SOC FORUM 2023

EDR vs Containers: актуальные проблемы

Москва 2023

SOC FORUM 2023

Владислав Лашкин

Руководитель отдела противодействия киберугрозам центра исследования Solar 4RAYS

ГК «СОЛАР»

Дмитрий Евдокимов

Founder&CTO

LUNTRY

План выступления

SOC FORUM 2023

[01] КОНТЕЙНЕРНЫЙ МИР

[02] ВИДЫ RUNTIME-ЗАЩИТЫ И АГЕНТОВ

[03] ПОДХОДЫ К ОБНАРУЖЕНИЮ УГРОЗ И ИХ ПРОБЛЕМЫ

[04] ПРИМЕРЫ ОБХОДА МЕТОДОВ ОБНАРУЖЕНИЯ

[05] ВЫВОДЫ

КОНТЕЙНЕРНЫЙ МИР

Linux-контейнеры (не Windows-контейнеры)

SOC FORUM 2023

Будем говорить только про Linux-контейнеры.
Пока Windows-контейнеры используются только для обхода средств защиты (antivirus, EDR) =)

"CONTAIN YOURSELF: STAYING UNDETECTED USING THE WINDOWS CONTAINERS ISOLATION FRAMEWORK", DANIEL AVINOAM, DEFCON 31

- Ransomware/Wiper protection bypass
- DLP/Secured folders write bypass
- ETW-based correlations bypass

Linux-контейнеры 101

SOC FORUM 2023

CONTAINER – ЭТО LINUX PROCESS С ОПРЕДЕЛЁННЫМИ СВОЙСТВАМИ/ОГРАНИЧЕНИЯМИ

- Что можно увидеть: namespaces (pid, user, uts, ipc, net, mnt), pivot_root (+ image)
- Что можно делать: Capabilities, seccomp, LSMs
- Что можно использовать: Control group (процессор, память, устройства…)

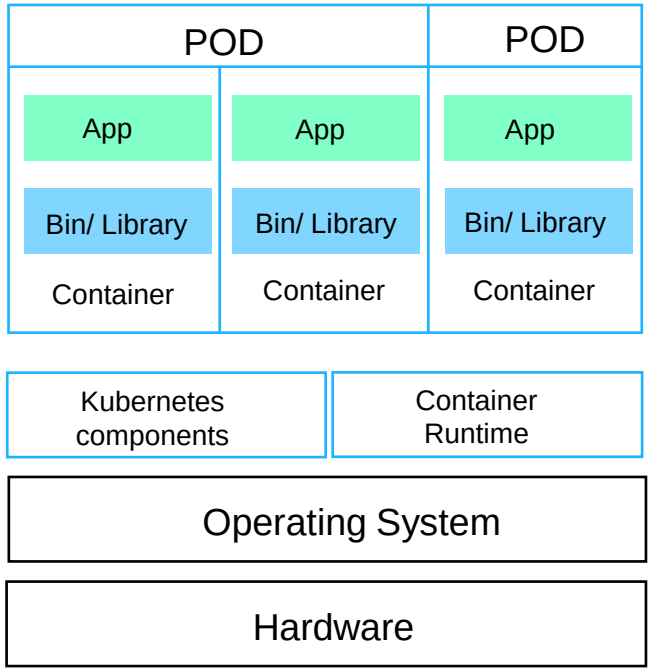
```
root     2966156  0.0  0.0 110128  5932 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2966174  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2966375  0.0  0.0 108720  6356 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
sadm     2966394  0.0  0.0 827512 19728 ?        Ssl  Nov19   0:00  |   \_ node /usr/bin/nodemon /src/index.js
sadm     2966421  0.0  0.0   4460    80 ?        S    Nov19   0:00  |       \_ sh -c node /src/index.js
sadm     2966422  0.0  0.0 967396 16596 ?        Sl   Nov19   0:00  |           \_ node /src/index.js
root     2988902  0.0  0.0 108720  5408 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2988922  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2989066  0.0  0.0 108720  5408 ?        Sl   Nov19   0:26  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989099  0.0  0.0  31000 23956 ?        Ss   Nov19   0:42  |   \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads 1 --
root     2989116  0.3  0.1 142092 48964 ?        Sl   Nov19  16:50  |       \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads
root     2989333  0.0  0.0 110128  5404 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989352  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root      596808  0.0  0.0 110128  6316 ?        Sl   Nov20   0:06  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      596827  0.0  0.0   1020     4 ?        Ss   Nov20   0:00  |   \_ /pause
root      598309  0.0  0.0 110128  6224 ?        Sl   Nov20   0:07  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      598334  1.4  5.5 7236340 1832196 pts/0 Ssl+ Nov20  39:39      \_ /docker-java-home/bin/java -Djava.util.logging.config.file=/opt/atlassian/conflue
root      599854  1.0  1.3 7007820 427956 pts/0  Sl+  Nov20  28:11      |   \_ /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -classpath /opt/atlassian/conf
root      701694  0.0  0.0   4288   764 ?        Ss+  Nov20   0:00      \_ /bin/sh
```

Kubernetes 101

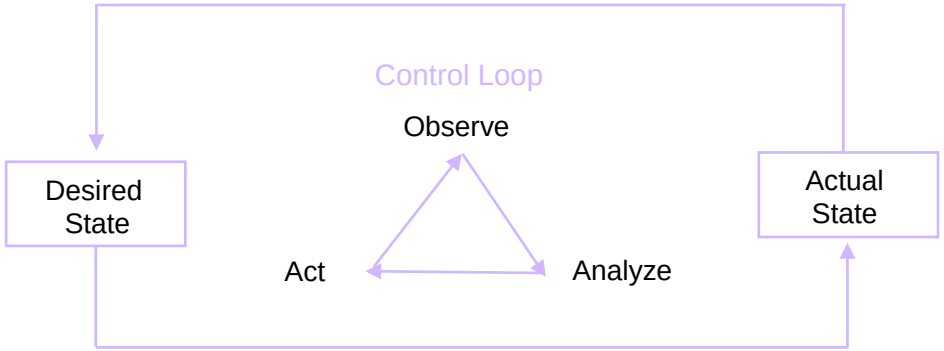
SOC FORUM 2023

PLATFORM AS
A SERVICE (PaaS)

- Configuration
- Function
- Applications
- Runtime
- Containers
- Operation Systems
- Hardware



Controller's Concept
Concept: Control Loop (Reconciliation Loop)
Read real resources



Change internal/external resources

Рекомендация: «Сочетание несочетаемого в Kubernetes: удобство, производительность, безопасность», Дмитрий Евдокимов

ВИДЫ RUNTIME-ЗАЩИТЫ И АГЕНТОВ

Виды защиты

SOC FORUM 2023

- Isolation
  - Дополнительный уровень изоляции от ядра хостовой ОС
  - WASM
    - WasmEdge, Wasmtime
  - Sandbox
    - gVisor
  - MicroVM
    - Kata containers

- Detection
  - Идентификация нежелательного действия
  - Сторонние решения

- Prevention
  - Невозможность выполнения нежелательного действия
  - seccomp
  - LSM
  - iptables/eBPF

- Reaction
  - Завершение процесса/контейнера/Pod постфактум после нежелательного события и/или дамп файловой системы и/или памяти
  - CRI интерфейс (stop/remove)
  - Kubelet Checkpoint API

Встроенные Runtime-защиты

SOC FORUM 2023

ВСТРОЕННЫЕ

- Linux capability
  - Определяет, какими правами обладает контейнер
  - Kubernetes SecurityContext

- Seccomp
  - Определяет, какие системные вызовы могут исполняться контейнером, а какие нет

- AppArmor (LSM)
  - Для Debian-based ОС
  - Определяет, какому файлу что можно, в терминах capabilities и файловых прав доступа в контейнере

- SeLinux (LSM)
  - Для RedHat-based ОС
  - Определяет, как контейнерные процессы взаимодействуют с хостовой ОС

- PARSEC (LSM)
  - AstraLinux
  - На текущий момент не отличает контейнерные процессы от хостовых

- NetworkPolicy
  - Реализуется CNI-плагином в Kubernetes (iptables, eBPF)
  - Определяет, что можно, а что нет только на уровне сети (L3/L4,L7)

Реализации сторонних средств защиты

SOC FORUM 2023

СТОРОННИЕ РЕШЕНИЯ

В КОНТЕЙНЕРЕ

- Библиотека
  - User mode
  - Подмена low-level runtime
  - LD_PRELOAD, dlopen, патчинг GOT-таблиц и т. д.

ЗА ПРЕДЕЛАМИ КОНТЕЙНЕРА

- Sidecar-контейнер
  - User mode
- Sensor-based (очень привилегированный компонент)
  - Kernel module
  - eBPF

Специализация

SOC FORUM 2023

СПЕЦИАЛИЗИРОВАННЫЕ – ЭТО ТЕ, КОТОРЫЕ ПОНИМАЮТ КОНТЕКСТ КОНТЕЙНЕРОВ И КОНТЕКСТ СУЩНОСТЕЙ KUBERNETES

НЕСПЕЦИАЛИЗИРОВАННЫЕ

Auditd, Sysmon for Linux, классические EDR, …

СПЕЦИАЛИЗИРОВАННЫЕ

Luntry, Falco (на его базе Sysdig, StackRox…), Aqua Security, PaloAlto Prisma, Tetragon, Tracee (на его базе KubeArmor)…

Обработка данных на user space

SOC FORUM 2023

C\C++

- Libbpf
- Сложности при многопоточности

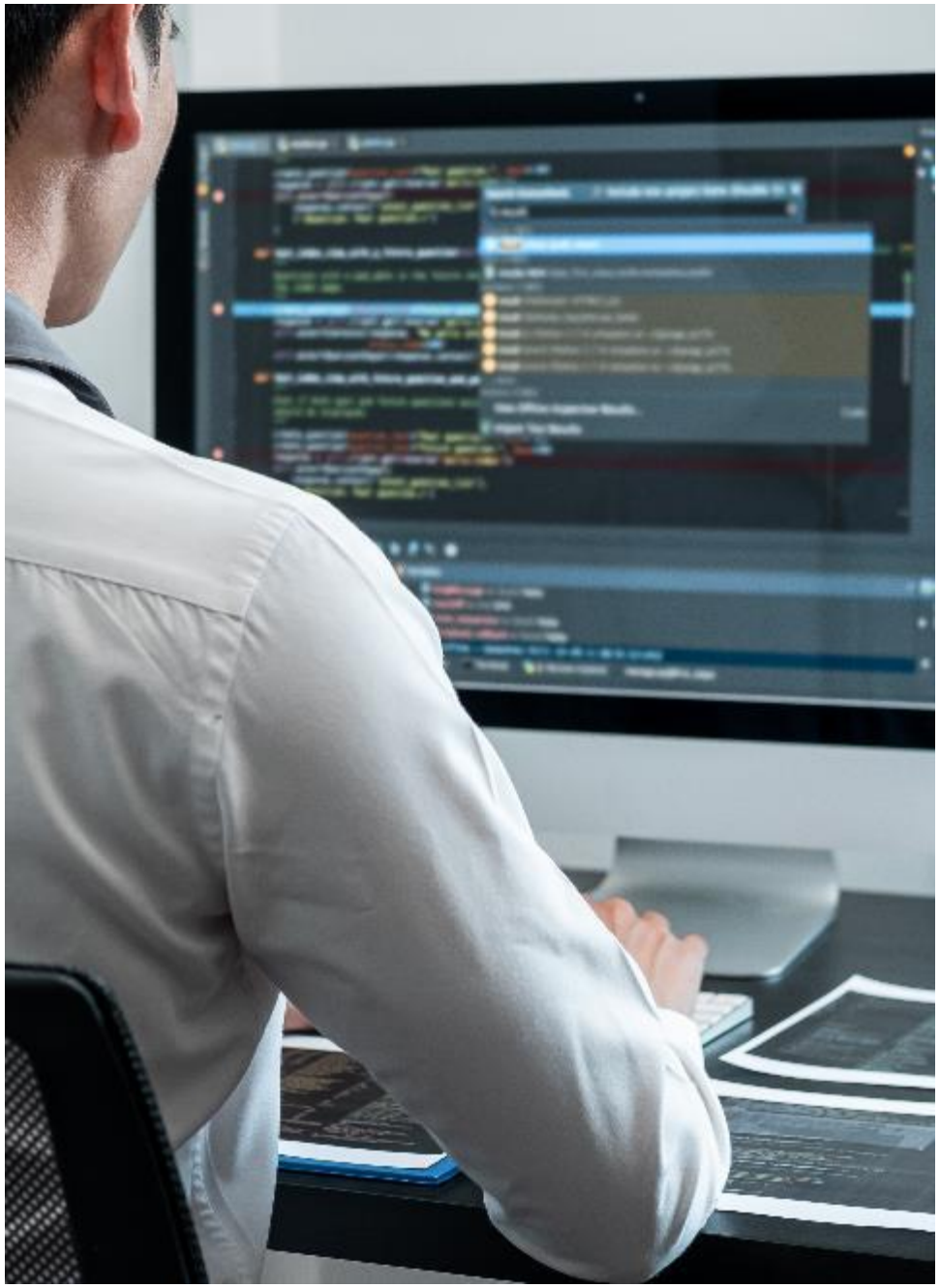
GO

- libbpf-go
- Сложности с приоритетами для потоков

RUST

- libbpf-rs
- Сложности с порогом вхождения

ПОДХОДЫ К ОБНАРУЖЕНИЮ УГРОЗ И ИХ ПРОБЛЕМЫ



Типы логики обнаружения

SOC FORUM 2023

СИГНАТУРНЫЙ

- Списки сущностей
  - Процесс, файловая операция, сетевая операция
- Списки системных вызов
  - sys_execve(), security_bprm_creds_from_file(), sys_write(), security_file_permission(), security_mmap_file(), ...

ГИБРИДНЫЙ

- Сочетание сигнатурного и поведенческого анализа

ПОВЕДЕНЧЕСКИЙ

- Не строгий профиль поведения
  - Список сущностей
- Строгий профиль поведения
  - Иерархия сущностей
- На уровне:
  - Образа
  - Контейнера
  - Микросервиса

Варианты размещения логического движка

SOC FORUM 2023

1 CLIENT-SIDE-ЛОГИКА

- Логика идентификации нежелательного действия находится на конечной точке
- С ростом числа базы знаний растет время и количество ресурсов, требуемых на обработку
- Доступна атакующему

2 SERVER-SIDE-ЛОГИКА

- Логика идентификации нежелательного действия находится на удаленной точке

Итоговая схема для сторонних решений

SOC FORUM 2023

	LUNTRY	FALCO (SYSDIG/ STACKROX/…)	AQUA SECURITY	PALOALTO PRISMA	TETRAGON	TRACEE (KUBEARMOR)
License type	Commercial	OpenSource	Commercial	Commercial	OpenSource	OpenSource
Protection type	Detection/Reaction	Detection/Reaction	Detection/Reaction /Prevention	Detection/Reaction/ Prevention	Detection/Reaction	Detection/Reaction
Installation type	Sensor	Sensor	Library	Library	Sensor	Sensor
Technology	eBPF	eBPF	User mode	User mode	eBPF	eBPF
Placement	Server-side	Client-side	Client-side	Client-side	Client-side	Client-side
Detection type	Поведенческий	Сигнатурный	Сигнатурный	Сигнатурный	Сигнатурный	Сигнатурный

ПРИМЕРЫ ОБХОДА МЕТОДОВ ОБНАРУЖЕНИЯ

Конфигурация тестового стенда

SOC FORUM 2023

1 MASTER NODE

- 4 CPU
- 8 GB RAM
- Ubuntu 22.04
- 6.2.0-34-generic

1 WORKER NODE

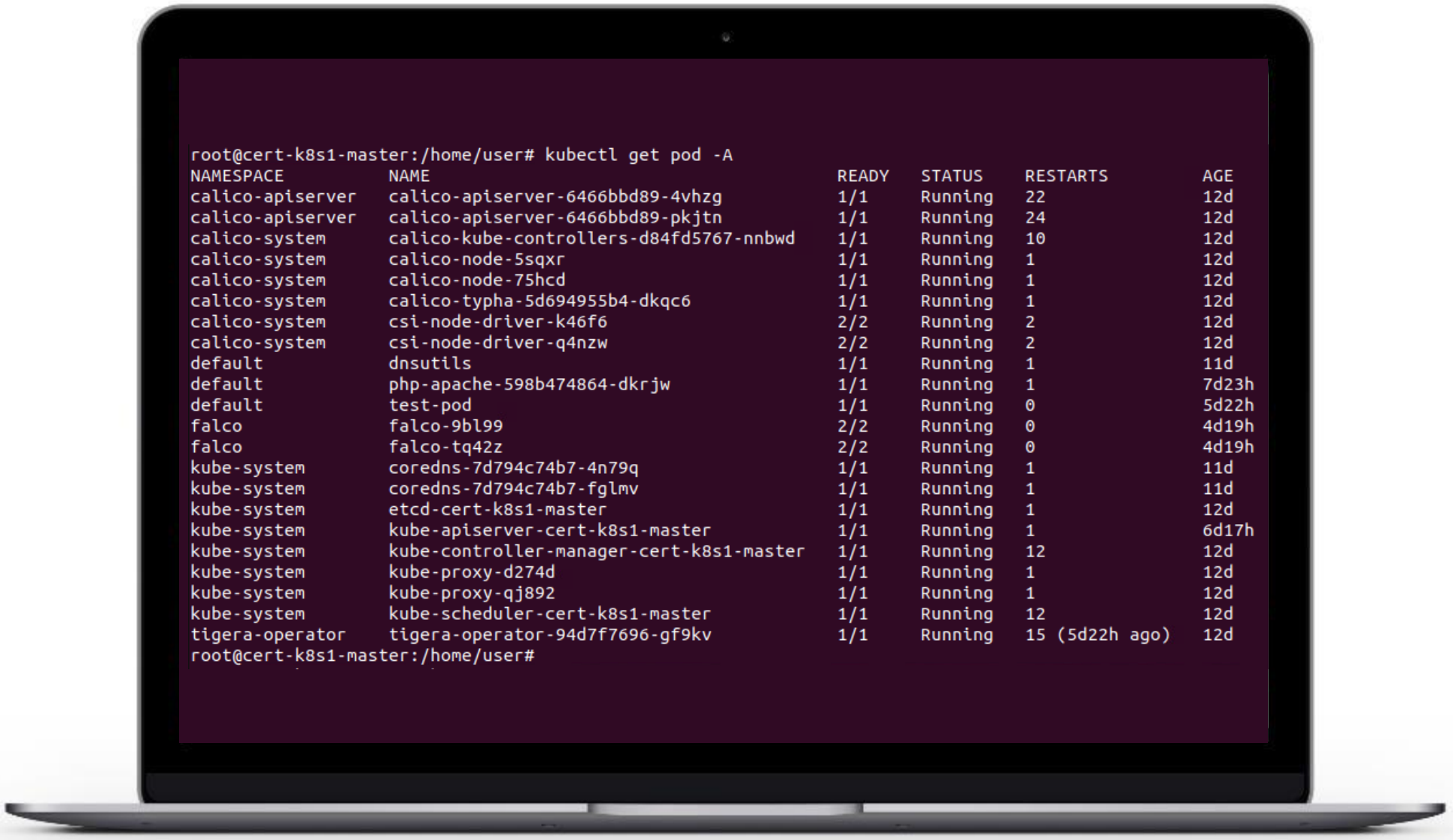
- 4 CPU
- 8 GB RAM
- Ubuntu 22.04
- 6.2.0-34-generic

K8S VERSION

- v1.28.2

FALCO VERSION

- v0.36.0 (последняя на момент выступления)
- default rules set



```
root@cert-k8s1-master:/home/user# kubectl get pod -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS          AGE
calico-apiserver   calico-apiserver-6466bbd89-4vhzg           1/1     Running   22                12d
calico-apiserver   calico-apiserver-6466bbd89-pkjtn           1/1     Running   24                12d
calico-system      calico-kube-controllers-d84fd5767-nnbwd    1/1     Running   10                12d
calico-system      calico-node-5sqxr                          1/1     Running   1                 12d
calico-system      calico-node-75hcd                          1/1     Running   1                 12d
calico-system      calico-typha-5d694955b4-dkqc6              1/1     Running   1                 12d
calico-system      csi-node-driver-k46f6                      2/2     Running   2                 12d
calico-system      csi-node-driver-q4nzw                      2/2     Running   2                 12d
default            dnsutils                                   1/1     Running   1                 11d
default            php-apache-598b474864-dkrjw                1/1     Running   1                 7d23h
default            test-pod                                   1/1     Running   0                 5d22h
falco              falco-9bl99                                2/2     Running   0                 4d19h
falco              falco-tq42z                                2/2     Running   0                 4d19h
kube-system        coredns-7d794c74b7-4n79q                   1/1     Running   1                 11d
kube-system        coredns-7d794c74b7-fglmv                   1/1     Running   1                 11d
kube-system        etcd-cert-k8s1-master                      1/1     Running   1                 12d
kube-system        kube-apiserver-cert-k8s1-master            1/1     Running   1                 6d17h
kube-system        kube-controller-manager-cert-k8s1-master   1/1     Running   12                12d
kube-system        kube-proxy-d274d                           1/1     Running   1                 12d
kube-system        kube-proxy-qj892                           1/1     Running   1                 12d
kube-system        kube-scheduler-cert-k8s1-master            1/1     Running   12                12d
tigera-operator    tigera-operator-94d7f7696-gf9kv            1/1     Running   15 (5d22h ago)    12d
root@cert-k8s1-master:/home/user#
```

Немного о правилах в целом

SOC FORUM 2023

В более ранних версиях Falco для syscall-правил существовал только один файл, который включал в себя все имеющиеся правила. В какой-то момент времени вендор разделил правила на 4 пака:

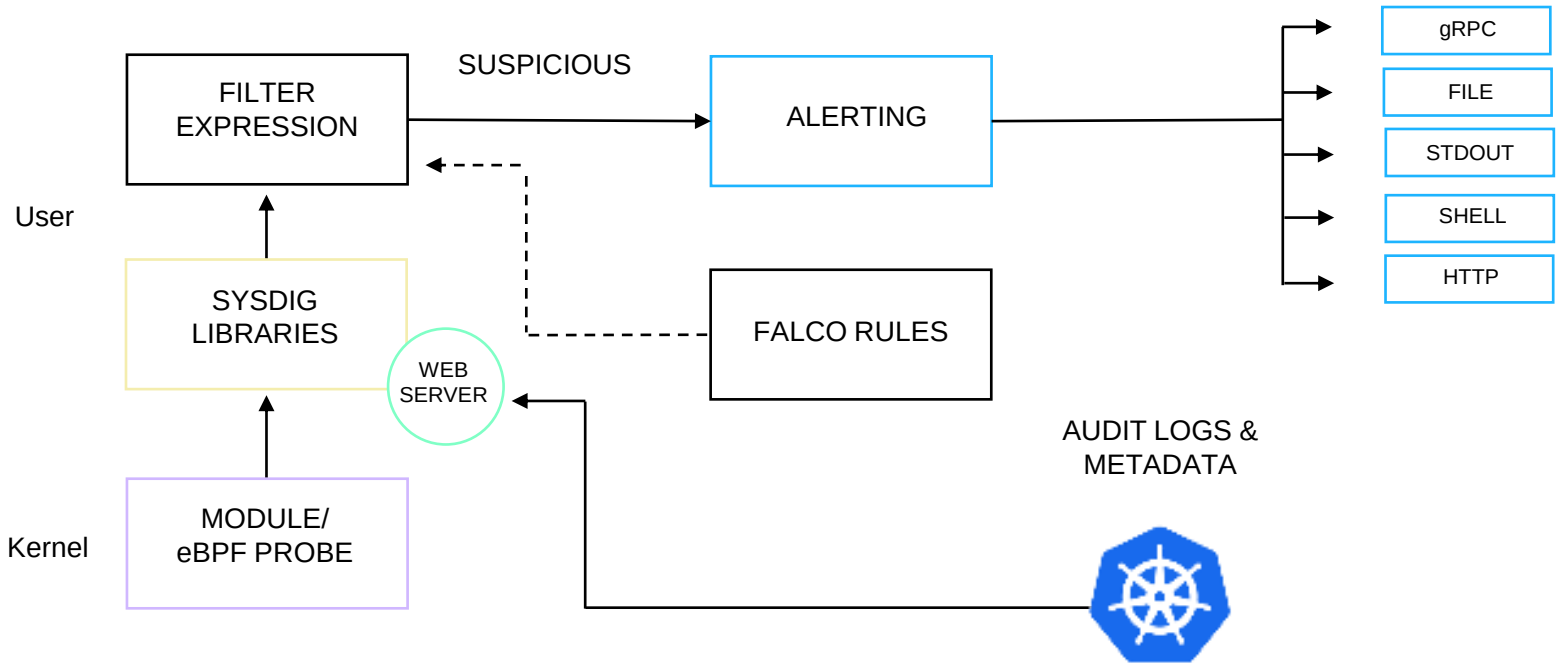
[01]
falco_rules.yaml

[02]
falco-incubating_rules.yaml

[03]
falco-sandbox_rules.yaml

[04]
falco-deprecated_rules.yaml

FALCO EXTENDED ARCHITECTURE

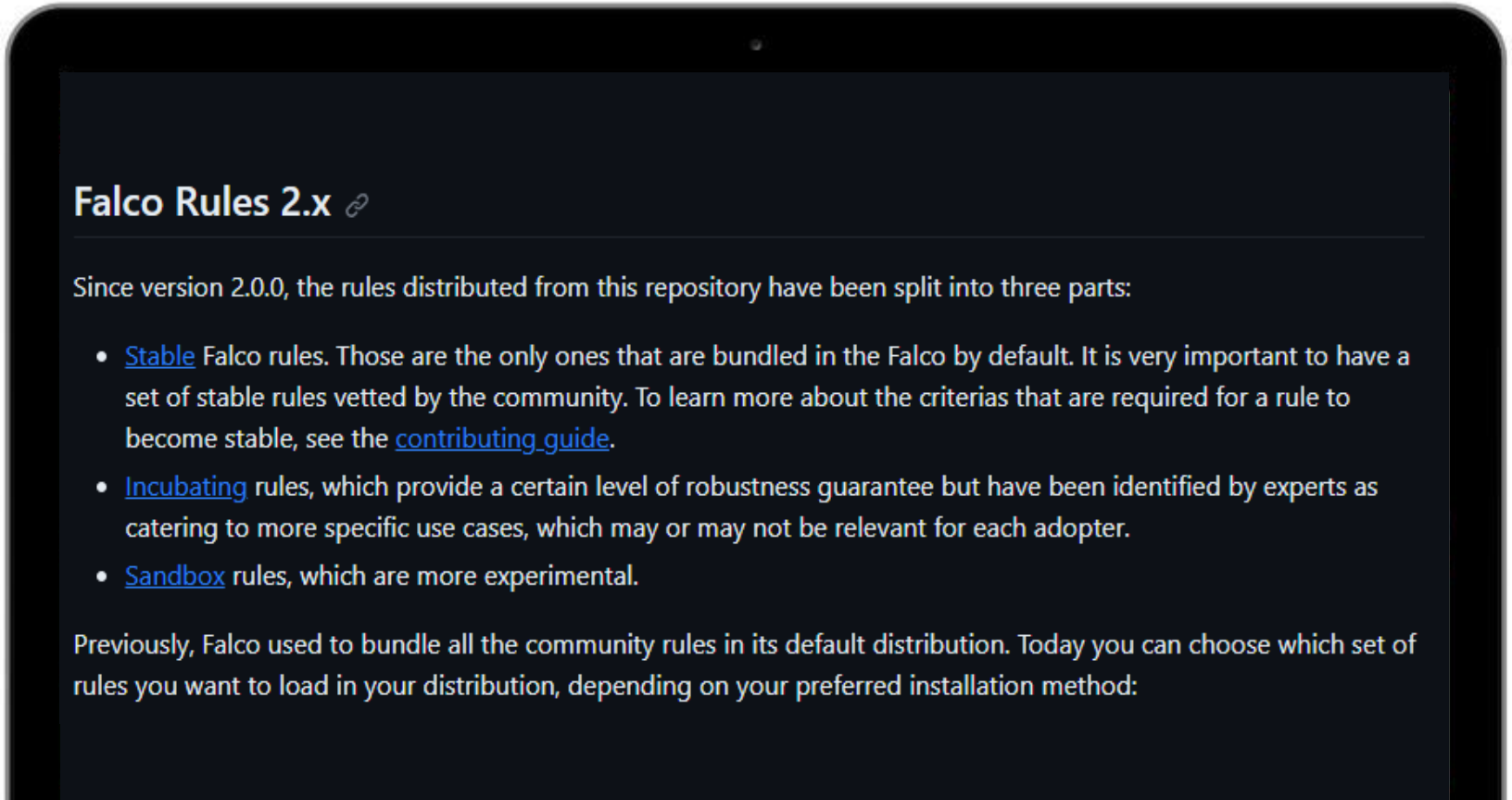


Немного о правилах в целом

SOC FORUM 2023

По умолчанию при установке через helm ставится только первый (если не указано иное в явном виде), который включает в себя (на данный момент) только 25 правил.

Дополнительный важный момент – правила обрабатываются последовательно
(в последних релизах добавлен флаг rule_matching, с которым каждый syscall проходит через все правила).



Обходы сигнатур

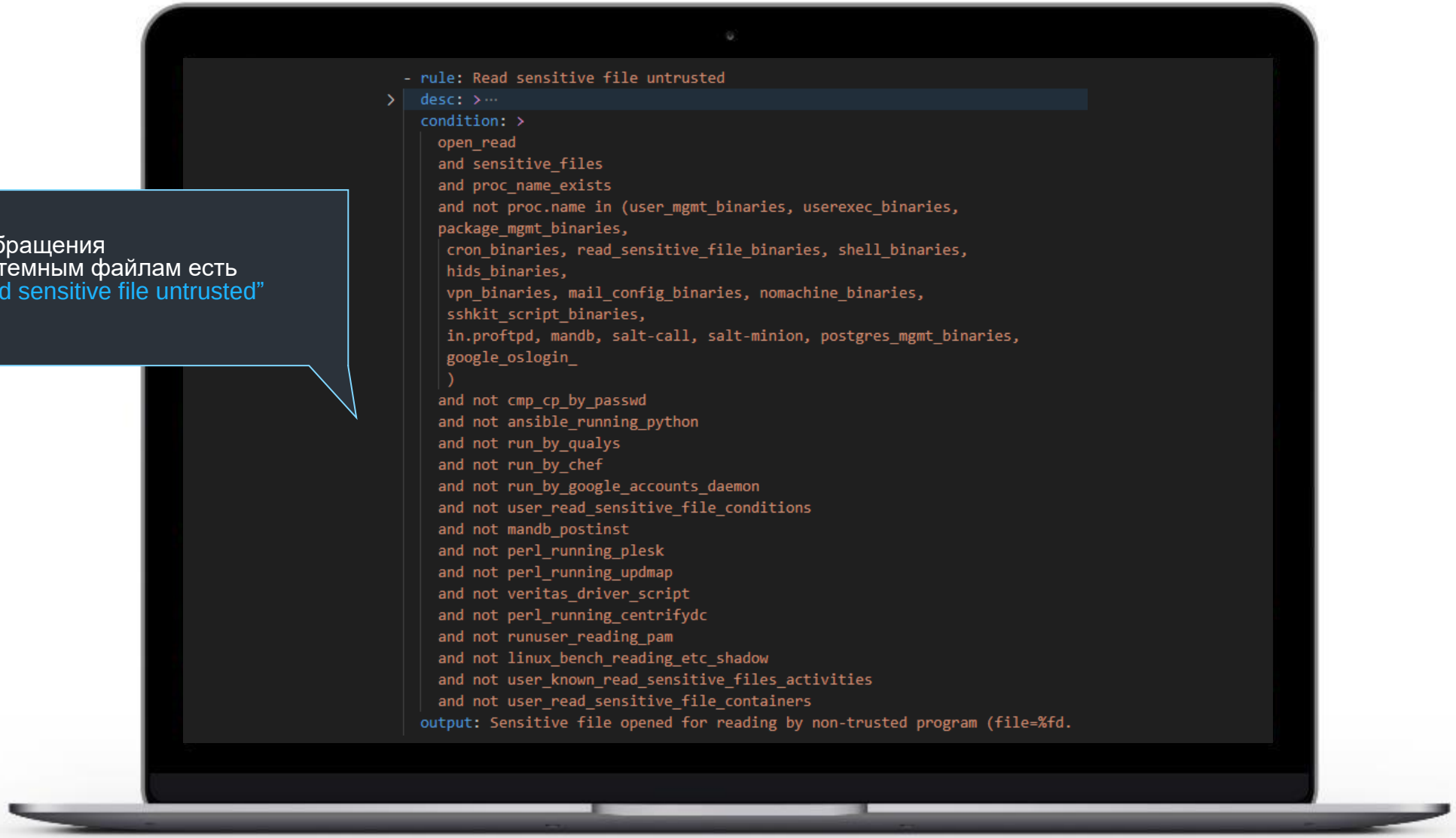
SOC FORUM 2023

1	2	3	4	5
Чтение важных системных файлов	Запуск новых исполняемых файлов в контейнере	Обнаружение запуска по аргументам	Очистка log-файлов	Sensitive container mounts

Чтение важных системных файлов [Rule]

SOC FORUM 2023

Для аудита обращения
к важным системным файлам есть
правило “Read sensitive file untrusted”



```
- rule: Read sensitive file untrusted
  desc: >...
  condition: >
    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
  output: Sensitive file opened for reading by non-trusted program (file=%fd.
```

Чтение важных системных файлов [Alert]

SOC FORUM 2023

Читаем файл в Pod (для теста – /etc/shadow), получаем сработку:

```
root@test-pod:/# cat /etc/shadow
root:*:19639:0:99999:7:::
daemon:*:19639:0:99999:7:::
bin:*:19639:0:99999:7:::
sys:*:19639:0:99999:7:::
sync:*:19639:0:99999:7:::
games:*:19639:0:99999:7:::
man:*:19639:0:99999:7:::
lp:*:19639:0:99999:7:::
mail:*:19639:0:99999:7:::
news:*:19639:0:99999:7:::
uucp:*:19639:0:99999:7:::
proxy:*:19639:0:99999:7:::
www-data:*:19639:0:99999:7:::
backup:*:19639:0:99999:7:::
list:*:19639:0:99999:7:::
irc:*:19639:0:99999:7:::
_apt:*:19639:0:99999:7:::
nobody:*:19639:0:99999:7:::
nginx:!:19642::::::
```


```
"priority": "Warning",
"rule": "Read sensitive file untrusted",
"source": "syscall",
"tags": [
  "T1555",
  "container",
  "filesystem",
  "host",
  "maturity_stable",
  "mitre_credential_access"
],
"time": "2023-10-18T09:23:23.260396305Z",
"output_fields": {
  "container.id": "b32ecb9505ac",
  "container.image.repository": "docker.io/library/nginx",
  "container.image.tag": "latest",
  "container.name": "test-pod",
  "evt.arg.flags": "O_RDONLY",
  "evt.time": 1697621003260396300,
  "evt.type": "openat",
  "fd.name": "/etc/shadow",
  "k8s.ns.name": "default",
  "k8s.pod.name": "test-pod",
  "proc.aname[2]": "runc",
  "proc.aname[3]": "crio",
  "proc.aname[4]": "systemd",
  "proc.cmdline": "cat /etc/shadow",
  "proc.exepath": "/usr/bin/cat",
  "proc.name": "cat",
  "proc.pname": "bash",
  "proc.tty": 34816,
  "user.loginuid": -1,
  "user.name": "root",
  "user.uid": 0
}
```

Чтение важных системных файлов [Evasion]

SOC FORUM 2023

Читаем внимательно текст правила и различных исключений и получаем,
что правило смотрит в директорию /etc (fd.name startswith /etc),
а также имеет ряд исключений.

Первый вариант обхода заключается в том, чтобы не читать файл напрямую, а использовать ссылки (смотрим внимательно на правило “Create Symlink Over Sensitive Files”,
чтобы не сгенерить сработку).

Во втором просто используем паттерн, который находится в исключениях: proc.cmdline startswith "perl /opt/VRTSsfmh/bin/mh_driver.pl"

```
root@test-pod:/# mkdir -p /opt/VRTSsfmh/bin/
root@test-pod:/# mv mh_driver.pl /opt/VRTSsfmh/bin/
root@test-pod:/# perl /opt/VRTSsfmh/bin/mh_driver.pl
root:*:19639:0:99999:7:::
daemon:*:19639:0:99999:7:::
bin:*:19639:0:99999:7:::
sys:*:19639:0:99999:7:::
sync:*:19639:0:99999:7:::
games:*:19639:0:99999:7:::
man:*:19639:0:99999:7:::
lp:*:19639:0:99999:7:::
mail:*:19639:0:99999:7:::
news:*:19639:0:99999:7:::
uucp:*:19639:0:99999:7:::
proxy:*:19639:0:99999:7:::
www-data:*:19639:0:99999:7:::
backup:*:19639:0:99999:7:::
list:*:19639:0:99999:7:::
irc:*:19639:0:99999:7:::
_apt:*:19639:0:99999:7:::
nobody:*:19639:0:99999:7:::
nginx:!:19642::::::
```

```
root@test-pod:/# ln -s /etc/fonts /lnk
root@test-pod:/#
root@test-pod:/#
root@test-pod:/# cat lnk/../shadow
root:*:19639:0:99999:7:::
daemon:*:19639:0:99999:7:::
bin:*:19639:0:99999:7:::
sys:*:19639:0:99999:7:::
sync:*:19639:0:99999:7:::
games:*:19639:0:99999:7:::
man:*:19639:0:99999:7:::
lp:*:19639:0:99999:7:::
mail:*:19639:0:99999:7:::
news:*:19639:0:99999:7:::
uucp:*:19639:0:99999:7:::
proxy:*:19639:0:99999:7:::
www-data:*:19639:0:99999:7:::
backup:*:19639:0:99999:7:::
list:*:19639:0:99999:7:::
irc:*:19639:0:99999:7:::
_apt:*:19639:0:99999:7:::
nobody:*:19639:0:99999:7:::
nginx:!:19642::::::
```

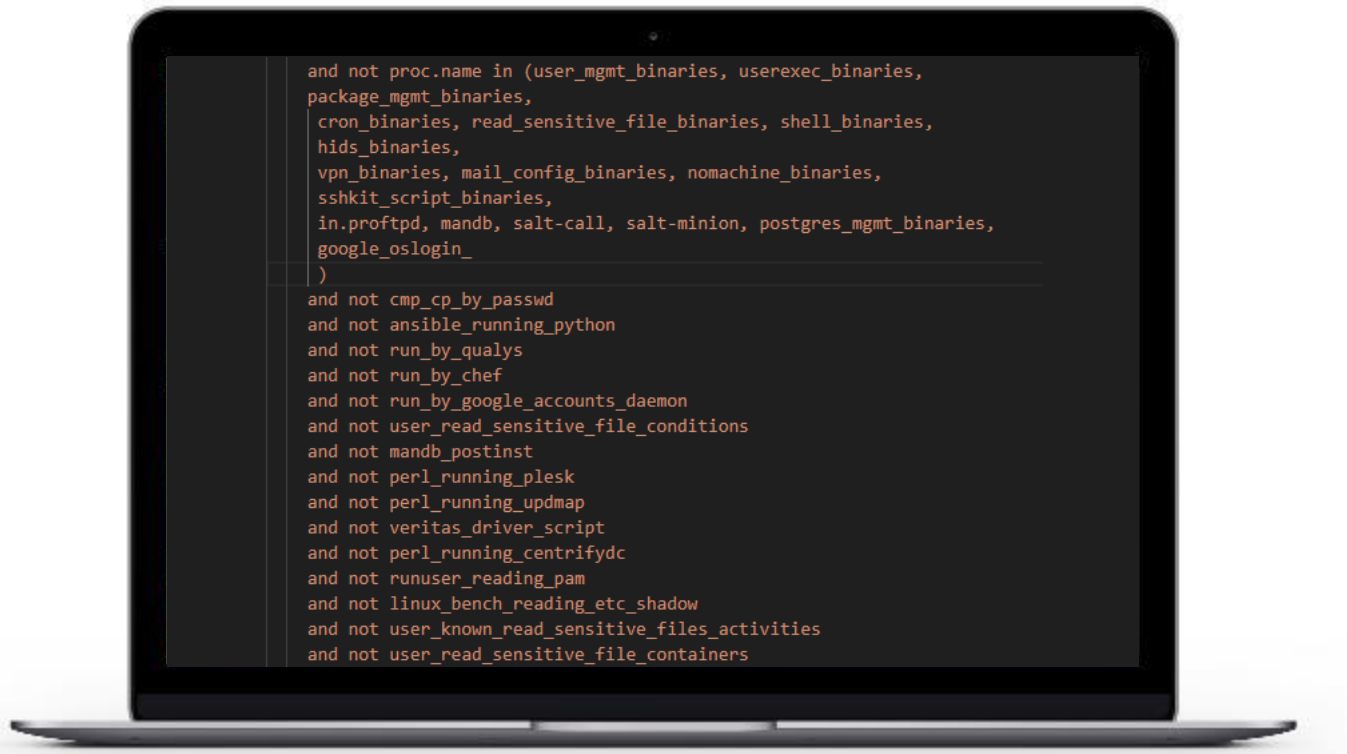
Чтение важных системных файлов [Mitigation]

SOC FORUM 2023

Потенциальный фикс первой проблемы – изменение правила “Create Symlink Over Sensitive Files” таким образом, чтобы в условии учитывались не конкретные файлы и папки, а их вхождение: лист “sensitive_directory_names” изменить на макрос, в котором учитывается вхождение тех же директорий (или не всех, а наиболее критичных).

```
- list: sensitive_directory_names
  items: [/, /etc, /etc/, /root, /root/]
```

> Адекватного фикса для второй проблемы нет, потому что в сигнатурном подходе всегда будут присутствовать исключения. Вопрос заключается только в том, насколько хорошо потенциальный злоумышленник знает коробочные правила.



Запуск новых исполняемых файлов в контейнере [Rule]

SOC FORUM 2023

В последнем релизе в Falco было добавлено правило, которое должно закрыть вопрос об обнаружении Initial Access: “Drop and execute new binary in container”.

Суть правила заключается в том, что детектируются запуски исполняемых файлов, которые первоначально не были в образе контейнера:

```
- rule: Drop and execute new binary in container
  desc: >···
  condition: >
    spawned_process
    and container
    and proc.is_exe_upper_layer=true
    and not container.image.repository in (known_drop_and_execute_containers)
  output: Executing binary not part of base image (proc_exe=%proc.exe
  proc_sname=%proc.sname gparent=%proc.aname[2] proc_exe_ino_ctime=%proc.
  exe_ino.ctime proc_exe_ino_mtime=%proc.exe_ino.mtime
  proc_exe_ino_ctime_duration_proc_start=%proc.exe_ino.
  ctime_duration_proc_start proc_cwd=%proc.cwd container_start_ts=%container.
  start_ts evt_type=%evt.type user=%user.name user_uid=%user.uid
  user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.exepath
  parent=%proc.pname command=%proc.cmdline terminal=%proc.tty exe_flags=%evt.
  arg.flags %container.info)
  priority: CRITICAL
  tags: [maturity_stable, container, process, mitre_persistence, TA0003,
  PCI_DSS_11.5.1]
```

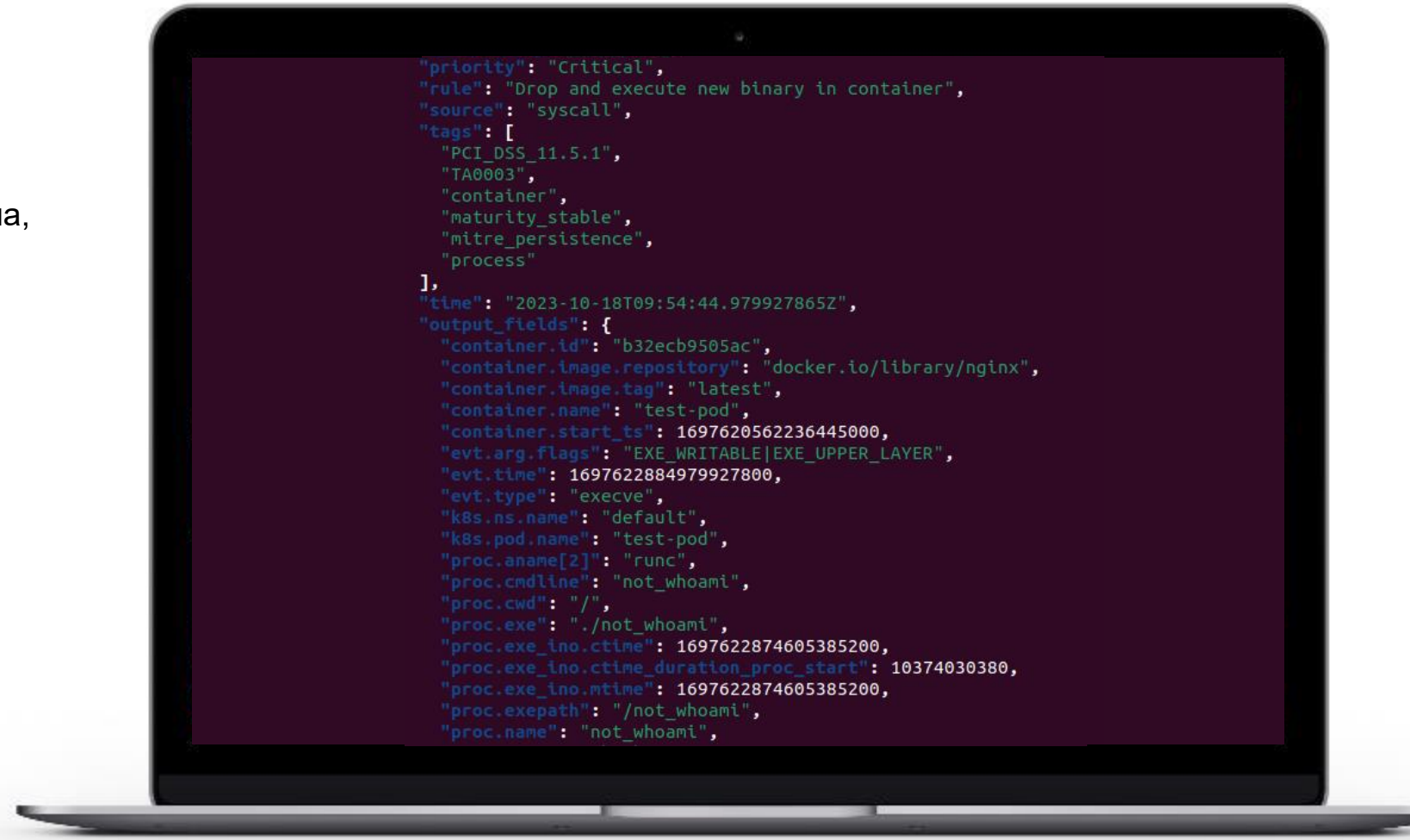
```
static __always_inline bool extract__exe_upper_layer(struct inode *inode, struct file *exe_file)
{
        unsigned long sb_magic = BPF_CORE_READ(inode, i_sb, s_magic);

        if(sb_magic == PPM_OVERLAYFS_SUPER_MAGIC)
        {
                struct dentry *upper_dentry = NULL;
                char *vfs_inode = (char *)inode;
                unsigned long inode_size = bpf_core_type_size(struct inode);
                if(!inode_size)
                {
                        return false;
                }

                bpf_probe_read_kernel(&upper_dentry, sizeof(upper_dentry), vfs_inode + inode_size);

                if(!upper_dentry)
                {
                        return false;
                }

                struct dentry *dentry = (struct dentry *)BPF_CORE_READ(exe_file, f_path.dentry);

                unsigned int d_flags = BPF_CORE_READ(dentry, d_flags);
                bool disconnected = (d_flags & DCACHE_DISCONNECTED);
                if(disconnected)
                {
                        return true;
                }
```

Запуск новых исполняемых файлов в контейнере [Alert]

SOC FORUM 2023

В нормальных условиях правило срабатывает на непосредственный запуск исполняемого файла, которого не было в первоначальном образе:

```
root@test-pod:/#
root@test-pod:/# cp /bin/whoami /not_whoami
root@test-pod:/#
root@test-pod:/# ./not_whoami
root
root@test-pod:/#
```



Запуск новых исполняемых файлов в контейнере [Evasion]

SOC FORUM 2023

Поскольку для работы правила требуется вызов execve/execveat (макрос “spawned_process”), нужно обойтись без него. Предлагаем два способа:

1 GTFObin ld.so

2 ddexec.sh

Проблема использования первого метода заключается в том, что при большом размере исполняемого файла все равно произойдет вызов execve/execveat, вторым методом можно выполнять файлы любого размера (хоть и не сильно быстро)

```
root@test-pod:/#
root@test-pod:/# /lib64/ld-linux-x86-64.so.2 /not_whoami
root
root@test-pod:/#
```

```
root@test-pod:/# base64 -w0 /not_whoami | bash ddexec.sh not_whoami
root
root@test-pod:/#
```

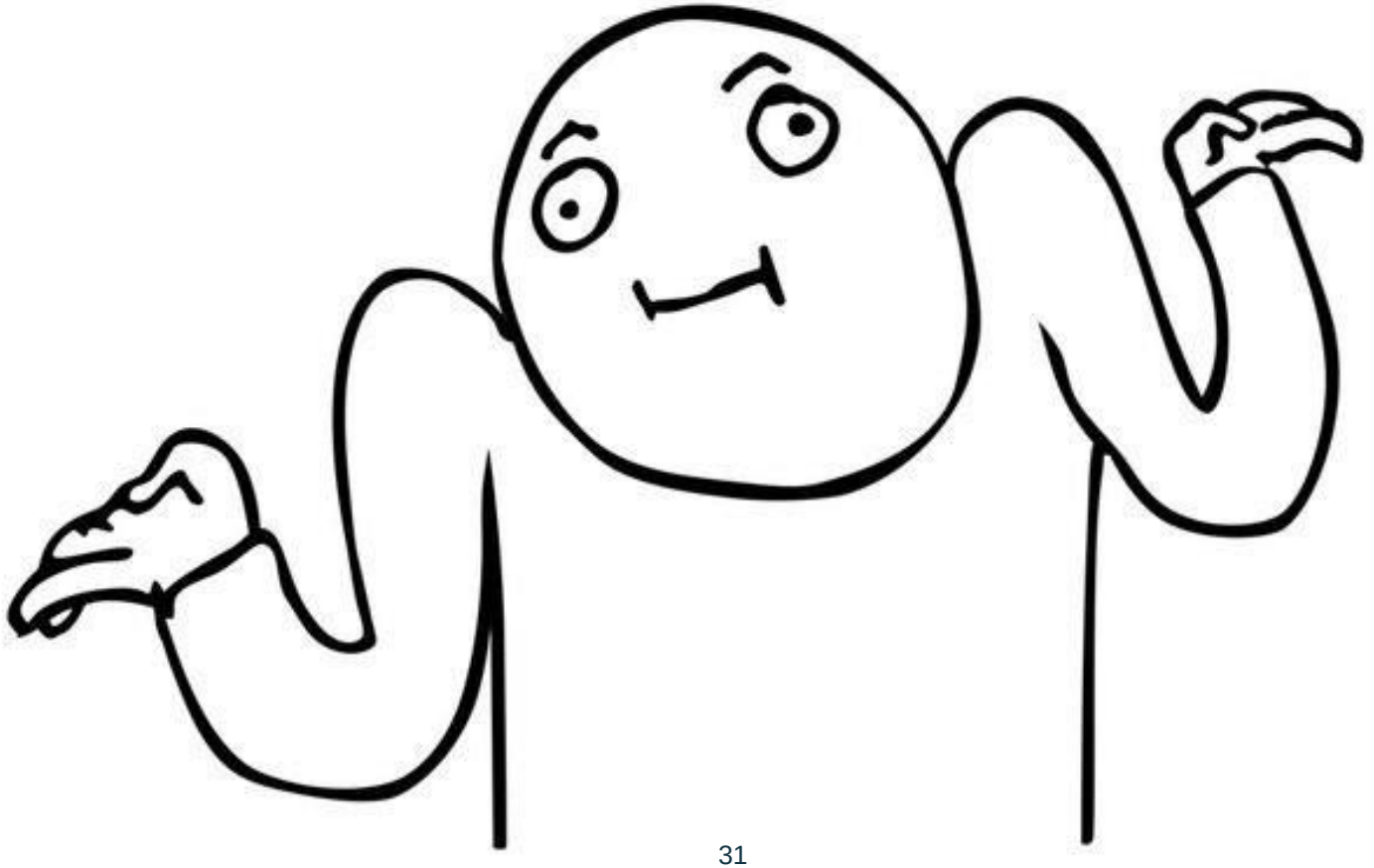
Input = [con10er]

Запуск новых исполняемых файлов в контейнере [Mitigation]

SOC FORUM 2023

Потенциальный фикс первого обхода: написание правила на конкретный GTFObin (будет отображаться как запуск процесса вида /lib64/ld-linux-x86-64.so.2).

Фикс второго обхода сигнатурными методами выглядит примерно так:



Обнаружение запуска по аргументам [Rule]

SOC FORUM 2023

Одно из самых проблемных и тонких мест сигнатурного анализа – обнаружение запуска чего-либо по аргументам (Falco – не исключение). Причин несколько:

1. Аргументов и их комбинаций очень много (особенно в Linux), нужно учитывать комбинации в регулярках/условиях.
2. Исполняемые файлы легко переименовываются (копируются).

Рассмотрим два правила схожей направленности: “Find AWS Credentials” и “Search Private Keys or Passwords”.

```
- rule: Find AWS Credentials
  desc: >···
  condition: >
    spawned_process
    and ((grep_commands and private_aws_credentials) or
         (proc.name = "find" and proc.args endswith ".aws/credentials"))
  output: Detected AWS credentials search activity (proc_pcmdline=%proc.
  pcmdline proc_cwd=%proc.cwd group_gid=%group.gid group_name=%group.name
  user_loginname=%user.loginname evt_type=%evt.type user=%user.name
  user_uid=%user.uid user_loginuid=%user.loginuid process=%proc.name
  proc_exepath=%proc.exepath parent=%proc.pname command=%proc.cmdline
  terminal=%proc.tty exe_flags=%evt.arg.flags %container.info)
  priority: WARNING
  tags: [maturity_stable, host, container, process, aws,
  mitre_credential_access, T1552]
```

```
- rule: Search Private Keys or Passwords
  desc: >···
  condition: >
    spawned_process
    and ((grep_commands and private_key_or_password) or
         (proc.name = "find" and (proc.args contains "id_rsa" or
                                  proc.args contains "id_dsa" or
                                  proc.args contains "id_ed25519" or
                                  proc.args contains "id_ecdsa"
         )
        ))
  output: Grep private keys or passwords activities found (evt_type=%evt.type
  user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid
  process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname
  command=%proc.cmdline terminal=%proc.tty exe_flags=%evt.arg.flags
  %container.info)
  priority:
    WARNING
  tags: [maturity_stable, host, container, process, filesystem,
  mitre_credential_access, T1552.001]
```

Обнаружение запуска по аргументам [Rule]

Ранее Сергей Канибор (Luntry) обнаружил обход запуска curl в контейнере (“Launch Ingress Remote File Copy Tools in Container”).

Интересная особенность паков правил от Falco – некоторые не самые удачные правила были перемещены из основного пака в дополнительный (требующий более точной настройки).
Названное выше правило – не исключение.

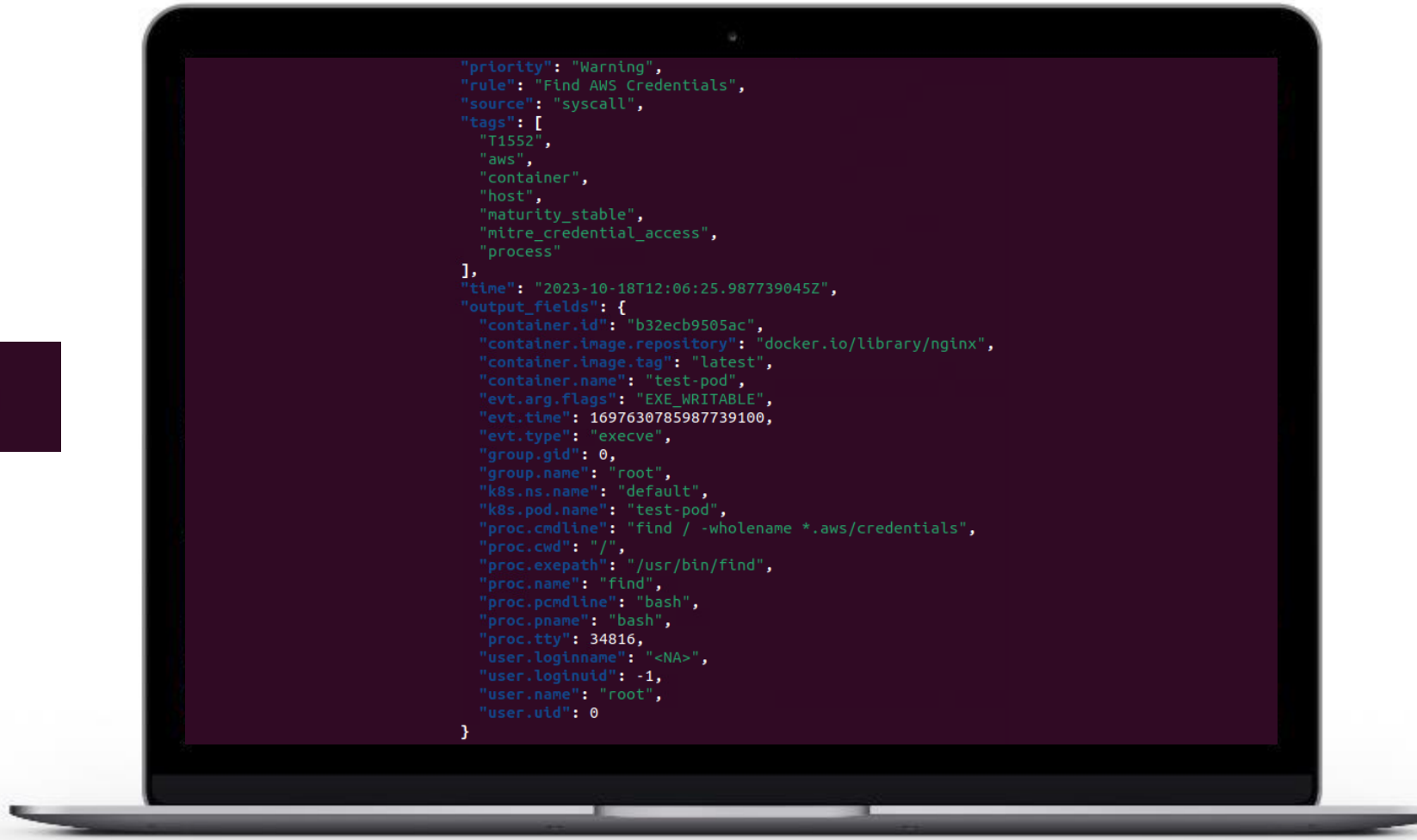
```
- macro: curl_download
  condition: (proc.name = curl and
                (proc.cmdline contains " -o " or
                proc.cmdline contains " --output " or
                proc.cmdline contains " -O " or
                proc.cmdline contains " --remote-name "))

- rule: Launch Ingress Remote File Copy Tools in Container
  desc: > ···
  condition: >
    spawned_process
    and container
    and (ingress_remote_file_copy_procs or curl_download)
    and not user_known_ingress_remote_file_copy_activities
  output: Ingress remote file copy tool launched in container
  (evt_type=%evt.type user=%user.name user_uid=%user.uid
  user_loginuid=%user.loginuid process=%proc.name proc_exepath=%proc.
  exepath parent=%proc.pname command=%proc.cmdline terminal=%proc.tty
  exe_flags=%evt.arg.flags %container.info)
  priority: NOTICE
  tags: [maturity_incubating, container, network, process,
  mitre_command_and_control, TA0011]
```

Обнаружение запуска по аргументам [Alert]

SOC FORUM 2023

Проблема указанных правил в том,
что они слишком точечные.
Посмотрим на генерируемые сработки:

```
root@test-pod:/#
root@test-pod:/# find / -wholename "*.aws/credentials"  2>/dev/null
/tmp/.aws/credentials
root@test-pod:/#
root@test-pod:/#
```



34

Обнаружение запуска по аргументам [Evasion]

SOC FORUM 2023

Конкретно эти два правила (как и многие другие с поиском аргументов) обходятся очень просто:

1 Используем * в именах файлов:

```
root@test-pod:/#
root@test-pod:/# find / -wholename "*.aws/cred*ntials"  2>/dev/null
/tmp/.aws/credentials
root@test-pod:/#
```

2 В правилах на grep также используем регулярки:

```
root@test-pod:/#
root@test-pod:/# grep -En "BEGIN PR.VATE" /* 2>/dev/null
/test_ssh_key:1:BEGIN PRIVATE
root@test-pod:/#
```

Забавная особенность
Atomic-тест также обходит правило:

Attack Commands: Run with `sh`!

```
find #{file_path} -name "credentials" -type f -path "*/.aws/*" 2>/dev/null
```

Обнаружение запуска по аргументам [Mitigation]

SOC FORUM 2023

Самый адекватный сигнатурный вариант фикса выглядит следующим образом:

- Тюним регулярки
- Профилируем активность, которая случайно под них попадает (false-positive)

Именно на этих правилах можно увидеть наиболее яркие проблемы сигнатурного подхода

Очистка log-файлов [Rule]

Рассмотрим правило “clear log activities”, направленное на обнаружение перезаписи логов

В условиях перечислены директории с log-файлами:

- /var/log
- /dev/log

И конкретные файлы с логами:

- syslog
- auth.log
- secure
- kern.log
- cron
- user.log
- dpkg.log
- last.log
- yum.log
- access_log
- mysql.log
- mysqld.log

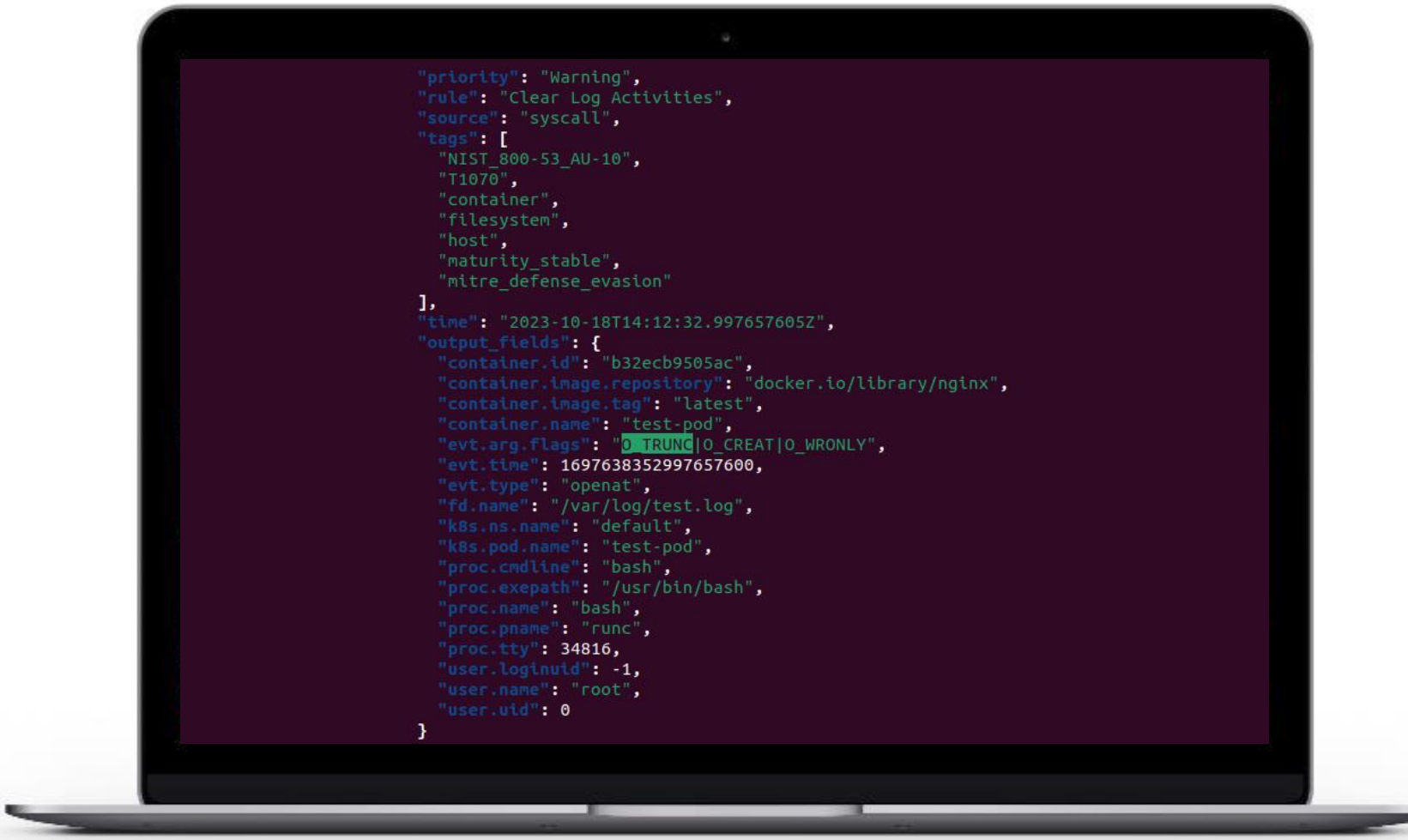
```
- rule: Clear Log Activities
  desc: > ···
  condition: >
    open_write
    and access_log_files
    and evt.arg.flags contains "O_TRUNC"
    and not containerd_activities
    and not trusted_logging_images
    and not allowed_clear_log_files
  output: Log files were tampered (file=%fd.name evt_type=%evt.type
  user=%user.name user_uid=%user.uid user_loginuid=%user.loginuid
  process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname
  command=%proc.cmdline terminal=%proc.tty exe_flags=%evt.arg.flags
  %container.info)
  priority:
    WARNING
  tags: [maturity_stable, host, container, filesystem, mitre_defense_evasion,
  T1070, NIST_800-53_AU-10]
```

Очистка log-файлов [Alert]

SOC FORUM 2023

Сработка происходит, если в аргументах системного вызова open/openat/openat2 есть флаг o_trunc:

```
root@test-pod:/#
root@test-pod:/# echo 123 > /var/log/test.log
root@test-pod:/#
```

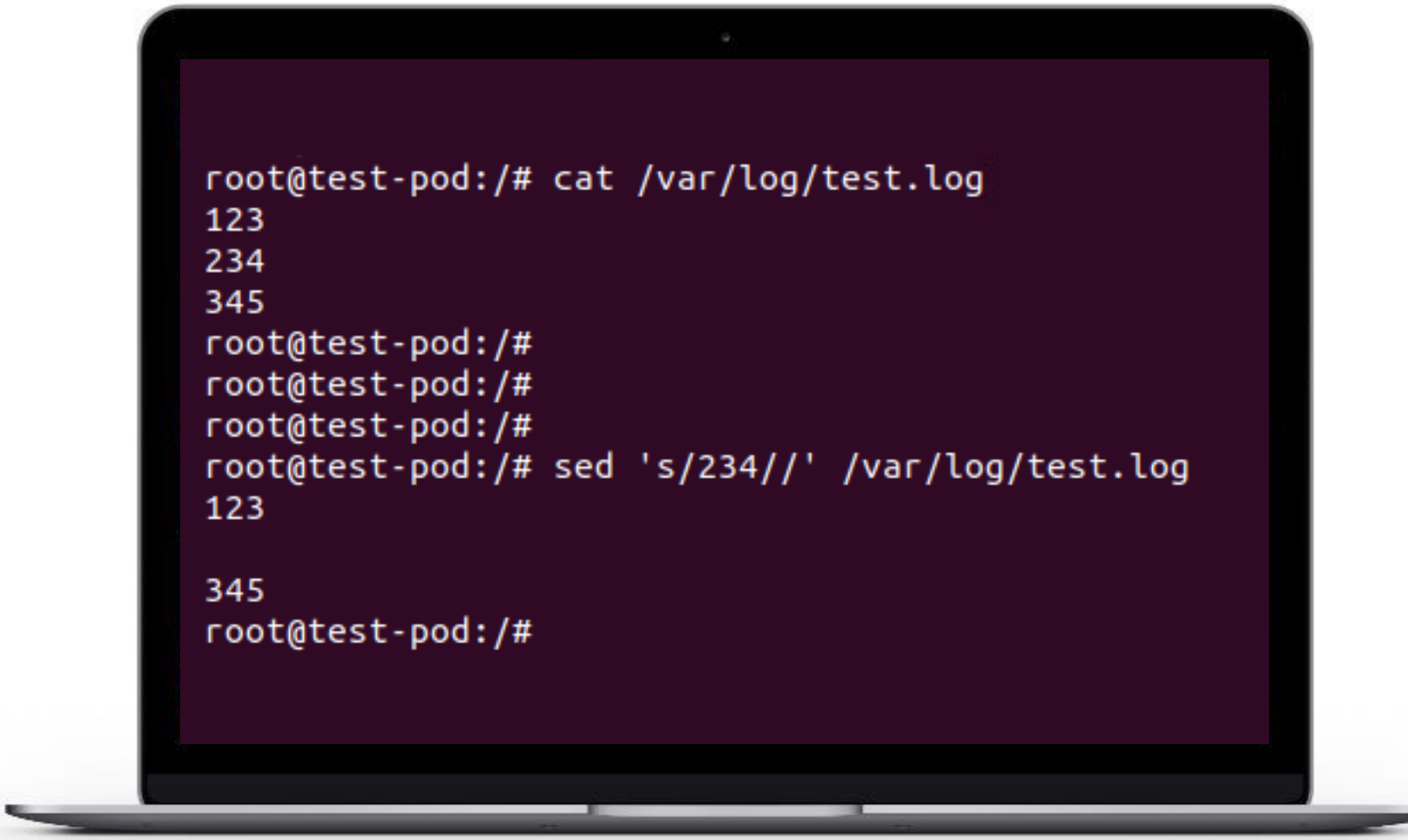


Очистка log-файлов [Evasion]

SOC FORUM 2023

Обход простой:
в стандартном паке правил не мониторится простое удаление файла.
Также можно удалить интересующие записи через использование утилиты sed.
В обоих случаях правило не срабатывает:

```
root@test-pod:/#
root@test-pod:/# rm /var/log/test.log
root@test-pod:/#
```



Очистка log-файлов [Mitigation]

Один из вариантов фикса данного правила:
пробовать ловить различные варианты запуска инструментов с путями до log-файлов в аргументах (да, скорее всего, будет не просто).
Также можно пробовать профилировать доступ к файлам (ближе к поведенческому анализу).

```
2
3  - macro: file_deletion_globs
4    condition: >
5      (evt.arg.path glob "/var/log/*" or
6      evt.arg.name glob "/var/log/*" or
7      evt.arg.path glob "/dev/log/*" or
8      evt.arg.name glob "/dev/log/*)
9
10
11 - rule: Logs deletion
12   desc: Detect log file deletion
13   condition: >
14     remove and
15     evt.dir=< and
16     file_deletion_globs
```

Sensitive container mounts [Rule]

Следующее правило - “launch sensitive mount container”.
Суть правила заключается в том, что проверяются маунты запущенного контейнера по заранее заданному списку:

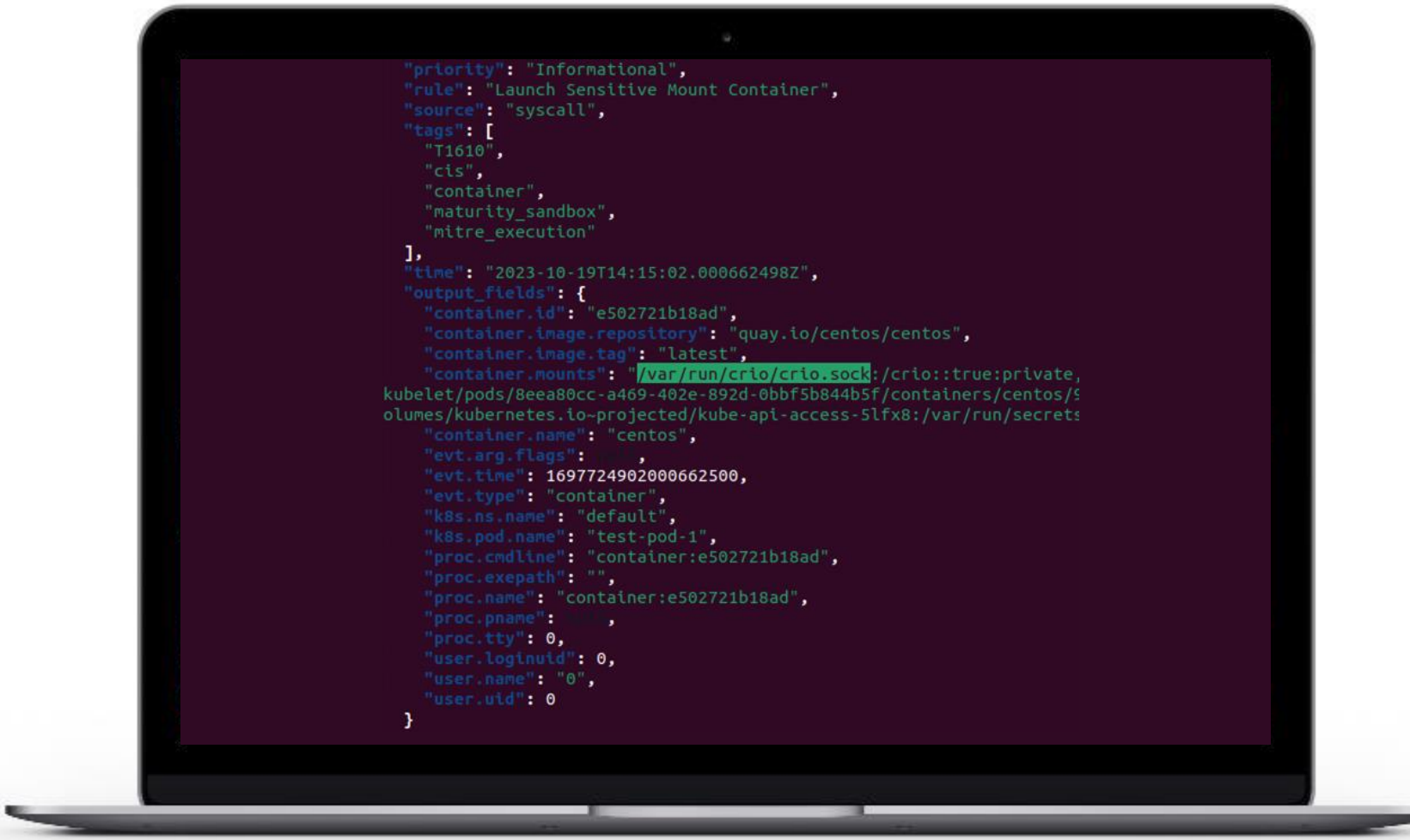
- /proc*
- /var/run/docker.sock
- /var/run/crio/crio.sock
- /run/containerd/containerd.sock
- /var/lib/kubelet
- /var/lib/kubelet/pki
- /
- /home/admin
- /etc
- /etc/kubernetes
- /etc/kubernetes/manifests
- /root*

```
- macro: sensitive_mount
  condition: (container.mount.dest[/proc*] != "N/A" or
              container.mount.dest[/var/run/docker.sock] != "N/A" or
              container.mount.dest[/var/run/crio/crio.sock] != "N/A" or
              container.mount.dest[/run/containerd/containerd.sock] != "N/A"
              or
              container.mount.dest[/var/lib/kubelet] != "N/A" or
              container.mount.dest[/var/lib/kubelet/pki] != "N/A" or
              container.mount.dest[/] != "N/A" or
              container.mount.dest[/home/admin] != "N/A" or
              container.mount.dest[/etc] != "N/A" or
              container.mount.dest[/etc/kubernetes] != "N/A" or
              container.mount.dest[/etc/kubernetes/manifests] != "N/A" or
              container.mount.dest[/root*] != "N/A")

- rule: Launch Sensitive Mount Container
  desc: >···
  condition: >
    container_started
    and container
    and sensitive_mount
    and not falco_sensitive_mount_containers
    and not user_sensitive_mount_containers
  output: Container with sensitive mount started (mounts=%container.mounts
  evt_type=%evt.type user=%user.name user_uid=%user.uid user_loginuid=%user.
  loginuid process=%proc.name proc_exepath=%proc.exepath parent=%proc.pname
  command=%proc.cmdline terminal=%proc.tty exe_flags=%evt.arg.flags
  %container.info)
  priority: INFO
  tags: [maturity_sandbox, container, cis, mitre_execution, T1610]
```

Sensitive container mounts [Alert]

SOC FORUM 2023

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod-1
  namespace: default
spec:
  containers:
  - name: centos
    image: centos
    command: ['sh', '-c', 'sleep 999']
    securityContext:
       privileged: false
    volumeMounts:
      - mountPath: /crio
        name: test-volume
  volumes:
    - name: test-volume
      hostPath:
        path: /var/run/crio/crio.sock
```

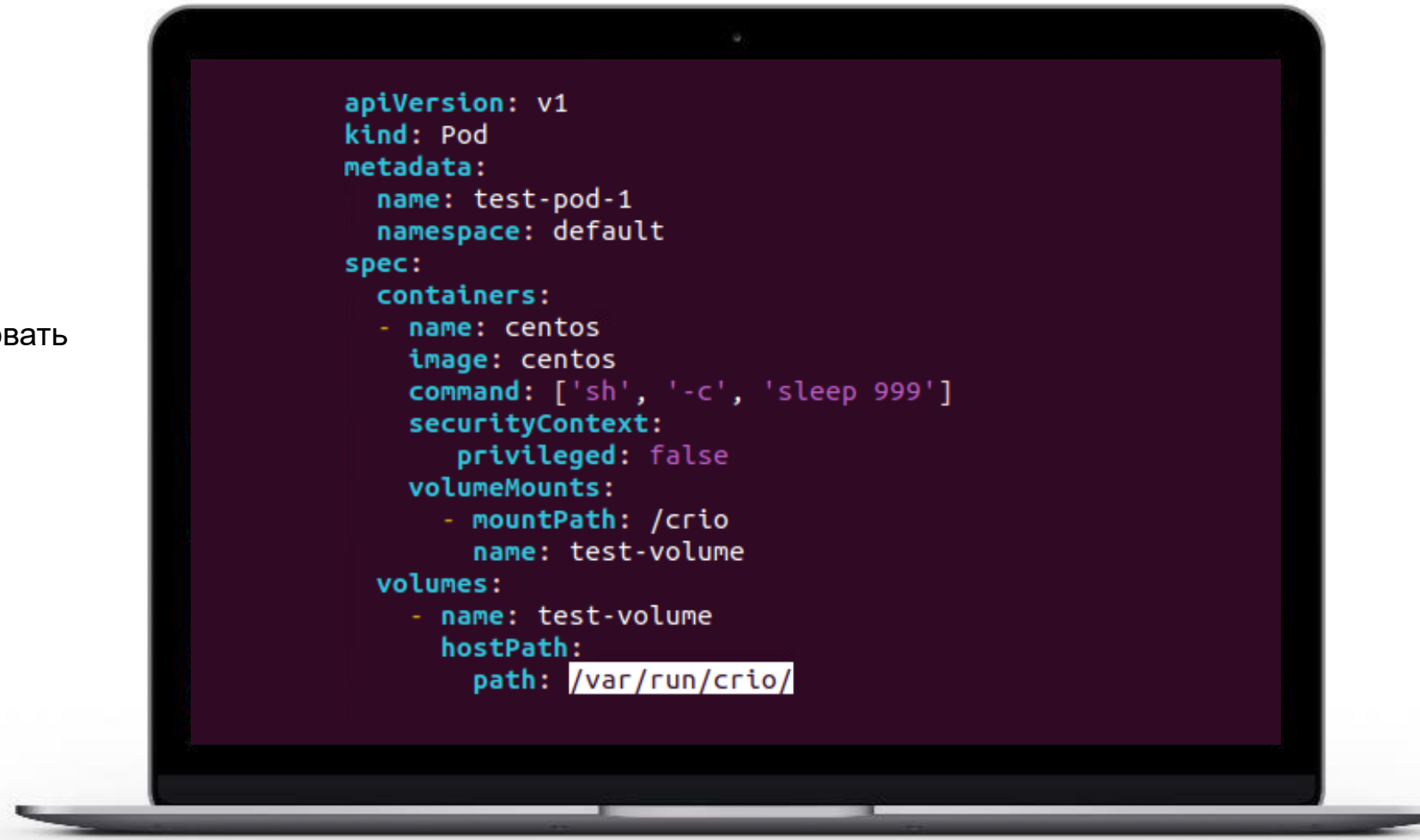


Sensitive container mounts [Evasion]

SOC FORUM 2023

Обойти правило в некоторых случаях довольно просто. Сокеты container runtime можно монтировать не напрямую, а монтировать директорию.

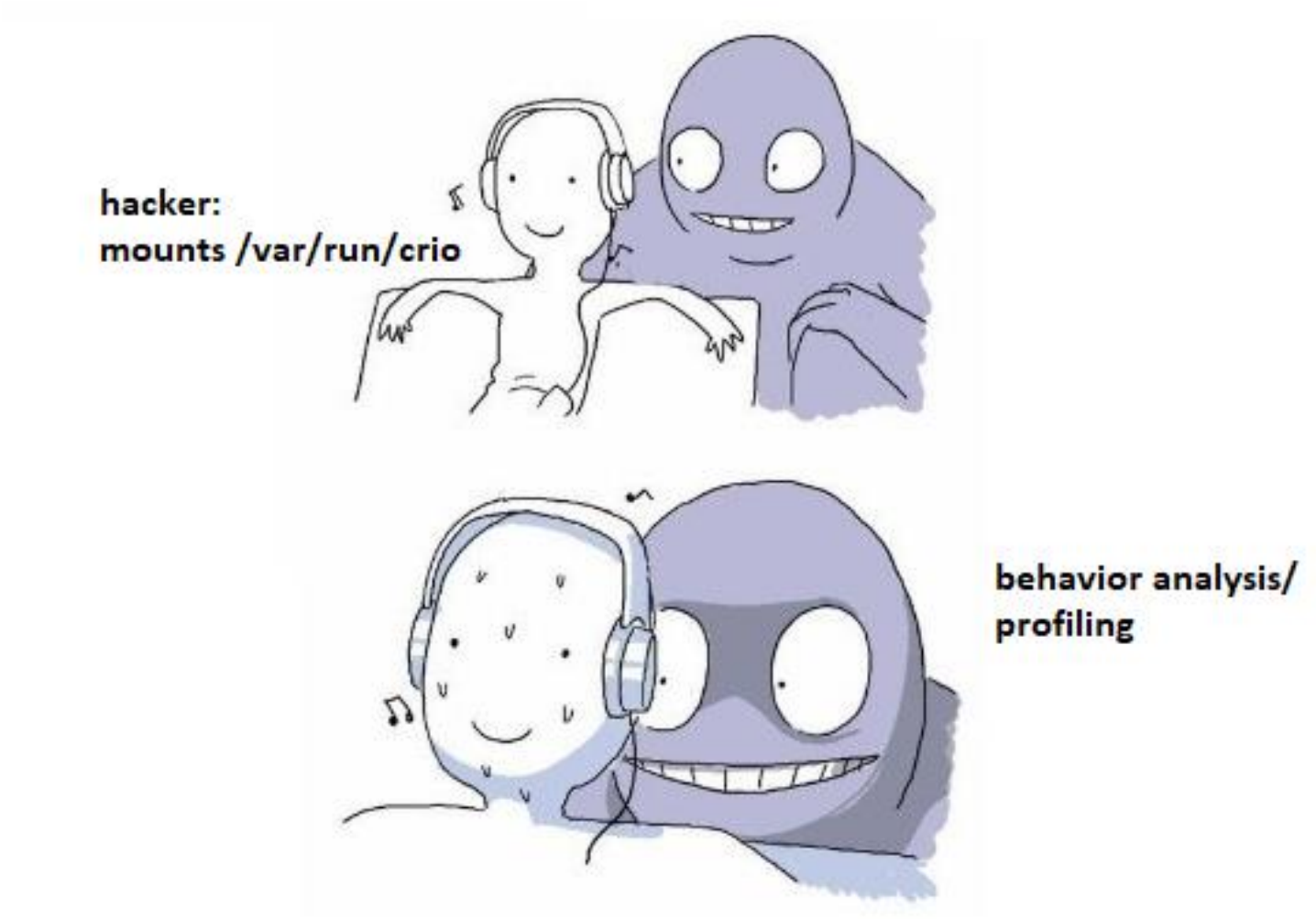
Пример: /var/run/crio/
В таком случае правило не отработает.



Sensitive container mounts [Mitigation]

SOC FORUM 2023

Проблема решается или более широкими условиями в правиле (частично и может вызывать множество ложных срабатываний), или профилированием и поведенческим анализом.



И это далеко не все…

SOC FORUM 2023

Также существует еще достаточное количество методов, которые позволяют обойти средства защиты:

[01]

Уязвимости eBPF, которые периодически появляются в публичном пространстве

[02]

TOCTOU (Time-of-check time-of-use) – некоторые продукты архитектурно подвержены данному типу атак;

[03]

Использование системных вызовов, которые находятся за скоупом мониторинга средства защиты

[04]

eBPF maps tampering

ДРУГИЕ РАБОТЫ В ДАННОЙ ОБЛАСТИ:

1. “Bypassing Falco: How to Compromise a Cluster without Tripping the SOC”
2. "Falco security audit"
3. "TOCTOU issue that could lead to rule bypass"
4. "Default rules can be bypassed with different techniques"
5. "How to hide your actions when every step is being monitored", Ivan Gavrilov, OFFZONE 2023
6. "Kubernetes Pentest All-in-One: The Ultimate Toolkit", Сергей Канибор, OFFZONE 2023
7. "Bypassing eBPF-based Security Enforcement Tools"
8. "Tetragon: A Lesson in Security Fundamentals"
9. "Tetragon: case study of security product's self-protection"
10. "A Wind of Change for Threat Detection“, Melissa Kilby, Apple, KubeCon + CloudNativeCon North America 2023

О других инструментах

SOC FORUM 2023

“Как скрыть свои действия, когда отслеживается каждый шаг”, Иван Гаврилов, OFFZONE 2023

Detection bypass summary

OFFZONE 2023

	Tetragon	Kubearmor
process	✕ - pass invalid utf-8 characters as arg - run a binary with large path - other cases depend on policy	✕ - just run a process ☹
file	✕ - run a process with a lot of args - other cases depend on policy	✕ - just open a file ☹
network	? depends on policy	✕ - TCP server in case filtering - UDP without sys_connect - ICMP
privileges	?	?

66

Detection bypass summary

OFFZONE 2023

1.

Security monitoring tool's flaws

- monitores incorrect kernel functions
- security sensitive functions are not monitored
- makes a lot of noise

2.

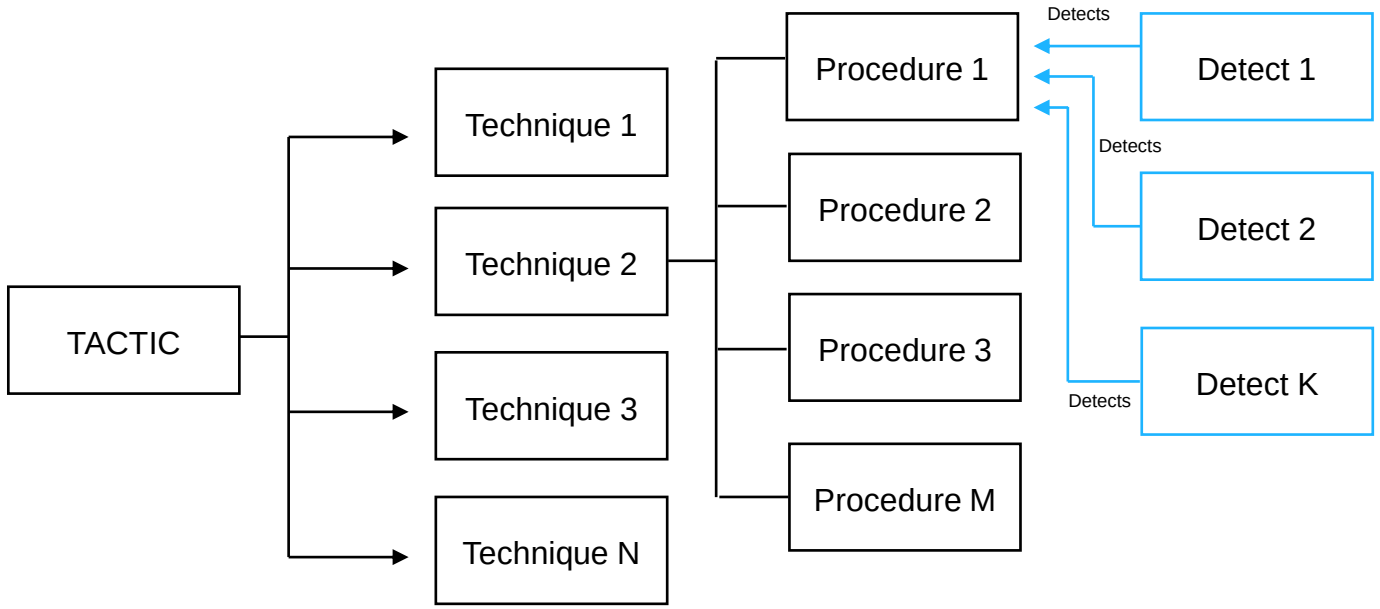
eBPF imitations

- loses events
- cannot provide full information about event in case large data
- allows to implement only simple logic in kernel part

3.

We

Не надо так…


TACTIC
Technique 1
Technique 2
Technique 3
Technique N
Procedure 1
Procedure 2
Procedure 3
Procedure M
Detects
Detect 1
Detects
Detect 2
Detects
Detect K

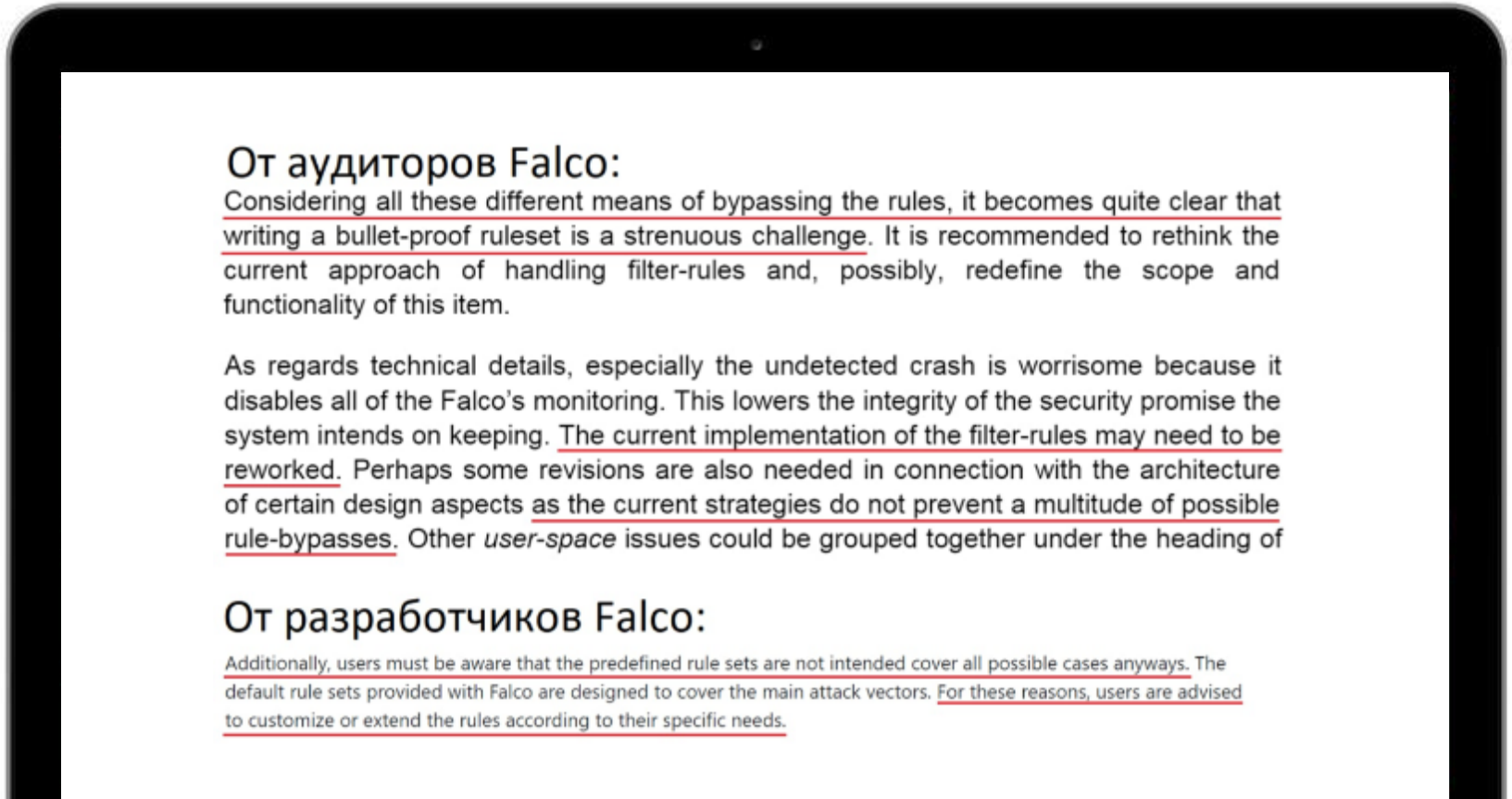

BLACK
FLEX
APE

Итог разбора сигнатур

SOC FORUM 2023

ЦЕЛЬЮ ВЫСТУПЛЕНИЯ НЕ БЫЛ РАЗБОР НЕДОСТАТКОВ FALCO КАК ИНСТРУМЕНТА, А ПОПЫТКА ПОКАЗАТЬ ОБЩИЕ ПРОБЛЕМЫ ПРИМЕНЕНИЯ СИГНАТУРНОГО ПОДХОДА В КОНТЕЙНЕРНЫХ СРЕДАХ.

ВСЕ ИНСТРУМЕНТЫ (КАК И FALCO) ИМЕЮТ СВОИ ПРЕИМУЩЕСТВА И НЕДОСТАТКИ, КОТОРЫЕ НУЖНО УЧИТЫВАТЬ.



ВЫВОДЫ

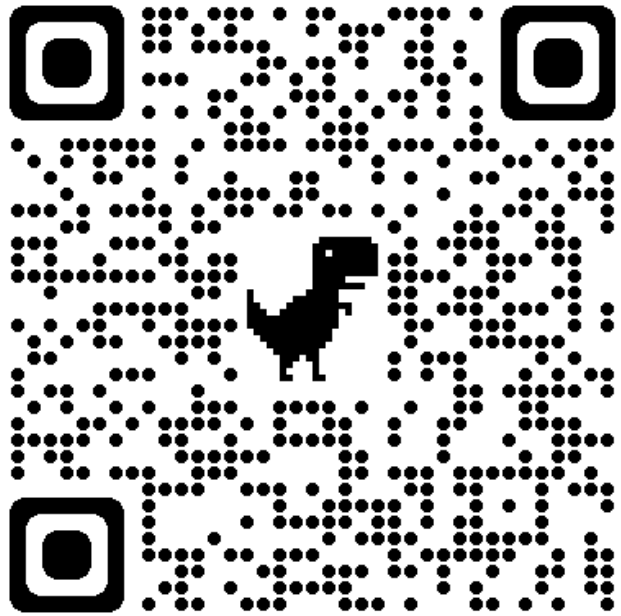
5

Выводы

SOC FORUM 2023

Общую ситуацию с безопасностью контейнеров (и не только) можно описать цитатой из официального блога Kubernetes:

"However, the year-by-year growth in cyber investments does not result in a parallel reduction in cyber incidents. Instead, the number of cyber incidents continues to grow annually. Evidently, organizations are doomed to fail in this struggle - no matter how much effort is made to detect and remove cyber weaknesses from deployed services, it seems offenders always have the upper hand."



Выводы

1. Для обеспечения безопасности Kubernetes-кластеров требуется специализированный инструментарий
2. Контекст контейнеров и Kubernetes очень важен
3. Сигнатурный подход для обнаружения малоэффективен в контейнерном окружении
4. Каждый контейнер – это свой уникальный маленький мир
5. Лучше вообще не доводить до инцидента
6. Предотвращение дешевле лечения

Больше практических кейсов от Solar 4RAYS

Другие полезные материалы от Luntry

SOC FORUM 2023

de@luntry.ru
v.lashkin@rt-solar.ru

ГК «Солар»
Luntry