can generate a SNMP trap, an email notification, or even an SMS message that notifies you as soon as the spike in traffic occurs.

One drawback to this approach is that it only tells you there is an increase in traffic, not the nature of the traffic. In order to determine the nature of the attack traffic, you have to log in to a device, apply a logging firewall filter, and then reexamine the logs. Frankly, having to do that manual work somewhat defeats the purpose of having an automated notification tool.

Another drawback to this approach is it only works for volumetric attacks where there is a significant increase in the volume of traffic. If the network operator is suffering from an application layer attack that does not require a large volume, it would go undetected.

Due to these drawbacks, newer DDoS detection tools are based on an analysis of network traffic.

Traffic Analysis

As discussed in Chapter 1, there are limitations to using SNMP statistics to detect DDoS attacks. In order to improve upon this situation, protocols like NetFlow, sFlow, and the IETF IP Flow Information Export protocol (IPFIX) were developed. All of these protocols take a sample of the actual traffic flowing through a router or switch. The header of the sampled traffic is placed in a flow record that is then exported to some kind of monitoring system. The type of information contained in the flow records varies based on the protocol, so let’s take a look at each of these.

NetFlow

The NetFlow protocol was originally designed by Cisco Systems as a way to sample the traffic that is flowing through a router in an IP network and export that information to a collection server for analysis. The three most common versions of NetFlow are version 5, version 8, and version 9. All of these are industry-standard protocols that are supported by a wide variety of vendors.

As of version 5, a flow is a unidirectional sequence of packets defined by the following seven values:

- Incoming interface (SNMP ifIndex)
- Source IP Address
- Destination IP Address
- IP Protocol
- Source Port
- Destination Port
- IP Type of Service

In newer versions of NetFlow, the user can determine the fields they want to define the flow (that will be discussed in more detail in the next section of this chapter).

Once a flow is defined, the router keeps track of the number of packets matching that unique flow. It then creates a flow record once the flow has completed. These flow records are exported via UDP packets sent to the defined flow collector for analysis.

The information available in the flow record varies by version. Some of the more common information includes:

- Input interface
- Output interface
- Flow start and stop timestamp
- Number of packets and bytes in the flow
- L3 header information
- TCP flags
- Source BGP AS
- Destination BGP AS

MORE? For more information on NetFlow, the different versions, what information is included in each version, and which vendors support them, check out NetFlow's Wikipedia page at: https://en.wikipedia.org/wiki/NetFlow.

sFlow

The sFlow protocol was originally developed by a company called InMon who develops network traffic monitoring solutions. They designed sFlow to be a lighter weight alternative to the NetFlow protocol, and sFlow is an industry standard protocol that is supported by many different networking vendors. It is defined in the IETF RFC 3176 (https://tools.ietf.org/html/rfc3176).

To work, sFlow records contain Layer 2 headers as well as packet counters not found in Netflow, which gives the operator the ability to match traffic based on things like 802.1q VLAN headers in a packet.

It uses the concept of an sFlow agent running on the router or switch that collects the sampled packets from the ASICs in the forwarding plane and exports them to the monitoring station. The goal of this design is to allow sFlow to achieve very high sampling rates in large bandwidth applications.

MORE? For more information on sFlow, check out this white paper at http://www.sflow.org/sFlowOverview.pdf.

IPFIX

The IPFIX protocol builds upon the work done in NetFlow. It was defined by the IETF in RFCs 7011 through 7015. The goal is to have a more comprehensive and standard way of exporting IP flow information.

In IPFIX, a *metering process* collects data packets at an *observation point*. An *exporter* then sends the data to a *collector*. One improvement made with IPFIX is that an exporter can send to more than one collector. In addition, a collector can receive data from more than one exporter.

Another enhancement that IPFIX provides to the network operator is the ability to customize what information they need to see. The user can create a template that defines the various headers that are of interest.

Finally, the transport mechanism for IPFIX is flexible. It can be defined by the user to be TCP, UDP, or SCTP (Stream Control Transport Protocol).

MORE? Once again, Wikipedia has a good amount of information to get a person started on IPFIX: https://en.wikipedia.org/wiki/NetFlow. For more detailed information, you can take a look at the IETF's IPFIX Working Group page: http://tools.ietf.org/wg/ipfix/.

Detection Tools

So, now that you are familiar with some of the protocols you can use to get information about the traffic in the network, what do you do with all of that information? How do you use this information to detect that an attack is underway? Let's take a look at a few of the tools that can be used to analyze this information and detect DDoS attacks.

Open Source

A lot of network operators prefer to use open source tools. These tools are free and can be customized by the user but come with less support. In the world of open source netflow analyzers, Nfdump and NfSen are two of the best tools available.

Nfdump (NetFlow Dump) is a set of applications that can run on a Linux server that collects and processes the flow records discussed in the previous section using a CLI. The goal is to give the user a way to see their flow data in a text-based format.

Netflow Sensor (NfSen) is a graphical front end to Nfdump. It allows the user to view the data present in Nfdump in a more graphical representation.

Once you have the data in these tools, you can write custom scripts to send notifications based on the information in the flow data. Note however, that this requires a fairly high degree of customization by the user.

MORE? For more information on Nfdump and NfSen, check out the Sourceforge pages available at: http://nfdump.sourceforge.net/ and http://nfsen.sourceforge.net/.

Arbor Networks SP

Arbor Networks is the leader in DDoS detection with their Arbor Networks SP solution (formerly Arbor Peakflow). While not free, like Nfdump and NfSen, it is user friendly and comes with more built-in functionality. Arbor Networks SP can analyze the flow records it receives and detect an attack in a network. Once an attack is detected, it can create an SNMP trap or generate BGP FlowSpec routes to automate the mitigation of the attack. This type of use case is discussed in future chapters.

NOTE Neither this book, nor Juniper Networks, endorses this networking product over any other product from any other company. The Arbor product was part of the *Day One* lab used in the book, and thus, is mentioned as one of the lab devices. If you want more information on Arbor Networks SP or any of their products, please, check out their website at: http://cdn.arbornetworks.com/.

InMon Traffic Sentinel

Another detection tool is InMon's Traffic Sentinel. As mentioned earlier, InMon created the sFlow protocol to help network operators get more visibility into their network traffic. Traffic Sentinel collects the sFlow data and analyzes it. They have since added the capability to collect NetFlow data as well. While it does not offer quite the robust features of Arbor Networks SP solution, it is a good option for those wanting to use sFlow.

MORE? For more information on InMon's Traffic Sentinel, check out their website at: http://www.inmon.com/products/trafficsentinel.php.

Summary

Take some time to research these tools and products in your lab setup. If you are just starting, use the free, open source tools. You can always move to commercial tools if your needs require that in the future.

Next, let's get the lab working on BGP FlowSpec. Remember, the whole point is to create an automated BGP FlowSpec response to any DDoS attack and protect your network.

Chapter 4

Inter-domain DDoS Mitigation

This chapter explores how to design a DDoS mitigation solution wherein a service provider allows their customers to advertise BGP FlowSpec routes to them. It starts off by taking a look at the overall design, then explores what the Junos OS configurations would look like on the various devices. It also discusses some best practices that should be considered before deploying this type of solution, and finally some of the Junos OS commands available to help you verify the solution is working properly.

Let’s jump right in.

Solution Design

Let's start by taking a look at how our solution is going to be set up. It's a similar network design to what was used in Chapter 1, however, a refresher may be in order.

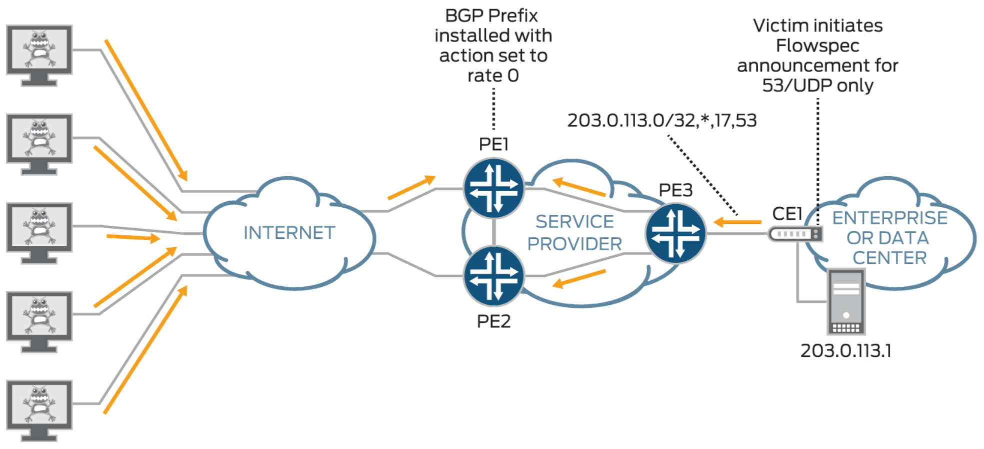
BGP Prefix installed with action set to rate 0
Victim initiates Flowspec announcement for 53/UDP only
203.0.113.0/32,*,17,53
PE1
PE3
CE1
INTERNET
SERVICE PROVIDER
ENTERPRISE OR DATA CENTER
PE2
203.0.113.1

Figure 4.1 Inter-Domain DDoS Mitigation Using BGP FlowSpec

On the right of Figure 4.1, you can see the enterprise customer who is advertising their IP address block via BGP to the service provider. The service provider in turn, advertises these routes to the rest of the Internet.

In this scenario, the attack is a DNS amplification attack. That means you are receiving a large volume of UDP port 53 packets that you do not need for normal web server operation. The attack fills up the circuit between the service provider and the enterprise customer and effectively takes the web server offline.

Once the bad actor decides to launch the attack, the enterprise customer can create a BGP FlowSpec route for just UDP port 53 packets, and advertise it to their service provider. The service provider's routers turn these routes into a firewall filter on their edge interfaces. This blocks the DNS amplification packets at the edge of the service provider network but allows legitimate web traffic to continue to reach the enterprise's web server.

CAUTION The current standard and Juniper implementation for BGP FlowSpec applies the firewall filter to all interfaces in the router. There is work being done in the IETF to provide the ability to only apply the firewall filters to a defined subset of interfaces.

Configuration

Let's start by taking a look at the configurations on the edge router, assuming that the router is already configured for normal BGP unicast routing.

Customer Edge (CE)

NOTE If you're following this in the lab, adding an address family to an existing BGP neighbor in the Junos OS will cause the BGP session to restart in order to renegotiate for the new address family.

To configure the router to advertise the BGP FlowSpec routes:

1. Add the BGP FlowSpec NLRI to the BGP session with the service provider:

```
protocols {
    bgp {
        group SP {
            neighbor 192.0.2.0 {
                family inet {
                    flow;
                }
            }
        }
    }
}
```

2. Make sure that you are using the RFC defined ordering of terms for BGP FlowSpec:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

3. Define the BGP FlowSpec route:

```
routing-options {
    flow {
        route dns {
            match {
                destination 203.0.113.1/32;
                protocol udp;
                destination-port 53;
            }
            then discard;
        }
    }
}
```

4. Create a policy to advertise the BGP FlowSpec route via BGP:

```
policy-options {
    policy-statement FLOW-TO-BGP {
        term 1 {
            from rib inetflow.0;
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

5. Apply the new policy as an export policy on the BGP session:

```
 protocols {
    bgp {
        group SP {
            export FLOW-TO-BGP;
        }
    }
}
```

Provider Edge (PE)

Okay, that's what is required from the customer, so let's move on and take a look at what the service provider will have to do. Again, let's assume that a unicast BGP session is already in place, so let's start by looking at the Provider Edge (PE) router that connects to PE3:

1. Add the BGP FlowSpec NLRI to the BGP session with the customer:

```
protocols {
    bgp {
        group CUST-FLOWSPEC {
            neighbor 192.0.2.1 {
                family inet {
                    flow;
                }
            }
        }
    }
}
```

2. Make sure to use the RFC defined ordering of terms for BGP FS:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

3. Configure all the iBGP sessions to advertise BGP FlowSpec type routes:

```
protocols {
    bgp {
        group ibgp {
            family inet {
                flow;
            }
        }
    }
}
```

Next, let's take a look at what the service provider must do on the other PE routers in their network (this would need to be done on all routers but let's only show PE1 for the sake of brevity):

1. Configure all the iBGP sessions to advertise BGP FlowSpec type routes:

```
protocols {
    bgp {
        group ibgp {
            family inet {
                flow;
            }
        }
    }
}
```

2. Make sure to use the RFC defined ordering of terms for BGP FS:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

That is the minimum configuration setup that is needed by both the customer and the service provider in order to mitigate this DDoS attack. However, let's review additional optional configurations that can be added in order to follow industry best practices.

Best Practices

There are best practices to protect this solution. Both the BGP Flow-Spec routes and the resulting firewall filters they create are finite resources on the router. Therefore, the provider will want to do inbound route filtering to make sure the customer does not send them too many routes or the wrong routes. One of the advantages of BGP FlowSpec is that you can reuse the same prefix limits and route policy structures to protect these routes that you use with inet unicast routes.

Prefix Limit

So the first thing to do is to set a prefix limit for the BGP FlowSpec routes from the customer. You could just set a single prefix limit for both inet unicast and inet flow routes, however, this example is going to set a separate limit for the inet flow routes. You want the customer to only be able to send a single flow route at a time, so let's set the prefix limit to 1. If you want the session to be torn down once the limit is reached, you can set the *teardown* configuration as well:

```
protocols {
    bgp {
        group CUST-FLOWSPEC {
            neighbor 192.0.2.1 {
                family inet {
                    flow {
                        prefix-limit {
                            maximum 1;
                        }
                    }
                }
            }
        }
    }
}
```

Route Policy

The next thing to do is to apply an inbound route policy. This policy is going to limit the customer to only sending a /32 prefix for BGP FlowSpec. Let's also add a community of 64496:86 so you can identify the routes as being customer-originated BGP FlowSpec routes. For all other routes, you can just filter them based on the customer's allocation of routes:

1. Create the policy definition:

```
policy-options {
    policy-statement CUST-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 203.0.113.0/24 prefix-length-range /32-/32;
            }
            then {
                community add CUST-FS-COMM;
                accept;
            }
        }
        term 2 {
            from {
                route-filter 203.0.113.0/24 orlonger;
            }
            then accept;
        }
        term 3 {
            then reject;
        }
    }
    community CUST-FS-COMM members 64496:86;
}

2. Apply the policy as an import policy on the customer's BGP session:
protocols {
    bgp {
        group CUST-FLOWSPEC {
            import CUST-IN;
        }
    }
}
```

Maximum Prefixes

The final thing to do is to set a maximum amount of BGP FlowSpec prefixes that can be installed in the routing table. This example sets a maximum of 10,000 routes but let's also configure the router to notify the administrator via a syslog message when a 90% threshold is reached. While only PE3 is shown here as an example, this setting should be applied on all routers in the service provider's network:

```
routing-options {
    rib inetflow.0 {
        maximum-prefixes 10000 threshold 90;
    }
}
```

It may not be intuitive to the reader, but the `threshold 90` option in the configuration is what tells the router that you want a syslog message generated when the 90% threshold is reached.

Verification

The first thing to verify is that BGP FlowSpec is configured correctly. Let's look at the NLRIs that are enabled on the BGP neighbor.

Verifying the NLRI

From operational mode on the PE3 router, enter the Junos OS `show bgp neighbor 192.0.2.1` command and look for the `inet-flow` capability in the output:

```
lab@pe3> show bgp neighbor 192.0.2.1
Peer: 192.0.2.1+45824 AS 64511 Local: 192.0.2.0+63720 AS 64496
  Type: External    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Import: [ CUST-IN ]
  Options: <Preference LocalAddress AddressFamily PeerAS PrefixLimit Refresh>
  Address families configured: inet-unicast inet-flow
  Local Address: 192.0.2.0 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 128.92.38.179   Local ID: 128.92.39.244     Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 0
  BFD: disabled, down
  Local Interface: ge-0/0/3.0
  NLRI for restart configured on peer: inet-unicast inet-flow
  NLRI advertised by peer: inet-unicast inet-flow
  NLRI for this session: inet-unicast inet-flow
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-unicast inet-flow
  NLRI of received end-of-rib markers: inet-unicast inet-flow
  NLRI of all end-of-rib markers sent: inet-unicast inet-flow
  Peer supports 4 byte AS extension (peer-as 64511)
  Peer does not support Addpath
  Table inetflow.0 Bit: 10000
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:             1
    Received prefixes:           1
    Accepted prefixes:           1
    Suppressed due to damping:   0
    Advertised prefixes:         0
  Table inet.0 Bit: 20000
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:             1
    Received prefixes:           1
    Accepted prefixes:           1
    Suppressed due to damping:   0
    Advertised prefixes:         0
  Last traffic (seconds): Received 13   Sent 28   Checked 4
  Input messages:  Total 392  Updates 4 Refreshes 0 Octets 7552
  Output messages: Total 386  Updates 0 Refreshes 0 Octets 7435
  Output Queue[0]: 0           (inetflow.0, inet-flow)
  Output Queue[1]: 0           (inet.0, inet-unicast)
```

Verifying Routes

The next thing to look at is the flow route configured on CE1:

```
lab@ce1> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): discard,count
Page 0 idx 0, (group SP type External) Type 1 val 0x966ed2c (adv_entry)
   Advertised metrics:
     Nexthop: Self
     AS path: [64511] I
     Communities: traffic-rate:0:0
Path 203.0.113.1,*,proto=17,dstport=53 Vector len 4.  Val: 0
        *Flow   Preference: 5
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active>
                Local AS: 64511
                Age: 3:22:30
                Validation State: unverified
                Task: RT Flow
                Announcement bits (2): 0-Flow 1-BGP_RT_Background
                AS path: I
                Communities: traffic-rate:0:0
```

You can see that the route is in the inetflow.0 table with a discard action and a traffic rate of 0 (discard). Now, let's take a look at the route on PE3:

```
lab@pe3> show route receive-protocol bgp 192.0.2.1 extensive table inetflow.0

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
     Accepted
     Nexthop: Self
     AS path: 64511 I
     Communities: traffic-rate:0:0
```

And here the route is being received by PE3. Note that the only community attached to the route is the one that specifies that you should set a rate of 0 for the traffic. Now let's take a look at the route in the routing table on PE3:

```
lab@pe3> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
TSI:
```

```
KRT in dfwd;
Action(s): discard,count
Page 0 idx 1, (group ibgp type Internal) Type 1 val 0x966ef08 (adv_entry)
   Advertised metrics:
     Nexthop: Self
     Localpref: 100
     AS path: [64496] 64511 I
     Communities: 64496:86 traffic-rate:0:0
Page 0 idx 2, (group RR-CLIENT-FLOWSPEC type Internal) Type 1 val 0x966f11c (adv_
entry)
   Advertised metrics:
     Nexthop: Self
     Localpref: 100
     AS path: [64496] 64511 I
     Communities: 64496:86 traffic-rate:0:0
Path 203.0.113.1,*,proto=17,dstport=53 from 192.0.2.1 Vector len 4.  Val: 1 2
        *BGP    Preference: 170/-101
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active Ext>
                Local AS: 64496 Peer AS: 64511
                Age: 3:27:43
                Validation State: unverified
                Task: BGP_64511.192.0.2.1+45824
                Announcement bits (2): 0-Flow 1-BGP_RT_Background
                AS path: 64511 I
                Communities: 64496:86 traffic-rate:0:0
                Accepted
                Validation state: Accept, Originator: 192.0.2.1
                Via: 203.0.113.0/24, Active
                Localpref: 100
                Router ID: 128.92.38.179
```

The route has been installed in the route table on PE3 and the customer community of 64496:86 that was assigned has been added to the route. Now let's go take a look at PE1 and make sure the route has been propagated through the network:

```
lab@pe1> show route table inetflow.0

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

203.0.113.1,*,proto=17,dstport=53/term:1
            *[BGP/170] 02:07:27, localpref 100, from 198.51.100.3
            AS path: 64511 I, validation-state: unverified
            Fictitious
```

Okay, the routes are propagating properly, so let's turn our attention to flow validation.

Verifying Flow Validation

As discussed previously, the BGP FlowSpec standard requires that flow routes be validated for authenticity before they can be converted to firewall filters:

```
lab@pe3> show route flow validation detail

inet.0:
203.0.113.0/24
                Internal node: no match, consistent, next-as: 64511
                Active unicast route
                    Dependent flow destinations: 1
                    Origin: 192.0.2.1, Neighbor AS: 64511
203.0.113.1/32
                Flow destination (1 entries, 1 match origin)
                    Unicast best match: 203.0.113.0/24
                    Flags: Consistent
```

You can see that the flow has been matched to the unicast route and validated for consistency. Let's move on.

Verifying Firewall Filters

Now that the flow routes have been validated, they get turned into firewall filters. Let's take a look at the firewall filter:

```
lab@pe3> show firewall filter __flowspec_default_inet__

Filter: __flowspec_default_inet__
Counters:
Name                                Bytes           Packets
203.0.113.1,*,proto=17,dstport=53       0                 0
```

Verifying System Logging

If you have followed best practices and set a prefix-limit on the BGP session, you will see a log message once the maximum is reached:

```
lab@pe3> show log messages | match BGP_PREFIX
Sep  5 17:51:04  pe3 rpd[2812]: BGP_PREFIX_LIMIT_
EXCEEDED: 192.0.2.1 (External AS 64511): Configured maximum prefix-
limit(1) exceeded for inet-flow nlri: 2 (instance master)
```

Summary

As you can see, the setup of an inter-domain BGP FlowSpec solution is not that hard. It just requires a little coordination between customer and service provider. There are enough tweaks and knob-turning in these configurations for you to experiment within your lab. When you're ready, let’s move on to the next chapter to find a solution for when your service provider does not allow their customers to advertise BGP FlowSpec routes.

Chapter 5

Intra-domain DDoS Mitigation

This chapter explores how to design a DDoS mitigation solution when a service provider *does not allow* their customers to advertise BGP FlowSpec routes to it. In this case, customers have to contact the service provider to request the attack be blocked.

Let's start off by taking a look at the overall design. Then we'll explore what the Junos OS configurations would be like on the various devices and some of the best practices that should be considered before deploying this type of solution. Finally, there's a few really good Junos OS CLI commands that can help you troubleshoot the solution. Let's get started.

Solution Design

There are a few changes to the topology that was used in Chapter 4 expressly for how this solution should be set up, and they are shown in Figure 5.1.

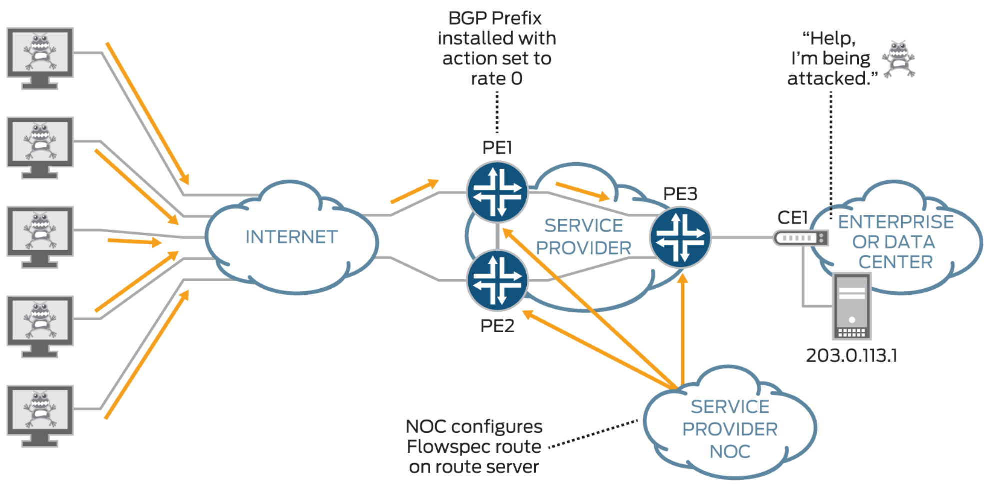
BGP Prefix installed with action set to rate 0
"Help, I'm being attacked."
PE1
PE3
CE1
INTERNET
SERVICE PROVIDER
ENTERPRISE OR DATA CENTER
PE2
203.0.113.1
NOC configures Flowspec route on route server
SERVICE PROVIDER NOC

Figure 5.1 Intra-Domain DDoS Mitigation Using BGP FlowSpec

Once again, you can see in Figure 5.1 that the enterprise customer on the right is advertising their IP block via BGP to the service provider, and the service provider in turn advertises these routes to the rest of the Internet.

And again, the attack is a DNS amplification attack. That means the receipt of a large volume of UDP port 53 packets that you do not need for normal web server operation. This fills up the circuit between the service provider and the enterprise customer and effectively takes the web server offline.

However, in this case, the enterprise customer must notify the service provider once the attack takes place. This notification could be a phone call to the service provider's Network Operations Center (NOC), or it could be via a web portal.

If the notification is done via a phone call, the service provider will then log in to their route server and advertise a BGP FlowSpec route to UDP port 53 traffic destined for the enterprise web server.

If the notification is done via a web portal, the service provider could automate the creation of the BGP FlowSpec route on the route server, giving the enterprise customer the ability to do self-service attack mitigation. The service provider could also develop intelligence in this web portal to limit what the enterprise customer is allowed to block, for example the number of routes or the prefix of the routes.

Another option is for the service provider to use a detection tool to detect and automatically block the attack in their network. They could then sell this as a value-added service to the enterprise customer.

No matter how the notification is done, the BGP FlowSpec route blocks the DNS amplification packets at the edge of the service provider network but allows legitimate web traffic to continue to reach the enterprise's web server.

Configuration

Let's start with the configurations for an intra-domain solution and establish an iBGP session between the route reflector (RR1) and all of our provider edge (PE1 – PE3) devices in order to receive the BGP FlowSpec routes.

Provider Edge (PE)

First let's configure the PE routers to make this work. Since the unicast routes are coming from the enterprise customer, BGP FlowSpec routes from the route reflector will fail to validate. Therefore, you must use the `no-validate` option to override this behavior. When you do, you have to create a route policy to use for validating routes instead. For brevity, let's just look at the configuration on PE1 (the other PEs in the service provider's network are similar);

1. Create the policy to validate the routes when they are received from the route reflector:

```
policy-options {
    policy-statement FS-RR-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

The astute reader might notice that the configuration is only allowing flow routes of length /32 and rejecting everything else.

2. Configure the iBGP session with the route reflector, enable the family inet flow NLRI type, and apply the policy through the use of the `no-validate` syntax:

```
protocols {
    bgp {
        local-address 198.51.100.1;
        group RR-CLIENT-FLOWSPEC {
            type internal;
            neighbor 198.51.100.5 {
                family inet {
                    flow {
                        no-validate FS-RR-IN;
                    }
                }
            }
        }
    }
}
```

3. And let's make sure you are using the RFC-defined ordering of terms for BGP FlowSpec:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

Route Reflector (RR)

Next, let's take a look at the configuration needed on the route reflector. The astute reader will notice the steps taken for the route reflector in this design are very similar to the steps taken on the customer edge router in the inter-domain design of the previous chapter. And they are:

1. Configure the iBGP group for the route reflector clients and enable the BGP FlowSpec NLRI:

```
protocols {
    bgp {
        local-address 198.51.100.5;
        group RR-CLIENT-FLOWSPEC {
            type internal;
            family inet {
                flow;
            }
            cluster 0.0.0.1;
            neighbor 198.51.100.1;
            neighbor 198.51.100.2;
            neighbor 198.51.100.3;
        }
    }
}
```

2. Make sure you are using the RFC-defined ordering of terms for BGP FlowSpec:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

3. Define the BGP FlowSpec route:

```
routing-options {
    flow {
        route dns {
            match {
                destination 203.0.113.1/32;
                protocol udp;
                destination-port 53;
            }
            then discard;
        }
    }
}
```

4. Create a policy to advertise the BGP FlowSpec route via BGP:

```
policy-options {
    policy-statement FLOW-TO-BGP {
        term 1 {
            from rib inetflow.0;
            then {
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
}
```

5. And apply the new policy as an export policy in the BGP route reflection group:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            export FLOW-TO-BGP;
        }
    }
}
```

Best Practices: Route Policy

The best practices needed for an *intra*-domain solution are probably less strict than what would be needed for an *inter*-domain solution where multiple organizations are involved. However, most operators need to put some checks in place to make sure a configuration mistake on the route reflector does not cause a major outage on the service provider backbone.

In this design, the service provider controls both the route reflector and the provider edge router. Let's take a look at the route policy that is needed on each device.

Route Reflector

Since the route reflector is originating the BGP FlowSpec routes into the service provider's network, this device attaches a BGP community to the routes. It uses a different community, 64496:85, than what was used in the previous design so it can distinguish the routes. Also, let's configure the route reflector to only send out routes and reject any routes sent to it. This helps protect from a configuration mistake on the provider edge router.

1. Create the policy definitions:

```
policy-options {
    policy-statement FLOW-TO-BGP {
        term 1 {
            from rib inetflow.0;
            then {
                community add INT-FS-COMM;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement NO-ROUTES-IN {
        term 1 {
            then reject;
        }
    }
    community INT-FS-COMM members 64496:85;
}
```

2. Apply the policy to the BGP group as import and export policies:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            import NO-ROUTES-IN;
            export FLOW-TO-BGP;
        }
    }
}
```

Provider Edge

As you may recall, you have to create a policy on the provider edge router in order to use the `no-validate` method. (You have already looked at this in the configuration section but let's take a look again just for completeness.)

1. Create the policy definition:

```
policy-options {
    policy-statement FS-RR-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

2. Apply the policy as an import policy on the route reflector's BGP session:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            neighbor 198.51.100.5 {
                family inet {
                    flow {
                        no-validate FS-RR-IN;
                    }
                }
            }
        }
    }
}
```

Maximum Prefixes

The final best practice is to set a maximum amount of BGP FlowSpec prefixes that can be installed in the routing table. This next example sets a maximum of 10,000 routes. The router will also be configured to notify the administrator via a syslog message when a 90% threshold is reached. While only PE1 is shown here as an example, this setting should be applied on all routers in the service provider's network:

```
routing-options {
    rib inetflow.0 {
        maximum-prefixes 10000 threshold 90;
    }
}
```

As noted in the previous chapter, the `threshold 90` piece of the configuration is what tells the router that you want a syslog message generated when the 90% threshold is reached.

Verification

While there are several verification steps, as in the previous chapter, the first thing to verify that BGP FlowSpec is configured correctly is to look at the NLRIs that are enabled on the BGP neighbor.

Verifying the NLRI

From operational mode on the PE1 router, enter this show command and look for the `inet-flow` capability in the output:

```
lab@pe1> show bgp neighbor 198.51.100.5
Peer: 198.51.100.5+45824 AS 64496 Local: 198.51.100.1+34802 AS 64496
  Type: Internal    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Options: <Preference LocalAddress AddressFamily Refresh>
  Address families configured: inet-flow
  Local Address: 198.51.100.1 Holdtime: 90 Preference: 170
  NLRI inet-flow: No-validate [ FS-RR-IN ]
  Number of flaps: 0
  Peer ID: 128.92.39.240   Local ID: 128.92.39.248     Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 0
  BFD: disabled, down
  NLRI for restart configured on peer: inet-flow
  NLRI advertised by peer: inet-flow
  NLRI for this session: inet-flow
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-flow
  NLRI of received end-of-rib markers: inet-flow
  NLRI of all end-of-rib markers sent: inet-flow
  Peer supports 4 byte AS extension (peer-as 64496)
  Peer does not support Addpath
  Table inetflow.0 Bit: 20001
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              1
    Received prefixes:            1
    Accepted prefixes:            1
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 6    Sent 21   Checked 47
  Input messages:  Total 1072 Updates 5 Refreshes 0 Octets 20577
  Output messages: Total 1071 Updates 0 Refreshes 0 Octets 20419
  Output Queue[1]: 0            (inetflow.0, inet-flow)
```

Verifying Routes

The next thing to look at is the flow route configured on the route reflector (RR1). Let's use this operational show command to verify:

```
lab@rr1> show route table inetflow.0 extensive

inetflow.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): discard,count
Page 0 idx 0, (group RR-CLIENT-FLOWSPEC type Internal) Type 1 val 0x966ee60 (adv_
entry)
   Advertised metrics:
     Nexthop: Self
     Localpref: 100
     AS path: [64496] I
     Communities: 64496:85 traffic-rate:0:0
Path 203.0.113.1,*,proto=17,dstport=53 Vector len 4.  Val: 0
        *Flow   Preference: 5
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 2
                State: <Active>
                Local AS: 64496
                Age: 8:09:16
                Validation State: unverified
                Task: RT Flow
                Announcement bits (2): 0-Flow 1-BGP_RT_Background
                AS path: I
                Communities: traffic-rate:0:0
```

You can see here that the route is in the inetflow.0 table on the route reflector with a discard action and a traffic rate set to 0 (discard). Next, let's take a look at the route on a provider edge router, PE1, using this operational show command:

```
lab@pe1> show route receive-protocol bgp 198.51.100.5 extensive

inet.0: 68 destinations, 68 routes (68 active, 0 holddown, 0 hidden)

iso.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

mpls.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)

inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
     Accepted
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: 64496:85 traffic-rate:0:0
```

And the route is being received by PE1. Note that the BGP community that specifies to set a rate of 0 as well as the local community of 64496:85 are both applied to the route. Next, let's take a look at the route in the routing table on PE1:

```
lab@pe1> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*,proto=17,dstport=53/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): discard,count
        *BGP    Preference: 170/-101
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active Int Ext>
                Local AS: 64496 Peer AS: 64496
                Age: 8:13:02
                Validation State: unverified
                Task: BGP_64496.198.51.100.5+45824
                Announcement bits (1): 0-Flow
                AS path: I
                Communities: 64496:85 traffic-rate:0:0
                Accepted
                Localpref: 100
                Router ID: 128.92.39.240
```

And you can see that route has been installed in the routing table on PE1.

Verifying Flow Validation

Now that the routes are being propagated correctly, let's turn our attention to the flow validation. Remember, that in this design, the default flow validation has been disabled using the `no-validate` option and a route policy was used instead. Due to this, there should *not* be any validated routes:

```
lab@pe1> show route flow validation detail

inet.0:
```

So, instead, you have to rely on the route checking done in the previous chapter.

Verifying Firewall Filters

Again, once the routes have been validated and propagated, they get turned in to firewall filters automatically. So check the output of the `show firewall filter` command:

```
__flowspec_default_inet__:

lab@pe1> show firewall filter __flowspec_default_inet__

Filter: __flowspec_default_inet__
Counters:
Name                                                 Bytes             Packets
203.0.113.1,*,proto=17,dstport=53                        0                   0
```

Verifying System Logging

The final thing to verify is if you are exceeding the number of prefixes set as the limit. In order to test this, set the maximum-prefix on PE1 down to 1 and advertise a second route from the route reflector:

```
routing-options {
    rib inetflow.0 {
        maximum-prefixes 1 threshold 90;
    }
}
```

Now let's take a look at the logs:

```
lab@pe1> show log messages | match PREFIX_LIMIT
Sep 14 14:45:04  pe1 rpd[2806]: RPD_RT_PREFIX_LIMIT_
REACHED: Number of prefixes (1) in table inetflow.0 reached configured maximum (1)
```

Now, let's go set the limit back to 10,000 so you can see the log clear:

```
routing-options {
    rib inetflow.0 {
        maximum-prefixes 100000 threshold 90;
    }
}
```

And then:

```
lab@pe1> show log messages | match PREFIX_LIMIT
Sep 14 14:46:34  pe1 rpd[2806]: RPD_RT_PREFIX_LIMIT_BELOW: Number of prefixes (1) in
table inetflow.0 is now less than the configured maximum (10000)
```

Summary

These are the two BGP FlowSpec solutions, one with service provider advertisement, and one without.

This book has taken the perspective of both the service provider customer and the service provider itself. Whether you are creating new services for your customers, or whether you are trying to automate DDoS detection and security with your service provider, there are plenty of options to explore in the lab.

The next chapter puts everything you've learned so far to use when your service provider uses a scrubbing center to remove the attack traffic instead of blocking it at the provider edge.

Chapter 6

Using a Scrubbing Center

This chapter explores how to design a DDoS mitigation solution when a service provider is using a Scrubbing Center to block the attack traffic while allowing legitimate traffic to reach the enterprise server. The service provider could use an intrusion detection and prevention (IDP) appliance or a third-party service to accomplish this task.

The chapter starts off by taking a look at how the solution is designed, then dives into the Junos OS configurations to make it work, and as in previous chapters, wraps up with a best practices section and verification of the chapter solution.

Solution Design

As done in previous chapters, let's start by taking a look at how our solution is going to be set up. The topology shown in Figure 6.1 is similar to what was used in previous chapters with only a few modifications.

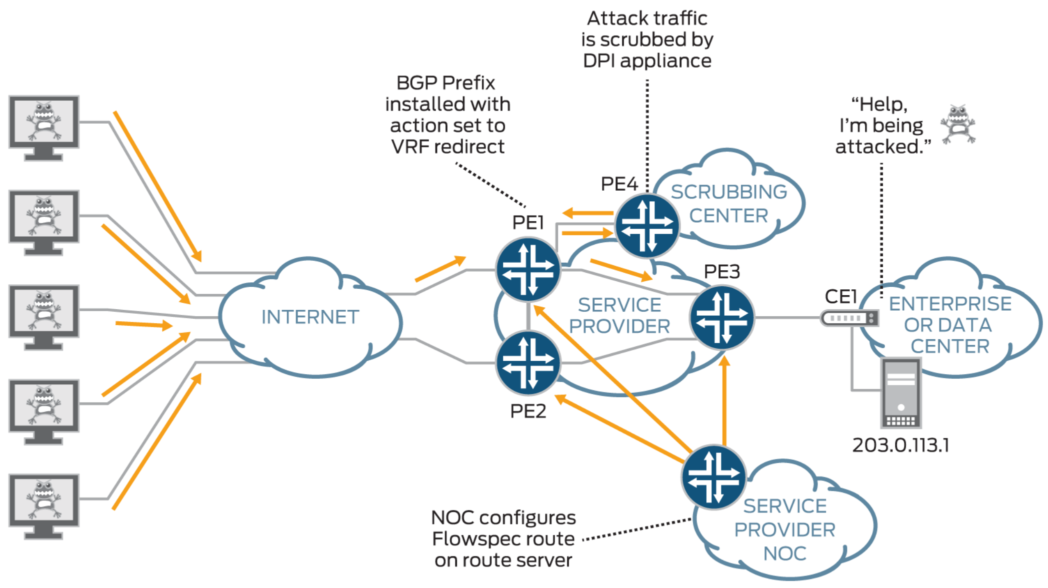
Attack traffic is scrubbed by DPI appliance
BGP Prefix installed with action set to VRF redirect
"Help, I'm being attacked."
PE4
SCRUBBING CENTER
PE1
PE3
INTERNET
SERVICE PROVIDER
CE1
ENTERPRISE OR DATA CENTER
PE2
203.0.113.1
SERVICE PROVIDER NOC
NOC configures Flowspec route on route server

Figure 6.1 DDoS Mitigation Using BGP FlowSpec with a Scrubbing Center

In Figure 6.1, the enterprise customer on the right is advertising their IP block via BGP to the service provider. The service provider in turn advertises these routes to the rest of the Internet.

Once again, the attack is a DNS amplification attack using a large volume of UDP port 53 packets to overwhelm the enterprise server.

For the sake of discussion, let's assume that the enterprise customer has to notify the service provider of the attack by any method discussed in Chapter 5. In any case, the BGP FlowSpec route is configured in the route reflector.

However, in this case, instead of blocking the traffic let's send the traffic to a special *DIRTY-VRF* that will be connected to our IDP. PE4 will act as the IDP appliance in this design and it is connected to the service provider network via PE1.

Once the traffic is injected into the DIRTY-VRF, it needs to reach PE4. You could put in a route for each prefix you are going to scrub, however, you really want to make this as simple as possible. Therefore, let's use a default out in the DIRTY-VRF that points at PE4. Figure 6.2 shows how all these pieces connect together using PE2 as an example.

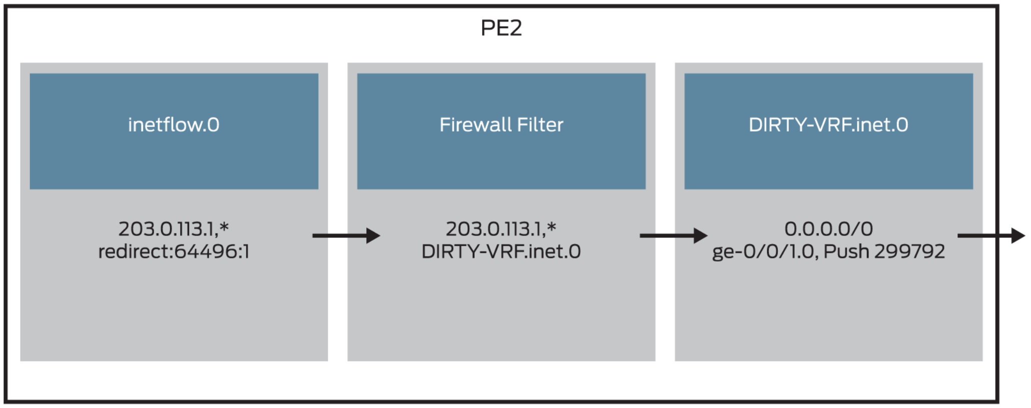

Figure 6.2 Logical Representation of Route Tables

So PE2 is receiving the BGP FlowSpec route into the inetflow.0 route table. As previously discussed, this is converted into a firewall filter, and this firewall filter performs filter-based forwarding to send the traffic to the DIRTY-VRF.inet.0 table. Once the traffic arrives in that table, a lookup is done and if the traffic matches the default route, PE2 will put on a label of 299792 on it and send it out via interface ge-0/0/1.

Once the attack traffic is removed by the IDP, the clean traffic will be re-injected back into the service provider's network in the global inet.0 routing table for delivery to the enterprise customer's server. Sounds cool, but how does it actually work? Let's take a look.

Configuration

In this design, there are three routers that are of interest: PE1, PE2, and RR1. PE3's configuration is similar to PE2, so PE3 will be ignored for the sake of brevity.

Provider Edge 1 (PE1)

First, let's set up MPLS in the core as well as configure our DIRTY-VRF L3 VPN.

1. Enable the MPLS family on the core-facing interfaces:

```
interfaces {
    ge-0/0/1 {
            family mpls;
        }
    }
    ge-0/0/3 {
        unit 0 {
            family mpls;
        }
    }
}
```

2. Turn on MPLS on the core-facing interfaces:

```
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

3. Turn on LDP on the core-facing interfaces:

```
protocols {
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

4. Make sure you are using the address on lo0 as the router ID:

```
routing-options {
    router-id 198.51.100.1;
}
```

5. Create the L3 VPN instance:

```
routing-instances {
    DIRTY-VRF {
        instance-type vrf;
        route-distinguisher 64496:1;
        vrf-target target:64496:1;
    }
}
```

6. Put the interface facing PE4 in the instance:

```
routing-instances {
    DIRTY-VRF {
        interface ge-0/0/2.0;
    }
}
```

7. Create a static default route inside the L3 VPN that points to PE4:

```
routing-instances {
    DIRTY-VRF {
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 192.0.2.7;
            }
        }
    }
}
```

8. Enable the `family inet-vpn` on the iBGP sessions:

```
protocols {
    bgp {
        group ibgp {
            family inet-vpn {
                unicast;
            }
        }
    }
}
```

NOTE Designing an MPLS and Layer 3 VPN network is beyond the scope of this book. For more details on how to do this, please refer to the *Day One* library at: http://www.juniper.net/dayone.

Now let's turn our attention to the BGP FlowSpec side of things. Again, the unicast routes are coming from the enterprise customer's CE1 device, so BGP FlowSpec routes from the route reflector will fail to validate. Use the `no-validate` option with a policy to override this behavior.

1. Create the policy in order to validate the routes when they are received from the route reflector:

```
policy-options {
    policy-statement FS-RR-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

2. Configure the iBGP session with the route reflector, enable the family inet flow NLRI type, and apply the policy through the use of the `no-validate` syntax:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            type internal;
            neighbor 198.51.100.5 {
                family inet {
                    flow {
                        no-validate FS-RR-IN;
                    }
                }
            }
        }
    }
}
```

3. And make sure you are using the RFC-defined ordering of terms for BGP FS:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

Provider Edge 2 (PE2)

With PE1's configuration complete, let's turn our attention to PE2. Since the scrubbing center is not connected to PE2, the configuration is slightly easier than it was on PE1. Again, start by configuring the MPLS and L3 VPN pieces you need in the core.

1. Enable the MPLS family on the core-facing interfaces:

```
interfaces {
    ge-0/0/1 {
        unit 0 {
            family mpls;
        }
    }
    ge-0/0/2 {
        unit 0 {
            family mpls;
        }
    }
}
```

2. Turn on MPLS on the core-facing interfaces:

```
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

3. Turn on LDP on the core-facing interfaces:

```
protocols {
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

4. Make sure you are using the address on lo0 as the router ID:

```
routing-options {
    router-id 198.51.100.2;
}
```

5. Create the L3 VPN instance:

```
routing-instances {
    DIRTY-VRF {
        instance-type vrf;
        route-distinguisher 64496:2;
        vrf-target target:64496:1;
    }
}
```

6. And, enable the `family inet-vpn` on the iBGP sessions:

```
protocols {
    bgp {
        group ibgp {
            family inet-vpn {
                unicast;
            }
        }
    }
}
```

There, now that you have completed the MPLS and L3 VPN portions, turn your attention to the BGP FlowSpec pieces of the configuration.

1. Create the policy to validate the routes when they are received from the route reflector:

```
policy-options {
    policy-statement FS-RR-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

2. Configure the iBGP session with the route reflector, enable the family inet flow NLRI type, and apply the policy through the use of the `no-validate` syntax:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            type internal;
            neighbor 198.51.100.5 {
                family inet {
                    flow {
                        no-validate FS-RR-IN;
                    }
                }
            }
        }
    }
}
```

3. And again, make sure you are using the RFC-defined ordering of terms for BGP FS:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

Route Reflector 1 (RR1)

Last but not least, let's take a look at the configuration on the service provider's route reflector (RR1).

1. Configure the iBGP group for the route reflector clients and enable the BGP FlowSpec NLRI:

```
protocols {
    bgp {
        local-address 198.51.100.5;
        group RR-CLIENT-FLOWSPEC {
            type internal;
            family inet {
                flow;
            }
            cluster 0.0.0.1;
            neighbor 198.51.100.1;
            neighbor 198.51.100.2;
            neighbor 198.51.100.3;
        }
    }
}
```

2. As always, make sure you are using the RFC-defined ordering of terms for BGP FlowSpec:

```
routing-options {
    flow {
        term-order standard;
    }
}
```

3. Define the BGP FlowSpec route:

```
routing-options {
    flow {
        route dns {
            match destination 203.0.113.1/32;
            then routing-instance target:64496:1;
        }
    }
}
```

4. Create a policy to advertise the BGP FlowSpec route via BGP:

```
policy-options {
    policy-statement FLOW-TO-BGP {
        term 1 {
            from rib inetflow.0;
            then {
                community add INT-FS-COMM;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    community INT-FS-COMM members 64496:85;
}
```

5. And, apply the new policy as an export policy in the BGP route reflection group:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            export FLOW-TO-BGP;
        }
    }
}
```

Best Practices

Similar to the design in Chapter 5, this solution is completely controlled by the service provider. Therefore, the best practices are less strict than designs where multiple organizations are involved. But there are checks a service provider could put into place to minimize the impact of a configuration mistake.

NOTE Again, best practices for MPLS and Layer 3 VPNs are beyond the scope of this book. The reader can get more information on these topics in the *Day One* library at: http://www.juniper.net/dayone.

Route Policy

For the purposes of best practices, there is no difference among the provider edge routers. Therefore, you can just take a look at the route reflector and a single provider edge router as an example.

Route Reflector

As with the previous designs, the route reflector is originating the BGP FlowSpec routes into the service provider's network. Therefore, this device attaches a BGP community to the routes for later identification. Use the same community used in the intra-domain solution: 64496:85. Also, configure the route reflector to reject any routes sent to it.

1. Create the policy definitions:

```
policy-options {
    policy-statement FLOW-TO-BGP {
        term 1 {
            from rib inetflow.0;
            then {
                community add INT-FS-COMM;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement NO-ROUTES-IN {
        term 1 {
            then reject;
        }
    }
    community INT-FS-COMM members 64496:85;
}
```

2. Apply the policy to the BGP group as import and export policies:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            import NO-ROUTES-IN;
            export FLOW-TO-BGP;
        }
    }
}
```

Provider Edge

Again, on the provider edge router, create a policy and apply it using the `no-validate` syntax.

1. Create the policy definition:

```
policy-options {
    policy-statement FS-RR-IN {
        term 1 {
            from {
                rib inetflow.0;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

2. Apply the policy as an import policy on the route reflector's BGP session:

```
protocols {
    bgp {
        group RR-CLIENT-FLOWSPEC {
            neighbor 198.51.100.5 {
                family inet {
                    flow {
                        no-validate FS-RR-IN;
                    }
                }
            }
        }
    }
}
```

Maximum Prefixes

To finish up, set the maximum amount of BGP FlowSpec prefixes that can be installed into the routing table. Set a maximum of 10,000 routes and configure the router to notify you via syslog message when a 90% threshold is reached. This configuration should be applied to all provider edge routers in the network:

```
routing-options {
   rib inetflow.0 {
      maximum-prefixes 10000 threshold 90;
   }
}
```

As mentioned previously, the `threshold 90` piece of the configuration is what tells the router that you want a syslog message generated when a 90% threshold is reached.

Verification

As you did in previous chapters, start by verifying that BGP FlowSpec is configured correctly by looking at the NLRIs that are enabled on the BGP neighbor.

Verifying the NLRI

From operational mode on the PE1 router, enter this show command and look for the inet-flow capability in the output:

```
lab@pe1> show bgp neighbor 198.51.100.5
Peer: 198.51.100.5+45824 AS 64496 Local: 198.51.100.1+34802 AS 64496
  Type: Internal    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Options: <Preference LocalAddress AddressFamily Refresh>
  Address families configured: inet-flow
  Local Address: 198.51.100.1 Holdtime: 90 Preference: 170
  NLRI inet-flow: No-validate [ FS-RR-IN ]
  Number of flaps: 0
  Peer ID: 128.92.39.240   Local ID: 128.92.39.248     Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 0
  BFD: disabled, down
  NLRI for restart configured on peer: inet-flow
  NLRI advertised by peer: inet-flow
  NLRI for this session: inet-flow
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-flow
  NLRI of received end-of-rib markers: inet-flow
  NLRI of all end-of-rib markers sent: inet-flow
  Peer supports 4 byte AS extension (peer-as 64496)
  Peer does not support Addpath
  Table inetflow.0 Bit: 20001
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              1
    Received prefixes:            1
    Accepted prefixes:            1
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 19   Sent 13   Checked 12
  Input messages:  Total 8974 Updates 10 Refreshes 0Octets 170922
  Output messages: Total 8975 Updates 0Refreshes 0Octets 170595
  Output Queue[1]: 0            (inetflow.0, inet-flow)
```

Verifying Routes

The next thing to look at is the flow route configured on the route reflector (RR1). Use the following command:

```
lab@rr1> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): ,count
Page 0 idx 0, (group RR-CLIENT-FLOWSPEC type Internal) Type 1 val 0x966ee60 (adv_
entry)
   Advertised metrics:
     Nexthop: Self
     Localpref: 100
     AS path: [64496] I
     Communities: 64496:85 redirect:64496:1
Path 203.0.113.1,* Vector len 4.  Val: 0
        *Flow   Preference: 5
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active>
                Local AS: 64496
                Age: 1d 23:59:47
                Validation State: unverified
                Task: RT Flow
                Announcement bits (2): 0-Flow 1-BGP_RT_Background
                AS path: I
                Communities: redirect:64496:1
```

The route is in the inetflow.0 table on the route reflector with a redirect community for our DIRTY-VRF instance. Next, look at the router on PE2 using this show command:

```
lab@pe2> show route receive-protocol bgp 198.51.100.5 extensive table inetflow.0

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 203.0.113.1,*/term:1 (1 entry, 1 announced)
     Accepted
     Nexthop: Self
     Localpref: 100
     AS path: I
     Communities: 64496:85 redirect:64496:1
```

You can see that the route is being received by PE2. Note that the BGP community that it specifies should redirect the traffic to DIRTY-VRF, as well as to our local community of 64496:85, as both are applied to the route. Next, let's take a look at the route in the routing table on PE2:

```
lab@pe2> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): ,count
        *BGP    Preference: 170/-101
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active Int Ext>
                Local AS: 64496 Peer AS: 64496
                Age: 1d 21:30:44
                Validation State: unverified
                Task: BGP_64496.198.51.100.5+44518
                Announcement bits (1): 0-Flow
                AS path: I
                Communities: 64496:85 redirect:64496:1
                Accepted
                Localpref: 100
                Router ID: 128.92.39.240
```

The route had been installed in the routing table on PE2 and will cause the traffic to be redirected into the DIRTY-VRF instance. Let's take a look at where the traffic will go from there:

```
lab@pe2> show route table DIRTY-VRF 203.0.113.1 extensive

DIRTY-VRF.inet.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
0.0.0.0/0 (1 entry, 1 announced)
TSI:
KRT in-kernel 0.0.0.0/0 -> {indirect(1048575)}
        *BGP    Preference: 170/-101
                Route Distinguisher: 64496:1
                Next hop type: Indirect
                Address: 0x9ab5f58
                Next-hop reference count: 6
                Source: 198.51.100.1
                Next hop type: Router, Next hop index: 594
                Next hop: 192.0.2.8 via ge-0/0/1.0, selected
                Label operation: Push 299776
                Label TTL action: prop-ttl
                Load balance label: Label 299776: None;
                Session Id: 0x1
                Protocol next hop: 198.51.100.1
                Label operation: Push 299776
                Label TTL action: prop-ttl
                Load balance label: Label 299776: None;
                Indirect next hop: 0x9854110 1048575 INH Session ID: 0xc
                State: <Secondary Active Int Ext ProtectionCand>
                Local AS: 64496 Peer AS: 64496
                Age: 22:08:04   Metric2: 1
                Validation State: unverified
                Task: BGP_64496.198.51.100.1+20729
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: target:64496:1
                Import Accepted
                VPN Label: 299776
                Localpref: 100
                Router ID: 128.92.39.248
                Primary Routing Table bgp.l3vpn.0
                Indirect next hops: 1
                        Protocol next hop: 198.51.100.1 Metric: 1
                        Label operation: Push 299776
                        Label TTL action: prop-ttl
                        Load balance label: Label 299776: None;
                        Indirect next hop: 0x9854110 1048575 INH Session ID: 0xc
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 192.0.2.8 via ge-0/0/1.0
                                Session Id: 0x1
             198.51.100.1/32 Originating RIB: inet.3
               Metric: 1      Node path count: 1
               Forwarding nexthops: 1
               Nexthop: 192.0.2.8 via ge-0/0/1.0
```

Traffic in the DIRTY-VRF destined for 203.0.113.1 will follow the default. It will get tagged with a label of 299776 and will be sent over ge-0/0/1 towards PE1 where the scrubbing center is connected. Let's take a look at this on PE1:

```
lab@pe1> show route table inetflow.0 extensive

inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
203.0.113.1,*/term:1 (1 entry, 1 announced)
TSI:
KRT in dfwd;
Action(s): routing-instance DIRTY-VRF,count
        *BGP    Preference: 170/-101
                Next hop type: Fictitious
                Address: 0x9358c04
                Next-hop reference count: 1
                State: <Active Int Ext>
                Local AS: 64496 Peer AS: 64496
                Age: 1d 21:48:33
                Validation State: unverified
                Task: BGP_64496.198.51.100.5+45824
                Announcement bits (1): 0-Flow
                AS path: I
                Communities: 64496:85 redirect:64496:1
                Accepted
                Localpref: 100
                Router ID: 128.92.39.240
```

Again, any traffic entering PE1 in the global inet.0 table will be sent over to DIRTY-VRF:

```
lab@pe1> show route table DIRTY-VRF 203.0.113.1 extensive

DIRTY-VRF.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
0.0.0.0/0 (1 entry, 1 announced)
TSI:
KRT in-kernel 0.0.0.0/0 -> {192.0.2.7}
Page 0 idx 0, (group ibgp type Internal) Type 1 val 0x966f368 (adv_entry)
   Advertised metrics:
     Flags: Nexthop Change
     Nexthop: Self
     Localpref: 100
     AS path: [64496] I
     Communities: target:64496:1
Path 0.0.0.0 Vector len 4.  Val: 0
        *Static Preference: 5
                Next hop type: Router, Next hop index: 559
                Address: 0x9ab5ae0
                Next-hop reference count: 4
                Next hop: 192.0.2.7 via ge-0/0/2.0, selected
                Session Id: 0x8
                State: <Active Int Ext>
                Age: 1d 20:58:57
                Validation State: unverified
                Task: RT
                Announcement bits (2): 0-KRT 2-BGP_RT_Background
                AS path: I
```

Once inside the VRF table, the traffic will follow the default route out of ge-0/0/2 to the IDP appliance:

```
lab@pe1> show route 203.0.113.1 table inet.0

inet.0: 63 destinations, 63 routes (63 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

203.0.113.0/24     *[BGP/170] 22:29:10, localpref 100, from 198.51.100.3
                      AS path: 64511 I, validation-state: unverified
to 192.0.2.3 via ge-0/0/3.0
```

Once the cleaned traffic arrives back on PE1 from the IDP appliance, it will follow the normal unicast route in inet.0 that is being advertised by the enterprise customer's CE1 device.

Verifying Flow Validation

Once you have checked that the routes are being propagated correctly, turn your attention to the flow validation. Remember that in this design, the default flow validation has been disabled using `no-validate` and a route policy was used instead. Due to this, you should *not* have any validated routes when using the operational `show route flow validation detail` command:

```
lab@pe1> show route flow validation detail

inet.0:
```

Instead, you have to rely on route checking as you did in the previous section, along with checking the firewall filters in the next section.

Verifying Firewall Filters

Again, once the routes have been validated and propagated, they automatically get turned into firewall filters. Look at this output command:

```
lab@pe1> show firewall filter __flowspec_default_inet__

Filter: __flowspec_default_inet__
Counters:
Name                                   Bytes      Packets
203.0.113.1,*                              0            0
```

In this particular case, the firewall filter does filter-based forwarding to send the packets over to the DIRTY-VRF instance instead of discarding them as it did in previous chapters.

Verifying System Logging

The final thing to check is to see if you are exceeding the number of prefixes you have set as the limit. In order to test this, set the maximum-prefix on PE1 down to 1 and advertise a second route from the route reflector:

```
routing-options {
   rib inetflow.0 {
      maximum-prefixes 1 threshold 90;
   }
}
```

Now take a look at the logs with this command:

```
lab@pe1> show log messages | match PREFIX_LIMIT
Sep 17 08:53:37  pe1 rpd[2806]: RPD_RT_PREFIX_LIMIT_REACHED: Number of prefixes (1) in
table inetflow.0 reached configured maximum (1)
```

And finally, set the limit back to 10,000 so you can see the log clear:

```
routing-options {
    rib inetflow.0 {
        maximum-prefixes 100000 threshold 90;
    }
}

lab@pe1> show log messages | match PREFIX_LIMIT
Sep 17 08:57:07  pe1 rpd[2806]: RPD_RT_PREFIX_LIMIT_BELOW: Number of prefixes (1) in
table inetflow.0 is now less than the configured maximum (10000)
```

Conclusion

The BGP FlowSpec protocol is the building block for many different DDoS mitigation designs. Whether you are an enterprise looking to automate mitigating DDoS attacks inside your network or a service provider looking to design a DDoS mitigation solution for your customers, BGP FlowSpec is a tool worth taking a look at.

Hopefully the information provided in this book will help you get started designing the perfect solution for your own organization.

MORE? Here are some additional links for further reading:

- Juniper Technical Documentation. *Example: Enabling BGP to Carry Flow-Specification Routes* (https://www.juniper.net/documentation/en_US/junos15.1/topics/example/routing-bgp-flow-specification-routes.html)
- Juniper Technical Documentation. *BGP Feature Guide for Routing Devices* (https://www.juniper.net/techpubs/en_US/junos15.1/information-products/pathway-pages/config-guide-routing/config-guide-routing-bgp.pdf)