Bypass Falco

Leonardo Di Donato - 20 Nov 2020

Whoami

Leonardo Di Donato

Open Source Software Engineer

Falco Maintainer

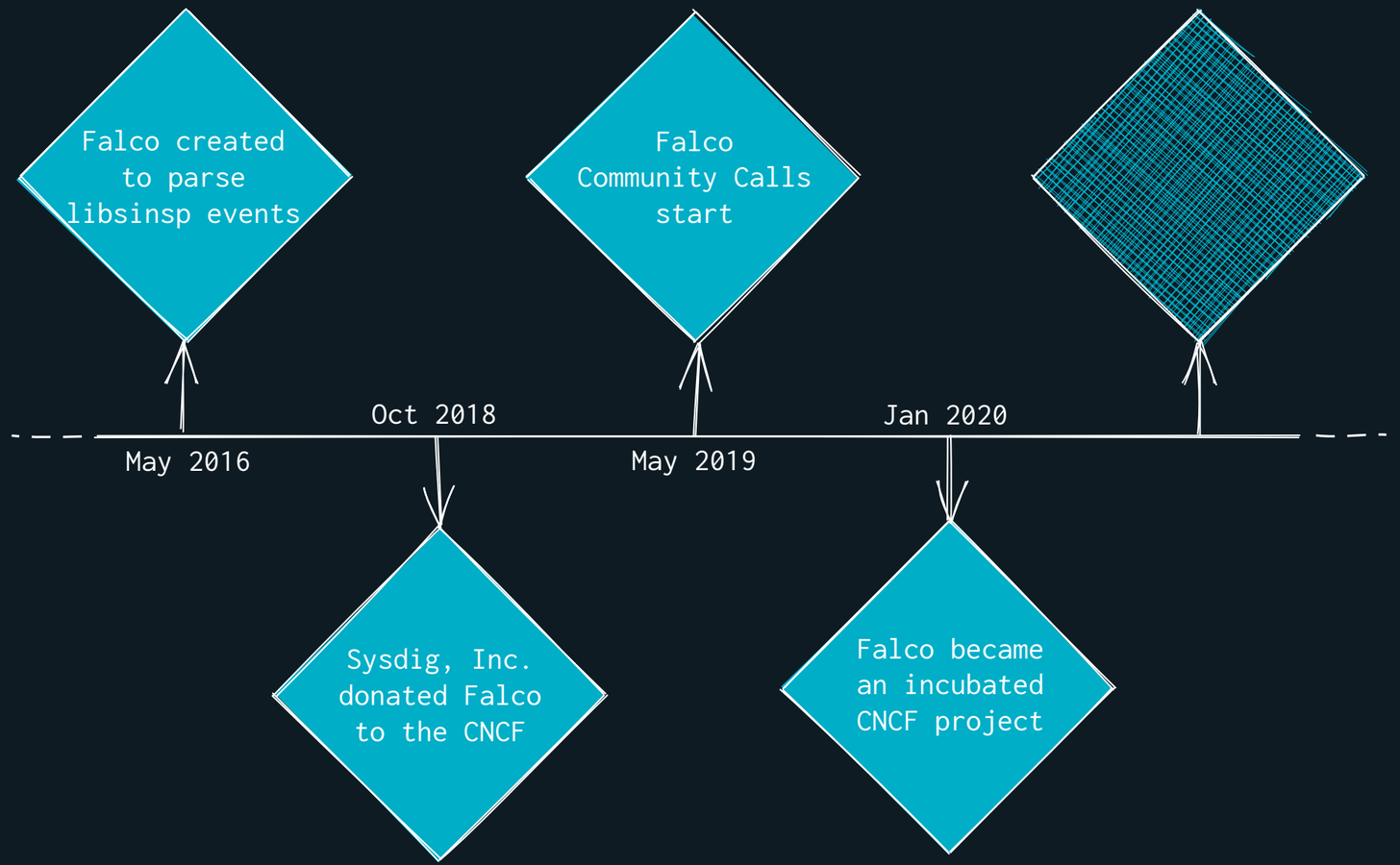
CLOUD NATIVE COMPUTING FOUNDATION
INCUBATING PROJECT

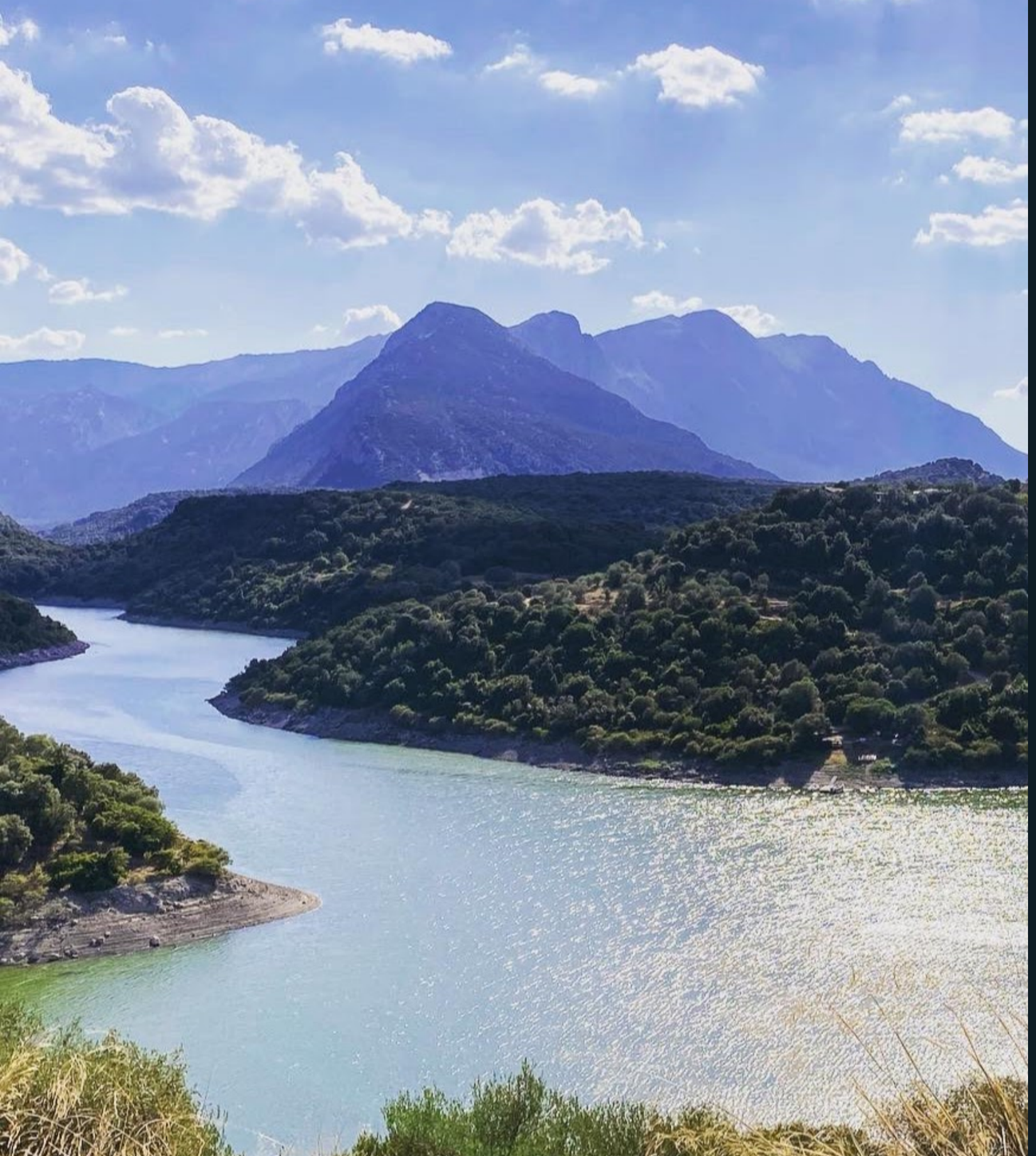
Falco

sysdig

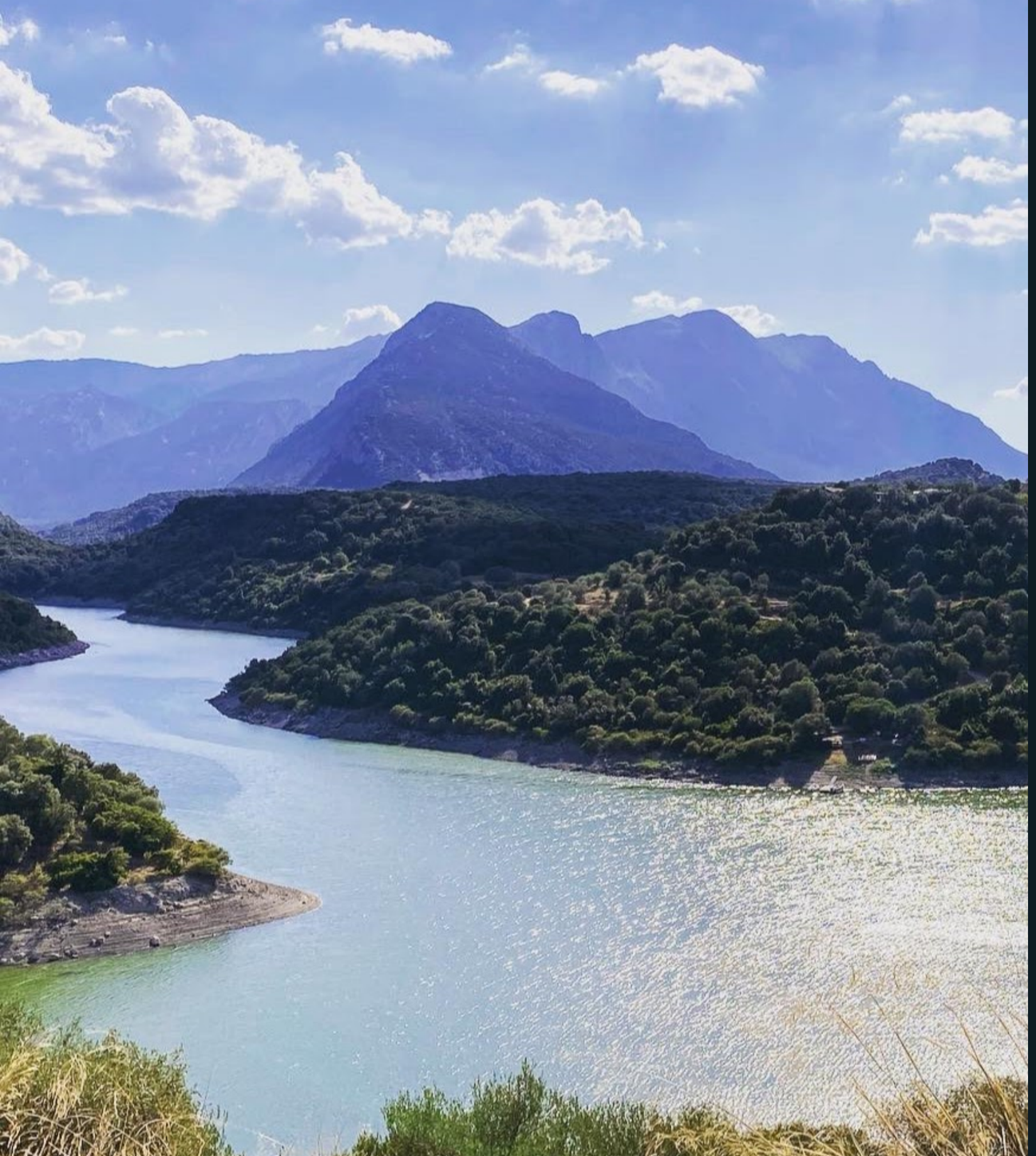
@leodido

A timeline always works fine



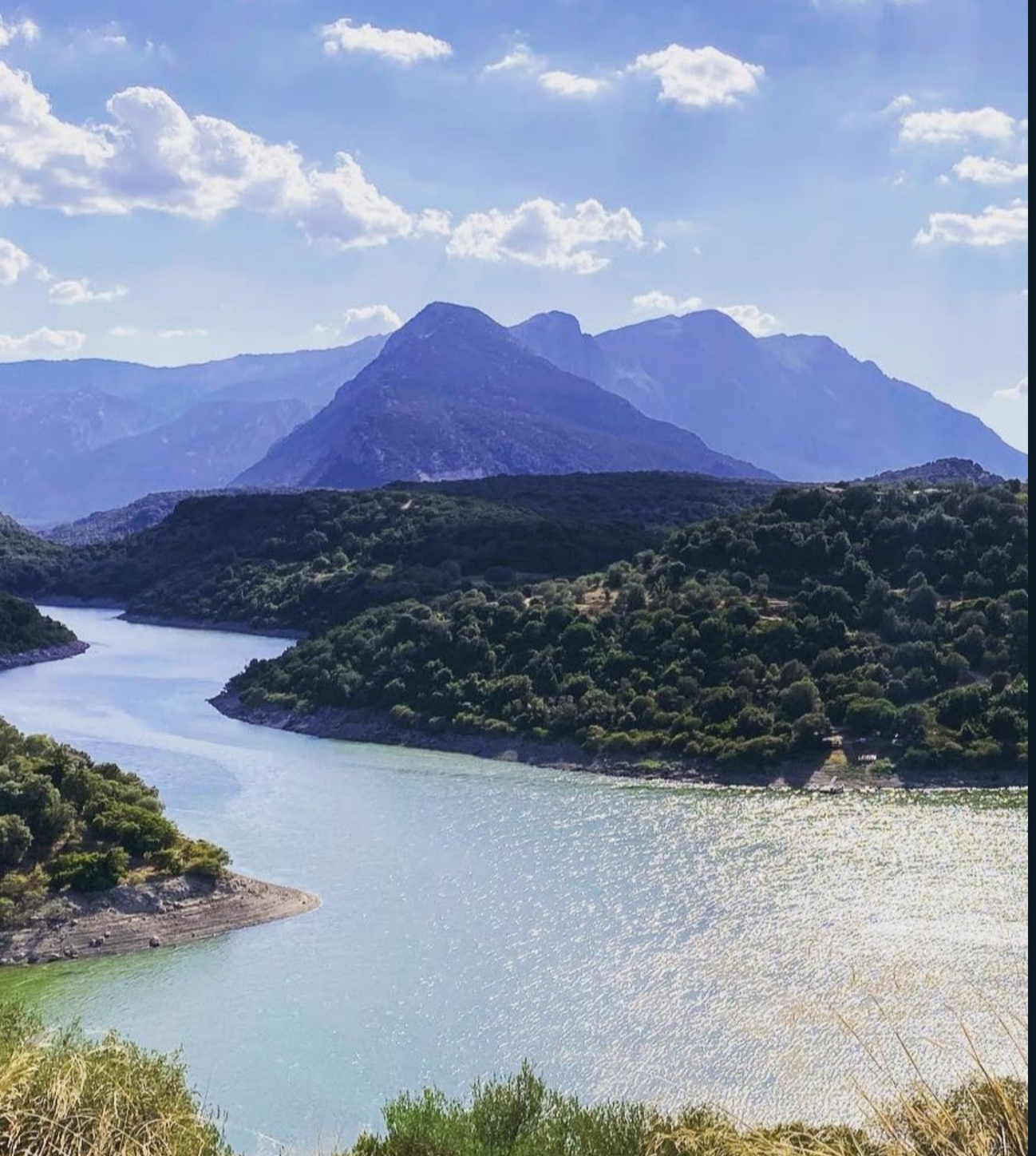


Contents



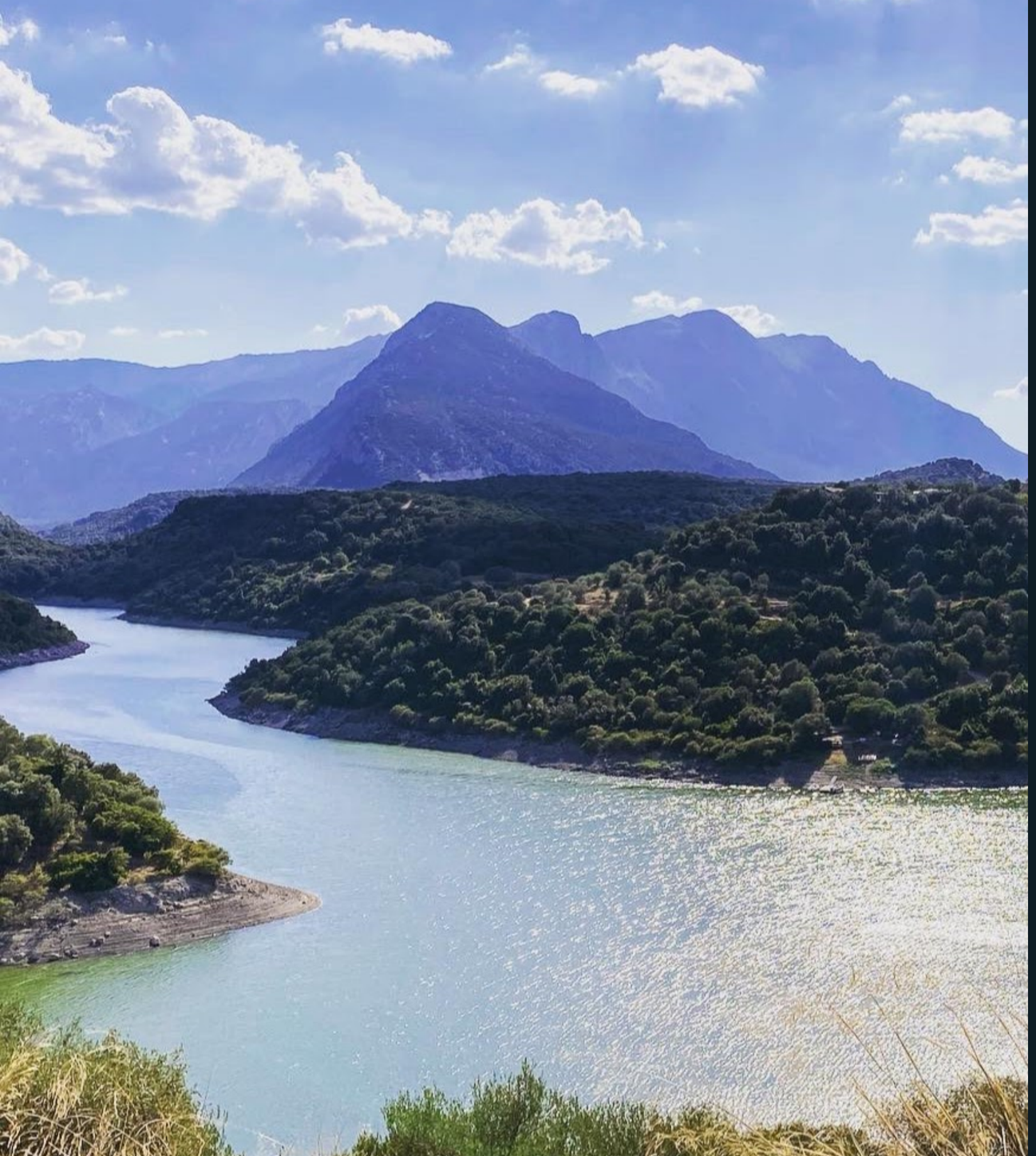
Contents

- Rationale



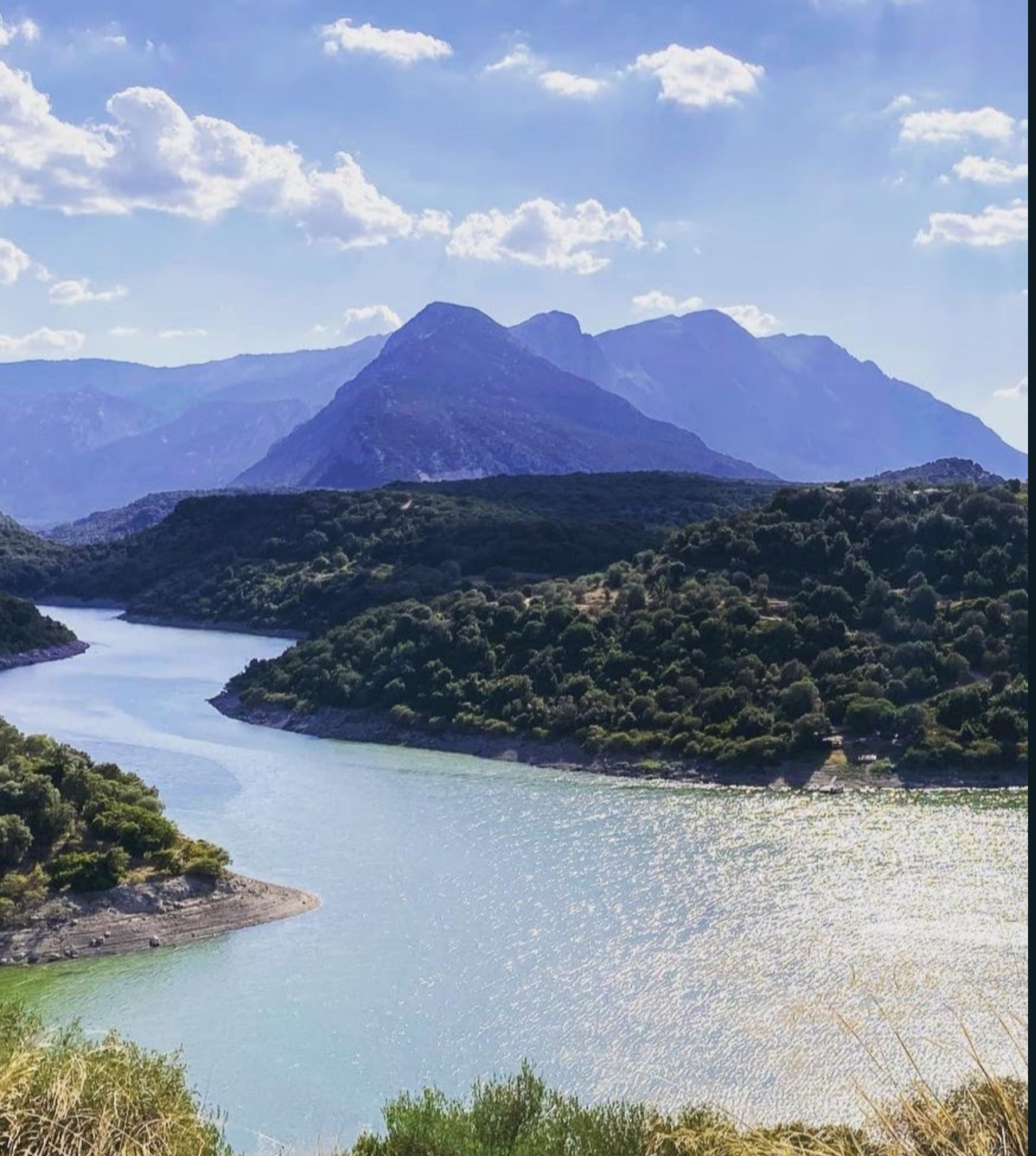
Contents

- Rationale
- Falco



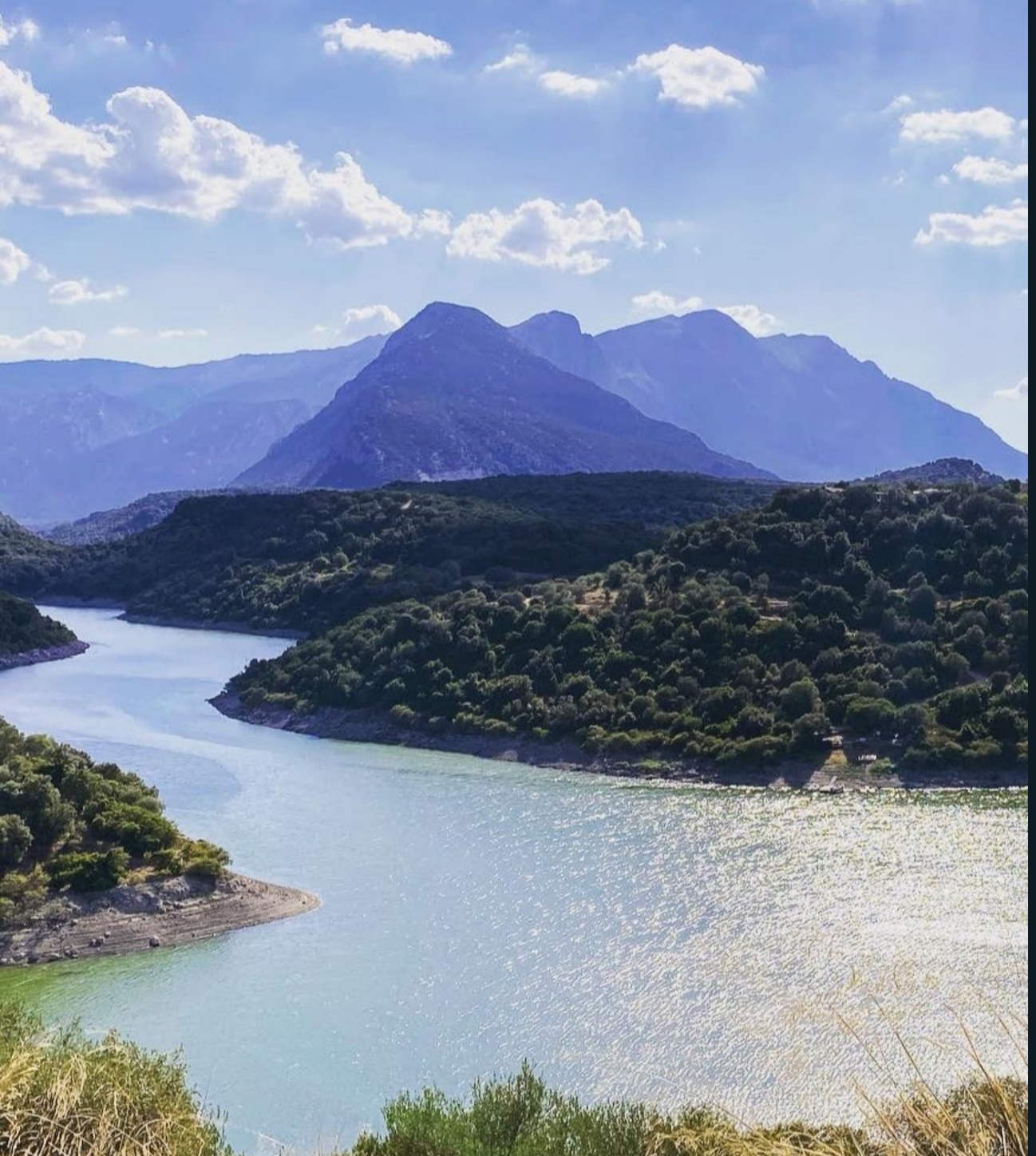
Contents

- Rationale
- Falco
  - What's runtime security?



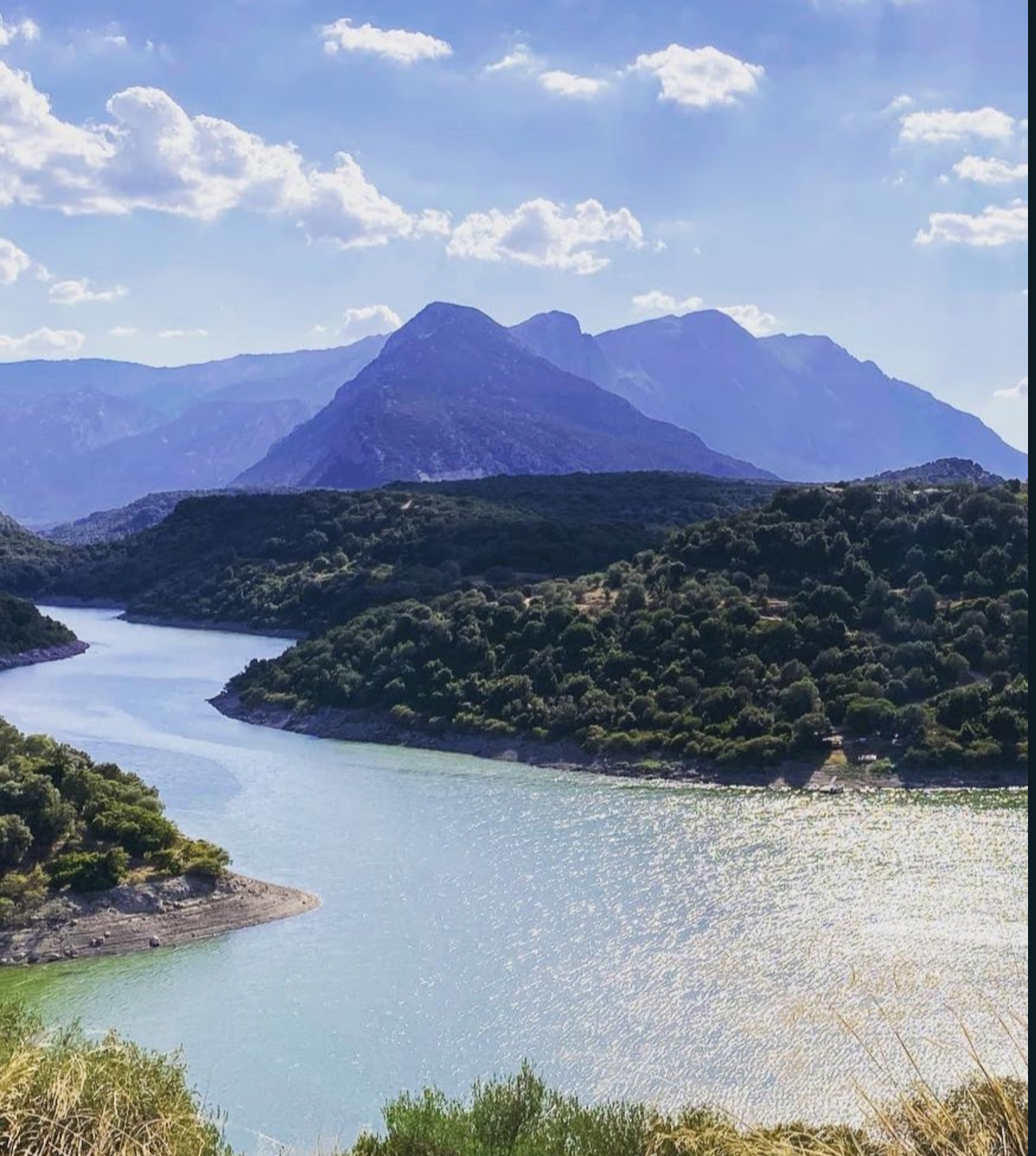
Contents

- Rationale
- Falco
  - What's runtime security?
  - How does it work?



Contents

- Rationale
- Falco
  - What's runtime security?
  - How does it work?
- Bypass!



Contents

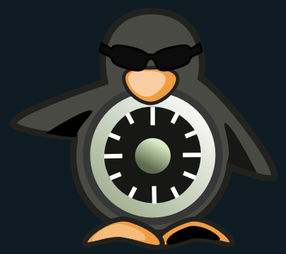
- Rationale
- Falco
  - What's runtime security?
  - How does it work?
- Bypass!
  - /honk

You gonna get fired for this. It's a mistake.

— my father.

Prevention + Detection

Use **policies** to change the behavior of a process by preventing syscalls from succeeding (also killing the process).



Use **policies** to monitor the behavior of a process and notify when its behavior steps outside the policy.

Prevention is not enough.

Combine with runtime detection tools. Use a defense-in-depth 🔗 strategy.

Runtime Security

She’s **Kelly**. 💔

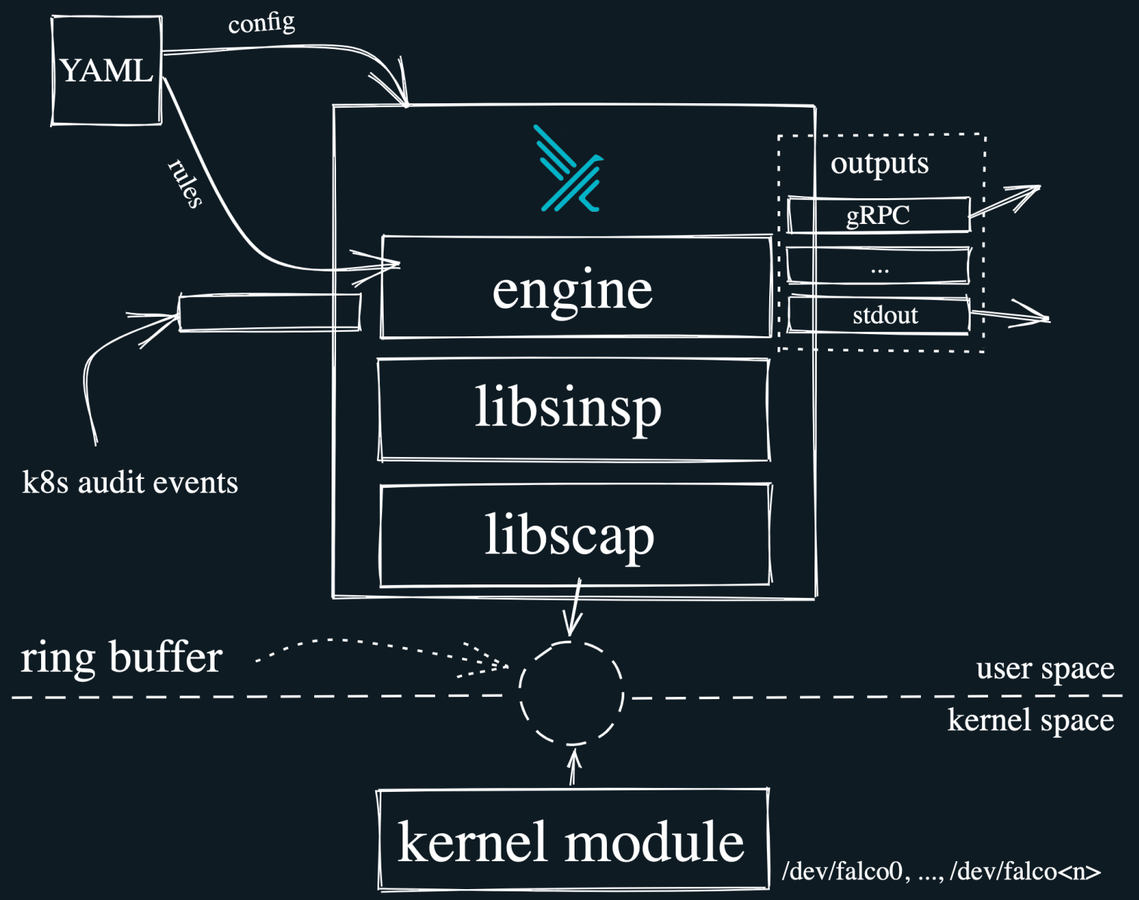
I have a lock on my front door and an alarm. She alerts me when things aren’t going right, when little bro is misbehaving or if there’s someone suspicious outside or nearby.

She detects **runtime anomalies** in my life at home.

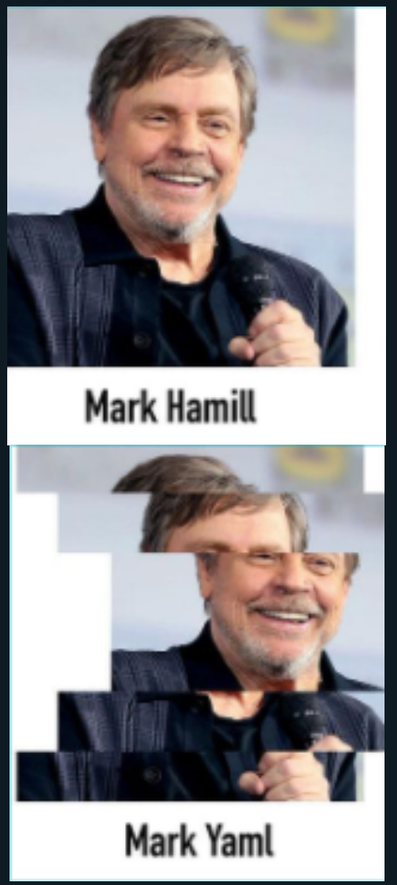
Still... Bad people were able to defy her and break into my house.


There is no such thing as perfect security.

How Falco works?



Falco rules are YAML![1]



[1] default rulesets 📋

@leodido

Falco rules are YAML![1]

- lists

[1] default rulesets 📋

@leodido

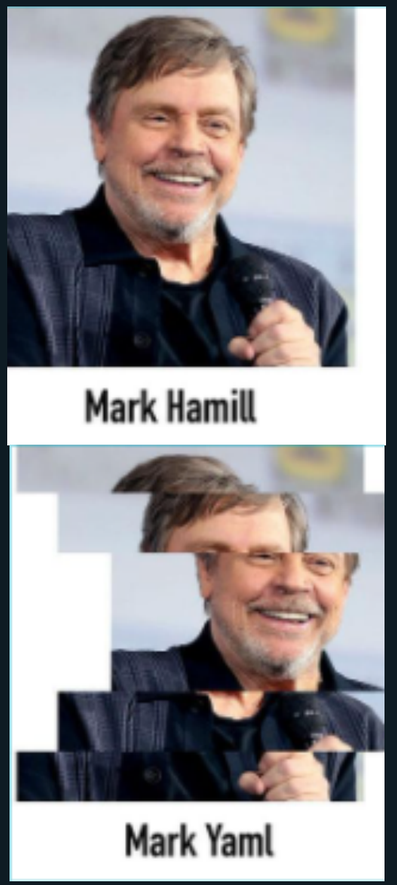
Falco rules are YAML![1]

- lists
- conditions

[1] default rulesets 📋

Falco rules are YAML![1]

- lists
- conditions
- macros



[1] default rulesets 📋

@leodido

@leodido

Falco rules are YAML![1]

- lists
- conditions
- macros
- priorities/severities

[1] default rulesets 📋

@leodido

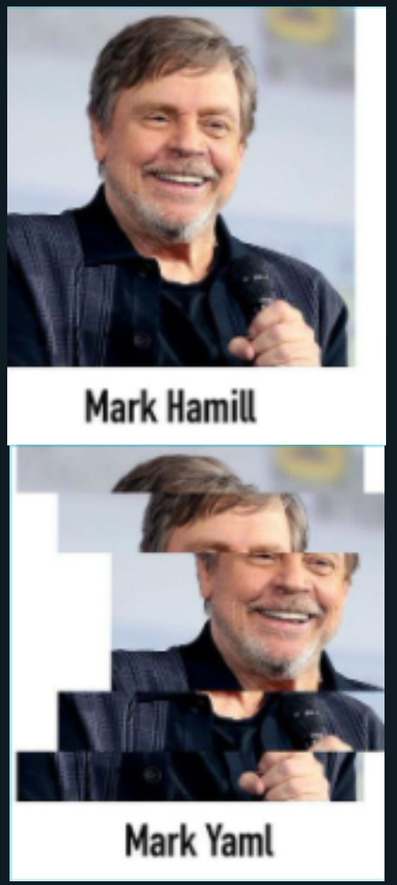
Falco rules are YAML![1]

- lists
- conditions
- macros
- priorities/severities
- (custom) output messages

[1] default rulesets 📋

Falco rules are YAML![1]

- lists
- conditions
- macros
- priorities/severities
- (custom) output messages
- tags

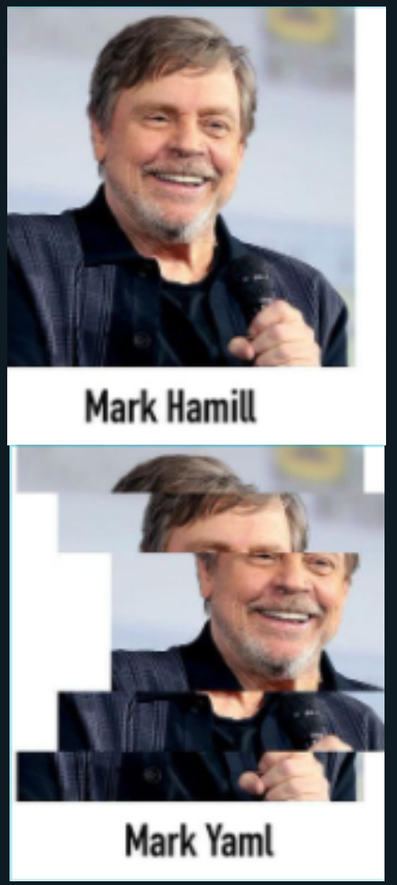


[1] default rulesets 📋

@leodido

Falco rules are YAML![1]

- lists
- conditions
- macros
- priorities/severities
- (custom) output messages
- tags
- overrides

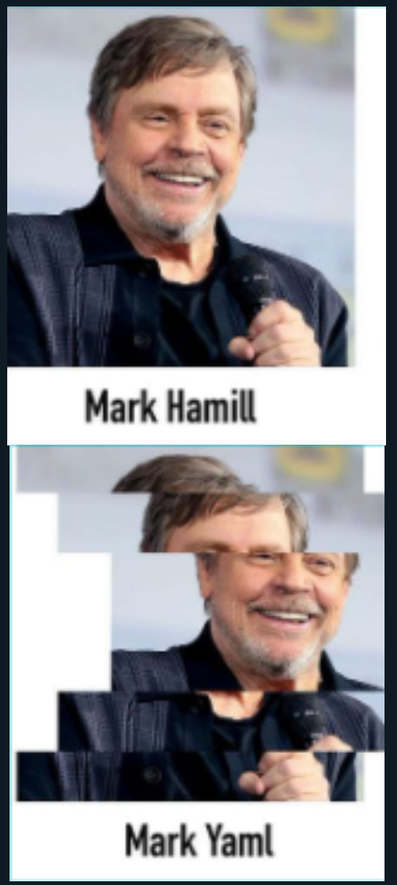


[1] default rulesets 📋

@leodido

Falco rules are YAML![1]

- lists
- conditions
- macros
- priorities/severities
- (custom) output messages
- tags
- overrides
- exceptions (soon)



[1] default rulesets 📋

Detect attempts to spawn a shell from non-shell applications[2]

```yaml
- rule: Run shell untrusted
  desc: >
  An attempt to spawn a shell below a non-shell application.
  Specific applications are monitored.
  condition: >
    spawned_process
    and shell_procs
    and proc.pname exists
    and protected_shell_spawner
    and not proc.pname in (shell_binaries, gitlab_binaries,
        cron_binaries, user_known_shell_spawn_binaries,
        needrestart_binaries, mesos_shell_binaries,
        erl_child_setup, exechealthz, PM2,
        PassengerWatchd, c_rehash, svlogd,
        logrotate, hhvm, serf, lb-controller,
        nvidia-installe, runsv, statsite, erlexec,
        calico-node, "puma reactor")
    and not proc.cmdline in (known_shell_spawn_cmdlines)
    and not ...
    and not user_shell_container_exclusions
  output: >
    Shell spawned by untrusted binary
    (user=%user.name user_loginuid=%user.loginuid
    shell=%proc.name parent=%proc.pname cmdline=%proc.cmdline
    pcmdline=%proc.pcmdline gparent=%proc.aname[2] ggparent=%proc.aname[3]
    aname[4]=%proc.aname[4] aname[5]=%proc.aname[5]
    aname[6]=%proc.aname[6] aname[7]=%proc.aname[7]
    container_id=%container.id image=%container.image.repository)
  priority: DEBUG
  tags: [shell, mitre_execution]
```

```yaml
- macro: spawned_process
  condition: evt.type = execve and evt.dir=<

- list: shell_binaries
  items: [ash, bash, csh, ksh, sh, tcsh, zsh, dash]

- macro: shell_procs
  condition: proc.name in (shell_binaries)

- list: protected_shell_spawning_binaries
  items: [
    http_server_binaries, db_server_binaries, nosql_server_binaries, mail_binaries,
    fluentd, flanneld, splunkd, consul, smbd, runsv, PM2
    ]

- macro: protected_shell_spawner
  condition: >
    (proc.aname in (protected_shell_spawning_binaries)
    or parent_java_running_zookeeper
    or ...
    or possibly_node_in_container)

- list: known_shell_spawn_cmdlines
  items: [
    '"sh -c uname -p 2> /dev/null"',
    '"sh -c uname -s 2>&1"',
    '"sh -c uname -r 2>&1"',
    '"sh -c uname -v 2>&1"',
    '"sh -c uname -a 2>&1"',
    '"sh -c ruby -v 2>&1"',
    ...
    '"sh -c /bin/sh -c ''date +%%s''"'
    ]
```

[2] rule definition 🔗

execveat

demo

Syscalls:
cross and delight

System call	Kernel	Notes
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	
accept(2)	2.0	See notes on socketcall(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bfin_spinlock(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
bind(2)	2.0	See notes on socketcall(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with __ARM_NR prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See chown(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
cmpxchg_badaddr(2)	2.6.36	Tile only (port removed in Linux 4.17)
connect(2)	2.0	See notes on socketcall(2)
copy_file_range(2)	4.5	
creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dma_memcpy(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	

Support them before Falco 1.0 🎯

👉 **falco#676**

@leodido

Syscalls:
cross and delight

◉ renameat2 ✅ (Falco >= 0.25)

Support them before Falco 1.0 🎯

👉 **falco#676**

@leodido

System call	Kernel	Notes
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	
accept(2)	2.0	See notes on socketcall(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bfin_spinlock(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
bind(2)	2.0	See notes on socketcall(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with __ARM_NR prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See chown(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
cmpxchg_badaddr(2)	2.6.36	Tile only (port removed in Linux 4.17)
connect(2)	2.0	See notes on socketcall(2)
copy_file_range(2)	4.5	
creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dma_memcpy(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	

Syscalls:
cross and delight

- renameat2 ✅ (Falco >= 0.25)
- copy_file_range ❌

Support them before Falco 1.0 🎯

👉 **falco#676**

@leodido

System call	Kernel	Notes
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	
accept(2)	2.0	See notes on socketcall(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bfin_spinlock(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
bind(2)	2.0	See notes on socketcall(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with __ARM_NR prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See chown(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
cmpxchg_badaddr(2)	2.6.36	Tile only (port removed in Linux 4.17)
connect(2)	2.0	See notes on socketcall(2)
copy_file_range(2)	4.5	
creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dma_memcpy(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	

Syscalls:
cross and delight

- renameat2 ✅ (Falco >= 0.25)
- copy_file_range ❌
- execveat ❌

Support them before Falco 1.0 🎯

👉 **falco#676**

@leodido

System call	Kernel	Notes
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	
accept(2)	2.0	See notes on socketcall(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bfin_spinlock(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
bind(2)	2.0	See notes on socketcall(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with __ARM_NR prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See chown(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
cmpxchg_badaddr(2)	2.6.36	Tile only (port removed in Linux 4.17)
connect(2)	2.0	See notes on socketcall(2)
copy_file_range(2)	4.5	
creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dma_memcpy(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	

Syscalls:
cross and delight

- renameat2 ✅ (Falco >= 0.25)
- copy_file_range ❌
- execveat ❌
- ...

Support them before Falco 1.0 🎯

👉 **falco#676**

@leodido

System call	Kernel	Notes
_llseek(2)	1.2	
_newselect(2)	2.0	
_sysctl(2)	2.0	
accept(2)	2.0	See notes on socketcall(2)
accept4(2)	2.6.28	
access(2)	1.0	
acct(2)	1.0	
add_key(2)	2.6.10	
adjtimex(2)	1.0	
alarm(2)	1.0	
alloc_hugepages(2)	2.5.36	Removed in 2.5.44
arc_gettls(2)	3.9	ARC only
arc_settls(2)	3.9	ARC only
arc_usr_cmpxchg(2)	4.9	ARC only
arch_prctl(2)	2.6	x86_64, x86 since 4.12
atomic_barrier(2)	2.6.34	m68k only
atomic_cmpxchg_32(2)	2.6.34	m68k only
bdflush(2)	1.2	Deprecated (does nothing) since 2.6
bfin_spinlock(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
bind(2)	2.0	See notes on socketcall(2)
bpf(2)	3.18	
brk(2)	1.0	
breakpoint(2)	2.2	ARM OABI only, defined with __ARM_NR prefix
cacheflush(2)	1.2	Not on x86
capget(2)	2.2	
capset(2)	2.2	
chdir(2)	1.0	
chmod(2)	1.0	
chown(2)	2.2	See chown(2) for version details
chown32(2)	2.4	
chroot(2)	1.0	
clock_adjtime(2)	2.6.39	
clock_getres(2)	2.6	
clock_gettime(2)	2.6	
clock_nanosleep(2)	2.6	
clock_settime(2)	2.6	
clone2(2)	2.4	IA-64 only
clone(2)	1.0	
clone3(2)	5.3	
close(2)	1.0	
cmpxchg_badaddr(2)	2.6.36	Tile only (port removed in Linux 4.17)
connect(2)	2.0	See notes on socketcall(2)
copy_file_range(2)	4.5	
creat(2)	1.0	
create_module(2)	1.0	Removed in 2.6
delete_module(2)	1.0	
dma_memcpy(2)	2.6.22	Blackfin only (port removed in Linux 4.17)
dup(2)	1.0	
dup2(2)	1.0	
dup3(2)	2.6.27	

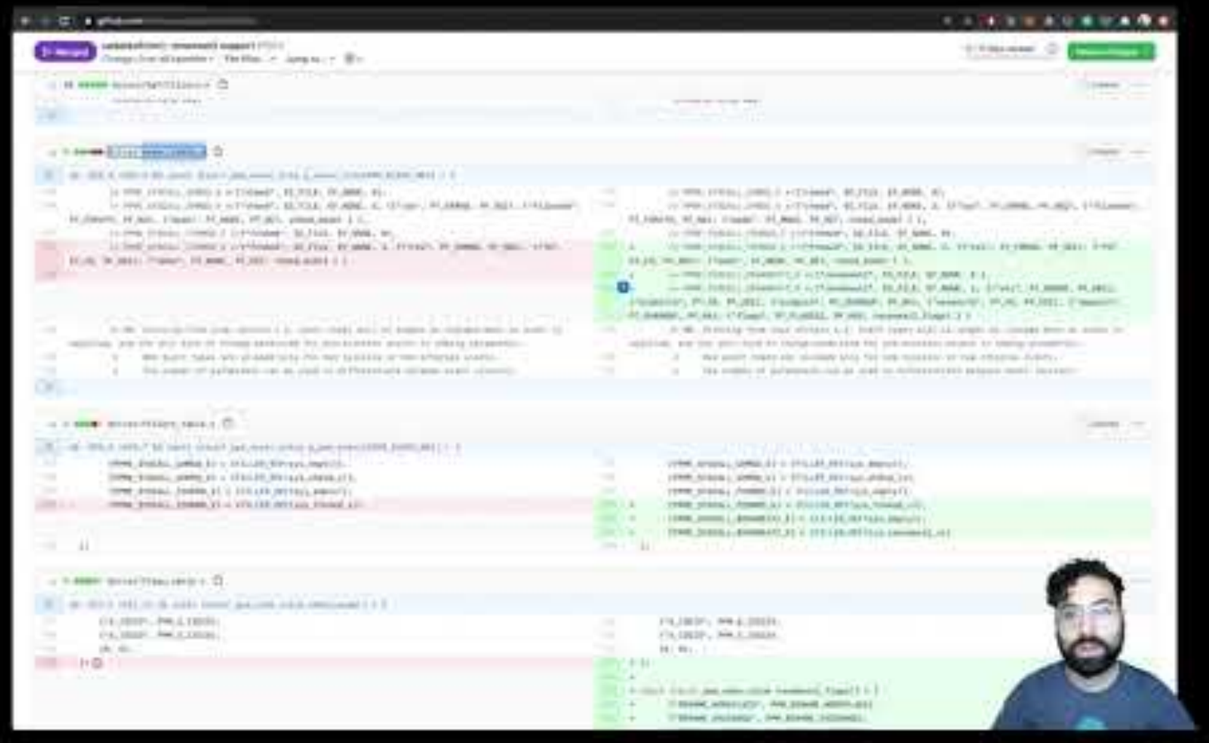
Missing syscalls

```
#!/usr/bin/env bash

DRIVER="/home/vagrant/workspace/draios/sysdig/"
HEADERS="/lib/modules/$(uname -r)/build/"
HEADERQUERY="asmlinkage long sys_"

SUP=$(grep -oh "__NR_\w*" "${DRIVER}/driver/syscall_table.c" | \
            grep -v ia32 | sed -e "s/__NR_//")
ALL=$(grep "${HEADERQUERY}" "${HEADERS}/include/linux/syscalls.h" | \
      awk '{print $3}' | sed -e "s/^sys_//" | \
      sed -e "s/(/ /g" | awk '{print $1}')

sdiff \
  <(echo "${SUP}" | sort | uniq) \
  <(echo "${ALL}" | sort | uniq)
```

```
11  chown16                         chown16
12  chroot                          chroot
13  ...
14  clone                           clone
15                                > clone3
16  close                           close
17  connect                         connect
18                                > copy_file_range
19  creat                           creat
20  ...
21  epoll_wait                      epoll_wait
22  eventfd                         eventfd
23  eventfd2                        eventfd2
24  execve                          execve
25                                > execveat
26  ...
27  fchdir                          fchdir
28  fchmod                          fchmod
29  fchmodat                        fchmodat
30  fchown                          fchown
31  fchown16                        fchown16
32  fchownat                        fchownat
33  ...
34  fork                            fork
35  ...
36  madvise                         madvise
37                                > mbind
38                                > membarrier
39                                > memfd_create
40                                > migrate_pages
41  mkdir                           mkdir
42  mkdirat                         mkdirat
43  ...
44  read                            read
45                                > readahead
46  readlink                        readlink
47  readlinkat                      readlinkat
48  ...
49  rename                          rename
50  renameat                        renameat
51  renameat2                       renameat2
```

Is tracing syscalls only enough?

👉 **io_uring**



@leodido

rename() renames a file, moving it between directories if required. Any other hard links to the file (as created using link(2)) are unaffected. Open file descriptors for *oldpath* are also unaffected.

How to support a new syscall

demo

renameat2 support

If *oldpath* refers to a symbolic link, the link is renamed; if *newpath* refers to a symbolic link, the link will be overwritten.

renameat()
The **renameat**() system call operates in exactly the same way as **rename**(), except for the differences described here.

If the pathname given in *oldpath* is relative, then it is interpreted relative to the directory referred to by the file descriptor *olddirfd* (rather than relative to the current working directory of the calling process, as is done by **rename**() for a relative pathname).

If *oldpath* is relative and *olddirfd* is the special value **AT_FDCWD**, then *oldpath* is interpreted relative to the current working directory of the calling process (like **rename**()).

If *oldpath* is absolute, then *olddirfd* is ignored.

The interpretation of *newpath* is as for *oldpath*, except that a relative pathname is interpreted relative to the directory referred to by the file descriptor *newdirfd*.

See openat(2) for an explanation of the need for **renameat**().

renameat2()
renameat2() has an additional *flags* argument. A **renameat2**() call with a zero *flags* argument is equivalent to **renameat**().

The *flags* argument is a bit mask consisting of zero or more of the following flags:

Detect package management process ran inside container...

Error Package management process launched in container (user=root user_loginuid=-1 command=apt update -y container_id=6640634d89d4 container_name=testdpkg image=ubuntu:18.04)

```
- macro: never_true
  condition: (evt.num=0)

- macro: spawned_process
  condition: evt.type = execve and evt.dir=<

- macro: container
  condition: (container.id != host)

- list: deb_binaries
  items: [
    dpkg, dpkg-preconfigu, dpkg-reconfigur, dpkg-divert, apt, apt-get, aptitude,
    frontend, preinst, add-apt-reposit, apt-auto-remova, apt-key,
    apt-listchanges, unattended-upgr, apt-add-reposit, apt-config, apt-cache
    ]

- list: package_mgmt_binaries
  items: [..., deb_binaries, alternatives, pip, pip3, apk, gem, snapd, ...]

- macro: package_mgmt_procs
  condition: proc.name in (package_mgmt_binaries)

- macro: package_mgmt_ancestor_procs
  condition: proc.pname in (package_mgmt_binaries) or
             proc.aname[2] in (package_mgmt_binaries) or
             proc.aname[3] in (package_mgmt_binaries) or
             proc.aname[4] in (package_mgmt_binaries)

- macro: user_known_package_manager_in_container
  condition: (never_true)
```

```
- rule: Launch Package Management Process in Container
  desc: Package management process ran inside container
  condition: >
    spawned_process
    and container
    and user.name != "_apt"
    and package_mgmt_procs
    and not package_mgmt_ancestor_procs
    and not user_known_package_manager_in_container
  output: >
    Package management process launched in container
    (user=%user.name user_loginuid=%user.loginuid command=%proc.cmdline
    container_id=%container.id container_name=%container.name
    image=%container.image.repository:%container.image.tag)
  priority: ERROR
  tags: [process, mitre_persistence]
```

Let's do it again

demo

Mitigations/Considerations

Advice

Mitigations/Considerations

- Monitor symlinks?

Advice

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic

Advice

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective

Advice

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

- Containers from scratch

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

- Containers from scratch
- Read-only entrypoint

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

- Containers from scratch
- Read-only entrypoint
- One data path with `no-exec` flag

Mitigations/Considerations

- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

- Containers from scratch
- Read-only entrypoint
- One data path with `no-exec` flag
- Falco rule to monitor that only the entrypoint executes

Mitigations/Considerations

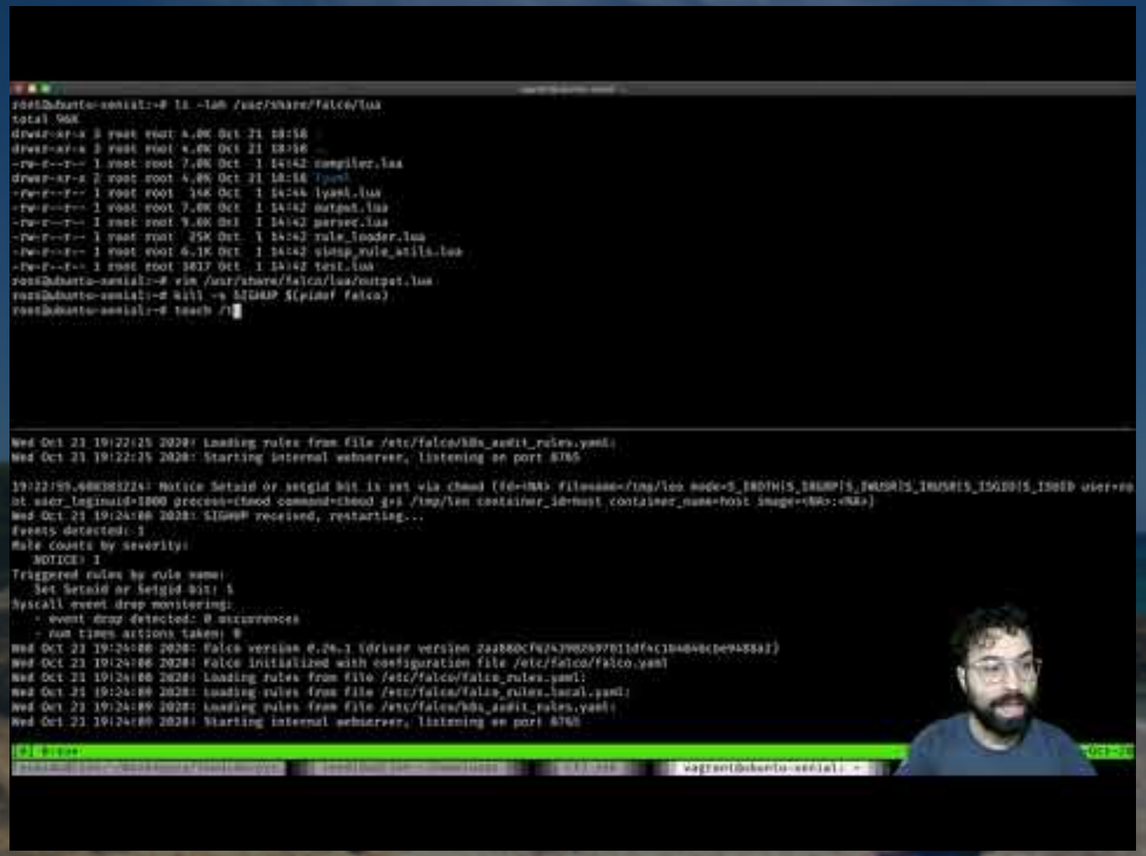
- Monitor symlinks?
  - Ok, but better if automatic
- Ruleset can be ineffective
  - The effectiveness depends on various rules because rules are interconnected

Advice

- Containers from scratch
- Read-only entrypoint
- One data path with `no-exec` flag
- Falco rule to monitor that only the entrypoint executes
- Monitor copies, renames, symlinks, open...

Close the gate of the (Lua) outputs?

demo



Solution?

Remove Lua.

Solution?

Remove Lua.

- Falco outputs refactoring

Solution?

Remove Lua.

- Falco outputs refactoring
- Falco outputs improvements

Solution?

Remove Lua.

- Falco outputs refactoring
- Falco outputs improvements
- **TODO**: rewrite Falco rule parser and engine in C++

Thanks and Honks!

Does anyone have any questions?

KubeCon | CloudNativeCon North America 2020 Virtual

- twitter.com/leodido
- gh:leodido
- gh:falcosecurity/falco
- slack.k8s.io, #falco channel