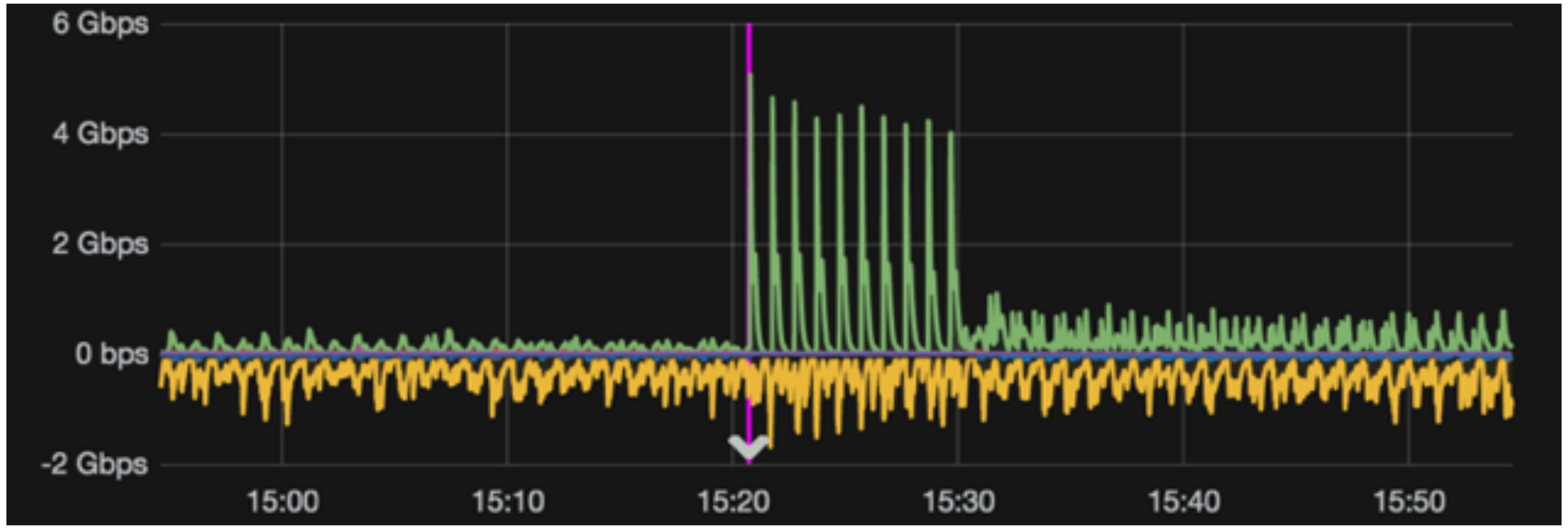
DDoS Attacks

An open-source recipe to improve fast detection and automate mitigation techniques



Vicente De Luca

Sr. Network Engineer

vdeluca@zendesk.com

AS21880 / AS61186

zendesk

Introduction

Tentative to solve:

#1 DDoS fast detection and better monitoring

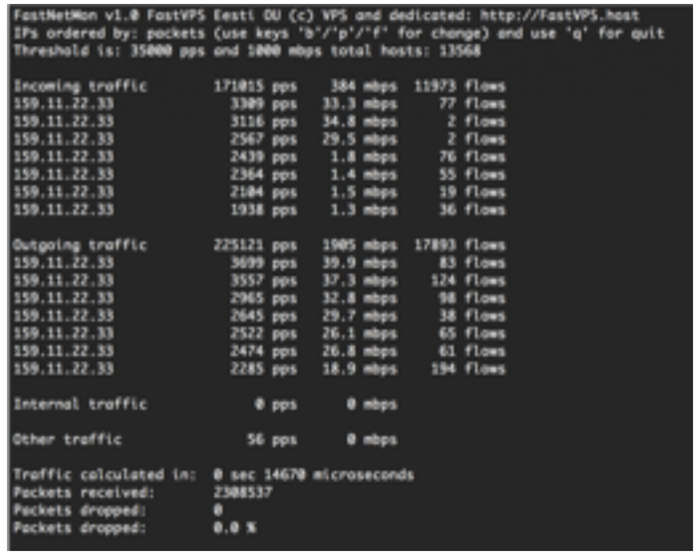
#2 Improve response time on mitigation triggering

Opensource recipe

- **FastNetMon**: main core of our solution. DDoS analyzer with sflow/netflow/mirror support
- **InfluxDB**: Scalable data store for metrics, events, and real-time analytics
- **Grafana**: Gorgeous metric viz, dashboards & editors
- **Redis**: An in-memory database that persists on disk
- **Morgoth**: Metric anomaly detection for Influx databases
- **BIRD**: a fully functional dynamic IP routing daemon
- **Net Healer**: experimental code to "glue" all moving parts, trigger actions and provide API queries

FastNetMon: very fast DDoS analyzer

- collects sFlow (v4/v5), NetFlow (v5/v9/v10), IPFIX and SPAN/mirror
- fast detect IPv4 host above certain threshold
- feed Graphite (compatible) time-series DB
- supports BGP daemons (ExaBGP, GoBGP, others)
- supports Lua processing net flows
- CLI client



```
FastNetMon v1.0 FastVPS Eesti OU (c) VPS and dedicated: http://FastVPS.host
IPs ordered by: packets (use keys 'b'/'p'/'f' for change) and use 'q' for quit
Threshold is: 35000 pps and 1000 mbps total hosts: 13568

Incoming traffic        171015 pps    384 mbps  11973 flows
159.11.22.33              3309 pps   33.3 mbps     77 flows
159.11.22.33              3116 pps   34.8 mbps      2 flows
159.11.22.33              2567 pps   29.5 mbps      2 flows
159.11.22.33              2439 pps    1.8 mbps     76 flows
159.11.22.33              2364 pps    1.4 mbps     55 flows
159.11.22.33              2104 pps    1.5 mbps     19 flows
159.11.22.33              1938 pps    1.3 mbps     36 flows

Outgoing traffic        225121 pps   1905 mbps  17893 flows
159.11.22.33              3699 pps   39.9 mbps     83 flows
159.11.22.33              3557 pps   37.3 mbps    124 flows
159.11.22.33              2965 pps   32.8 mbps     98 flows
159.11.22.33              2645 pps   29.7 mbps     38 flows
159.11.22.33              2522 pps   26.1 mbps     65 flows
159.11.22.33              2474 pps   26.8 mbps     61 flows
159.11.22.33              2285 pps   18.9 mbps    194 flows

Internal traffic             0 pps      0 mbps

Other traffic               56 pps      0 mbps

Traffic calculated in:  0 sec 14670 microseconds
Packets received:       2308537
Packets dropped:        0
Packets dropped:        0.0 %
```

available for CentOS / Ubuntu / Debian / Vyatta / FreeBSD / source / Docker Image

tested with Juniper, Cisco, Extreme, Huawei and Linux (ipt_NETFLOW)

https://github.com/pavel-odintsov/fastnetmon

FastNetMon

Detection Logic:

- number of **pps, mbps and flows** to/from a /32
- number of **fragmented packets** to/from a /32
- number of **tcp syn / udp** to/from a /32
- global / per protocol (udp/tcp/icmp) / per host group (CIDR)
- nDPI support (SPAN/mirror)

Complete support most popular attacks for channel overflow:

- **SYN Flood**
- **UDP Flood** (amplified SSDP, Chargen, DNS, SNMP, NTP, etc)
- **IP Fragmentation**

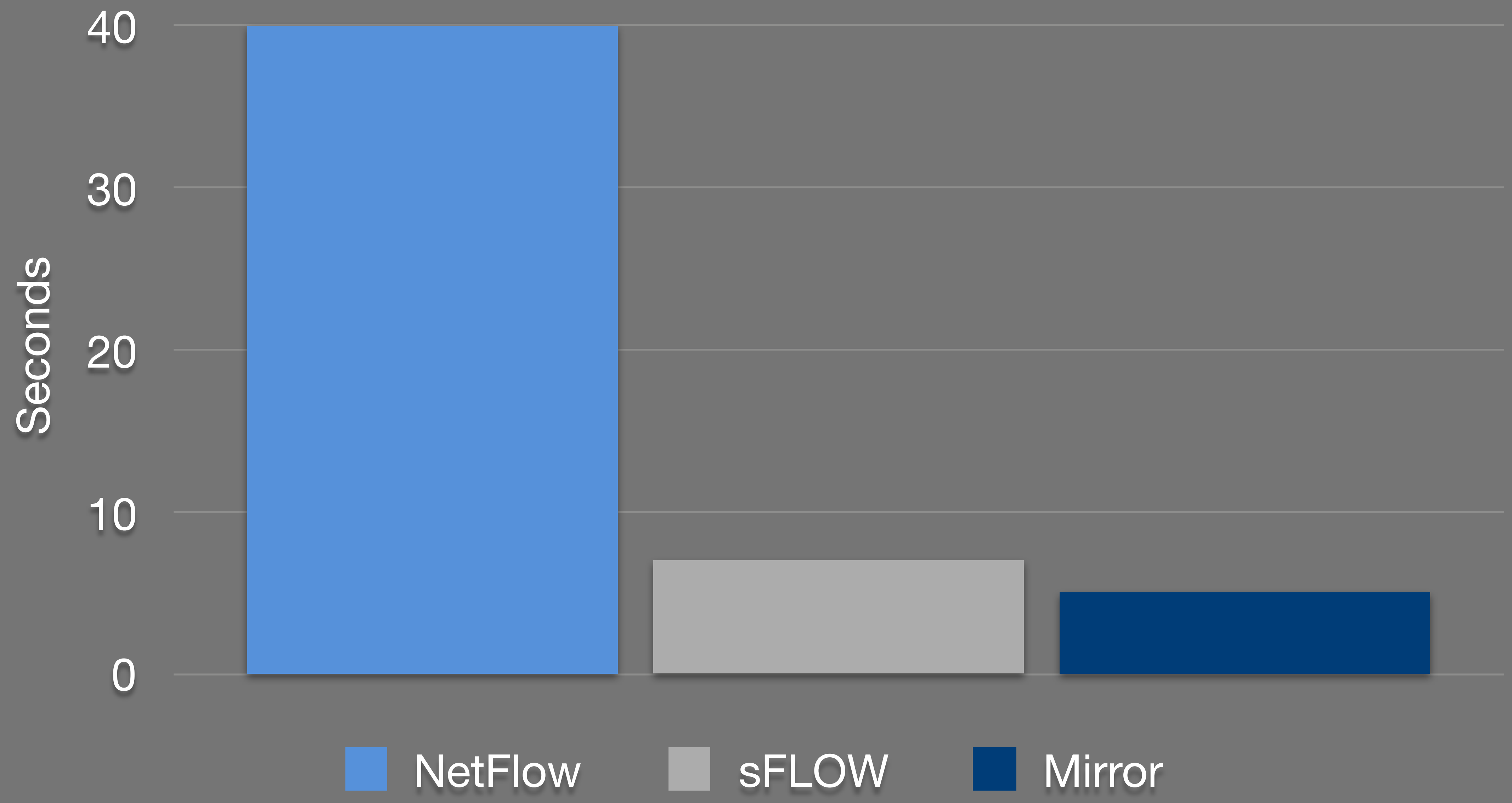
FastNetMon

How it can react during an attack ?

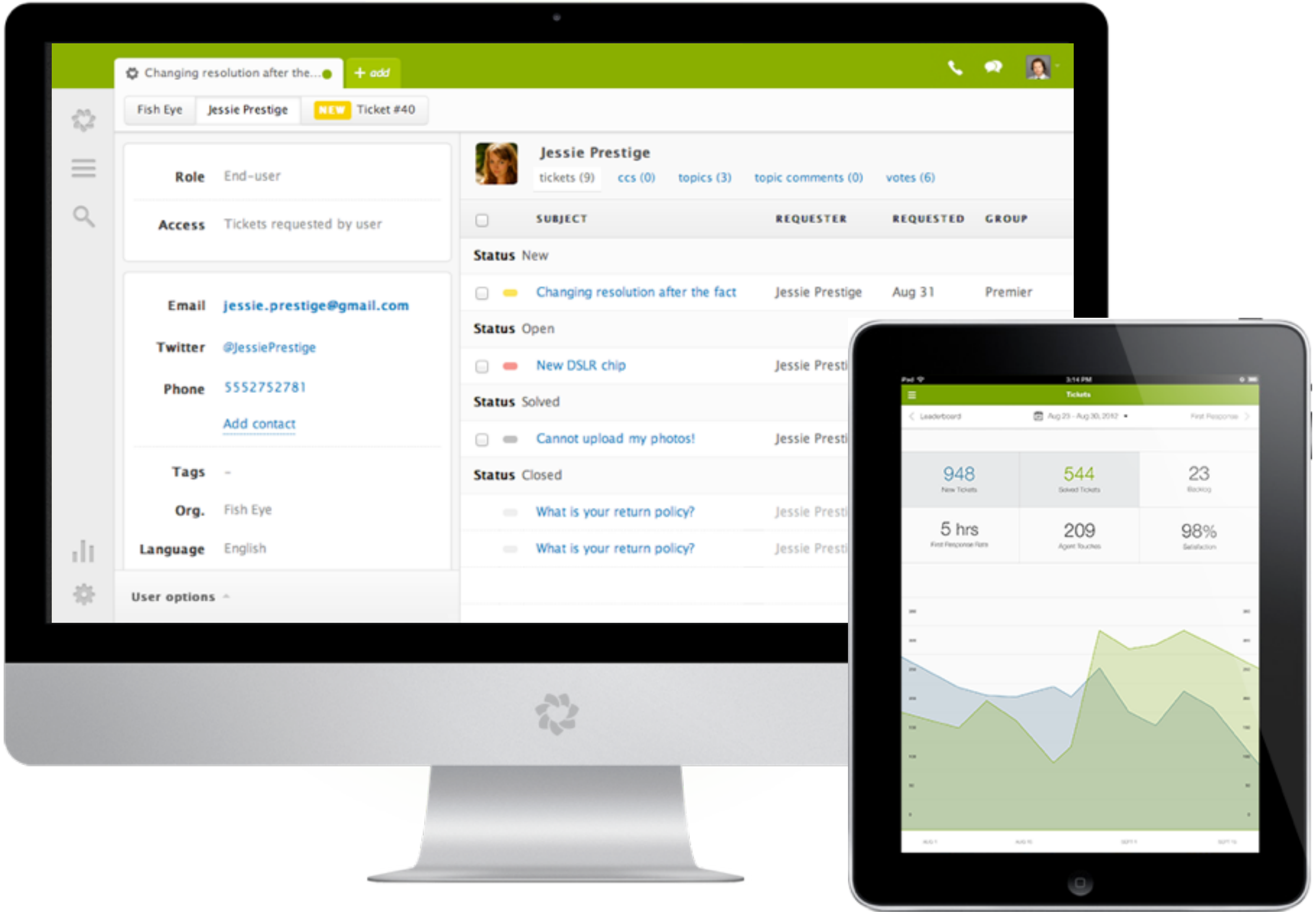
- Custom script (send email, apply an ACL, shutdown a VM, etc etc etc…)
- BGP Announce (community, blackhole, selective blackhole, cloud mitigation)
- BGP Flow Spec (**RFC 5575**) for selective traffic blocking
- Populate Redis DB (target, type, attack peak, tcpdump during attack, etc)

FastNetMon

Detection time per capture backends



our proof-of-concept



Role	End-user
Access	Tickets requested by user
Email	jessie.prestige@gmail.com
Twitter	@JessiePrestige
Phone	5552752781
	Add contact
Tags	-
Org.	Fish Eye
Language	English

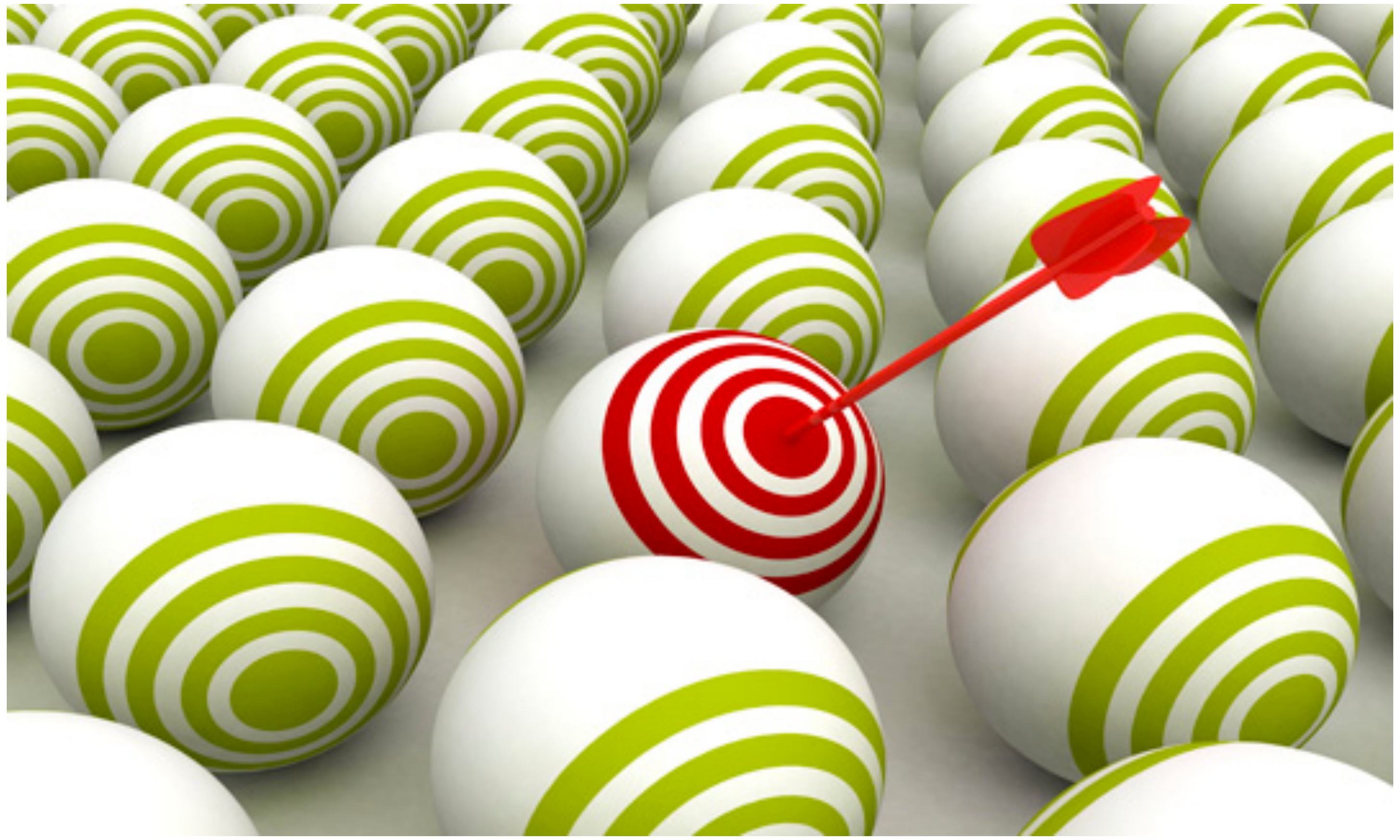
Jessie Prestige

tickets (9) ccs (0) topics (3) topic comments (0) votes (6)

SUBJECT	REQUESTER	REQUESTED	GROUP
Status New			
Changing resolution after the fact	Jessie Prestige	Aug 31	Premier
Status Open			
New DSLR chip	Jessie Presti		
Status Solved			
Cannot upload my photos!	Jessie Presti		
Status Closed			
What is your return policy?	Jessie Presti		
What is your return policy?	Jessie Presti		

948 New Tickets	544 Solved Tickets	23 Backlog
5 hrs First Response Rate	209 Agent Touches	98% Satisfaction

Are we targets?



support.acme.com

CNAME

acme.zendesk.com

The good, the bad and the ugly

The good: mitigation

via cloud provider (BGP)

- multiple scrubbing centers across the globe
- Lots of Tbps of mitigation bandwidth capacity
- presence in IXPs - GRE tunnel established in a safer circuit

some cons:

- Reaction time: Internet route convergence (BGP) —not that bad
- mitigation occurs on incoming only
- always on = $$$

The bad

NOC paged with a site-down alert :(
Troubleshoot to identify an ongoing attack

The ugly

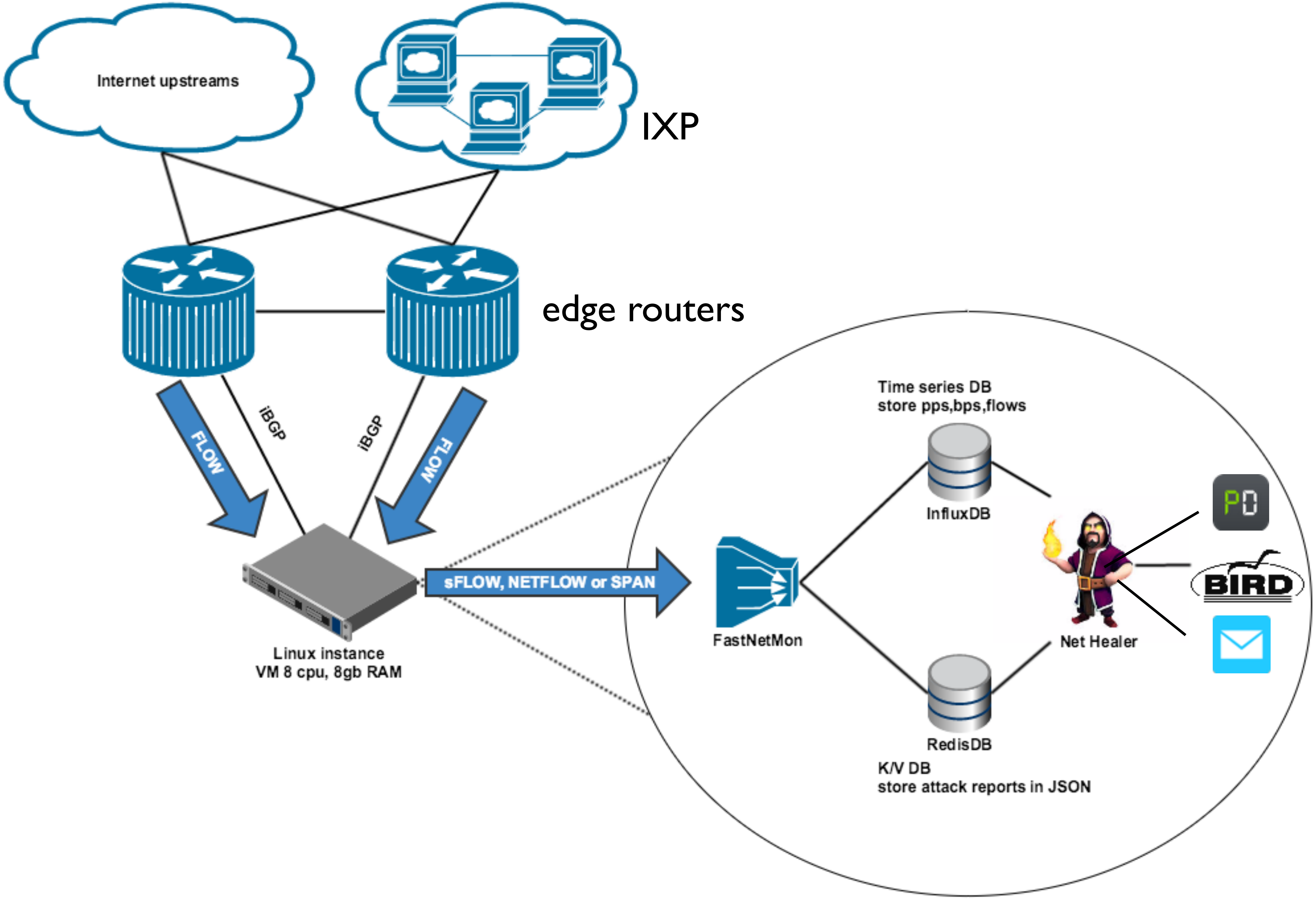
detecting takes "too long", dependent on humans :(

trigger mitigation also needs manual config change

Why not simply buy an already existent and reliable DDoS mitigation appliance?

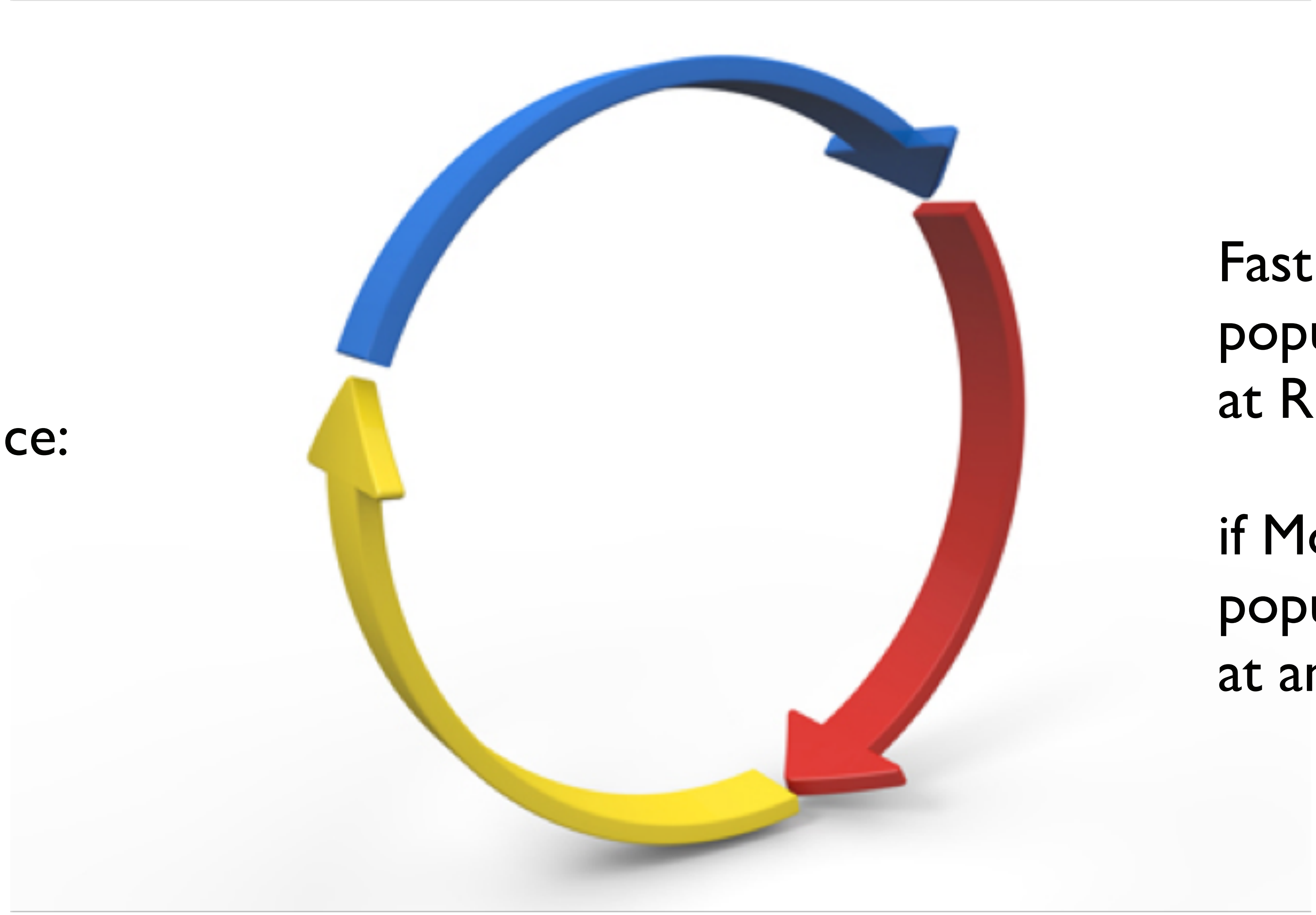
- mostly demands almost dedicated and qualified engineers
- Mitigation available = useless in case of volumetric attack
- High investment for multiple sites ($$$)

Architecture Diagram


Internet upstreams
IXP
edge routers
FLOW
iBGP
iBGP
FLOW
Linux instance
VM 8 cpu, 8gb RAM
sFLOW, NETFLOW or SPAN
Time series DB
store pps,bps,flows
InfluxDB
FastNetMon
Net Healer
BIRD
RedisDB
K/V DB
store attack reports in JSON

DDoS Attack cycle

Attack started



FastNetMon:
populate /32 details
at RedisDB

if Morgoth detects:
populate timestamp
at anomaly InfluxDB

FNM quiescence:
15s per /32

Net Healer watches RedisDB and InfluxDB
if the current attack reports match any policy, trigger the associated action

Net Healer Policies example:

(in a time period of 5 min)

if attack reports = 2 then trigger on call
if attack reports >=4 then inject /24 route

if attack report = 2 + anomaly detected (morgoth)
then trigger on call + inject /24 route

time window / policies can be customized

Why Net Healer ?

- FastNetMon supports all I need, but relies on pre-configured thresholds

- Hard to predict realistic thresholds since our traffic is influenced by our customers activity (out of our control)

- To avoid false positives we prefer to trigger different actions based on each attack cycle phase

- Allow quick integrations like Morgoth x FNM consensus, or API calls such as Pagerduty, etc

Why InfluxDB ?

- Speaks graphite protocol (compatible with FastNetMon)
- Drop in binary - simple install
- Supports cluster mode - easy to scale

Note: Use version >=0.9.6.1 - with tsm1 engine with no batching

InfluxDB

Why Morgoth ?

- Implements non-gaussian algorithm (MGOF) to detect anomaly on data stream metrics

- Takes InfluxDB (bps/pps) fingerprints every chunk of 10s

- Compares the actual fingerprint with the past learned traffic

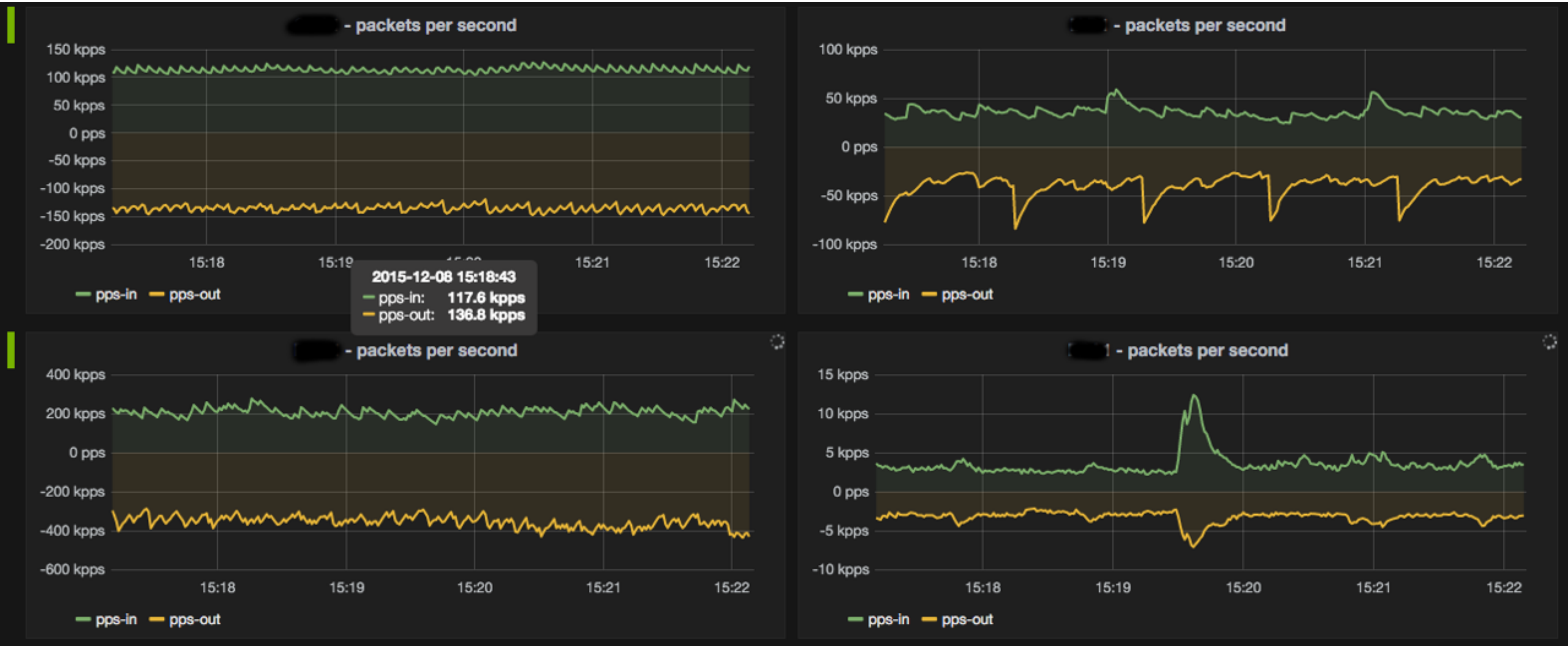
- Anomaly found: Create an alert entry with timestamp

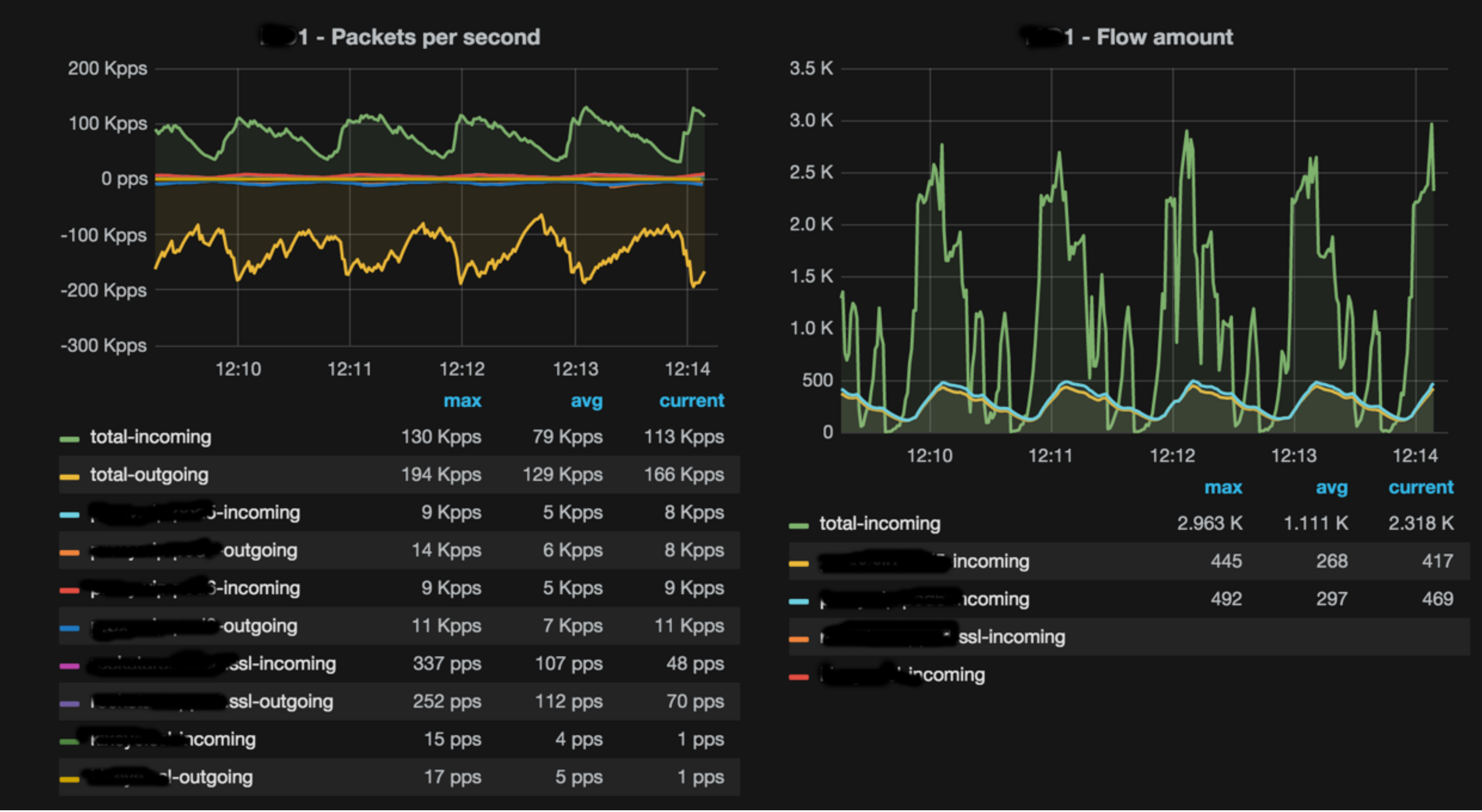
Note: At the time we started developing this project, we were unaware of Influx T.I.C.K stack — We'd love to try Influx Kapacity

Why BIRD ?

- syncing with kernel routing tables (blackhole, mitigate)
- iBGP with edge routers
- Routing policies will decide if RTBH or Advertise to mitigation provider
- friendly to Network Engineers (birdc)

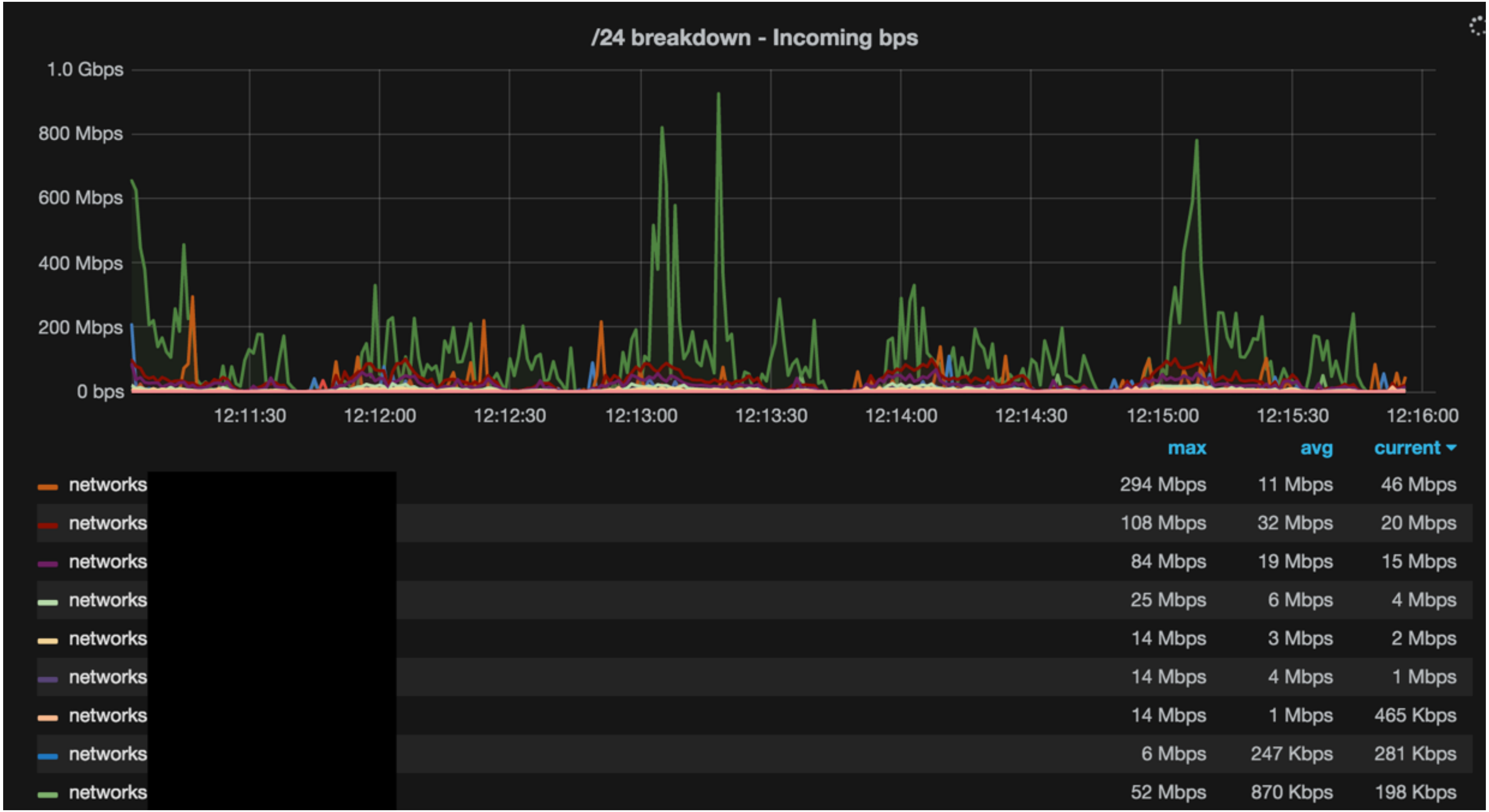
How does it look ?


[redacted] - packets per second
150 kpps
100 kpps
50 kpps
0 pps
-50 kpps
-100 kpps
-150 kpps
-200 kpps
15:18
15:19
15:21
15:22
pps-in
pps-out
2015-12-08 15:18:43
pps-in: 117.6 kpps
pps-out: 136.8 kpps
[redacted] - packets per second
100 kpps
50 kpps
0 pps
-50 kpps
-100 kpps
15:18
15:19
15:20
15:21
15:22
pps-in
pps-out
[redacted] - packets per second
400 kpps
200 kpps
0 pps
-200 kpps
-400 kpps
-600 kpps
15:18
15:19
15:20
15:21
15:22
pps-in
pps-out
[redacted] - packets per second
15 kpps
10 kpps
5 kpps
0 pps
-5 kpps
-10 kpps
15:18
15:19
15:20
15:21
15:22
pps-in
pps-out

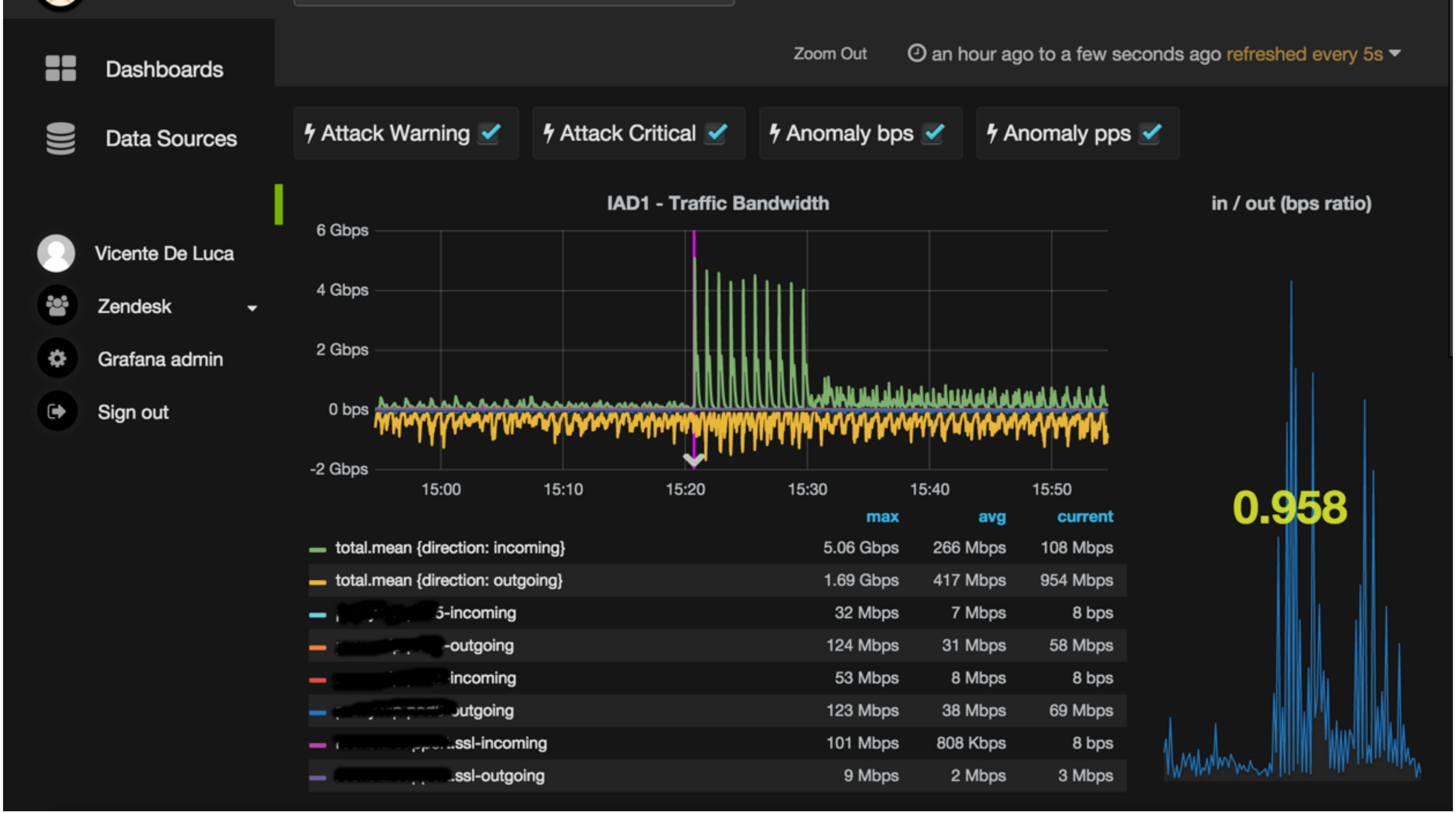


	max	avg	current
🟩 total-incoming	130 Kpps	79 Kpps	113 Kpps
🟨 total-outgoing	194 Kpps	129 Kpps	166 Kpps
🟦 [redacted]-incoming	9 Kpps	5 Kpps	8 Kpps
🟧 [redacted]outgoing	14 Kpps	6 Kpps	8 Kpps
🟥 [redacted]-incoming	9 Kpps	5 Kpps	9 Kpps
🟦 [redacted]-outgoing	11 Kpps	7 Kpps	11 Kpps
🟪 [redacted]ssl-incoming	337 pps	107 pps	48 pps
🟪 [redacted]ssl-outgoing	252 pps	112 pps	70 pps
🟩 [redacted]ncoming	15 pps	4 pps	1 pps
🟨 [redacted]l-outgoing	17 pps	5 pps	1 pps

	max	avg	current
🟩 total-incoming	2.963 K	1.111 K	2.318 K
🟨 [redacted] incoming	445	268	417
🟦 [redacted]ncoming	492	297	469
🟧 [redacted] ssl-incoming			
🟥 [redacted]ncoming			

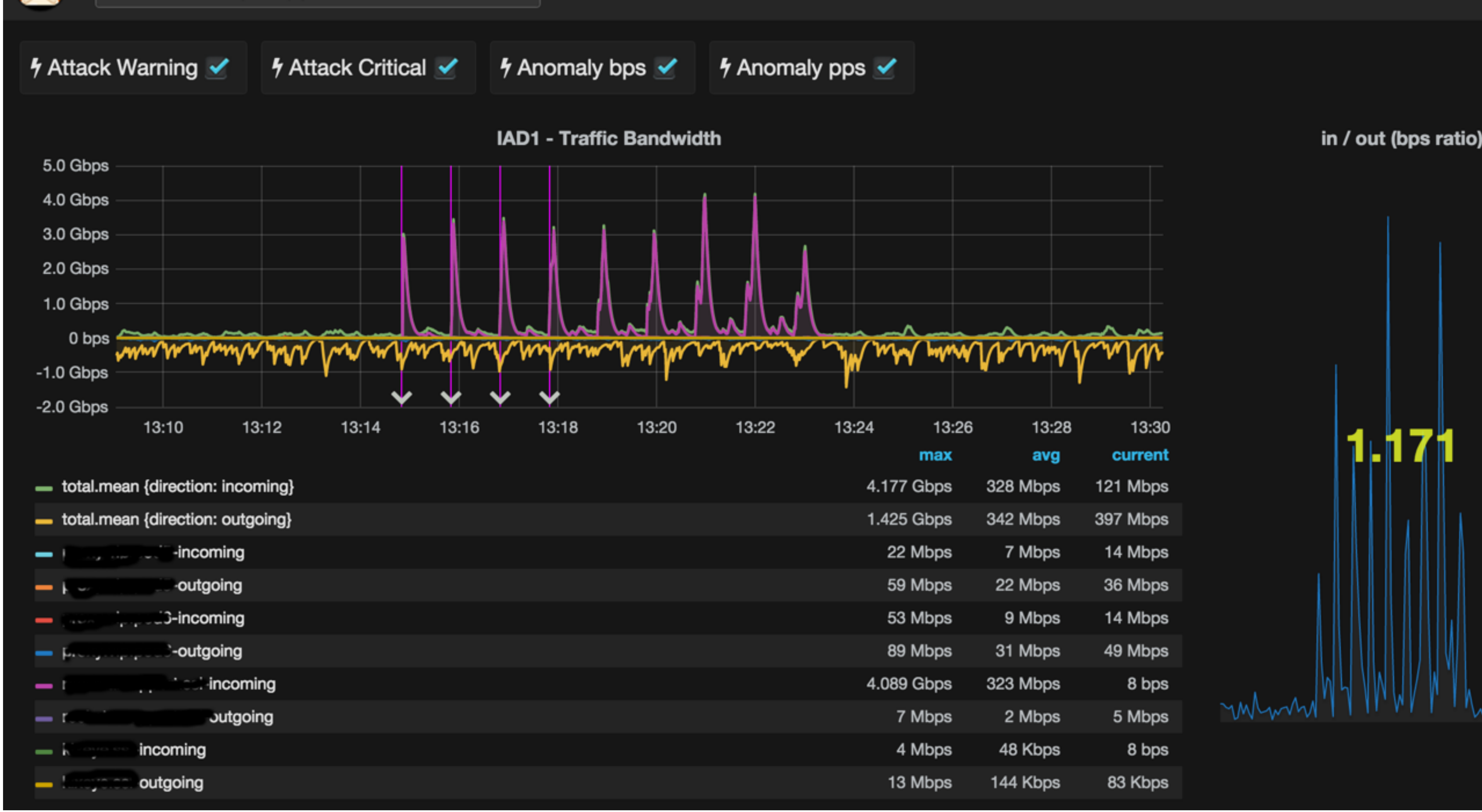


	max	avg	current ▾
🟧 networks [redacted]	294 Mbps	11 Mbps	46 Mbps
🟥 networks [redacted]	108 Mbps	32 Mbps	20 Mbps
🟪 networks [redacted]	84 Mbps	19 Mbps	15 Mbps
🟩 networks [redacted]	25 Mbps	6 Mbps	4 Mbps
🟨 networks [redacted]	14 Mbps	3 Mbps	2 Mbps
🟪 networks [redacted]	14 Mbps	4 Mbps	1 Mbps
🟧 networks [redacted]	14 Mbps	1 Mbps	465 Kbps
🟦 networks [redacted]	6 Mbps	247 Kbps	281 Kbps
🟩 networks [redacted]	52 Mbps	870 Kbps	198 Kbps



IAD1 - Traffic Bandwidth

	max	avg	current
🟩 total.mean {direction: incoming}	5.06 Gbps	266 Mbps	108 Mbps
🟨 total.mean {direction: outgoing}	1.69 Gbps	417 Mbps	954 Mbps
🟦 [redacted]5-incoming	32 Mbps	7 Mbps	8 bps
🟧 [redacted]-outgoing	124 Mbps	31 Mbps	58 Mbps
🟥 [redacted]incoming	53 Mbps	8 Mbps	8 bps
🟦 [redacted]utgoing	123 Mbps	38 Mbps	69 Mbps
🟪 [redacted].ssl-incoming	101 Mbps	808 Kbps	8 bps
🟪 [redacted].ssl-outgoing	9 Mbps	2 Mbps	3 Mbps

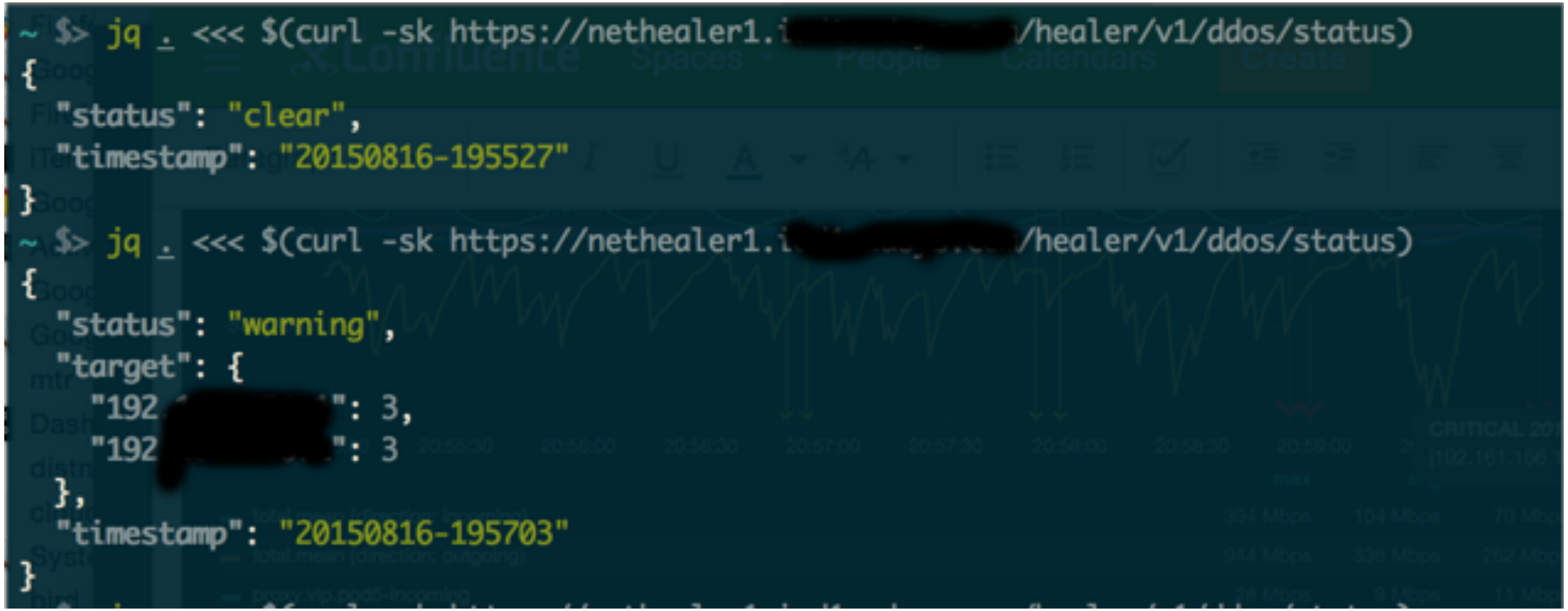


IAD1 - Traffic Bandwidth

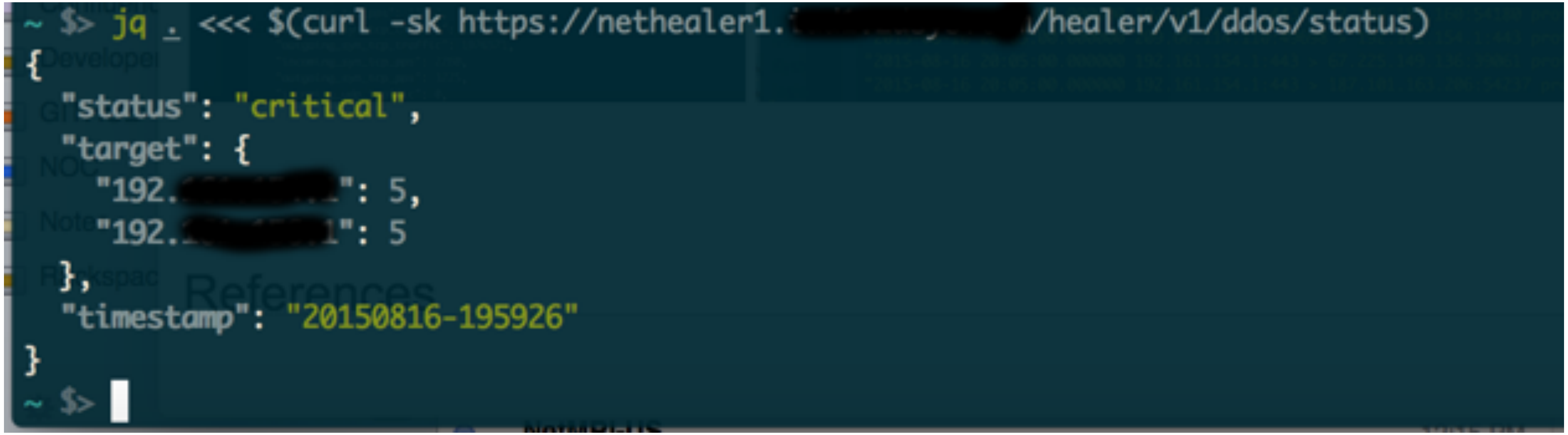
in / out (bps ratio)

		max	avg	current
🟩	total.mean {direction: incoming}	4.177 Gbps	328 Mbps	121 Mbps
🟨	total.mean {direction: outgoing}	1.425 Gbps	342 Mbps	397 Mbps
🟦	[redacted]-incoming	22 Mbps	7 Mbps	14 Mbps
🟧	[redacted]-outgoing	59 Mbps	22 Mbps	36 Mbps
🟥	[redacted]-incoming	53 Mbps	9 Mbps	14 Mbps
🟦	[redacted]-outgoing	89 Mbps	31 Mbps	49 Mbps
🟪	[redacted]-incoming	4.089 Gbps	323 Mbps	8 bps
🟪	[redacted]outgoing	7 Mbps	2 Mbps	5 Mbps
🟩	[redacted] incoming	4 Mbps	48 Kbps	8 bps
🟨	[redacted] outgoing	13 Mbps	144 Kbps	83 Kbps

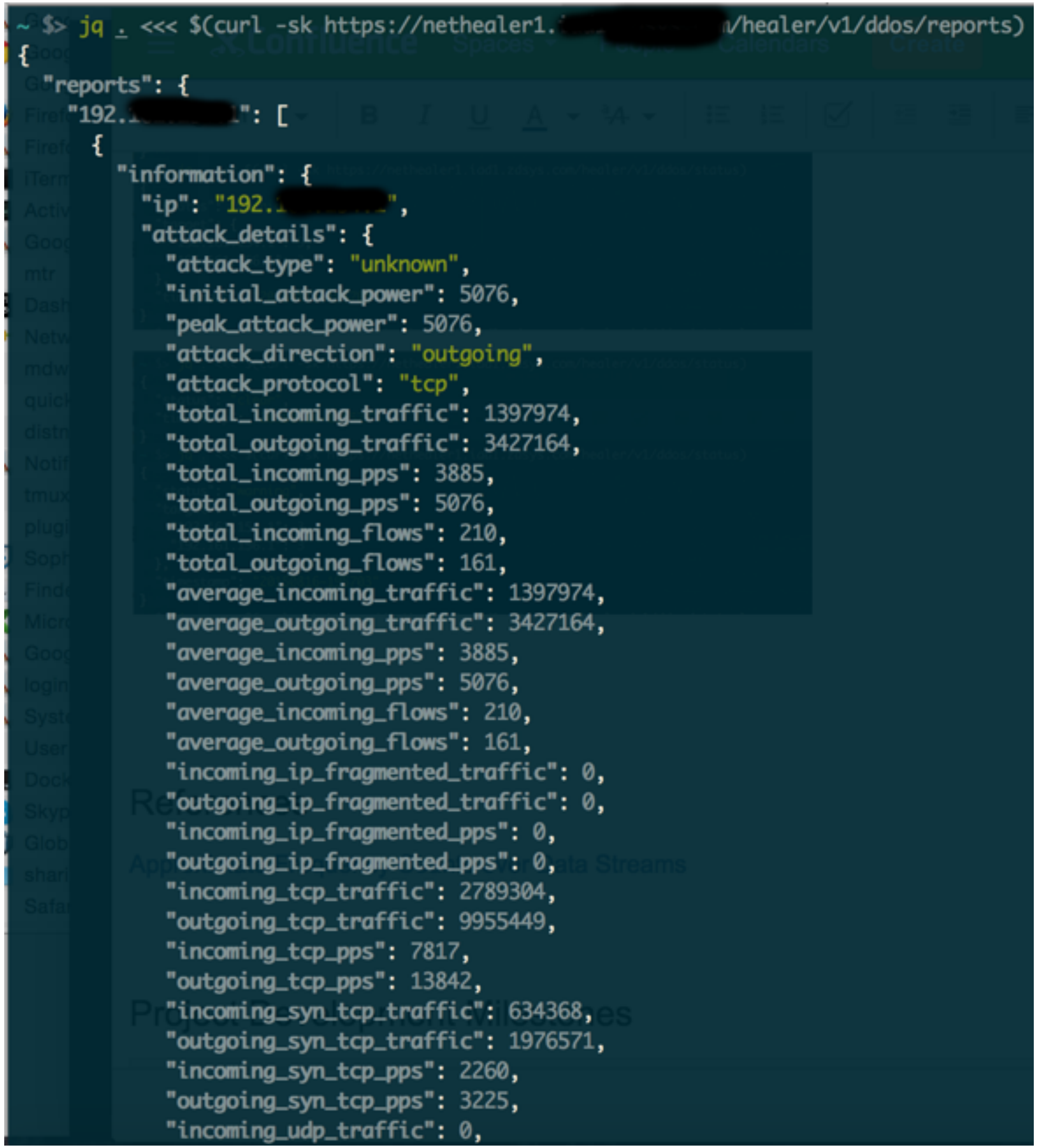
REST API queries



```
~ $> jq . <<< $(curl -sk https://nethealer1.[redacted]/healer/v1/ddos/status)
{
  "status": "clear",
  "timestamp": "20150816-195527"
}
~ $> jq . <<< $(curl -sk https://nethealer1.[redacted]/healer/v1/ddos/status)
{
  "status": "warning",
  "target": {
    "192[redacted]": 3,
    "192[redacted]": 3
  },
  "timestamp": "20150816-195703"
}
```



```
~ $> jq . <<< $(curl -sk https://nethealer1.[redacted]/healer/v1/ddos/status)
{
  "status": "critical",
  "target": {
    "192.[redacted]": 5,
    "192.[redacted]": 5
  },
  "timestamp": "20150816-195926"
}
~ $> 
```



```
~ $> jq . <<< $(curl -sk https://nethealer1.[redacted]/healer/v1/ddos/reports)
{
  "reports": {
    "192.[redacted]": [
      {
        "information": {
          "ip": "192.[redacted]",
          "attack_details": {
            "attack_type": "unknown",
            "initial_attack_power": 5076,
            "peak_attack_power": 5076,
            "attack_direction": "outgoing",
            "attack_protocol": "tcp",
            "total_incoming_traffic": 1397974,
            "total_outgoing_traffic": 3427164,
            "total_incoming_pps": 3885,
            "total_outgoing_pps": 5076,
            "total_incoming_flows": 210,
            "total_outgoing_flows": 161,
            "average_incoming_traffic": 1397974,
            "average_outgoing_traffic": 3427164,
            "average_incoming_pps": 3885,
            "average_outgoing_pps": 5076,
            "average_incoming_flows": 210,
            "average_outgoing_flows": 161,
            "incoming_ip_fragmented_traffic": 0,
            "outgoing_ip_fragmented_traffic": 0,
            "incoming_ip_fragmented_pps": 0,
            "outgoing_ip_fragmented_pps": 0,
            "incoming_tcp_traffic": 2789304,
            "outgoing_tcp_traffic": 9955449,
            "incoming_tcp_pps": 7817,
            "outgoing_tcp_pps": 13842,
            "incoming_syn_tcp_traffic": 634368,
            "outgoing_syn_tcp_traffic": 1976571,
            "incoming_syn_tcp_pps": 2260,
            "outgoing_syn_tcp_pps": 3225,
            "incoming_udp_traffic": 0,
```

Work in progress

** all the ingredients used on this recipe are open source **
** how to build yourself **

Read Documentation https://github.com/pavel-odintsov/fastnetmon/tree/master/docs

Download https://github.com/pavel-odintsov/fastnetmon

Join mail list https://groups.google.com/forum/#!forum/fastnetmon

About **FastNetMon**:
Thanks to **Pavel Odintsov**
for the amazing gift he made available the open source community

About **NetHealer**: experimental (alpha) Ruby code.
ideas, issues and pull requests are more than welcome.
https://github.com/zenvdeluca/net_healer

First
Attempt
In
Learning

Thank you!

Questions?

vdeluca@zendesk.com