

OCCAID

Remote-Triggered BGP Blackhole For IPv6



<http://www.occaid.org>

Discussion of a security tool to handle current and future IPv6 network abuse.

October 5, 2004

Automated BGP Blackhole isn't new..

- Some operators long before AS701 did this to remotely trigger routing of traffic to a null adjacency or to a GRE tunnel for interception. It wasn't until Chris Morrow and Brian Gemberling who showed the world of this methodology in NANOG where it grabbed attention of many people.
- More operators after AS701 have started doing this as well. Rumored to include AS3549, AS3356, AS4436, AS3561, and so on...
- Helped their downstream customers greatly in saving the network during a massive DDoS attack by remotely filtering the destination address using a BGP community, instead of waiting back and forth in the support telephone queue.
- Today it just recently have become an industry best practice: RFC3882 or see <http://www.faqs.org/rfcs/rfc3882.html>
- Also evolved into integrated use of a sink hole or "darknet" as an extended security tool for IDS or packet analysis of the attack flow. Furthermore, many operators have also begun using combination of uRPF and remote blackhole trigger to create a pseudo automatic source address based firewalls in their AS.

Automated BGP Blackhole for IPv6?

Why??

- Sooner or later, security issues of IPv4 will become common for IPv6 as well.
- We have actually heard of rumors that sophisticated bot-nets exist -- where the infected worm can remotely issue "ipv6 install" command over DOS prompt in Windows machines, then join an IRC network for DDoS flooding through use of 6to4 relays.
- And we *think* we might have seen one.. It wasn't until just recently for us to find out that DDoS for IPv6 does indeed exist:

```
19:14:37.286554 2002:da0c:8b93::da0c:8b93 > 3ffe:401d:2004::2: frag (33264|1232)
19:14:37.286580 2002:da0c:ec81::da0c:ec81 > 3ffe:401d:2004::2: frag (17248|1232)
19:14:37.286692 2002:da0c:8b93::da0c:8b93 > 3ffe:401d:2004::2: frag (32032|1232)
19:14:37.287044 2002:ddc3:44b::ddc3:44b > 3ffe:401d:2004::2: frag (50512|1232)
```

- The above sample attack had sustained maximum rate of 30Mbps, which is fairly small attack for many IPv4 network operators. However, it was big enough to affect the downstream network, furthermore the traffic pattern and source of the traffic was very well distributed to our surprise. It certainly does seem like a bot net of some sort that is now capable of IPv6 DDoS, by abusing public 6to4 anycast relays.
- We believe it is the time for not just us, but other active IPv6 operators to consider installing certain set of security tools for dealing with sudden network abuse like these, just like how they have set it up in their IPv4 networks.

- Fortunately, at the time of the above attack, we had already completed setup of an IPv6 remote blackhole community for our BGP routing. The community is 30071:666.
- Using the remote trigger method, we were able to null-route the destination and neutralize the attack with a very rapid response.
- In the next pages, we will discuss how we have the remote triggered backhole mechanism configured throughout our network, geared for IPv6.

On Every Router..

- We have chosen 666::/64 as our blackhole next-hop prefix to be used over BGP.

```
ipv6 route 666::/64 Null0
```

In case of FreeBSD based routers running Quagga, at the time we implemented this, we had no Null0 capability. Therefore, we assigned "665::/64" under "interface ds0", then did this:

```
ipv6 route 666::/64 ds0
```

This is a temporary hack for now, but later as blackhole routing in our platform gets fixed, we will correct this to use the first "Null0" method above.

At The Core.. (iBGP)

- For triggering blackhole by injecting a null-route at one of the core routers over iBGP, we have inserted the following route-map to each router's running BGP configuration:

```
!
route-map null-comm permit 5
  set community 30071:666 30071:9001
  set local-preference 400
  set ipv6 next-hop local 666::1
  set ipv6 next-hop global 666::1
!
```

- The important part is that we are setting both local and global IPv6 next-hops to any address within the 666::/64 static blackhole range -- that is what does the trick. The community 30071:666 is applied for identification that this prefix is a null-route, and 30071:9001 in our network means AS30071 internal originated prefix. And local-preference of 400 is there because we use local pref of 400 on any internally originated routes.

The Triggering Charge..

- To actually trigger a null-route, it is pretty simple. Here, we will use the 3ffe:401d:2004::2/128 address as an example, since this address was shown in the DDoS snippet previously. All you have to do is just advertise the network with the null-comm route-map:

```
conf t
ipv6 route 3ffe:401d:2004::2/128 Null0
router bgp 30071
address-family ipv6
network 3ffe:401d:2004::2/128 route-map null-comm
```

- And moments later, suddenly every router on the network begins discarding packets destined to 3ffe:401d:2004::2/128 :-)

At The Edge.. (eBGP)

Allowing your customers to null-route their prefixes

Configuration at the Service Provider's Router

- This is vastly similar to the above setup, but the key difference is that we are seeking for prefixes received from the customer with the 30071:666 community. We have constructed a route-map that we apply in 'inbound' direction on each downstream member's BGP session at our core backbone:

```
!
ip community-list standard c666 permit 30071:666
!
ipv6 prefix-list customer seq 5 permit 3ffe:401d:2004::/48 le 64
!
route-map 30071-MEMBERS6-IN permit 60
  match community c666
  set ipv6 next-hop local 666::1
  set ipv6 next-hop global 666::1
  set local-preference 350
!
route-map 30071-MEMBERS6-IN permit 65
!
!
router bgp 30071
  address-family ipv6
    neighbor < customer peering address > route-map 30071-MEMBERS6-IN in
    neighbor < customer peering address > prefix-list customer in
    neighbor < customer peering address > maximum-prefix 50
  !
```

- Notice how this is similar to the previous setup, except that we are matching for the 30071:666, then upon match, we set our next-hop to the same 666::1 static null route. Now take a closer look at the 30071-MEMBERS6-IN route-map, on "permit 65" statement. We do an implicit permit after matching for 30071:666, so that any regular prefixes that do not match the 30071:666 condition will be permitted through.
- We are accepting up to /64 prefix length from the customer in prefix-list. Depending on your network configuration, you may want to hear up to as long as /128 host route from the customer for blackhole purposes. However you should at least limit the prefixes received by using "maximum-prefix <number>" to prevent accidental BGP meltdown. Because there are 65535 /64's in a /48 prefix, the need for maximum-prefix protection in IPv6 is probably more real than in IPv4 BGP world. However, always adjust this value as necessary and required to YOUR network needs. Furthermore, you should communicate to your customers that you have a prefix limit on their BGP session, so that they can be aware of the fact that their BGP session may get torn down, if they do not be careful with outbound route advertisements.
- Now let's take a look at what to configure on the customer's edge router in the next page.

At The Edge.. (eBGP)

Allowing your customers to null-route their prefixes

Configuration at the Customer Edge Router

- Customer side is pretty simple, just set up a route-map that can be used in the event of an attack, and ensure he is sending communities to his upstream.

```
!  
!  
ipv6 prefix-list ToOccaId seq 5 permit 3ffe:401d:2004::/48 le 64  
!  
route-map null-comm permit 5  
    set community 30071:666  
!  
router bgp 64512  
    address-family ipv6  
        neighbor < upstream provider peering addr > send-community both  
        neighbor < upstream provider peering addr > prefix-list ToOccaId out  
!
```

- Same as above, we are allowing up to /64 to be advertised out. As mentioned previously, feel free to extend this to your desirable prefix length that meets your requirements and needs.
- The key here however is that we have a route-map that will set the community when used using the "network" configuration statement, as well as the "send-community" syntax, which will make the router send its BGP communities to its upstream. This is critical.

At The Edge.. (eBGP)

Allowing your customers to null-route their prefixes

Triggering the Blackhole at the Customer Edge

- Exactly same as how we did this on service provider's core router at our previous "Triggering Charge" slide. Watch this:

Customer Side:

```
Boston-Core1# conf t
Boston-Core1(config)#
Boston-Core1(config)# ipv6 route 3ffe:401d:2004::/64 Null0
Boston-Core1(config)# router bgp 64512
Boston-Core1(config-router)# address-family ipv6
Boston-Core1(config-router-af)# net 3ffe:401d:2004::/64 route-map null-comm
Boston-Core1(config-router-af)#
```

Provider Side:

```
cr1.sfo2> sh bgp ipv6 3ffe:401d:2004::  
BGP routing table entry for 3ffe:401d:2004::/64  
Paths: (1 available, best #1, table Default-IP-Routing-Table)  
Not advertised to any peer  
64512  
666::1 from 3ffe:401d:f00::1 (65.126.230.2)  
(666::1)  
Origin IGP, metric 77, localpref 350, valid, internal, best  
Community: 30071:666 30071:9003  
  
cr1.sfo2> sh ipv6 route 3ffe:401d:2004::  
Routing entry for 3ffe:401d:2004::/64  
Known via "bgp", distance 200, metric 0, best  
Last update 00:03:52 ago  
* 666::1, via lo0 (recursive is directly connected, ds0)
```

Voila! :-)

Any Questions?

Please feel free to join the mailing list at www.occaid.org and ask any questions you have that are related to this topic.