Comparison of eBPF, XDP and DPDK for packet inspection

Marian Marinov <mm@yuhu.biz>
Chief System Architect of SiteGround
Who am I?

- Chief System Architect of SiteGround.com
- Sysadmin since 1996
- Organizer of OpenFest, BG Perl Workshops, LUG-BG, RailsGirls and others
- Teaching Network Security and Linux System Administration courses in Sofia University and SoftUni
Why do we need this?

Frequency of DoS/DDoS attacks to our infrastructure

- 4-10 Gbps 6-8 times a month
- 10-40 Gbps maybe 2-3 times a month
- 100+ Gbps around 2 times a month
Attacks resulting in service degradation:

➢ for the past 276 days we had 31 DDoS attacks
➢ some of the months, no attacks
➢ but some months, up to 9

➢ 2019 – 31 attacks
➢ 2018 – 75 attacks
➢ 2017 – 69 attacks
➢ 2016 – 84 attacks

Note: I have manually counted the e-mails. The numbers can be slightly inaccurate.
Most attacks are basic

- 20k pps toward ISC Bind can consume up to 30 CPU cores
- A child can generate that on its laptop, at home
General solutions

➢ Buy additional bandwidth
➢ Buy a very expensive scrubbing device

OR

➢ Offload this task to other companies, like CloudFlare
Hosted solution issues

➢ Not every DataCenter is willing to invest in these devices
➢ Shared devices
➢ Attacks can be larger than the capacity of the device
➢ Larger attacks almost always result in null route
➢ Attacks saturating the uplinks can affect other machines in the rack and/or row
Cloud solution issues

- You have to point your DNS to the service provider
- Controlling your DNS is now only API based
- Large DNS updates become an issue
- Not suitable for hosting companies
Requirements?

➢ Build a VM that can handle 10Gbps with ~8Mpps
➢ Why a VM?
➢ scrub UDP DNS and NTP traffic
➢ scrub TCP traffic by implementing SYN cookies
➢ scrub all unrelated traffic
➢ cache HTTP responses (wishful thinking) :)
Linux Network Flow
10M packet drop

➢ in 2018 CloudFlare published the article: How to drop 10m packets
➢ I confirm their results with a few additions:

➢ iptables can drop at best 2m pps
   Note: with only one entry in the PREROUTING chain of the mangle table
➢ heaving multiple entries in that chain easily becomes a problem
➢ even if you use ipset with that, you have a big problem when updating that information
CloudFlare results

Packet dropping performance

- IPv4
- IPv6

Application (contrack)
Application (NOTRACK)
BPF on socket
iptables INPUT
iptables PREROUTING
nftables ingress
tc ingress
CloudFlare results XDP

Packet dropping performance

Processed packets per second on single CPU

IPv4
IPv6

Application (contrack) Application (NOTRACK) BPF on socket iptables INPUT iptables PREROUTING nftables ingress tc ingress XDP
CloudFlare demo code can be found on GitHub
So, how I started?

- I already knew about XDP
- But I decided to be “smart ass” and wrote an iptables module...
- It could handle between 260k and 280k pps
Not good enough... eBPF

➢ I also knew I can use eBPF for that...
➢ from the talk of Daniel Borkmann from FOSDEM 2016

➢ It was better, but not enough...
➢ 320-350k pps drop rate
➢ with 2000 domains and UDP packet checking
➢ no checksums thou
I had previous experience with DPDK

So I ordered one Intel and one SolarFlare NICs

With both I managed to drop anything that was below the 10G limit of the cards

With SolarFlare I even tested uploading code into the NIC itself
Nobody, except me, was interested in supporting DPDK code
Writing and updating DPDK is not trivial
DPDK required specific HW that may not be available in the DataCenter
A friend (Boyan Krosnov) told me about P4
P4 made updating the logic and content of the filter program a lot simpler for me...
➢ P what?
➢ If we were to use DPDK with P4, everyone had to learn the language :(
And then came XDP

- received packet
- XDP eBPF
- alloc_skb
- ingress (qdisc)
- bridge check
- broute brouting
- nat prerouting
- raw prerouting
- bridging decision
- nat prerouting
- mangle prerouting
- conntrack
- filter forward
- mangle forward
- nat postrouting
- mangle postrouting
- nat postrouting
- filter input
- routing decision
- mangle forward
- nat postrouting
- mangle postrouting
- nat postrouting
- egress (qdisc)
- bdrige
- netfilter
- other net
And then came XDP

- Extremely fast and closest to the NIC, same as DPDK
- Supported by many drivers
- Extendable with eBPF functions
- Developed by Jasper Brouer
What I ended up with?

- A filter similar to what CF did with their DROP example
- instead of comparing a single prefix, I'm extracting the UDP data if the packet is UDP
- then the extracted data is compared with a BPF map
- I wrote a simple user space tool, that updates the map in the kernel
- voila I had a fast scrubber
the UDP scrubber

- if the DNS request is not for a domain that is within the list in the map I drop the packet
- ToDo: add caching of responses with TTL
This is where I had to stop :(

- compare the packet's dst port and allow it only if it is:
  - SYN to a port that is allowed
  - send and receive SYN cookies here
  - part of already existing connection by examining its own db of tuples and the supplied by the user space (other VMs)
This is where I had to stop :(  

- It should handle the SYN cookie for the servers behind and replay the initial SYN if correct SYN,ACK is received
Testing the bastard

I knew I was able to drop packets fast…
But I needed a proof ;)

➢ I had a talk with Jasper at Linux Plumbers 2019
➢ He pointed me to

his patched version of pktgen on GitHub :)

SiteGround
Now...

How to get from 10Gbps to 200Gbps?
Now...

How to get from 10Gbps to 200Gbps?

➢ Combining multiple VMs with ECMP
➢ I did that directly on the switch :)

SiteGround
How to drop 10 million packets per second
https://blog.cloudflare.com/how-to-drop-10-million-packets/

XDP tutorial
https://github.com/xdp-project/xdp-tutorial

More XDP materials:
https://www.iovisor.org/technology/xdp

Enhanced pktgen by Jasper
https://github.com/netoptimizer/network-testing
Links

Linux tc and eBPF
man pages
http://man7.org/linux/man-pages/man2/bpf.2.html

SolarFlare AOR firmware development kit

Data Plane Development Kit
https://www.dpdk.org/

P4 Language Specification
https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.pdf

P4 meets DPDK
Thank you!