DDoS Damage Control
Cheap & effective

Job Snijders
job@instituut.net

RIPE68

Who am I?

Job Snijders

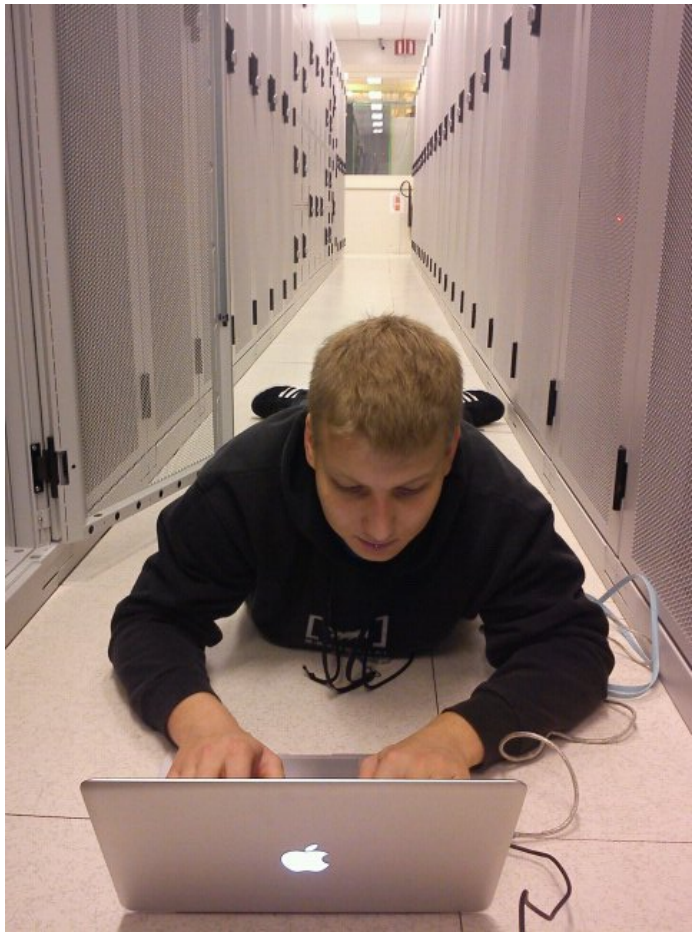
Independent Network Architect

Founder of NLNOG RING

Twitter: @JobSnijders
Email: job@instituut.net

Hobbies: IP Routing, LISP, MPLS, IPv6, RPSL

Shoe size: 45/EU



Agenda

- What is *"selective blackholing"*?
  - Definition
  - Examples based on RIPE ATLAS
- How to set up selective blackholing as a carrier
  - Defining scopes
  - Route-maps
  - Some python

What is selective blackholing?

Selective blackholing ~ selective discarding

1. Use BGP communities to instruct your Service Provider to **discard packets when certain conditions are met**.

2. A region of space-time from which gravity prevents anything, including light, from escaping, **except the colour purple.**

What does it matter?!

Content is most often the victim (webshop, gameserver, webserver)

Most prefixes/content have a geographical significance which *decreases* as distance between the sender and receiver *increases*.

(theorem stems from sFlow data gathered at global ISP).

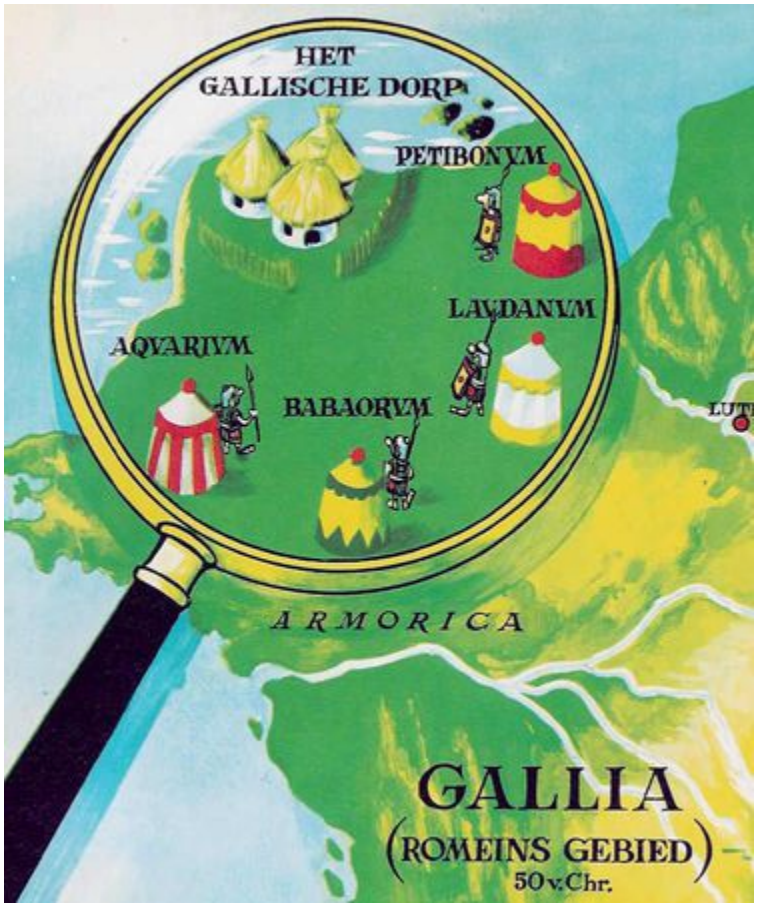
In other words: Chances are a Polish web-shop owner cares most about Polish eyeballs.

What's wrong with normal blackholing?

Classic blackholing is an **all or nothing proposition**:

you throw away all revenue generated by the victim IP address, in order to avoid congesting your upstream links.

Scope is relevant!



Damage control is not mitigation

Selective blackholing should be considered as *yet another tool* in the toolbox when under duress.

Assertion #1:

"it is better to remain partially reachable than not reachable at all during a DDoS attack"

Assertion #2:

"I can take a percentage of the DDoS traffic, but not all"

Effects:
Discard outside 1000 KM radius

Customer connects in Amsterdam, Netherlands
White dot means traffic cannot reach destination
Colored dot implies reachability

Effects:
Discard outside ‘this’ country

White dot means traffic cannot reach destination
Color dot implies reachability, Customer connected in Amsterdam, NL

‘discard outside NL’ is perfect reachability inside NL

Part 2: How to set this up as carrier

Focus on four features:

Scope	End-user BGP community
Outside 'This' country	15562:664
Outside 'This' continent	15562:660
Outside 1000 KM radius	15562:663
Outside 2500 KM radius	15562:662

'This' means 'where the customer interconnection is located'

Distance is from Edge router to Edge router in the SP's network "as the crow flies" (not actual optical fiber path length!). Can only be guaranteed for own backbone

Assign your routers some integers

name	Continent id	ISO31661	City ID	Latitude, Longitude
tky.jp	3	392	46	35.65671, 139.80342
sjo.us	1	840	29	37.44569,-122.16111
dal.us	1	840	33	32.80096, -96.81962
nyc.us	1	840	26	40.71780, -74.00885
lon.uk	2	276	23	51.51173, -0.00197
ams.nl	2	528	20	52.35600, 4.95068
sto.se	2	752	22	59.36264, 17.95560

Router specific configuration – IOS'ish

nyc.us:

```
ip community-list THIS:METRO     seq 5 permit 65123:10026
ip community-list THIS:COUNTRY   seq 5 permit 65123:840
ip community-list THIS:CONTINENT seq 5 permit 65123:1000
```

lon.uk:

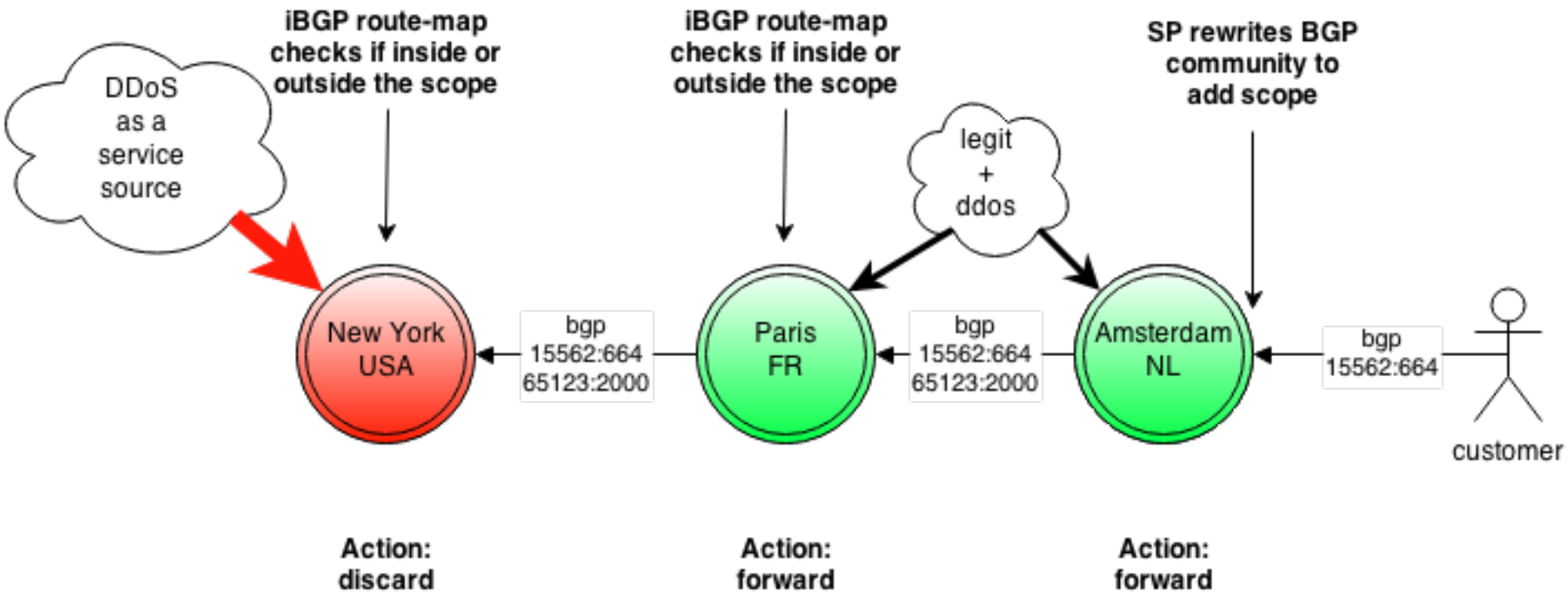
```
ip community-list THIS:METRO     seq 5 permit 65123:20023
ip community-list THIS:COUNTRY   seq 5 permit 65123:276
ip community-list THIS:CONTINENT seq 5 permit 65123:2000
```

ams.nl:

```
ip community-list THIS:METRO     seq 5 permit 65123:20020
ip community-list THIS:COUNTRY   seq 5 permit 65123:528
ip community-list THIS:CONTINENT seq 5 permit 65123:2000
```

…… etc!

What happens where?



iBGP inbound route-map

```
ip route 10.0.0.1 255.255.255.255 null0

route-map INBOUND-IBGP permit 100
  match community  15562:666              ! classic blackhole community
  set ip next-hop 10.0.0.1                ! discard

route-map INBOUND-IBGP permit 200
  match community  15562:660 15562:662 15562:663 15562:664
  continue 1100                           ! Jump over regular 'accept' @ 1000
                                          ! towards scope checking

route-map INBOUND-IBGP permit 1000
                                          ! No match statement == accept anything

route-map INBOUND-IBGP permit 1100
  match community  THIS:METRO THIS:COUNTRY THIS:CONTINENT
                                          ! If match is found, accept prefix and stop
                                          ! evaluating the route-map


route-map INBOUND-IBGP permit 1101        ! Anything that arrives here: discard
  set ip next-hop 10.0.0.1
```

Customer facing route-map

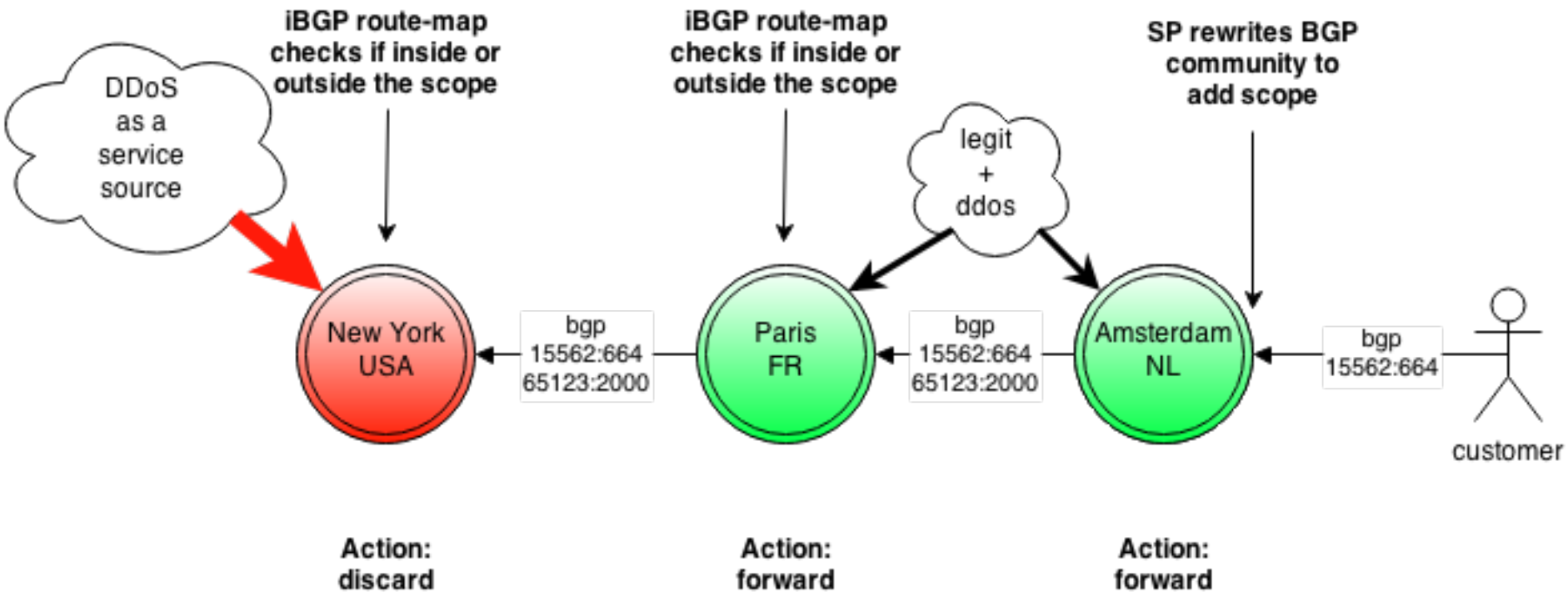
```
01.  route-map IMPORT:FROM:CUSTOMER-A permit 200
02.    match ip address prefix-list CUSTOMER-A-PREFIXES
03.    match community 15562:666
04.    set community  no-export additive
05.    set ip next-hop 10.0.0.1

06.  route-map IMPORT:FROM:CUSTOMER-A permit 300
07.    match ip address prefix-list CUSTOMER-A-PREFIXES
08.    match community SCOPED:ACTION
09.    continue 600              ! Remember this jump !

10.  route-map IMPORT:FROM:CUSTOMER-A permit 400
11.    match ip address prefix-list CUSTOMER-A-PREFIXES
12.    set local-preference 650

13.  route-map IMPORT:FROM:CUSTOMER-A deny 500
```

Customer facing (cont.)

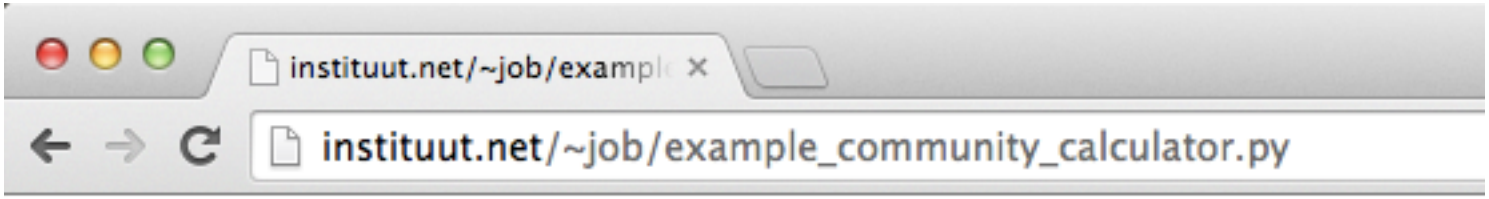
Add/Rewrite scoping information when a ‘scoped action’ is used

```
14.  route-map IMPORT:FROM:CUSTOMER-A permit 600          ! Here is 600 again
15.    match community  OUTSIDE:1000KM:RADIUS:DISCARD    ! 15562:663
16.    set community  65123:10029 additive

17.  route-map IMPORT:FROM:CUSTOMER-A permit 700
18.    match community  OUTSIDE:2500KM:RADIUS:DISCARD    ! 15562:662
19.    set community  65123:10033 65123:10029 additive

20.  route-map IMPORT:FROM:CUSTOMER-A permit 900
21.    match community  OUTSIDE:THIS:COUNTRY:DISCARD     ! 15562:664
22.    set community  65123:840 additive

23.  route-map IMPORT:FROM:CUSTOMER-A permit 1100
24.    match community  OUTSIDE:THIS:CONTINENT:DISCARD   ! 15562:660
25.    set community  65123:1000 additive
```

What happens where?



But wait a second… how do you figure out what needs to be rewritten to… what?




```
#!/usr/bin/env python

import sys
from haversine import haversine
from itertools import combinations

distances_matrix = {}
community_matrix = {}

cmdb = {'r1.tky.jp': {'cont': 3, 'country': 392, 'metro': 46,
                      'latlon': (35.65671, 139.80342)},
        'r1.sjo.us': {'cont': 1, 'country': 840, 'metro': 29,
                      'latlon': (37.44569, -122.16111)},
        'r1.dal.us': {'cont': 1, 'country': 840, 'metro': 33,
                      'latlon': (32.80096, -96.81962)},
        'r1.nyc.us': {'cont': 1, 'country': 840, 'metro': 26,
                      'latlon': (40.71780, -74.00885)},
```

Gratis download!
http://instituut.net/~job/example_community_calculator.py

Proof: Software is cool – SDN finally arrived!

```
derp:~ job$ wget -q http://instituut.net/~job/example_community_calculator.py

derp:~ job$ python example_community_calculator.py

r1.lon.uk - rewrite targets:
1000 km: 65123:20020 65123:276
2500 km: 65123:2000

r1.dal.us - rewrite targets:
1000 km: 65123:10033
2500 km: 65123:840

r1.sjo.us - rewrite targets:
1000 km: 65123:10029
2500 km: 65123:10033 65123:10029

r1.nyc.us - rewrite targets:
1000 km: 65123:10026
2500 km: 65123:10026 65123:10033

<snip>
```

The integers in essence provide groupings of routers, which the software/route-maps use

COMM /RTR	:276	:2000	:10033	:840	:10029	:10026	:30046	:20022	:20020
ams.nl		X							X
lon.uk	X	X							
sto.se		X						X	
nyc.us				X		X			
dal.us			X	X					
sjo.us				X	X				
tky.jp							X		

(incomplete table, but you get the gist…)

Process flow diagram

Considerations

- Automate ***all*** route-map deployments (actually, automate everything!)
- Use or make a CMDB where you store integers
- Selective Blackholing is a pretty advanced feature..
  with very little router specific configuration ☺
- Can be deployed on any vendor. Crappy vendors are
  not an excuse. This requires no extra CAPEX
- Customers don't ask for this
  feature because they don't
  know it exists (yet)
- Saves **both** the service provider
  and customer money: win/win

Questions?

Resources & Credits

Technical narrative in text form:

http://mailman.nanog.org/pipermail/nanog/2014-February/064381.html

I want to thank Saku Ytti, Torsten Blum and Peter van Dijk for contributing to this methodology.