



eBPF



Makes the kernel programmable

@lizrice




Run custom code in the kernel

app

system calls

Files 

Networking 

Memory 

Processes 

Interesting for security

event



eBPF

program

eBPF Hello World

```
hello(void *ctx) {
```

```
SEC("kprobe/sys_execve") int
```

@lizrice

program

+ userspace code to load eBPF

```
bpf_trace_printk("Hello World!");
```

```
return 0;
```

```
}
```

Info about process that
called `execve` syscall

```
$ sudo ./hello
```

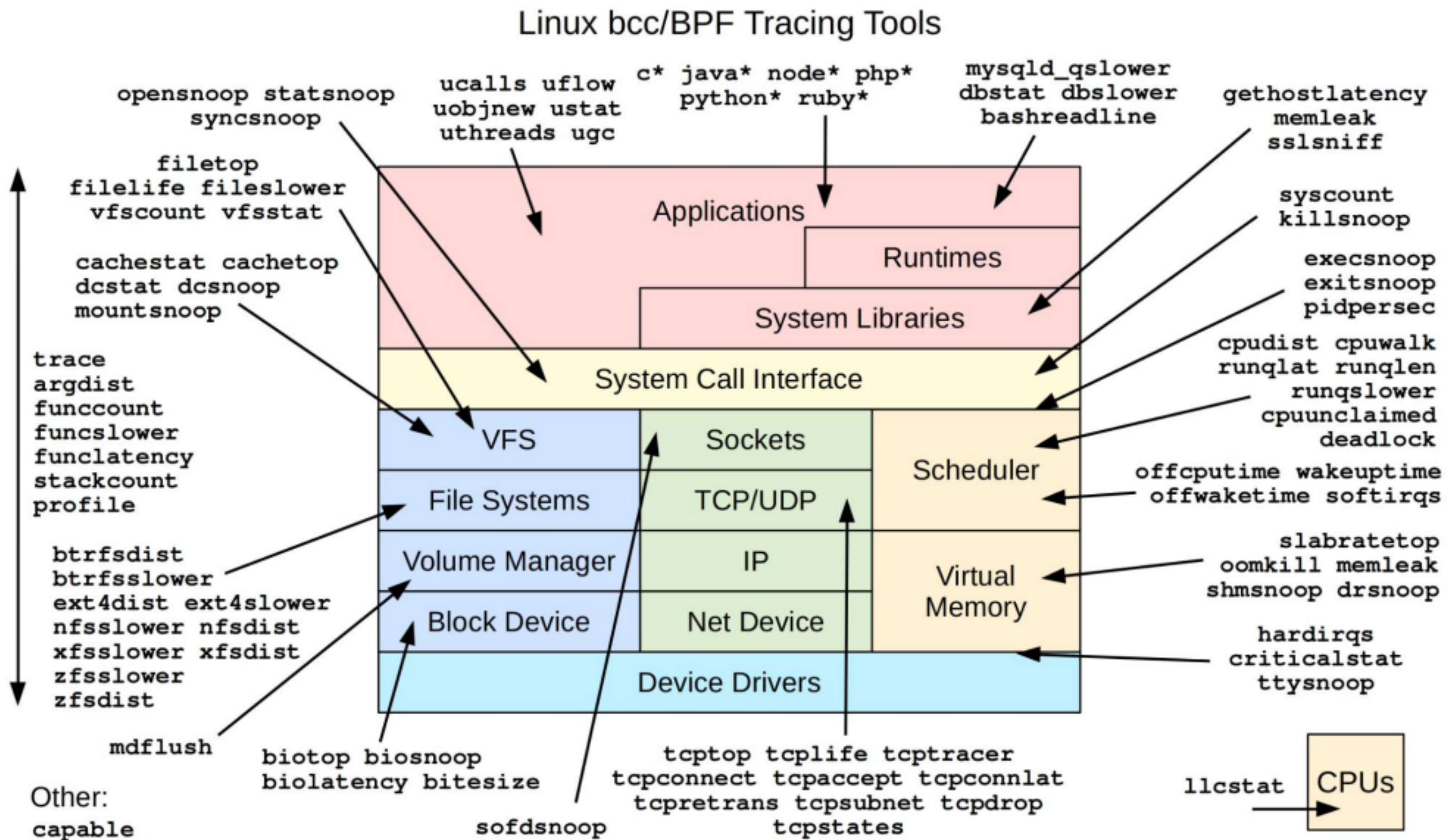
```
bash-20241 [004] d... 84210.752785: 0: Hello World!  
bash-20242 [004] d... 84216.321993: 0: Hello World!  
bash-20243 [004] d... 84225.858880: 0: Hello World!
```

 **@lizrice**

ISOVALENT



eBPF tracing tools from iovisor/bcc



<https://github.com/iovisor/bcc#tools> 2019

@lizrice

eBPF tracing - opensnoop

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

```
PID COMM FD ERR PATH
```

```
5040 node 21 0 /proc/5132/cmdline
```

```
5040 node 21 0 /proc/6460/cmdline
```

```
5040 node 21 0 /proc/6460/cmdline
```

```
6461 opensnoop 18 0 /etc/localtime
```

```
5040 node 21 0 /proc/5132/cmdline
```

```
5040 node 21 0 /proc/6460/cmdline
```

```
5060 node 23 0 /home/liz/.vscode-server/data/User/workspaceStorage/48b53 5040 node 21 0  
/proc/5132/cmdline
```

```
5040 node 21 0 /proc/6460/cmdline
```

```
5040 node 21 0 /proc/5132/cmdline
```

```
5040 node 21 0 /proc/6460/cmdline
```

```
...
```

 **lizrice**

ISOVALENT

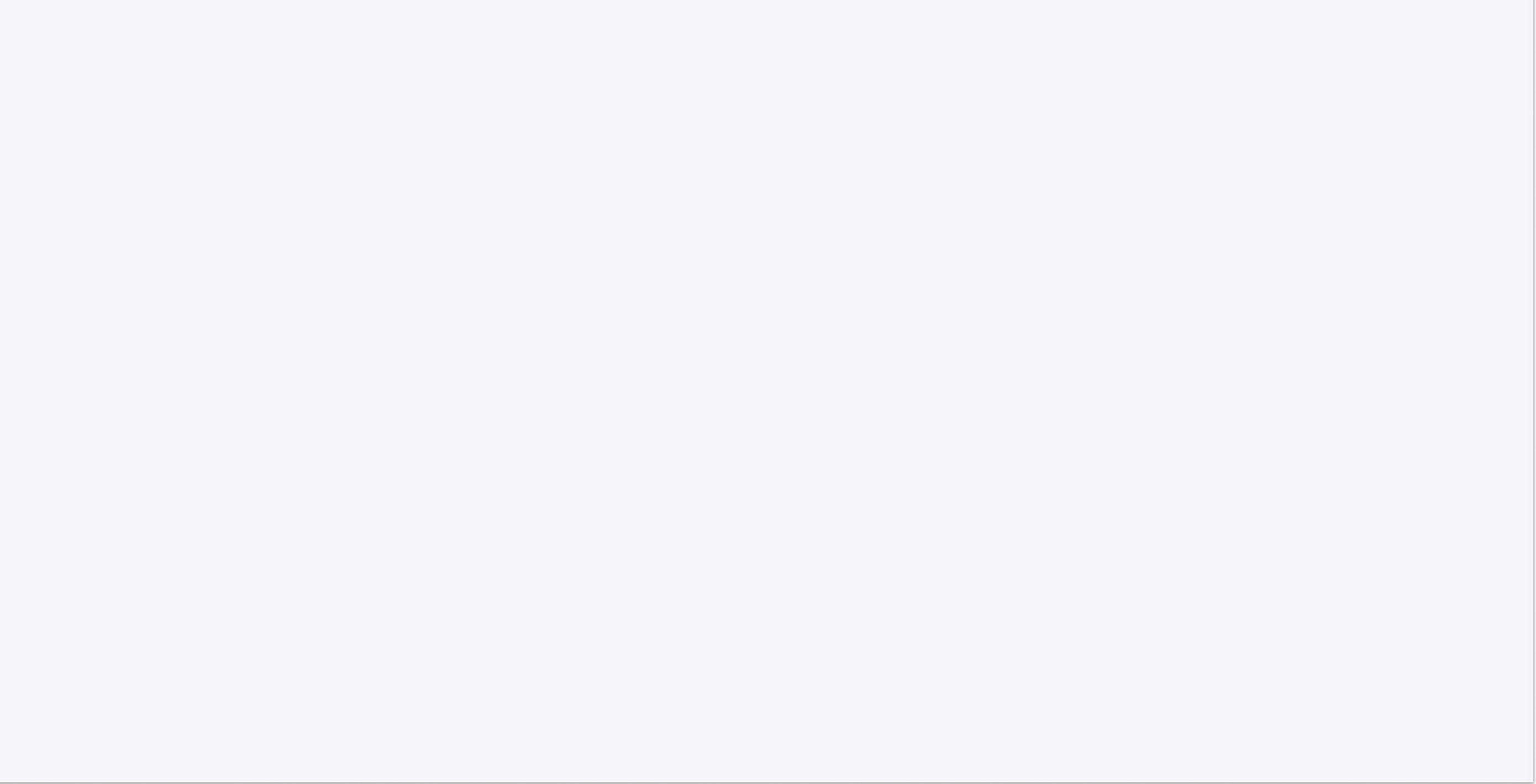


eBPF and Kubernetes

pod container




One kernel per host



pod container container

access files

**networking create
containers**



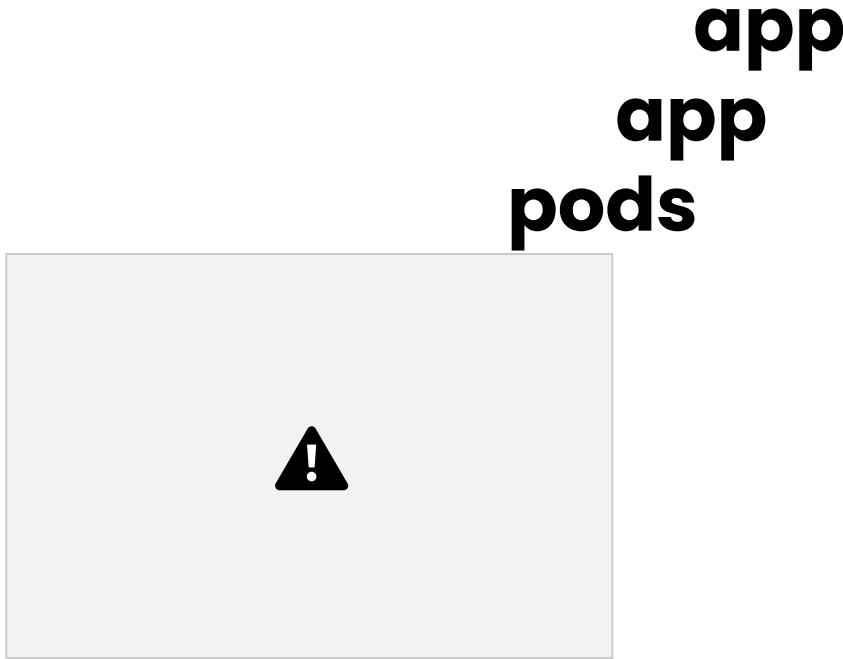
**app
app
pods**

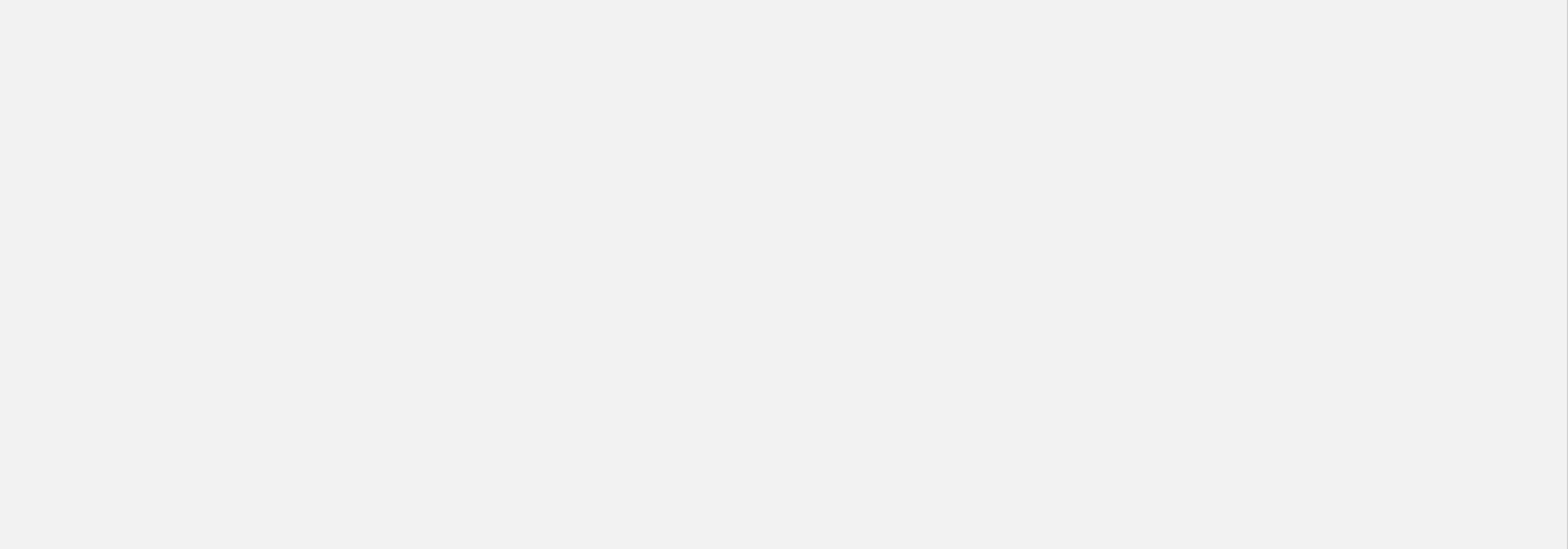
create **Kernel aware**
Of containers
everything on
the host

networking

access files

No changes to
apps or config
needed





create eBPF programs

access files

**containers can be
aware of
everything**



networking



eBPF tracing on Kubernetes - Inspektor Gadget

```
$ kubectl gadget trace open
```

```
NODE NAMESPACE POD CONTAINER PID COMM FD ERR PATH kind-2-control-plane default xwing  
spaceship 361876 vi 3 0 /etc/passwd
```

Kubernetes info

@lizrice







@lizrice



eBPF security observability

Observability





@lizrice



Security observability

@lizrice



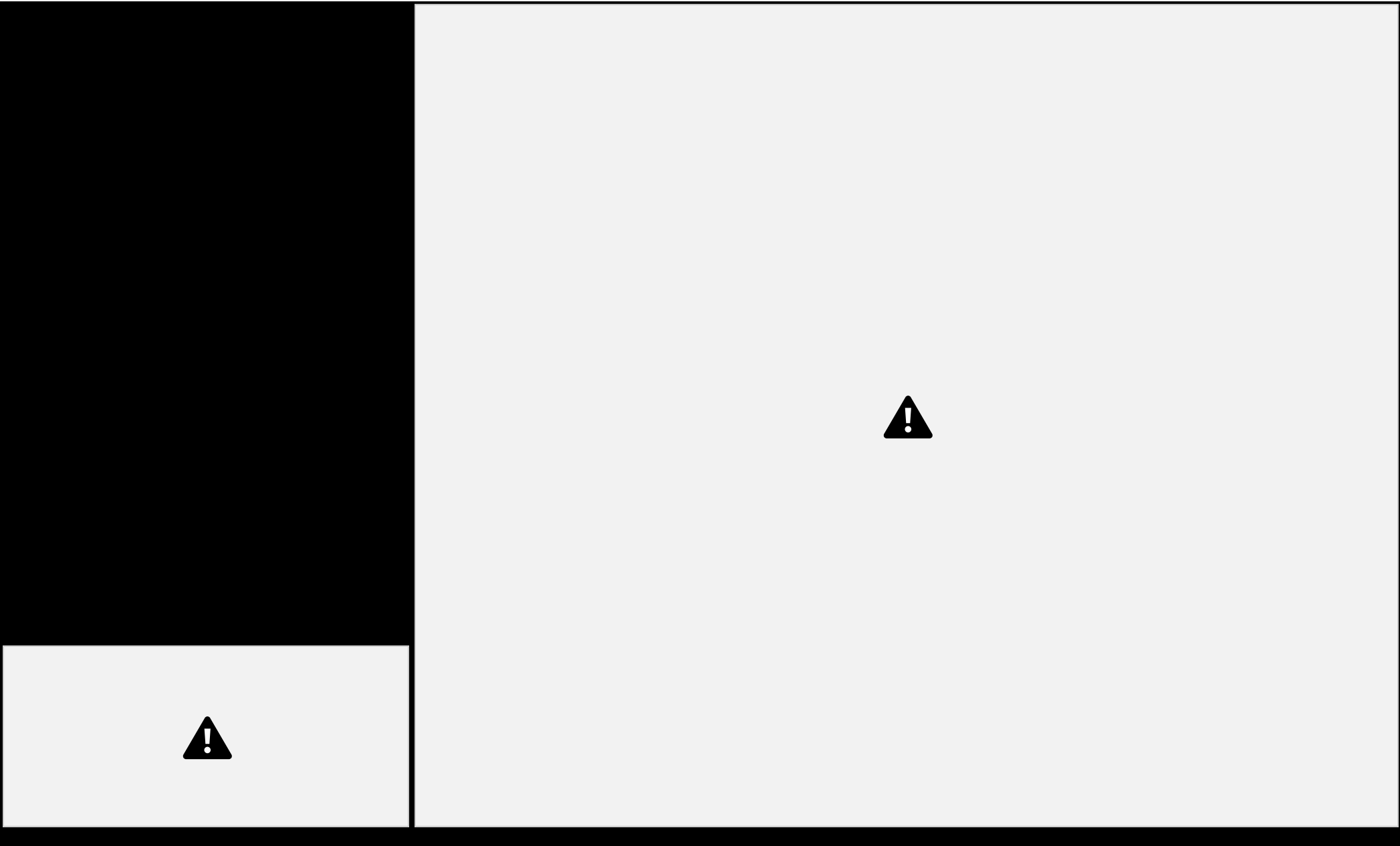
What activity do we care about for



eBPF programs



Syscall checks within the kernel





TOCTTOU vulnerabilities with syscalls



*Attacker changes syscall
args after inspection*

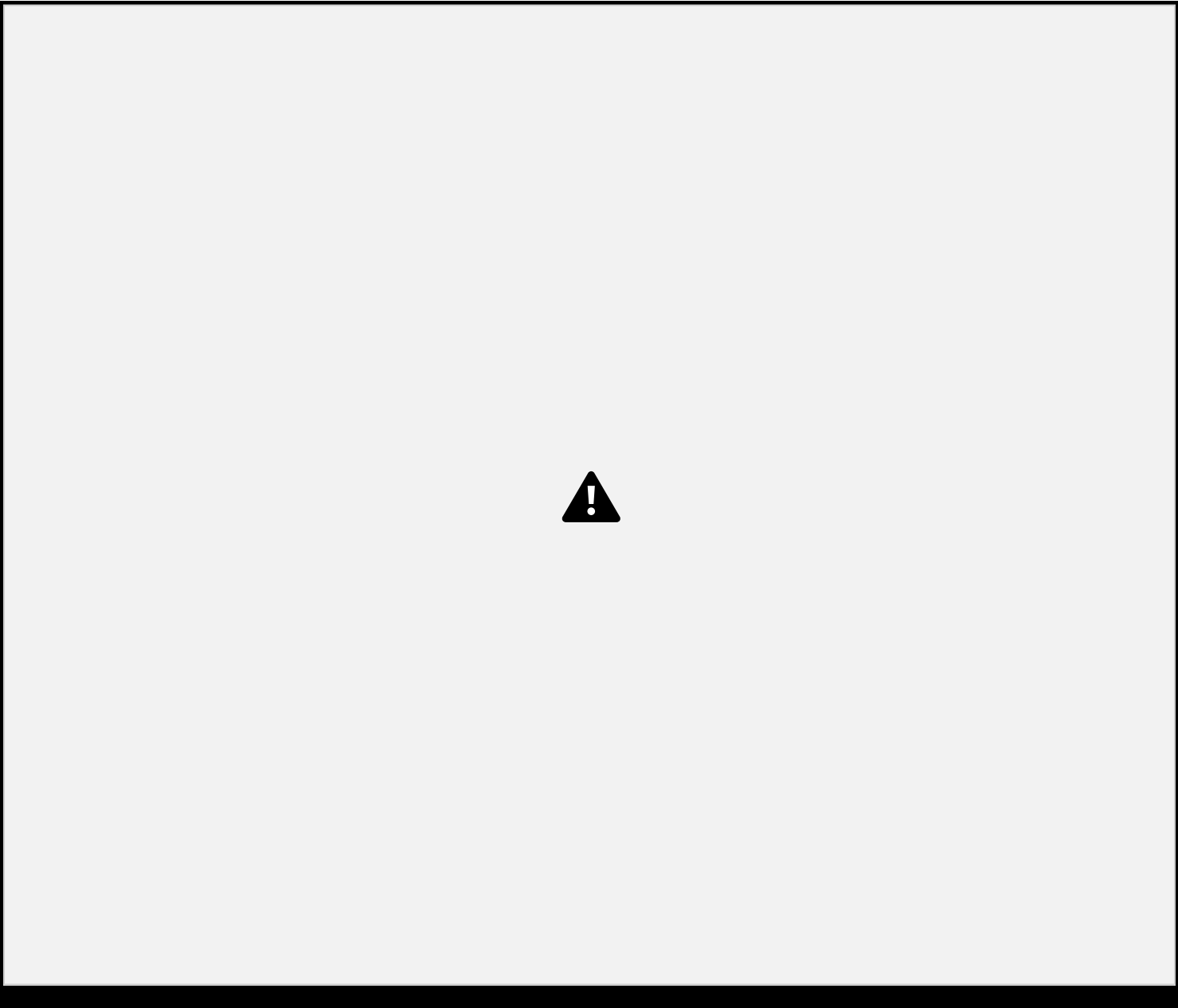
For more details

- Leo Di Donato & KP Singh at CN eBPF Day 2021
- Rex Guo & Junyuan Zeng at DEFCON 29 on Phantom attacks

@lizrice



Need to make the check at the right place







- Stable interface

- Safe places to make checks



@lizrice





- Stable interface

- Safe places to make checks

+ eBPF benefits

- Dynamic
- Protect pre-existing processes



BPF LSM hook has kernel info populated

`SEC("lsm/path_chmod")`

```
int BPF_PROG(path_chmod, const struct path *path, umode_t mode)
{
    bpf_printk("lsm path_chmod %s\n", path->dentry->d_iname);
return 0;
}
```

} Filename known

```
$ sudo ./chmoddemo &
[1] 7631
$ sudo cat
```

```
/sys/kernel/debug/tracing/trace_pip
e
to kernel
```

```
chmod-7776 [001] d... 38197.342160: bpf_trace_printk: lsm path_chmod liz
```






- Stable interface

- Safe places to make checks

+ eBPF benefits

- Dynamic
- Protect pre-existing processes

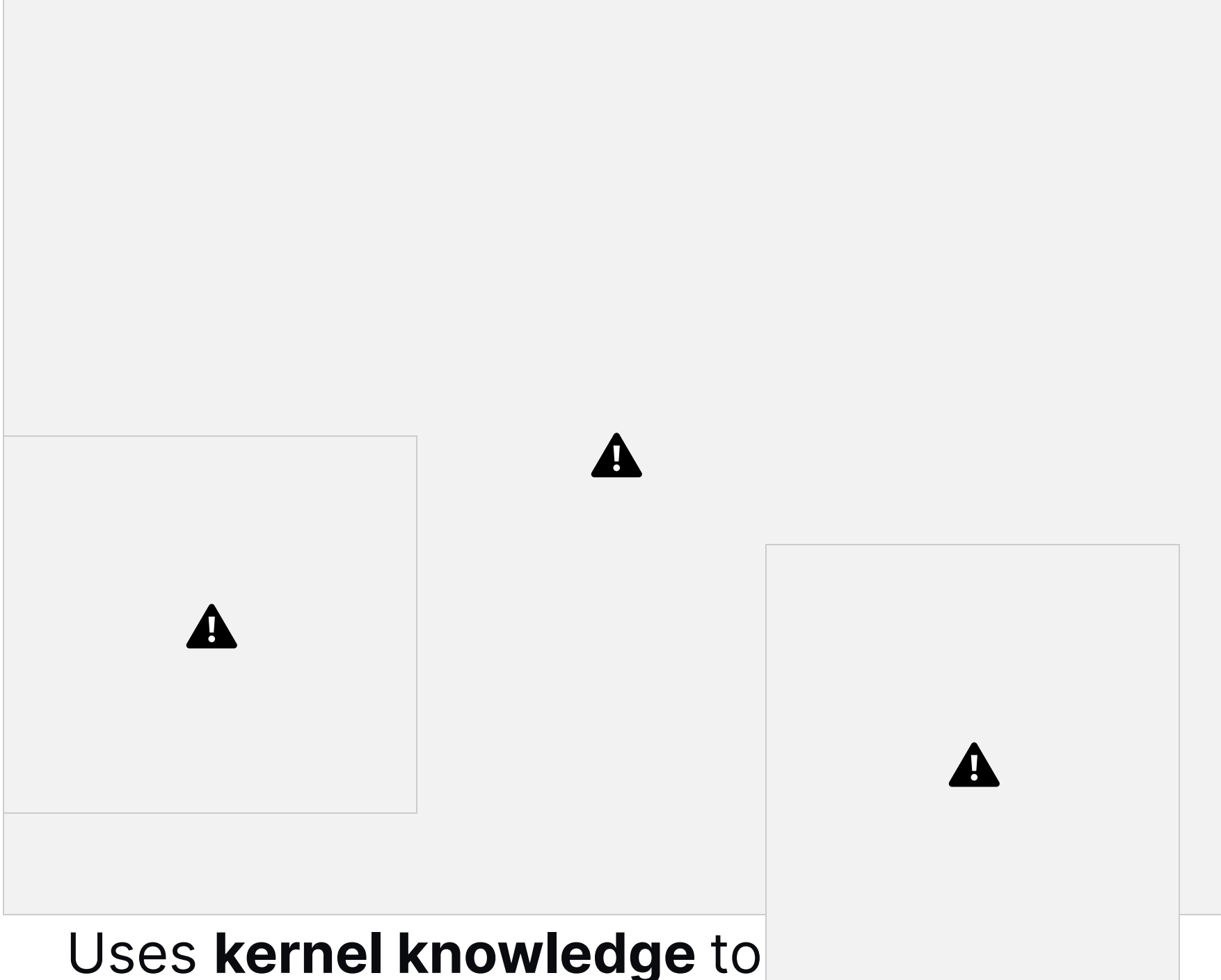
But needs **kernel 5.7+**
& Kubernetes context?



How stable is the Linux kernel?



Cilium Tetragon



Uses **kernel knowledge** to

• Safe places to make checks

+ eBPF benefits

- Dynamic
- Protect pre-existing processes

hook into
sufficiently stable functions

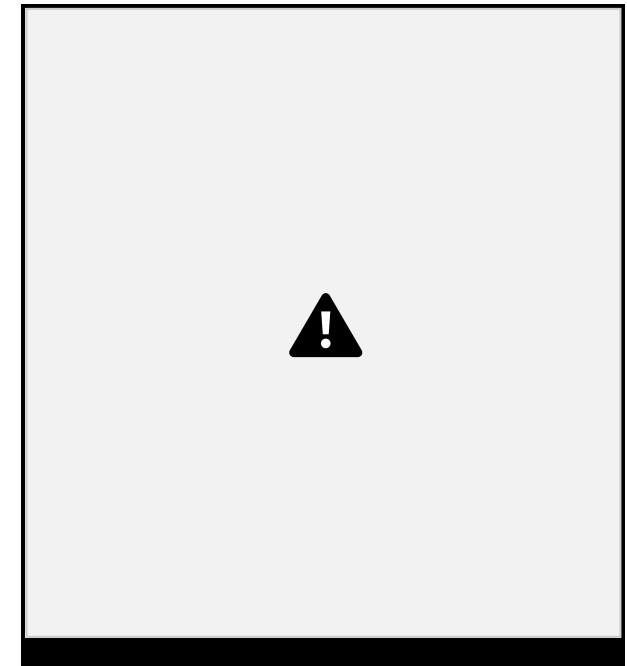
Adds Kubernetes context



A *Tetragonisca angustula* bee

guarding the nest-entrance

Bibafu



@lizrice

Cilium Tetragon tracing policy


```
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name: "etc-files"
spec:
kprobes:
  - call: "fd_install" ...
matchArgs:
  - index: 1
operator: "Prefix"  values:
  - "/etc/"
...
```



+ Policy “follows” file
descriptor through read, write
& close events

@lizrice



Cilium Tetragon observe security events

```
$ kubectl logs ds/tetragon -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  
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

@lizrice



Combined network and runtime visibility



@lizrice



eBPF preventative runtime security



Cilium network policy → eBPF programs drop packets



@lizrice



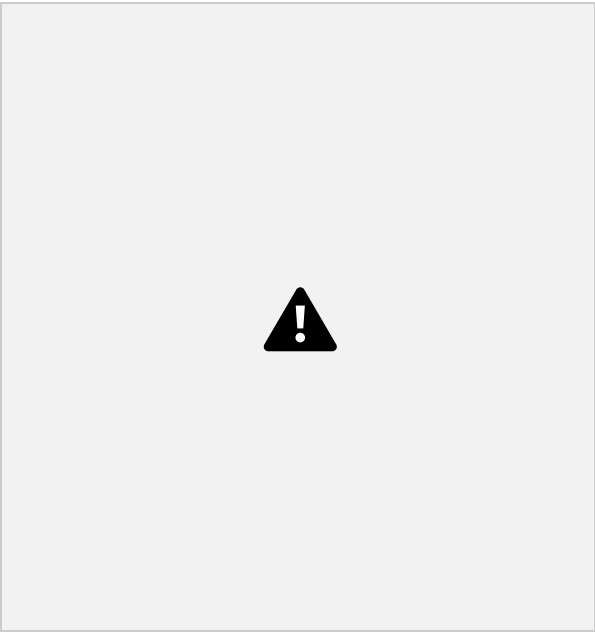
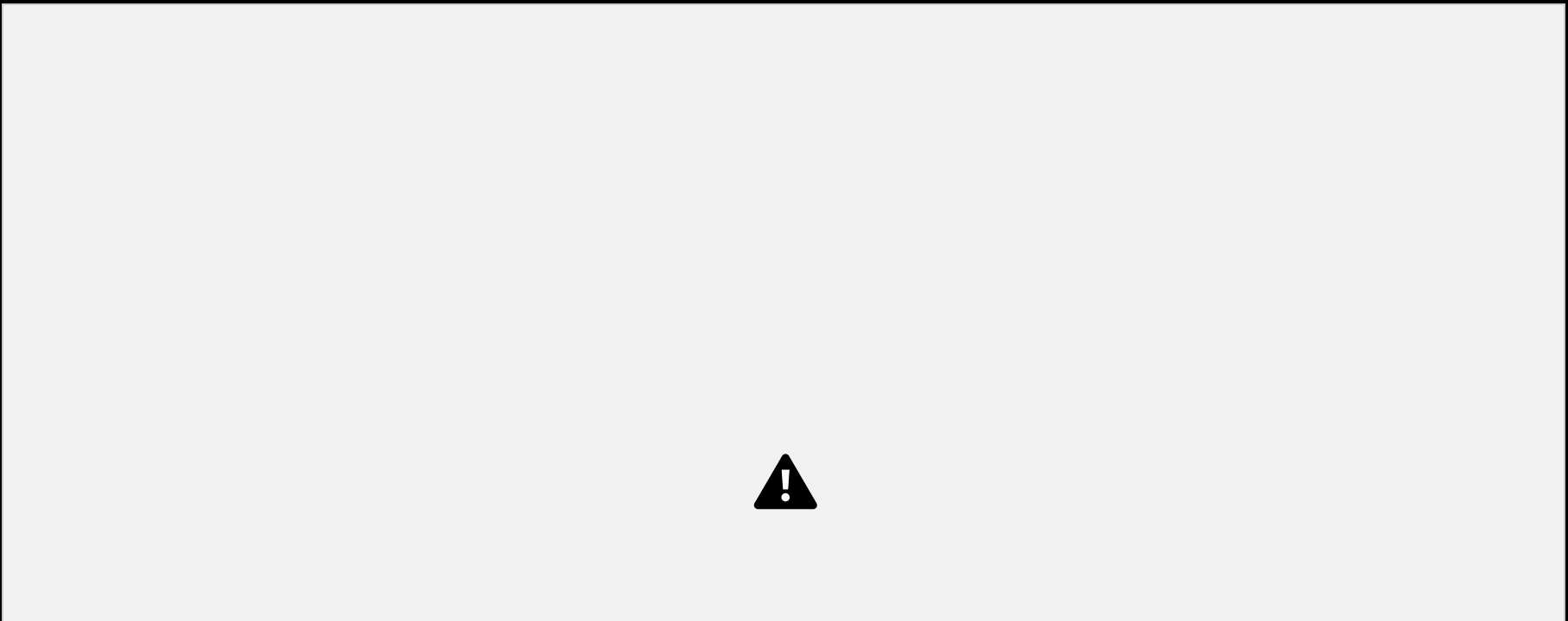
Preventative actions from user space



@lizrice



Preventative actions from kernel





Cilium Tetragon observe

```
$ kubectl logs ds/tetragon -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  
write default/xwing /usr/bin/vi /etc/passwd 1269 bytes  
exit default/xwing /usr/bin/vi /etc/passwd SIGKILL
```

Killed before write

@lizrice



eBPF security









Thank you



