eBPF Superpowers for SRE

Liz Rice | @lizrice
Chief Open Source Officer, Isovalent
Emeritus Chair, CNCF Technical Oversight Committee | CNCF & OpenUK boards
What is eBPF?

Makes the kernel programmable
Run custom code in the kernel

**ISOVALENT**

- **userspace**
  - app

- **kernel**
  - Files
  - Networking
  - Memory
  - Processes

- **event**

- **eBPF program**

**System calls**
ISOVALENT

eBPF Hello World

SEC("kprobe/sys_execve")
int hello(void *ctx)
{
    bpf_printk("I'm alive!");
    return 0;
}

$ sudo ./hello
bash-20241 [004] d... 84210.752785: 0: I'm alive!
bash-20242 [004] d... 84216.321993: 0: I'm alive!
bash-20243 [004] d... 84225.858880: 0: I'm alive!

Info about process that called execve syscall
SRE practical use case #1
Event tracing
eBPF tracing tools from iovisor/bcc
## eBPF tracing - opensnoop

```bash
~/bcc/libbpf-tools$ sudo ./opensnoop

<table>
<thead>
<tr>
<th>PID</th>
<th>COMM</th>
<th>FD</th>
<th>ERR</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/5132/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/6460/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/6460/cmdline</td>
</tr>
<tr>
<td>6461</td>
<td>opensnoop</td>
<td>18</td>
<td>0</td>
<td>/etc/localtime</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/5132/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/6460/cmdline</td>
</tr>
<tr>
<td>5060</td>
<td>node</td>
<td>23</td>
<td>0</td>
<td>/home/liz/.vscode-server/data/User/workspaceStorage/48b53</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/5132/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/6460/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/5132/cmdline</td>
</tr>
<tr>
<td>5040</td>
<td>node</td>
<td>21</td>
<td>0</td>
<td>/proc/6460/cmdline</td>
</tr>
</tbody>
</table>
```
Dynamically change kernel behaviour
Application Developer:

I want this new feature to observe my app.

Hey kernel developer! Please add this new feature to the Linux kernel.

OK! Just give me a year to convince the entire community that this is good for everyone.

1 year later...

I'm done. The upstream kernel now supports this.

But I need this in my Linux distro.

5 years later...

Good news. Our Linux distribution now ships a kernel with your required feature.

OK but my requirements have changed since...
Application Developer:

I want this new feature to observe my app

eBPF Developer:

OK! The kernel can’t do this so let me quickly solve this with eBPF.

A couple of days later...

Here is a release of our eBPF project that has this feature now. BTW, you don’t have to reboot your machine.
Couldn’t I do this with kernel modules?
eBPF verification ensures program safety
SRE practical use case #2
Kernel vulnerability mitigation
Packet of Death
Packet of Death

host

eth0

Discard?
eBPF Packet Drop

SEC("xdp/bye")
int goodbye_ping(struct xdp_md *ctx)
{
    ...
    if (iph->protocol == IPPROTO_ICMP)
        return XDP_DROP;
    return XDP_PASS;
}
eBPF with Kubernetes
eBPF started a **whole new infrastructure movement** in the cloud native space

- Daniel Borkmann
One kernel per host
Kernel aware of everything on the host
eBPF programs can be aware of everything
eBPF tools instrument the system without any app or config changes
My other sidecar is a kernel

“Get in loser. We’re going tracing”

- Nathan LeClaire @dotpem
A sidecar has a view across one pod
Sidecars need YAML

```
containers:
  - name: my-app
   ...
  - name: my-app-init
    ...
  - name: my-sidecar
    ...
```
eBPF does not need any app changes

my-app.yaml
containers:
- name: my-app
  ...
- name: my-app-init
  ...

isovalent
eBPF can see ALL activity on the node

```
my-app.yaml
containers:
  - name: my-app
    ...
  - name: my-app-init
    ...
```
SRE practical use case #3
Cloud native observability tools
Observability is the fundamental basis for all SRE

- Mario Biemans
$ kubectl gadget trace open

<table>
<thead>
<tr>
<th>NODE</th>
<th>NAMESPACE</th>
<th>POD</th>
<th>CONTAINER</th>
<th>PID</th>
<th>COMM</th>
<th>FD</th>
<th>ERR</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind-2-control-plane</td>
<td>default</td>
<td>xwing</td>
<td>spaceship</td>
<td>361876</td>
<td>vi</td>
<td>3</td>
<td>0</td>
<td>/etc/passwd</td>
</tr>
</tbody>
</table>

Kubernetes info
SRE practical use case #4
High performance networking
host

pod
- iptables conntrack
- iptables PREROUTING
- Linux routing
- iptables

app
- socket

veth

veth

Linux routing

cilium

eth0

eBPF
Unleashing the Power of Cilium CNI to Propel Trendyol’s Performance Up to 40%!

Trendyol implemented Cilium as the default CNI for the Kubernetes Cluster starting from version 1.26. Discover our journey.

Throughput - Higher is better

CPU Usage - Lower is better

Unleashing the Power of Cilium CNI to Propel Trendyol's Performance Up to 40%! 20 July 2023
ISOVALENT

Cilium Hubble observability

- Network flow logs
- Prometheus metrics
- Service map
- L3/4 & L7 (HTTP, DNS, Kafka, ...)
- Aware of Kubernetes identities

@lizrice
SRE practical use case #5
Cloud native security tools
68% of SREs say they expect their role in security to become even more central

- Dynatrace State of SRE report 2022
ISOVALENT

eBPF for security-relevant events

userspace

app

system calls

kernel

Files

Networking

Memory

Processes

eBPF program

event
Cilium network policy → eBPF programs drop packets
ISOVALENT

Security observability

Observe events

Policy → Filter events → Alerts

What is the cause?
What is affected?
Taking eBPF observability tools as-is and using them for security monitoring would be like driving your car into the ocean and expecting it to float

- Brendan Gregg
Security observability

Observe events

Filter events

Alerts

What is the cause?
What is affected?
Cilium Tetragon observes security events

$ kubectl logs tetragon-74fffc -c export-stdout -f | tetra getevents -o compact

- process default/xwing /usr/bin/vi /etc/passwd
- open default/xwing /usr/bin/vi /etc/passwd
- close default/xwing /usr/bin/vi
- open default/xwing /usr/bin/vi
- write default/xwing /usr/bin/vi /etc/passwd 1275 bytes
- close default/xwing /usr/bin/vi
- exit default/xwing /usr/bin/vi /etc/passwd 0

Policy events

Kubernetes info
I S O V A L E N T

Preventative actions from kernel

Pod

app.go

Exploit / malicious attempt

SIGKILL

Kernel

Tetragon

@lizrice
$ kubectl logs tetragon-74ffc -c export-stdout -f | tetra getevents -o compact

🚀 process default/xwing /usr/bin/vi /etc/passwd
↾ open    default/xwing /usr/bin/vi /etc/passwd
↘ close   default/xwing /usr/bin/vi
↾ open    default/xwing /usr/bin/vi /etc/passwd
ราว write default/xwing /usr/bin/vi /etc/passwd 1269 bytes
ราว write default/xwing /usr/bin/vi /etc/passwd 1269 bytes
💥 exit    default/xwing /usr/bin/vi /etc/passwd SIGKILL

Killed before write
eBPF enables powerful Cloud Native tools

- High performance observability, networking and security
- Dynamic instrumentation - zero app modifications
- Contextual information, Kubernetes identity-aware
eBPF in the CNCF

cilium
Pixie
hubble
Falco
Tetragon
Inspektor Gadget
Thank you

cilium/cilium  @ciliumproject  cilium.io

Download from isovalent.com