#ebpf
You Cannot Stop This

David S. Miller, Red Hat Inc.


davem_dokebi

Van, “The Man”

Ultimately, the true father of container networking and security

Invented BPF in 1992 with Steven McCanne at LBL

Recognized the need for programmability inside of the OS

Used for tcpdump/libpcap during the initial development of TCP congestion control

We refer to this as Classical BPF

But wait, there’s more...

Alexei, Openning Up Pandora's Box

Invented #eBPF

The realization of full OS programmability

Full 64-bit instruction set, registers, atomics

Need data structures? #ebpf MAPs...

Need to access kernel objects? #ebpf helpers...

Dynamic code generation, only what you use...

#ebpf brain implants… Sign me up!!!

Thomas, Cooking Up The Recipe

Creator of the Cilium Project

API aware networking and security

Leverages #ebpf related technologies like XDP, cls_bpf, sockmap and also things like kTLS

Back to the Unix Philosophy of individual tools that do one thing well and are used together to solve a problem

Forget these monsters like OVS and DPDK which trend towards monolithic designs

Daniel, Up All Night

Co-maintains #ebpf GIT tree with Alexei

Works 14+ hours a day to make #ebpf a success

Pull requests coming in at 4:30am Zurich time

Daniel, get some rest dude!

Jesper, Makin’ It Fast

Nanosecond level performance analysis

Page recycling and sharing

Driver model

XDP Redirect

World wide advocacy for #ebpf and XDP

Educating the world with Andy Gospodarek

My Danish coffee importer...

Simon, Offloadin’ It

Strong proponent of #ebpf at Netronome

Making everything #ebpf offload based

Even OVS et al. configurations

Unified firmware strategy

My Dutch coffee importer...

Me

Someone has to coordinate all of this

Container Networking Security

Key question: Where does security belong in the container network?

Many solutions put it at the namespace and/or network boundary.

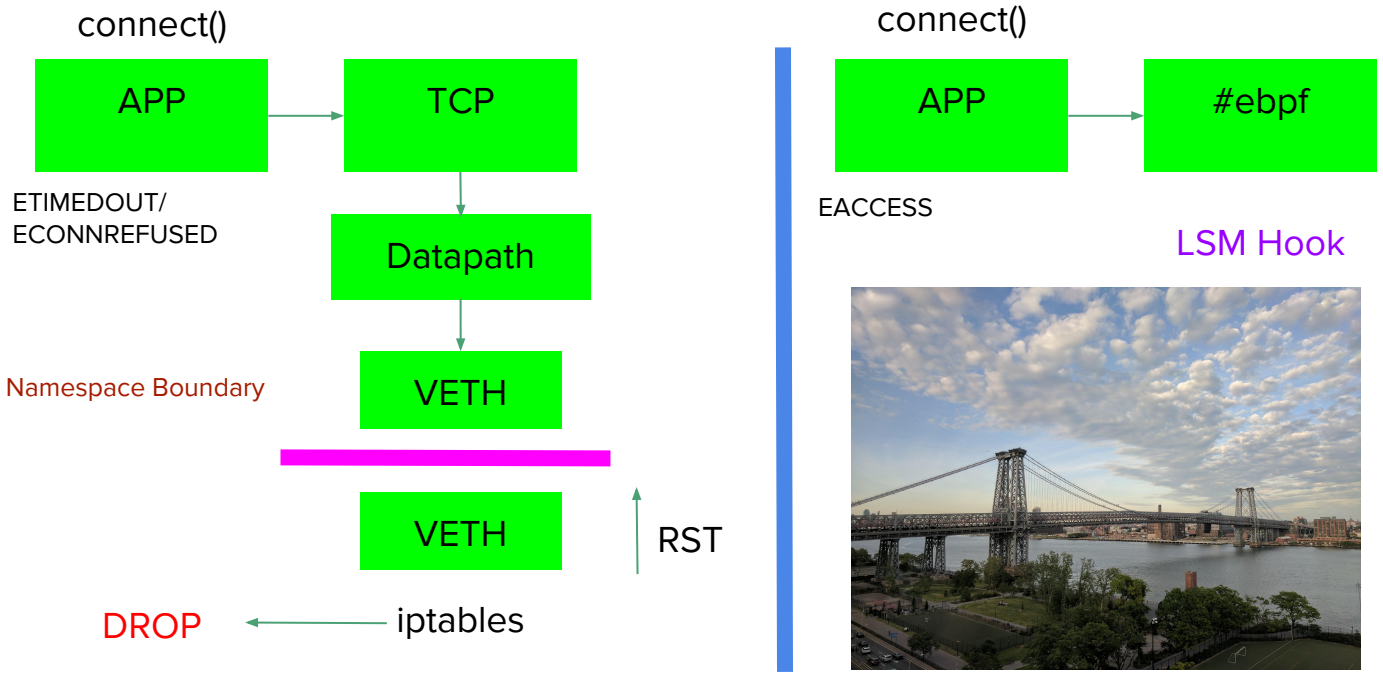
Two problems:

1) Inefficient
2) Far removed from the application context, less information

Well, there is also a third problem:

3) TLS

Example: IPTables vs. #ebpf



Why Not DPDK For Containers?

Unnecessary layer crossing

Userland is not the realm for solutions to this space

The discrete path between containers is inside of the kernel, not outside

If you want every inter-container packet to be copied twice, feel free to use DPDK

Don’t get me wrong, DPDK is great for solving some problems, just not this one

What About OVS?

OVS can be applied to the solution of many problems, perhaps too many. Why?

OVS's design tends towards monolithic solutions

OVS feature set is huge and complicated, misses processed in userspace

A cache controllable by remote entities is by definition DDoS'able

Conntack support added as a hack for flow lookup performance

Now people want conntrack offloaded in HW, yikes!

Containers And The Unix Philosophy

Make small components that do one specific job well

These components are put together to solve a specific task

Everyone can see this in how shell pipelines work

This also applies to fundamental facilities like #ebpf

Sockmap, XDP, kTLS, and cls_bpf as tools in a “networking pipeline.”

Oh yes, kTLS: Transparent encryption for inter-container communication

Replacing IPTables

Built for the age of IP and port based security. We are beyond that

We need “identity” based security and high performance load balancing

Each kubernetes service definition requires 6 IPTables rules

IPTables evaluates rules linearly, this simply does not scale

IPTables updates are “replace entire table” operations

People burn entire cores on cluster nodes just for IPTables

The Future of Networking and Security

Socket level BPF

Security and routing as close to the application as possible

And XDP provides integrated DDoS mitigation:

- Allows exposing kubernetes clusters directly on the internet without requiring expensive hardware load balancers
- Hardware offloadable, more cpu/memory cycles for microservices

The Power Of Sockmap

Traditional packet filtering cannot wholly match any criteria

Filtering is done on a packet by packet basis

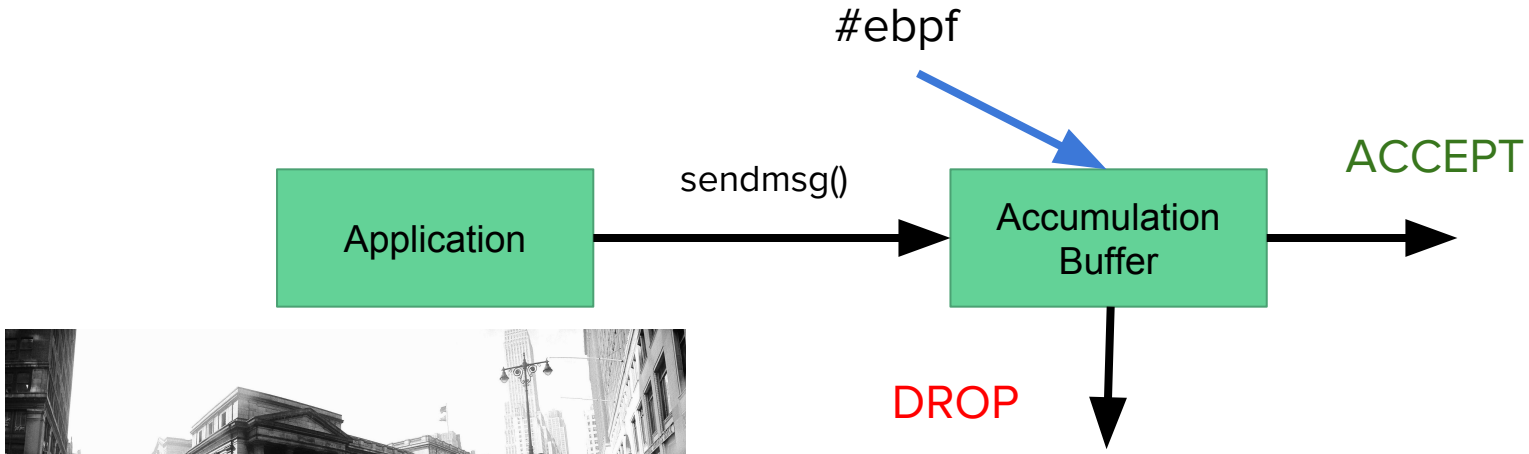
Attacker just has to segment the data stream, thwarting the filter

Not so with sockmap, data is queued until enough context is available

Zero cost, unlike other schemes, because socket does the queueing

Sockmap works right at the recvmsg()/sendmsg() point

Sockmap At A Glance



Similarly for recvmsg()

Service Mesh Acceleration

BPF provides the power to perform service mesh acceleration

What is service mesh:

A new forwarding model where all communication is proxied through EnVoy/HAPRoxy/nginx. This is for enabling transparent mTLS, L7 tracing and latency information, and much more

BPF socket map can accelerate persistent HTTP connections

kTLS + #ebpf Sockmap

Can give visibility into SSL encrypted data

Right now people inject a fake root CA into apps

This is done in order to terminate SSL connections on behalf of 3rd party sites

We’re talking about applications using 3rd party APIS, outside of user’s control:

Cloud provider APIs, auth0, github.com APIs, analytics APIs, etc.

kTLS provides a complete unique angle, avoiding fake root CA injection

Cilium 1.4.0 Is Out!

Released Tuesday, Feb 12th, 2019

Has all the features mentioned in this presentation

Amazing stuff, go check it out!

Thank You

Linda Wang

Daniel Borkmann

Jesper Dangaard Brouer

Toke Høiland-Jørgensen

John Fastabend

Simon Horman

Alexei Starovoitov, and the rest of the Facebook #ebpf team

Thomas Graf