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
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

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

Seconds

NetFlow
sFLOW
Mirror
our proof-of-concept
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 ($$$$)
edge routers

internet upstreams

IXP

Linux instance
VM 8 cpu, 8gb RAM

eFLOW, NETFLOW or SPAN

FastNetMon

InfluxDB

Net Healer

Redis DB

K/V DB
store attack reports in JSON
DDoS Attack cycle
Net Healer watches RedisDB and InfluxDB

if the current attack reports match any policy, trigger the associated action

FNM quiescence:
15s per /32

FastNetMon:
populate /32 details at RedisDB

if Morgoth detects:
populate timestamp
at anomaly InfluxDB

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 $\geq 0.9.6.1$ - with tsm1 engine with no batching
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?
REST API queries
$ jq - <<< $(curl -sk https://nethealer1.institute/api/healer/v1/ddos/status)
{
  "status": "clear",
  "timestamp": "20150816-195527"
}

$ jq - <<< $(curl -sk https://nethealer1.institute/api/healer/v1/ddos/status)
{
  "status": "warning",
  "target": {
    "192.168.1.1": 3,
    "192.168.1.2": 3
  },
  "timestamp": "20150816-195703"
}

$ jq - <<< $(curl -sk https://nethealer1.institute/api/healer/v1/ddos/status)
{
  "status": "critical",
  "target": {
    "192.168.1.1": 5,
    "192.168.1.2": 5
  },
  "timestamp": "20150816-195926"
}
{
    "reports": {
        "information": {
            "ip": "192.
            "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](https://github.com/pavel-odintsov/fastnetmon/tree/master/docs)

**Download** [https://github.com/pavel-odintsov/fastnetmon](https://github.com/pavel-odintsov/fastnetmon)

**Join mail list** [https://groups.google.com/forum/#!forum/fastnetmon](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](https://github.com/zenvdeluca/net_healer)
Thank you!

Questions?

vdeluca@zendesk.com