BGP Flowspec Tutorial

Mark Brochu
Internet2
DDoS Mitigation Evolution

• Manual ACLs and Null Routes
  – Victim detects incoming attack, either by automation or reacting to an outage
  – Usually there is an upstream segment that’s congested, and usually this link is not under the victim’s control (ie ISP link)
  – Victim contacts the ISP to install an ACL or block the traffic.
  – Downsides: ACL’s aren’t granular enough. Even when an ideal ACL is created that minimizes fallout, the attacker can usually change vectors.
  – Human communication chain takes time and introduces the probability for error.

• Remote Trigger Black Hole (RFC3882/7999)
  – Victim sends a bgp prefix containing the destination of the attack, usually a /32. The prefix is tagged with a community, usually identified ahead of time in provider documentation, a well known community is 65535:666, although not necessarily supported by all ISPs.
  – ISP modifies next hop of prefix to a designated non-routable prefix which in turn gets routed to discard.
  – Still useful in emergency situations to protect the network as a whole, but effectively completes the DOS attack by taking the target IP address(s) offline.
DDoS Mitigation Evolution

  – Combines the benefits of ACLs with Null Routing. Think of it as light QoS with remote signaling.
  – Allows for filtering based on several types of traffic masks:
    • Destination Prefix
    • Source Prefix
    • IP Protocol (can also be used with operands such as AND/OR/Length/Less Than/Greater Than)
    • TCP/UDP Port (source/dst/either)
    • ICMP Type
    • ICMP Code
    • TCP Flags
    • Packet Length
    • Diffserv Codepoint
    • Fragmentation
  – Actions
    • Rate Limit - in bytes per second (set to zero to drop)
    • Traffic-action - Sample
    • Redirect - Extended Community (Route target) used to redirect traffic to a VRF.
    • Traffic-marking - Modify DSCP Values.
BGP Flowspec Advantages

  - Integration with BGP - Offers several advantages
    - Rules are easily propagated upstream via standard routing rules
    - Able to filter routes based on standard prefix lists and policy maps
    - Allows for centralized route server to generate policies
    - BGP Monitoring Protocol (BMP) offers ability to track events
    - Automation solutions such as PyEz
  - Multi-Vendor Support
    - Cisco
    - Juniper
    - Nokia (Alcatel/Lucent)
  - Leverages the routers ACL/Firewall Subsystems - Granular/High Performance
Configuration of BGP Flowspec - Juniper

- Signaling is done via well known protocol (MPBGP)
- New AFI/SAFI (inetflow) with NLRI’s corresponding to the types of traffic to match, and actions
- Configurable via standard CLI by using intuitive syntax (at least on Juniper anyway)

**Juniper Configuration** - Installing a flowspec route (placed into `<vrfname>.inetflow.0`)
```
set routing-options flow route block-10.131.1.1 match destination 10.131.1.1/32
set routing-options flow route block-10.131.1.1 match protocol icmp
set routing-options flow route block-10.131.1.1 match icmp-type echo-request
set routing-options flow route block-10.131.1.1 then discard
set routing-options flow term-order standard
```

- **Juniper Configuration** - BGP Configuration for route advertisement
```
user@host# show protocols
bgp {
    group core {
        family inet {
            unicast;
            flow;
        }
    }
}
```
Configuration of BGP Flowspec - Cisco

- **BGP Configuration**
  
  ```
  router bgp 65000
  address-family ipv4 flowspec
  
  neighbor 10.99.99.99
  remote-as 65001
  address-family ipv4 flowspec
  
  ```

- **Flowspec Rule Creation**
  
  ```
  class-map type traffic match-all block-10.131.1.1
  match destination-address ipv4 10.131.1.1/32
  match protocol 1
  match ipv4 icmp
  
  end-class-map
  
  policy-map type pbr block 10.131.1.1
  class type traffic block-10.131.1.1
  drop
  
  ```

  ```
  class class-default
  
  end-policy-map
  
  flowspec-map
  local-install interface-all
  address-family ipv4
  service-policy type pbr block-10.131.1.1
  ```

- **Cisco**

- **TechEX18**
  
  ORLANDO FL | OCTOBER 15-18
Classic DNS Amplification Attack

Attacker 10.131.1.1 (Spoofed) -> x.x.x.x
dig ANY isc.org @x.x.x.x

Service Provider Network

x.x.x.x:53 -> 10.131.1.1:1234
3K+ Byte DNS response

1-100G of Amplified DNS responses

Campus or Enterprise Network

Victim 10.131.1.1
Flowspec Advertisement - Step 1
Victim Network Creates flowspec rule

Prefix | Nexthop | MED | Lclpref | AS path
--- | --- | --- | --- | ---
10.131.1.1 | * | * | * | *

Self 100 1

1-100G of Amplified DNS responses

*,10.131.1.1,proto=17,prot=53/term:2 (1 entry, 1 announced)
*Flow Preference: 5
Next hop type: Fictitious
Address: 0x904d5e4
Next-hop reference count: 2
State: <Active>
Local AS: 65000
Age: 38
Validation State: unverified
Task: RT Flow
Announcement bits (2): 0-Flow 1-BGP_RT_Bacground
AS path: 1
AS path: Recorded
Communities: traffic-rate:0.1875

x.x.x.x:53 -&gt; 10.131.1.1:1234
3K+ Byte DNS response

dig ANY isc.org @x.x.x.x
Flowspec Advertisement - Step 2
ISP Receives and validates rule propagates throughout backbone

Prefix  | Nexthop   | MED  | Lclpref | AS path
10.131.1.1,*,proto=17,port=53,term:1 | Self 100 | I

Prefix  | Nexthop   | MED  | Lclpref | AS path
10.131.1.1,*,proto=17,port=53,term:1 | Self 100 | I

*,10.131.1.1,proto=17,port=53,term:2 (1 entry, 1 announced)
Flow Preference: 5
Next hop type: Fictitious
Address: 0x904d5e4
Next-hop reference count: 2
State: <Active>
Local AS: 65000
Age: 38
Validation State: unverified
Task: RT Flow
Announcement bits (2): 0-Flow 1-BGP_RT_Background
AS path: I
AS path: Recorded
Communities: traffic-rate:0:1875

Attacker 10.131.1.1 (Spoofed) -> x.x.x.x
dig ANY isc.org @x.x.x.x

x.x.x.x:53 -> 10.131.1.1:1234
3K+ Byte DNS response
BGP Flowspec - validation of routes

Because creating a flowspec rule for dst=8.8.8.8/32 could be bad.

Per RFC:

A flow specification NLRI must be validated such that it is considered feasible if and only if:

a) “The originator of the flow specification matches the originator of the best-match unicast route for the destination prefix embedded in the flow specification.”

In other words - If I advertise a flowspec route for a.a.a.a/24, I also need to be listed as the best-match for a.a.a.a/24.

b) 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 a).

In other words - If there exists a a route for a.a.a.B/32 from a different ASN (Say a DDOS Mitigation Provider), the flowspec rule will not be honored.
BGP Flowspec - validation of routes

Most (all?) implementations of flowspec allow you to disable validation.

Why would you want to do this?

- You might want to create a dedicated peering only using the flowspec afi/safi. This might be the same router connected to your upstream, or a route server within your network. In this situation, no route will ever be validated since there will never be any corresponding unicast routes from that peer.
- You’re an ISP and present a portal or some API for customers to insert flowspec routes.
BGP Flowspec - Monitoring

```bash
dmpr@R2-RE0# run show route table inetflow.0 detail
inetflow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
10.3.255.2,130.130/16,proto=1/term:1 (1 entry, 1 announced)
  *Flow   Preference: 5
    Next hop type: Fictitious
    Address: 0x8df4664
    Next-hop reference count: 1
    State: <Active>
    Local AS: 4259905000
    Age: 8:34
    Task: RT Flow
    Announcement bits (2): 0-Flow 1-BGP_RT_Background
    AS path: I
    Communities: traffic-rate:0:0• show firewall filter
__flowspec_default_inet__
```

```bash
dmpr@R1-RE0#run show firewall filter __FlowSpec_default_inet__
Filter: __FlowSpec_default_inet__
Counters:
Name                                                Bytes              Packets
*,10.1.1.1,proto=17,port=53                             0                    0
Policers:
Name                                                Bytes              Packets
15K_*,10.1.1.1,proto=17,port=53                         0                    0
```
BGP Flowspec - Monitoring

user@host# run show system resource-monitor fpc
FPC Resource Usage Summary

Free Heap Mem Watermark : 20 %
Free NH Mem Watermark : 20 %
Free Filter Mem Watermark : 20 %

* - Watermark reached

<table>
<thead>
<tr>
<th>Slot #</th>
<th>Heap % Free</th>
<th>ENCAP mem</th>
<th>NH mem % Free</th>
<th>FW mem % Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>94</td>
<td>NA</td>
<td>83</td>
<td>99</td>
</tr>
</tbody>
</table>

NPC7(gallon vty)# sho jnh 0 pool usage

EDMEM overall usage:

[NH////////////////|FW////////|CNTR//////////|HASH//////////|ENCAPS////////|--------]
0 8.0 12.0 18.0 24.7 28.8 32.0M

Next Hop

[*******************************************************************] 8.0M (84% | 16%)

Firewall

[*]-------------------[RRRRRRRRRRRRRRRRRRR] 4.0M (3% | 97%)

Counters

[*******************************************************************] 6.0M (38% | 62%)

HASH

[*******************************************************************] 6.7M (100% | 0%)

ENCAPS

[*******************************************************************] 4.1M (100% | 0%)

Shared Memory - NH/FW/CNTR/HASH/ENCAPS

[*******************************************************************] 7.2M (48% | 52%)

DMEM overall usage:

[0 0.0M]
BGP Flowspec - Concerns with Adoption

• SP side: Customers might advertise 8.8.8.8.
• Enterprise/DC side - Security team might advertise 8.8.8.8.
• Turning routers into security devices - potentially unpredictable behavior, especially if automated.
• New protocol means learning curve for operational staff - one more thing to check if there are outages
• Installing flowspec rules = resources used - high speed TCAM/memory for firewall rules. Need to monitor
Practical Example: Automation of Flowspec using PyEz

What is PyEz?


Junos PyEZ is a microframework for Python that enables you to manage and automate devices running the Junos operating system (Junos OS). Junos PyEZ is designed to provide the capabilities that a user would have on the Junos OS command-line interface (CLI) in an environment built for automation tasks. Junos PyEZ does not require extensive knowledge of Junos OS or the Junos XML APIs.

Junos PyEZ enables you to manage devices running Junos OS using the familiarity of Python. However, you do not have to be an experienced programmer to use Junos PyEZ. Non-programmers can quickly execute simple commands in Python interactive mode, and more experienced programmers can opt to create more complex, robust, and reusable programs to perform tasks.

PyEz will allow us to connect to our Juniper router remotely using the NETCONF protocol, and push the appropriate configuration to the device using a template file (jinja2). This will allow us to add and delete flowspec routes.
Practical Example: Automation of Flowspec using PyEz

What we’ll need:

- A management server running Python 2.6, Python 2.7, or Python 3.4
- The following OS python/dev packages:
  - Ubuntu: python-pip, python-dev, libxml2-dev, libxslt-dev, libssl-dev, libffi-dev
  - Windows: pip, pycrypto, ecdsa
- The PyEz package
- The python script, jinja2 template, and YAML file for adding and deleting route
  - https://forums.juniper.net/jnet/attachments/jnet/Archive/60/2/set-flow-routes.zip
  - https://forums.juniper.net/jnet/attachments/jnet/Archive/60/3/delete-flow-routes.zip

root@nms:~# apt-get install python-pip python-dev libxml2-dev libxslt-dev libssl-dev libffi-dev
root@nms:~# pip install junos-eznc
set-flow-routes.py

# $Id$
# Simple Python script using YAML and Junos PyEz
# to configure local flow routes as per template

from jnpr.junos.utils.config import Config
from jnpr.junos import Device
from pprint import pprint
from jnpr.junos.factory import loadyaml
from jnpr.junos.op import *

import yaml
import sys
from glob import glob
from jinja2 import Template

# YAML file.
with open(glob('set-flow-route.yml')[0]) as fh:
    data = yaml.load(fh.read())

# Jinja2 template file.
with open(glob('set-flow-route.j2')[0]) as t_fh:
    t_format = t_fh.read()

routesnippet = Template(t_format)
print(routesnippet.render(data))
Set-flow-routes.py (contd)

# Open netconf connection with RR
dev = Device(host='r6', user='juniper', password='Clouds')
dev.open()

# Bind and lock configuration and load it
dev.bind(cfg=Config)
dev.cfg.lock()
dev.cfg.load(template_path='set-flow-route.j2', template_vars=data, format='text', merge=True)

# Commit and unlock
dev.cfg.commit()
dev.cfg.unlock()

# Close netconf connection
dev.close()
---
# $Id$
# YAML file covering all possible variables

# Name of flow route
flow_route_name: foo2

# Destination prefix and mask in format: A.B.C.D/Z
destination_address_and_mask: 198.51.100.1/32

# DSCP in decimal format
dscp:

# Destination port as alias or in decimal format
destination_port: 53

# Fragment Junos OS knobs
fragment:

# ICMP code as alias or in decimal format
icmp_code:

# ICMP type as alias or in decimal format
icmp_type:
Set-flow-route.yml (contd)

# Full L3 length in bytes
packet_length:

# Source or destination port as alias or in decimal format
port:

# Protocol as alias or in decimal format
protocol: udp

# Source prefix and mask in format: A.B.C.D/Z
source_address_and_mask:

# Source port as alias or in decimal format
source_port:

# TCP flags with Junos OS knobs
tcp_flags:

# Include 'accept' action
accept:

# Include existing community to be tagged action
community_name:

# Include 'discard' action
discard: discard

# Include 'next-term' action
next_term:

# Include policer name to be applied as action
rate_limit:

# Include existing RT community to select VRF for redirection
redirect_RT:

# Include 'sample' action
sample:
```java
set-flow-route.j2

# Flow-route configure via vRR

routings-options {

  flow {
    route {{flow_route_name}} {
      match {
        destination {{destination_address_and_mask}};
        {% if destination_port is defined and destination_port !=None %}
          destination-port {{destination_port}};
        {% endif %}
        {% if dscp is defined and dscp !=None %}
          dscp {{dscp}};
        {% endif %}
        {% if fragment is defined and fragment !=None %}
          fragment {{fragment}};
        {% endif %}
        {% if icmp_code is defined and icmp_code !=None %}
          icmp-code {{icmp_code}};
        {% endif %}
        {% if icmp_type is defined and icmp_type !=None %}
          icmp-type {{icmp_type}};
        {% endif %}
        {% if packet_length is defined and packet_length !=None %}
          packet-length {{packet_length}};
        {% endif %}
        {% if port is defined and port !=None %}
          port {{port}};
        {% endif %}
        {% if protocol is defined and protocol !=None %}
          protocol {{protocol}};
        {% endif %}
        {% if source_address_and_mask is defined and source_address_and_mask !=None %}
          source {{source_address_and_mask}};
        {% endif %}
        {% if source_port is defined and source_port !=None %}
          source-port {{source_port}};
        {% endif %}
        {% if tcp_flags is defined and tcp_flags !=None %}
          tcp-flags {{tcp_flags}};
        {% endif %}
      }
    }
  }
}
```
then {
    {% if accept is defined and accept != None %}
    {{accept}};
    {% endif %}
    {% if community_name is defined and community_name != None %}
    community {{community_name}};
    {% endif %}
    {% if discard is defined and discard != None %}
    {{discard}};
    {% endif %}
    {% if next_term is defined and next_term != None %}
    {{next_term}};
    {% endif %}
    {% if rate_limit is defined and rate_limit != None %}
    rate-limit {{rate_limit}};
    {% endif %}
    {% if redirect_RT is defined and redirect_RT != None %}
    routing-instance {{redirect_RT}};
    {% endif %}
    {% if sample is defined and sample != None %}
    {{sample}};
    {% endif %}
}