7 июня 2023 Москва, МЦК ЗИЛ

БЕКОН'23

Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

Как приручить Linux capabilities в Kubernetes

Николай Панченко

Тинькофф

Угрозы безопасности K8s – capabilities

- Запуск процессов с повышенными привилегиями
- Захват мощностей инфраструктуры, майнинг
- Контроль сетевого трафика
- Получение sensitive информации
- Побег из контейнера на Worker ноду
- Побег из контейнера на Master ноду
- Получение прав “Cluster Admin”

Инфраструктура компании

Угрозы безопасности K8s – capabilities

- Запуск процессов с повышенными привилегиями
- Захват мощностей инфраструктуры, майнинг
- Контроль сетевого трафика
- Получение sensitive информации
- Побег из контейнера на Worker ноду
- Побег из контейнера на Master ноду
- Получение прав “Cluster Admin”

Где посмотреть примеры:
“Container escapes: Kubernetes edition”
Дмитрий Евдкоимов (ZeroNights 2021)
https://luntry.ru/video/doklad-container-escapes-kubernetes-edition-na-zeronights-2021

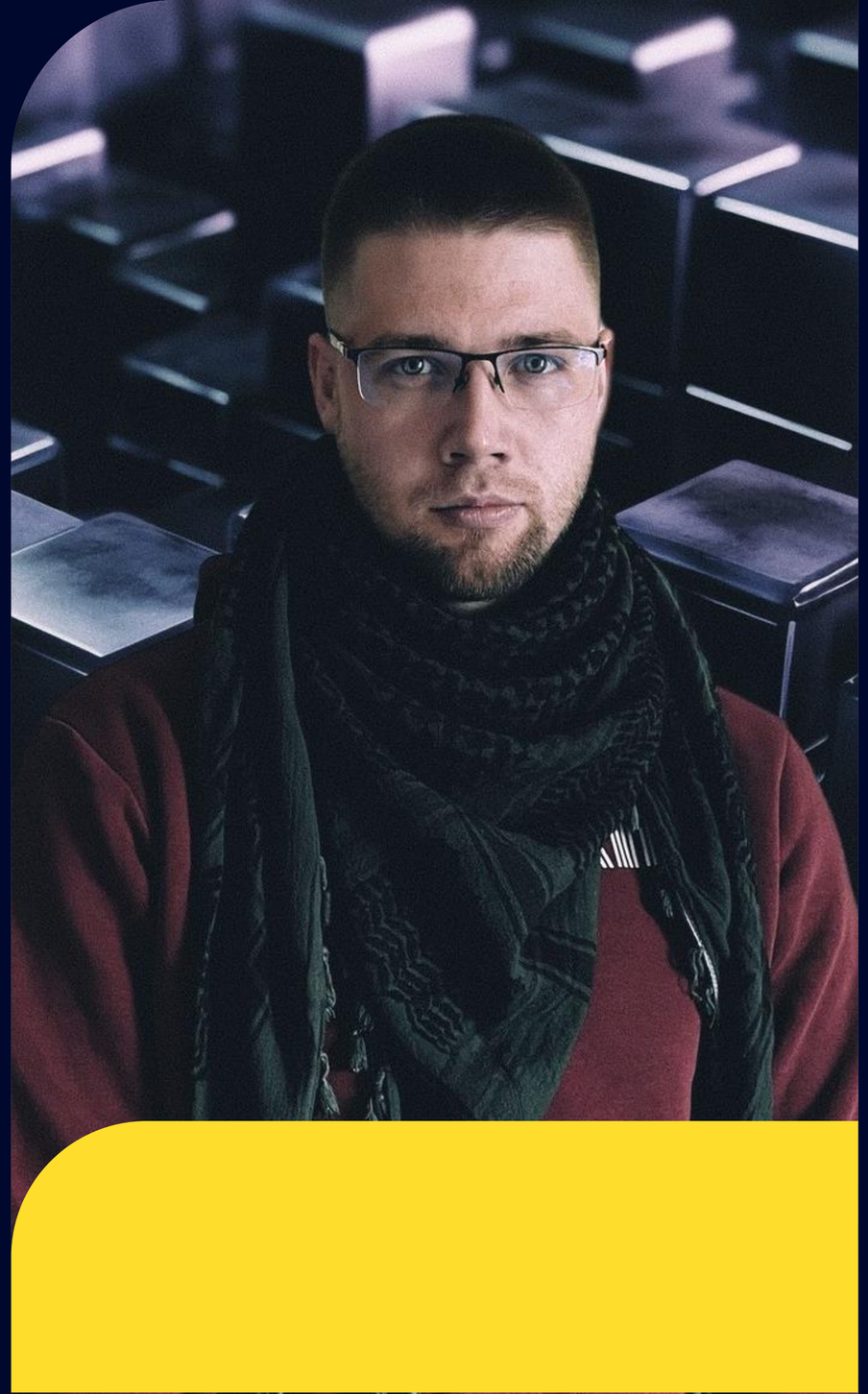
Capabilities – история про “бреши” в безопасности K8s

Что сделает злоумышленник если получит права “Cluster Admin”?

Capabilities – история про “бреши” в безопасности K8s

Что сделает злоумышленник если получит права “Cluster Admin”?

1. Начнет использовать мощности в своих целях
2. Попросит вознаграждение вместо нанесения ущерба
3. Нанесет ущерб через отказ в обслуживании
4. Нанесет ущерб через “сливы” данных



Панченко Николай

Ведущий специалист ИБ

Специализация – K8s

IT's TINKOFF

Скачать Презентацию

k8s (in)security

nickrzaion@gmail.com

n.s.panchenko@tinkoff.ru

Telegram: @yours_rage

План

1. Что такое capabilities в OS Linux
2. Capabilities в k8s
3. Констрейнты для capabilities в k8s
4. Seccomp профили и capabilities
5. Процесс работы с capabilities в Тинькофф
6. Рекомендации

Что такое capabilities в OS Linux

Linux capabilities – что это ?

Именованные привилегии суперпользователя (root)

Linux capabilities – что это ?

- Именованные привилегии суперпользователя (root)
- Предоставляют только ограниченную часть прав

Linux capabilities – что это ?

- Именованные привилегии суперпользователя (root)
- Предоставляют только ограниченную часть прав
- Выставляются на конкретный поток

Linux capabilities – что это ?

- Именованные привилегии суперпользователя (root)
- Предоставляют только ограниченную часть прав
- Выставляются на конкретный поток
- Имя всегда начинается с CAP_<name>

Linux capabilities – что это ?

- Именованные привилегии суперпользователя (root)
- Предоставляют только ограниченную часть прав
- Выставляются на конкретный поток
- Имя всегда начинается с CAP_<name>
- В ядре Linux начиная с версии 2.2

Linux capabilities – как выглядят?

Capabilities list

The following list shows the capabilities implemented on Linux, and the operations or behaviors that each capability permits:

CAP_AUDIT_CONTROL (since Linux 2.6.11)
Enable and disable kernel auditing; change auditing filter rules; retrieve auditing status and filtering rules.

CAP_AUDIT_READ (since Linux 3.16)
Allow reading the audit log via a multicast netlink socket.

CAP_AUDIT_WRITE (since Linux 2.6.11)
Write records to kernel auditing log.

CAP_BLOCK_SUSPEND (since Linux 3.5)
Employ features that can block system suspend (**epoll**(7) **EPOLLWAKEUP**, /proc/sys/wake_lock).

CAP_CHOWN
Make arbitrary changes to file UIDs and GIDs (see **chown**(2)).

CAP_DAC_OVERRIDE
Bypass file read, write, and execute permission checks. (DAC is an abbreviation of "discretionary access control".)

CAP_DAC_READ_SEARCH
* Bypass file read permission checks and directory read and execute permission checks;
* invoke **open_by_handle_at**(2);
* use the **linkat**(2) **AT_EMPTY_PATH** flag to create a link to a file referred to by a file descriptor.

Capabilities “брешь” – CAP_DAC_OVERRIDE

Linux capabilities – как работают?

Linux capabilities – Ограничения

01

Для всех привилегированных операций ядро должно проверять, есть ли у потока требуемые capabilities в его рабочем наборе capabilities

Linux capabilities – Ограничения

01

Для всех привилегированных операций ядро должно проверять, есть ли у потока требуемые capabilities в его рабочем наборе capabilities

02

В ядре для запускаемого потока должны быть доступны **все** запрашиваемые системные вызовы.

Linux capabilities – Ограничения

01

Для всех привилегированных операций ядро должно проверять, есть ли у потока требуемые capabilities в его рабочем наборе capabilities

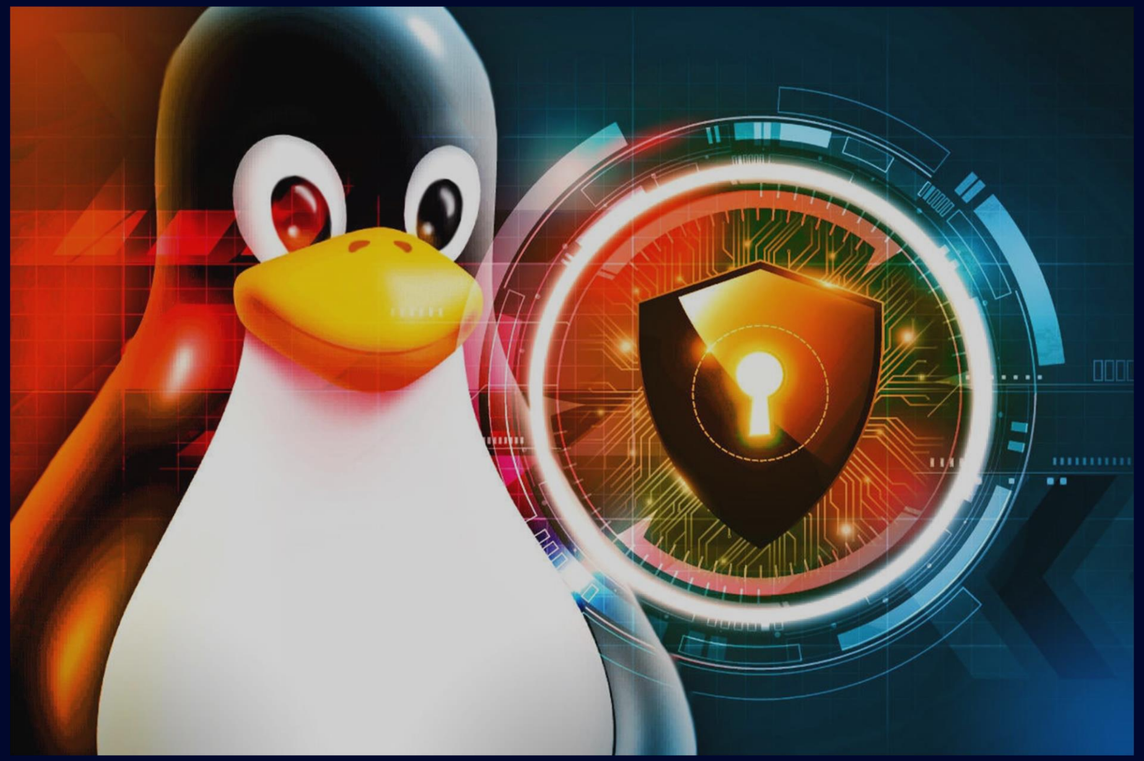
02

В ядре для запускаемого потока должны быть доступны **все** запрашиваемые системные вызовы.

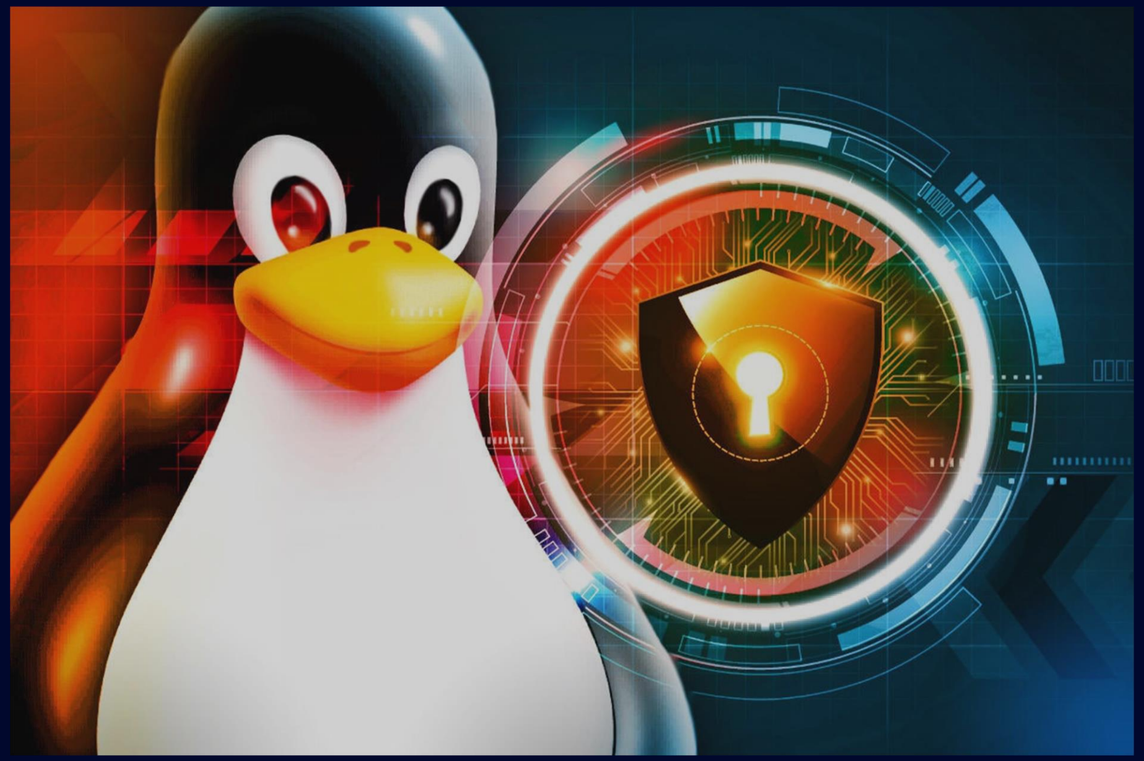
03

Файловая система должна поддерживать присоединение capabilities к исполняемому файлу, чтобы процесс получал эти capabilities при исполнении файла (добавлено с версии ядра Linux 2.6.24).

Linux capabilities – как выставляются?

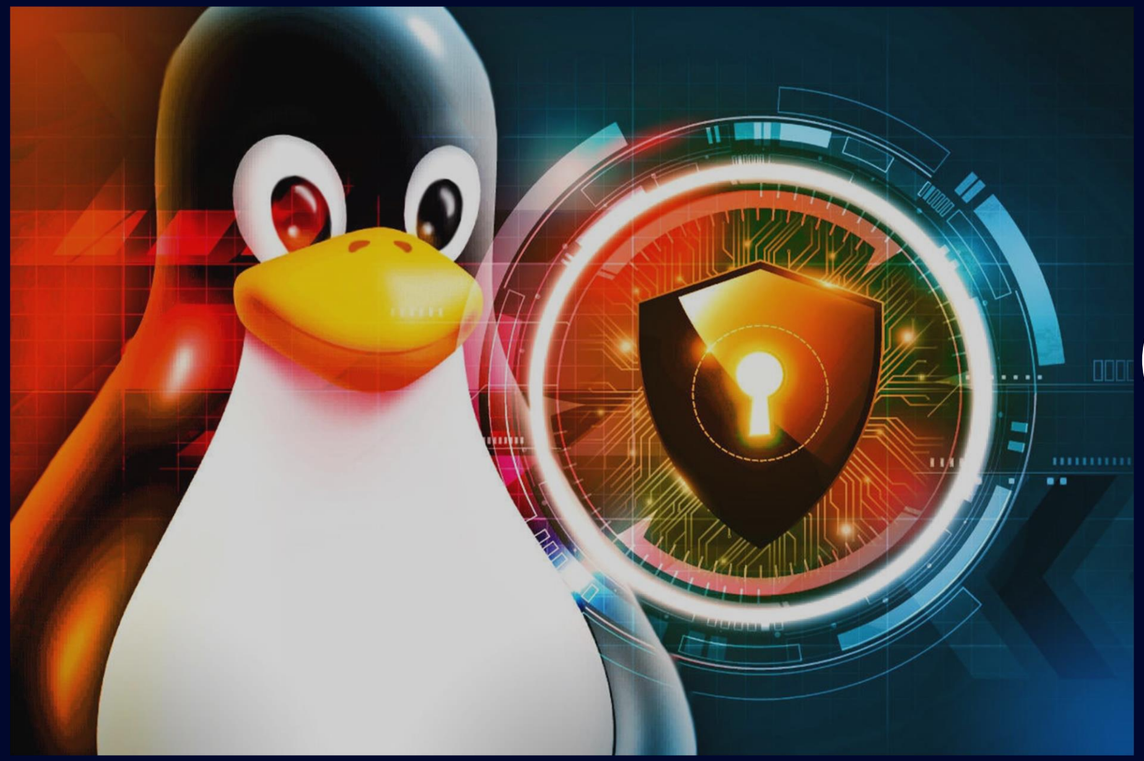


Linux capabilities – как выставляются?



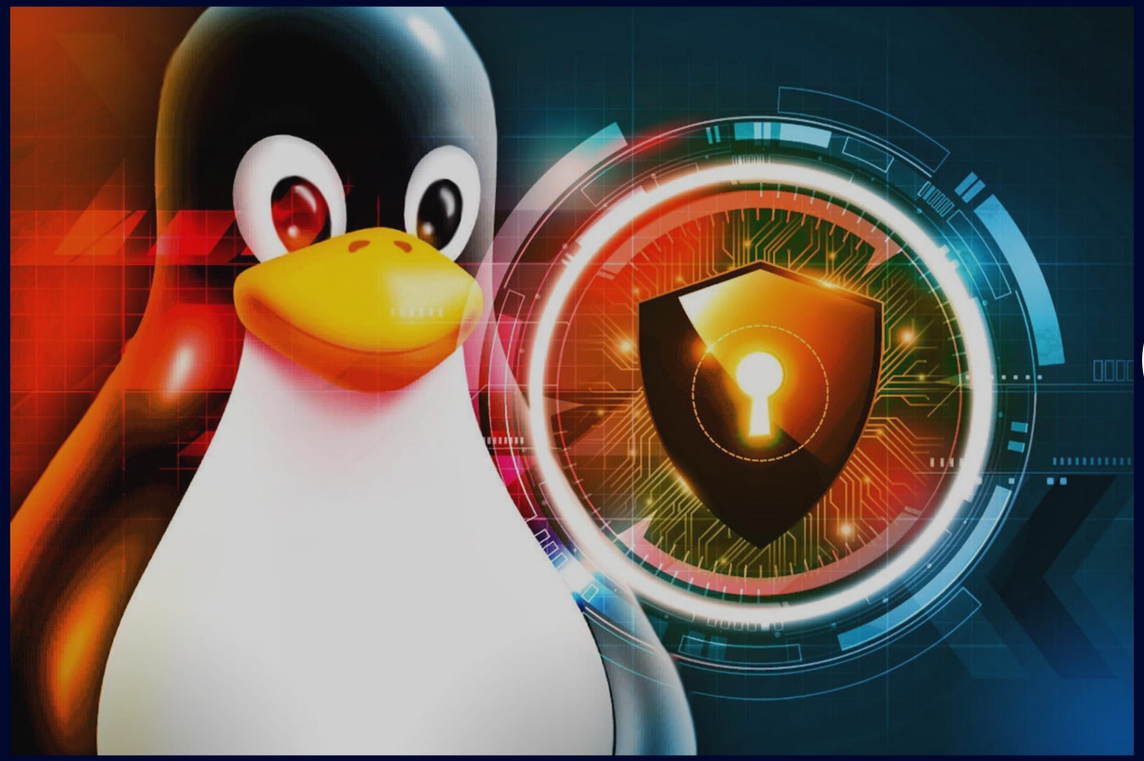
Прямо из потока, при наличии у родительского потока капы CAP_SETCAP

Linux capabilities – как выставляются?



- Прямо из потока, при наличии у родительского потока капы CAP_SETCAP
- Через File capabilities – специальный файл набора capabilities прикручеваемый к исполняемому файлу

Linux capabilities – как выставляются?



- Прямо из потока, при наличии у родительского потока капы CAP_SETCAP
- Через File capabilities – специальный файл набора capabilities прикручеваемый к исполняемому файлу
- Через Namespaced file capabilities – специальный неймспейсный файл набора capabilities для всех запускаемых процессов в namespace (начиная с ядра Linux 4.14)

Linux capabilities – как найти?

```
root@host ~
# ls /proc/325667
arch_status     fd              ns              setgroups
attr            fdinfo          numa_maps       smaps
autogroup       gid_map         oom_adj         smaps_rollup
auxv            io              oom_score       stack
cgroup          limits          oom_score_adj   stat
clear_refs      loginuid        pagemap         statm
cmdline         map_files       patch_state     status
comm            maps            personality     syscall
coredump_filter mem             projid_map      task
cpuset          mountinfo       root            timers
cwd             mounts          sched           timerslack_ns
environ         mountstats      schedstat       uid_map
exe             net             sessionid       wchan
```

```
root@host ~
# grep Cap /proc/325667/status
CapInh: 0000000000000000
CapPrm: 0000003fffffffff
CapEff: 0000003fffffffff
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

Linux capabilities – как распознать?

```
root@host ~
# grep Cap /proc/325667/status
CapInh: 0000000000000000
CapPrm: 0000003fffffffff
CapEff: 0000003fffffffff
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

* Нужен пакет **libcap2-bin** или аналоги!

```
root@host ~
# capsh --decode=0000003fffffffff
0x0000003fffffffff=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin
```

Linux capabilities – как посмотреть?

```
┌─root@DevOps-server ~
└─# getcap /usr/bin/ping
/usr/bin/ping = cap_net_raw+ep
┌─root@DevOps-server ~
└─# getcap /usr/bin/vim.basic
/usr/bin/vim.basic = cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,
cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chro
ot,cap_mknod,cap_audit_write,cap_setfcap+ep
┌─root@DevOps-server ~
└─# getcap /sbin/dash
/sbin/dash = cap_sys_admin+ep
```

* Нужен пакет **libcap2-bin** или аналоги!

“Злые” capabilities – кто какие знает?

“Злые” capabilities

- CAP_SYS_ADMIN
- CAP_SYS_MODULE
- CAP_DAC_OVERRIDE
- CAP_DAC_READ_SEARCH
- CAP_NET_ADMIN
- CAP_NET_RAW
- CAP_SYS_CHROOT
- CAP_SYS_PTRACE
- CAP_SYS_RAWIO
- CAP_SYS_BOOT
- CAP_SYSLOG

Полезная статья – Excessive Capabilities:
https://0xn3va.gitbook.io/cheat-sheets/container/escaping/excessive-capabilities

Capabilities в k8s

Linux capabilities – в k8s

- В k8s можно (нужно) контролировать **capabilities**!
- Можно выставить напрямую через securityContext
- Можно выставить через PSP (до версии k8s 1.25)
- Можно добавить или дропнуть

Linux capabilities – в k8s

- В k8s можно (нужно) контролировать **capabilities**!
- Можно выставить напрямую через securityContext
- Можно выставить через PSP (до версии k8s 1.25)
- Можно добавить или дропнуть

```
securityContext:
        capabilities:
          drop: ["ALL"]
          add: ["NET_BIND_SERVICE", “CAP_SETCAP"]
```

* Пользуемся принципом наименьших привилегий

** Правильнее дропать все, а потом добавлять нужное – так нагляднее =)

Какие capabilities в runtime по умолчанию?

```
apiVersion: v1
kind: Pod
metadata:
    name: security-context-demo-4
spec:
    containers:
    -   name: sec-ctx-4
        image: gcr.io/google-samples/node-hello:1.0
```

Полезная статья – k8s docs:
https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Какие capabilities в runtime по умолчанию?

```
apiVersion: v1
kind: Pod
metadata:
    name: security-context-demo-4
spec:
    containers:
    -   name: sec-ctx-4
       image: gcr.io/google-samples/node-hello:1.0
```

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000a80425fb
CapEff: 00000000a80425fb
...
```

Полезная статья – k8s docs:
https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Какие capabilities в runtime по умолчанию?

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
```

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000a80425fb
CapEff: 00000000a80425fb
...
```

```
root@DevOps-server ~
# capsh --decode=00000000a80425fb
0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid
,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_a
udit_write,cap_setfcap
```

Полезная статья – k8s docs:
https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
    name: security-context-demo-4
spec:
    containers:
    -   name: sec-ctx-4
        image: gcr.io/google-samples/node-hello:1.0
        securityContext:
            capabilities:
                add: ["NET_ADMIN", "SYS_TIME"]
```

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
         capabilities:
            add: ["NET_ADMIN", "SYS_TIME"]
```

Default caps – CapPrm: 00000000a80425fb

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
         capabilities:
            add: ["NET_ADMIN", "SYS_TIME"]
```

Default caps – CapPrm: 00000000a80425fb

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
...
```

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
         capabilities:
            add: ["NET_ADMIN", "SYS_TIME"]
```

Default caps – CapPrm: 00000000a80425fb

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
...
```

Сравниваем:

00000000a80425fb – по умолчанию в образе

00000000aa0435fb – после подключения

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
         capabilities:
           add: ["NET_ADMIN", "SYS_TIME"]
```

Default caps – CapPrm: 00000000a80425fb

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
...
```

Сравниваем:

00000000a80425fb – по умолчанию в образе

00000000aa0435fb – после подключения

Вывод: после подключения capabilities для рабочего процесса биты привилегий меняются!

Linux capabilities – в k8s (пример 1)

```
apiVersion: v1
kind: Pod
metadata:
   name: security-context-demo-4
spec:
   containers:
   -  name: sec-ctx-4
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
         capabilities:
           add: ["NET_ADMIN", "SYS_TIME"]
```

Default caps – CapPrm: 00000000a80425fb

```
$ kubectl exec -it security-context-demo-4 – sh

$ cd /proc/1
$ cat status
...
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
...
```

Сравниваем:

00000000a80425fb – по умолчанию в образе

00000000aa0435fb – после подключения

Вывод: после подключения capabilities для рабочего процесса биты привилегий меняются!

Какой бит соответствует выбранной Вами капе можно посмотреть прямо у Торвальдса =) Linux Kernel 6.4

https://github.com/torvalds/linux/blob/master/include/uapi/linux/capability.h

Спойлер – на момент создания презентации в мастер-ветке была 41 константа =)

Linux capabilities – в k8s (пример 2)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: drop-all-caps-for-nginx
spec:
  containers:
  - name: drop-all-caps-for-nginx
    image: nginx:1.22.0-bullseye
    securityContext:
      capabilities:
        drop: ["ALL"]
```

Заработает ли?

Linux capabilities – в k8s (пример 2)

```yaml
apiVersion: v1
kind: Pod
metadata:
   name: drop-all-caps-for-nginx
spec:
   containers:
   -   name: drop-all-caps-for-nginx
      image: nginx:1.22.0-bullseye
      securityContext:
         capabilities:
            drop: ["ALL"]
```

Заработает ли?

```
$ kubectl exec -it drop-all-caps-for-nginx – sh

$ cd /proc/1
$ cat status
...
CapPrm: 0000000000000000
CapEff: 0000000000000000
...
```

Вывод: capabilities вообще нет!

Linux capabilities – в k8s (пример 2)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: drop-all-caps-for-nginx
spec:
  containers:
  - name: drop-all-caps-for-nginx
    image: nginx:1.22.0-bullseye
    securityContext:
      capabilities:
        drop: ["ALL"]
```

Заработает ли?
НЕТ!
Нужно добавить капы!
Какие?

Полезная статья – DATADOG security labs:
https://securitylabs.datadoghq.com/articles/container-security-fundamentals-part-3/

Linux capabilities – в k8s (пример 2)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: drop-all-caps-for-nginx
spec:
  containers:
  - name: drop-all-caps-for-nginx
    image: nginx:1.22.0-bullseye
    securityContext:
      capabilities:
        drop: ["ALL"]
          add:
            - NET_BIND_SERVICE
            - SYS_CHROOT
```

Заработает ли?
НЕТ!
Нужно добавить капы!
Какие?

- NET_BIND_SERVICE
- SYS_CHROOT

Полезная статья – DATADOG security labs:
https://securitylabs.datadoghq.com/articles/container-security-fundamentals-part-3/

Linux capabilities – в k8s (пример 3)

```
apiVersion: v1
kind: Pod
metadata:
   name: drop-all-caps-for-nginx
spec:
   containers:
   -   name: drop-all-caps-for-nginx
      image: nginx:1.22.0-bullseye
     securityContext:
          privileged: true
        capabilities:
          drop: ["ALL"]
             add: ["NET_BIND_SERVICE", "SYS_CHROOT"]
```

Что произойдет?

Linux capabilities – в k8s (пример 3)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: drop-all-caps-for-nginx
spec:
  containers:
  - name: drop-all-caps-for-nginx
    image: nginx:1.22.0-bullseye
    securityContext:
      privileged: true
      capabilities:
        drop: ["ALL"]
          add: ["NET_BIND_SERVICE", "SYS_CHROOT"]
```

Что произойдет?
Правильно!

Нас запустит от пользователя **root** со **всеми** capabilities!

Вывод: в securityContext нужно следить за приоритетом!

Linux capabilities – k8s PSP

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: drop-all-caps-and-priv-false
spec:
  privileged: false
  requiredDropCapabilities:
    - ALL
```

Linux capabilities – k8s PSP

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: drop-all-caps-and-priv-false
spec:
  privileged: false
  requiredDropCapabilities:
    - ALL
```

Какие параметры бывают:

- RequiredDropCapabilities
- AllowedCapabilities
- DefaultAddCapabilities

Linux capabilities – k8s PSP

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: drop-all-caps-and-priv-false
spec:
  privileged: false
  requiredDropCapabilities:
    - ALL
```

Какие параметры бывают:

- RequiredDropCapabilities
- AllowedCapabilities
- DefaultAddCapabilities

Минусы?

- Бесконечные приколы с сервисными аккаунтами
- Нужно добавлять принудительно
- v1beta1 – были в бете (значит могут быть баги)

Linux capabilities – k8s PSP

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: drop-all-caps-and-priv-false
spec:
  privileged: false
  requiredDropCapabilities:
    - ALL
```

Какие параметры бывают:

- RequiredDropCapabilities
- AllowedCapabilities
- DefaultAddCapabilities

Минусы?

- Бесконечные приколы с сервисными аккаунтами
- Нужно добавлять принудительно
- v1beta1 – были в бете (значит могут быть баги)

> **Removed feature**
>
> PodSecurityPolicy was deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25.

Констрейнты для capabilities в k8s

Констрейнты безопасности K8s, PolicyEngines

VS

OPA Gatekeeper

Kyverno

Linux capabilities – Kyverno (пример)

Идем на сайт Kyverno в раздел политики > лучшие практики:
https://kyverno.io/policies/best-practices/require_drop_all/require_drop_all/

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: drop-all-capabilities
  annotations:
    policies.kyverno.io/title: Drop All Capabilities
    policies.kyverno.io/category: Best Practices
    policies.kyverno.io/severity: medium
    policies.kyverno.io/minversion: 1.6.0
    policies.kyverno.io/subject: Pod
    policies.kyverno.io/description: >-
      Capabilities permit privileged actions without giving full root access. All
      capabilities should be dropped from a Pod, with only those required added back.
      This policy ensures that all containers explicitly specify the `drop: ["ALL"]`
      ability. Note that this policy also illustrates how to cover drop entries in any
      case although this may not strictly conform to the Pod Security Standards.
```

Linux capabilities – Kyverno (пример)

```
spec:
  validationFailureAction: audit
  background: true
  rules:
    - name: require-drop-all
      match:
        any:
        - resources:
            kinds:
              - Pod
      preconditions:
        all:
        - key: "{{ request.operation || 'BACKGROUND' }}"
          operator: NotEquals
          value: DELETE
      validate:
        message: >-
                  Containers must drop `ALL` capabilities.
        foreach:
          - list: request.object.spec.[ephemeralContainers, initContainers, containers][]
            deny:
              conditions:
                all:
                - key: ALL
                  operator: AnyNotIn
                  value: "{{ element.securityContext.capabilities.drop[].to_upper(@) || `[]` }}"
```

Linux capabilities – OPA Gatekeeper (пример)

```
---
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sDenyCapabilitiesViaPSP
metadata:
  name: deny-all-capabilities-via-psp
spec:
  match:
    kinds:
      - apiGroups: ["apps"]
        kinds: ["Pod"]
      - apiGroups: ["apps"]
        kinds: ["Deployment"]
      - apiGroups: ["apps"]
        kinds: ["DaemonSet"]
      - apiGroups: ["apps"]
        kinds: ["StatefulSet"]
    excludedNamespaces:
      - gatekeeper-system
      - kube-system
  parameters:
        requiredDropCapabilities: ["ALL"]
        allowedCapabilities: []
```

Linux capabilities – OPA Gatekeeper (пример)

```
---
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8sdenycapabilitiesviapsp
spec:
  crd:
    spec:
      names:
        kind: K8sDenyCapabilitiesViaPSP
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8sdenycapabilitiesviapsp

        missing_drop_capabilities(container) {
          must_drop := {c | c := input.parameters.requiredDropCapabilities[_]}
          dropped := {c | c := container.securityContext.capabilities.drop[_]}
          count(must_drop - dropped) > 0
        }

        ...
```

Linux capabilities – OPA Gatekeeper (пример)

```
...
    has_disallowed_capabilities(container) {
      allowed := {c | c := input.parameters.allowedCapabilities[_]}
      not allowed["*"]
      capabilities := {c | c := container.securityContext.capabilities.add[_]}
      count(capabilities - allowed) > 0
    }

    get_default(obj, param, _default) = out {
      out = obj[param]
    }
    get_default(obj, param, _default) = out {
      not obj[param]
      not obj[param] == false
      out = _default
    }
    violation[{"msg": msg}] {
      container := input_object_container_spec.initContainers[_]
      has_disallowed_capabilities(container)
      msg := sprintf("init container <%v> has a disallowed capability.\
      Allowed capabilities are %v", [container.name, get_default(input.parameters, "allowedCapabilities", "NONE")])
    }      ... И так далее – описываем все violations с необходимым текстом и если нужно с сылками на внутреннюю Wiki
```

Кто победил?

VS

OPA Gatekeeper

Kyverno

WINS!

Seccomp профили и capabilities

Linux capabilities and Seccomp profiles

01

Capabilities и доступность
системных вызовов –
это разные вещи!

Linux capabilities and Seccomp profiles

01

Capabilities и доступность системных вызовов – это разные вещи!

02

Для рантайма существуют специальные профили, разрешающие только определенные вызовы

Linux capabilities and Seccomp profiles

01

Capabilities и доступность системных вызовов – это разные вещи!

02

Для рантайма существуют специальные профили, разрешающие только определенные вызовы

03

Определенные вызовы можно запихивать под определенные capabilities

Linux capabilities and Seccomp profiles

01

Capabilities и доступность системных вызовов – это разные вещи!

02

Для рантайма существуют специальные профили, разрешающие только определенные вызовы

03

Определенные вызовы можно запихивать под определенные capabilities

04

Можно разделять профили на весь рантайм или на конкретные нагрузки через securityContext

Linux capabilities and Seccomp profiles

01

Capabilities и доступность системных вызовов – это разные вещи!

02

Для рантайма существуют специальные профили, разрешающие только определенные вызовы

03

Определенные вызовы можно запихивать под определенные capabilities

04

Можно разделять профили на весь рантайм или на конкретные нагрузки через securityContext

05

В кластере на kubelet можно выставить seccomp RuntimeDefault, как профиль по умолчанию для всех нагрузок на этой ноде

Linux capabilities and Seccomp profiles

01

Capabilities и доступность системных вызовов – это разные вещи!

02

Для рантайма существуют специальные профили, разрешающие только определенные вызовы

03

Определенные вызовы можно запихивать под определенные capabilities

04

Можно разделять профили на весь рантайм или на конкретные нагрузки через securityContext

05

В кластере на kubelet можно выставить seccomp RuntimeDefault, как профиль по умолчанию для всех нагрузок на этой ноде

06

Профиль больше нельзя назначить по умолчанию через функции containerd

Seccomp profile operator

Seccomp profile operator

Плюсы

- Его применение уменьшает поверхность атаки
- Умеет сам раскидывать профили прямо к Kubelet
- Умеет black/white листы на уровне кластера
- Умеет записывать профиль приложения
- Поддерживает репозитории профилей (наконец-то!)

Seccomp profile operator

Плюсы

- Его применение уменьшает поверхность атаки
- Умеет сам раскидывать профили прямо к Kubelet
- Умеет black/white листы на уровне кластера
- Умеет записывать профиль приложения
- Поддерживает репозитории профилей (наконец-то!)

Минусы

- ЭТО ОПЕРАТОР! – повышенные права в кластере
- Нужно оборачивать в процесс (Нужна команда, которая будет готова поддерживать эту историю)
- Для DEV команд нужно объяснить, что это важно =)
- Для DEV команд нужна доработка процесса QA
- * в ранних версиях нужно было бороться с доставкой профилей из тест кластера на прод

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

02

Включайте по умолчанию **RuntimeDefault** для **Kubelet** (с k8s v1.22)

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

02

Включайте по умолчанию **RuntimeDefault** для **Kubelet** (с k8s v1.22)

03

Старайтесь максимально минимизировать профиль и применяйте его прямо к нагрузке через SecutiryContext (stable с k8s v1.19)

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

02

Включайте по умолчанию **RuntimeDefault** для **Kubelet** (с k8s v1.22)

03

Старайтесь максимально минимизировать профиль и применяйте его прямо к нагрузке через SecutiryContext (stable с k8s v1.19)

04

Проводите достаточное количество тестов проверки функциональности приложения, чтобы его не поломать запретами

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

02

Включайте по умолчанию **RuntimeDefault** для **Kubelet** (с k8s v1.22)

03

Старайтесь максимально минимизировать профиль и применяйте его прямо к нагрузке через SecutiryContext (stable с k8s v1.19)

04

Проводите достаточное количество тестов проверки функциональности приложения, чтобы его не поломать запретами

05

Контролируйте **SecutiryContext** на подключение недопустимых для приложения профилей через **констрейнты** вашего **PolicyEngine**

Seccomp profiles - рекомендации

01

Включите поддержку **Seccomp** в настройках кластера k8s

02

Включайте по умолчанию **RuntimeDefault** для **Kubelet** (с k8s v1.22)

03

Старайтесь максимально минимизировать профиль и применяйте его прямо к нагрузке через SecutiryContext (stable с k8s v1.19)

04

Проводите достаточное количество тестов проверки функциональности приложения, чтобы его не поломать запретами

05

Контролируйте **SecutiryContext** на подключение недопустимых для приложения профилей через **констрейнты** вашего **PolicyEngine**

06

Используйте в работе **Seccomp profile operator** - (https://github.com/kubernetes-sigs/security-profiles-operator)

Процесс работы с capabilities в Тинькофф

Процесс работы с capabilities в Тинькофф

По умолчанию: всем - DROP ALL

Процесс работы с capabilities в Тинькофф

По умолчанию: всем - DROP ALL

Почему?

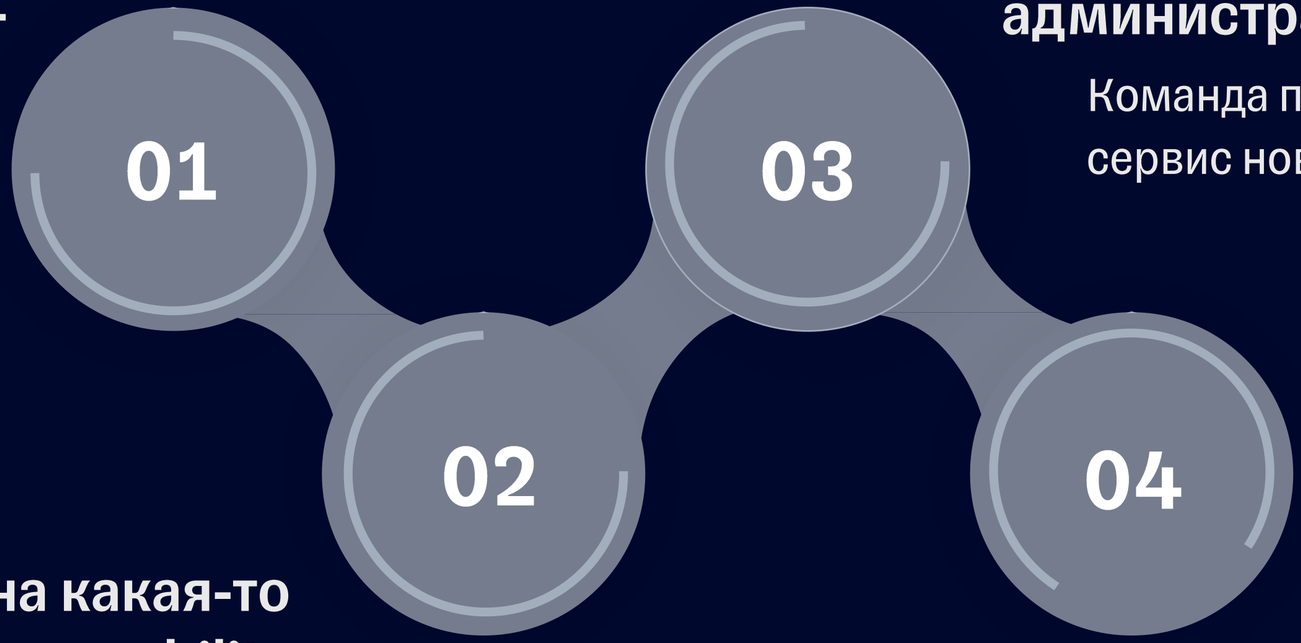
Процесс работы с capabilities в Тинькофф

По умолчанию: всем - DROP ALL

Процесс работы с capabilities + OPA Gatekeeper

По умолчанию всем - DROP ALL

Это обусловлено тем, что большинству микросервисов повышенные привилегии по умолчанию не нужны!



Команда идет к администраторам k8s

Команда просит выставить на их сервис новую капу

Команде нужна какая-то capability

Привилегии нужны, вопрос какие?
Команда проводит предварительные тесты

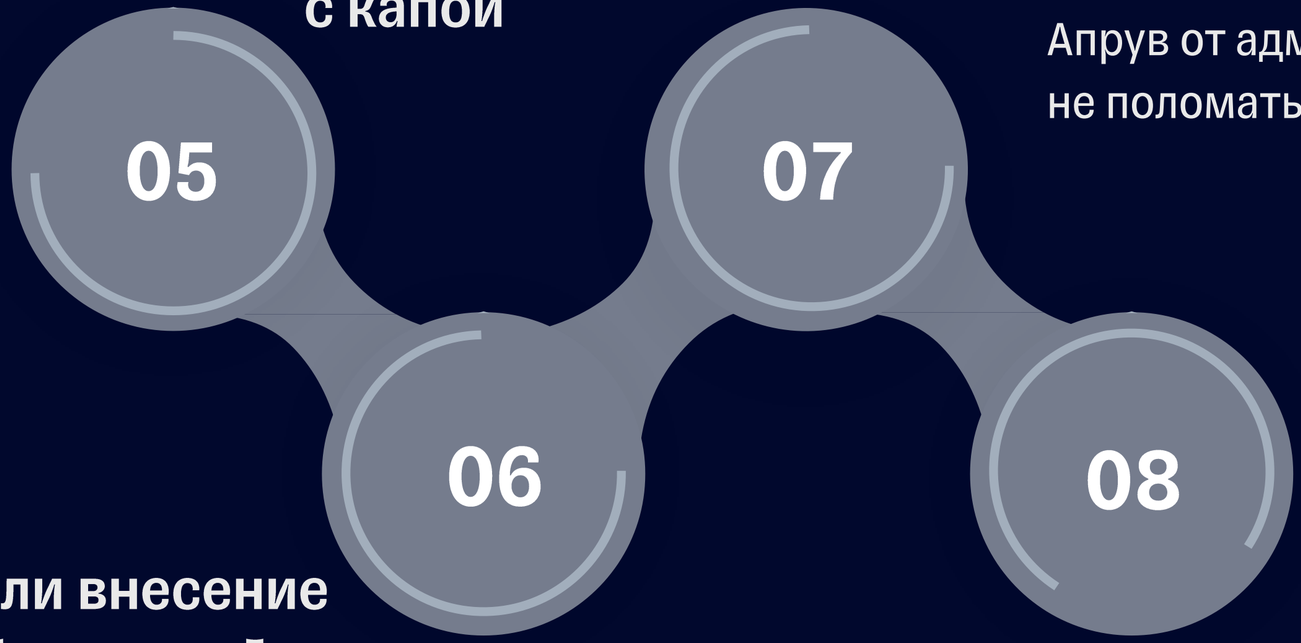
Админы k8s проверяют есть ли в репе констрейнтов нужный с нужными ограничениями?

- Если находят, то идут в Security и ссылаются на найденный констрейнт
- если не находят, идут в Security и просят создать новый констрейнт

Процесс работы с capabilities + OPA Gatekeeper

Security получает запрос от админов k8s на проверку или создание констрейнта с капой

Security проверяет насколько сервис **действительно** нуждается в этой капе, если не нуждается – отказ с подробным объяснением почему



Deploy в TEST –> ОК
MR в PROD –> подтверждение от Админов k8s –> Deploy в PROD

Апрув от админов нужен, чтобы не поломать прод, что логично =)

Написание или доработка тестов над констрейнтами

Написание нового или внесение правок в старый констрейнт MR и Deploy в TEST

История про работу с OPA Gatekeeper, репами и MR

Если админы k8s отвечают, что на проде все ок, то последним штрихом командой Security дорабатывается автоматический тест правильной работы констрейнтов на тестовом и продовом окружении

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

02

Для работы с capabilities сформирован специальный рабочий процесс

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

02

Для работы с capabilities сформирован специальный рабочий процесс

03

В процессе всегда участвуют 3 команды:

- Команда DEV
- Админы K8s
- Команда Security

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

02

Для работы с capabilities сформирован специальный рабочий процесс

03

В процессе всегда участвуют 3 команды:

- Команда DEV
- Админы K8s
- Команда Security

04

Для всех по умолчанию DROP ALL
+ для каждого кластера на каждый констрейнт в нужном окружении есть автотест

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

02

Для работы с capabilities сформирован специальный рабочий процесс

03

В процессе всегда участвуют 3 команды:

- Команда DEV
- Админы K8s
- Команда Security

04

Для всех по умолчанию DROP ALL
+ для каждого кластера на каждый констрейнт в нужном окружении есть автотест

05

При отказах всегда подробно объясняем почему не стоит применять выбранную капу и предлагаем нужные

Процесс работы с capabilities + OPA Gatekeeper

01

В качестве policyEngine для констрейнтов используется OPA Gatekeeper

02

Для работы с capabilities сформирован специальный рабочий процесс

03

В процессе всегда участвуют 3 команды:

- Команда DEV
- Админы K8s
- Команда Security

04

Для всех по умолчанию DROP ALL
+ для каждого кластера на каждый констрейнт в нужном окружении есть автотест

05

При отказах всегда подробно объясняем почему не стоит применять выбранную капу и предлагаем нужные

06

Иногда капы **действительно** нужны, но обычно достаточно дополнительного харденинга OS в образе или доработки самого сервиса

Linux capabilities – побеги

Linux capabilities – побеги

Google – побег через unshare()

Исходные данные:

- K8s кластер
- Deployment с DROP ALL CAPABILITIES
- Не выставлен SeccompProfile

Linux capabilities – побеги

Google – побег через unshare()

Исходные данные:

- K8s кластер
- Deployment с DROP ALL CAPABILITIES
- Не выставлен SeccompProfile

Риск:

Злоумышленник внутри контейнера благодаря доступной команде unshare из-за отсутствия дефолтного SeccompProfile, может получить дополнительные опасные capabilities, которые могут увеличить поверхность атаки для побега через уязвимости ядра.

Linux capabilities – побеги

Google – побег через unshare()

```
kubectl exec -it pod -- bash

root@pod:/run# unshare -r
# bash
root@pod:/run# capsh --print
Current: =ep
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_
linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys
_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_t
ty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_ala
rm,cap_block_suspend,cap_audit_read
Ambient set =
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=0(root) euid=0(root)
gid=0(root)
groups=
Guessed mode: UNCERTAIN (0)
```

Linux capabilities – побеги

Google – побег через unshare()

Получаем:

- Наличие SYS_MODULE, CAP_SYS_PTRACE, CAP_SYS_ADMIN
- Бежим из контейнера через CVE-2022-0185 =)

Linux capabilities – побеги

Google – побег через unshare()

Получаем:

- Наличие SYS_MODULE, CAP_SYS_PTRACE, CAP_SYS_ADMIN
- Бежим из контейнера через CVE-2022-0185 =)

Пример данного приема можно посмотреть в статье:
“CVE-2022-0185 - Winning a $31337 Bounty after Pwning Ubuntu and Escaping Google's KCTF Containers”
(https://www.willsroot.io/2022/01/cve-2022-0185.html)

Linux capabilities – побеги

Google – побег через unshare()

Рекомендации:

Запретить возможность запуск unshare через:

- настройку специализированного Seccomp профиля на нагрузку,
- принудительного RuntimeDefault на нагрузку через констрейнты
- или SeccompDefault для kubelet (Alpha1.22-1.24, Beta1.25, Stable1.27).

* Полезная информация – где можно найти подобные примеры побегов?

@k8security – канал о (не)безопасности Kubernetes + микросервисных, контейнеризированных приложений.

Рекомендации по работе с Capabilities в K8s

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

02

По умолчанию **дропайте все** повышенные привилегии (DROP CAPABILITIES ALL)

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

02

По умолчанию **дропайте все** повышенные привилегии (DROP CAPABILITIES ALL)

03

Там где нельзя доропать все, пользуйтесь принципом предоставления наименьших привилегий. Добавляйте только то, что **реально** нужно!

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

02

По умолчанию **дропайте все** повышенные привилегии (DROP CAPABILITIES ALL)

03

Там где нельзя доропать все, пользуйтесь принципом предоставления наименьших привилегий. Добавляйте только то, что **реально** нужно!

04

Проводите достаточное количество тестов, чтобы не блокировать нормальную работу приложений!

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

02

По умолчанию **дропайте все** повышенные привилегии (DROP CAPABILITIES ALL)

03

Там где нельзя доропать все, пользуйтесь принципом предоставления наименьших привилегий. Добавляйте только то, что **реально** нужно!

04

Проводите достаточное количество тестов, чтобы не блокировать нормальную работу приложений!

05

Не забывайте про **Seccomp профили**. Разрешенные вызовы тоже очень важны!

Linux capabilities – рекомендации

01

Контролировать capabilities – **очень важно!** Удобнее это делать через **констрейнты**. Уменьшайте поверхность атаки!

02

По умолчанию **дропайте все** повышенные привилегии (DROP CAPABILITIES ALL)

03

Там где нельзя доропать все, пользуйтесь принципом предоставления наименьших привилегий. Добавляйте только то, что **реально** нужно!

04

Проводите достаточное количество тестов, чтобы не блокировать нормальную работу приложений!

05

Не забывайте про **Seccomp профили**. Разрешенные вызовы тоже очень важны!

06

Настраивайте в своей компании собственный удобный для вас процесс работы с capabilities и Seccomp профилями

* В видео использованы материалы из игры GTA: San Andreas от разработчика Rockstar North

Вопросы?

7 июня 2023 Москва, МЦК ЗИЛ

Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

Contacts:

nickrzaion@gmail.com

n.s.panchenko@tinkoff.ru

Telegram: @yours_rage

Capabilities – история про “бреши” в безопасности K8s

Linux capabilities sets – типы наборов

```
root@host ~
# grep Cap /proc/325667/status
CapInh: 0000000000000000
CapPrm: 000000ffffffffff
CapEff: 000000ffffffffff
CapBnd: 000000ffffffffff
CapAmb: 0000000000000000
```

1. Inheritable - наследуемые
2. Permitted - разрешенные
3. Effective - действующие
4. Bounding - прикрепленные
5. Ambient - сохраняемые

Linux capabilities – как установить?

```
┌──root@DevOps-server ~
└─# getcap /sbin/dash
/sbin/dash =
┌──root@DevOps-server ~
└─# setcap 'cap_sys_admin=+ep' "/sbin/dash"
┌──root@DevOps-server ~
└─# getcap /sbin/dash
/sbin/dash = cap_sys_admin+ep
```

* Нужен пакет **libcap2-bin** или аналоги!

Linux capabilities – как дропнуть?

```
┌──root@DevOps-server ~
└─# setcap 'cap_sys_admin=-ep' "/sbin/dash"
┌──root@DevOps-server ~
└─# getcap /sbin/dash
/sbin/dash =
```

* Нужен пакет **libcap2-bin** или аналоги!

Linux capabilities – в Docker

Runtime privilege and Linux capabilities

Option	Description
`--cap-add`	Add Linux capabilities
`--cap-drop`	Drop Linux capabilities
`--privileged`	Give extended privileges to this container
`--device=[]`	Allows you to run devices inside the container without the `--privileged` flag.

Как это выглядит в консоли:

$ docker run --cap-add=SYS_ADMIN ...

$ docker run --cap-drop=CAP_SYS_ADMIN ...

Linux capabilities – в Docker (пример)

```
$ docker run --rm -it --cap-add SYS_ADMIN sshfs sshfs sven@10.10.10.20:/home/sven /mnt

fuse: failed to open /dev/fuse: Operation not permitted

$ docker run --rm -it --device /dev/fuse sshfs sshfs sven@10.10.10.20:/home/sven /mnt

fusermount: mount failed: Operation not permitted

$ docker run --rm -it --cap-add SYS_ADMIN --device /dev/fuse sshfs

# sshfs sven@10.10.10.20:/home/sven /mnt
The authenticity of host '10.10.10.20 (10.10.10.20)' can't be established.
ECDSA key fingerprint is 25:34:85:75:25:b0:17:46:05:19:04:93:b5:dd:5f:c6.
Are you sure you want to continue connecting (yes/no)? yes
sven@10.10.10.20's password:

root@30aa0cfaf1b5:/# ls -la /mnt/src/docker

total 1516
drwxrwxr-x 1 1000 1000   4096 Dec  4 06:08 .
drwxrwxr-x 1 1000 1000   4096 Dec  4 11:46 ..
-rw-rw-r-- 1 1000 1000     16 Oct  8 00:09 .dockerignore
-rwxrwxr-x 1 1000 1000    464 Oct  8 00:09 .drone.yml
drwxrwxr-x 1 1000 1000   4096 Dec  4 06:11 .git
-rw-rw-r-- 1 1000 1000    461 Dec  4 06:08 .gitignore
....
```

Linux capabilities – Kyverno

Плюсы

- Большое комьюнити
- Вменяемая документация
- Есть уже готовые хорошо описанные правила
- Писать правила достаточно просто
- Правила на YAML - Сеньоры-YAML-девелоперы аплодируют стоя =)

Минусы

- Нужно учить примитивы и операторы
- Ты ограничен операторами типа AnyNotIn, что не позволяет писать сложные конструкции
- Разработчики не всегда идут на контакт
- Есть баги

Linux capabilities – OPA Gatekeeper

Open Policy Agent

Плюсы

- Можно писать очень сложные правила и они будут работать
- Меньшее количество багов чем у Kyverno
- Удобно работать с шаблонами
- Все, что нужно для работы – есть на Github

Минусы

- Да, грустная история про “Учите REGO”
- Не самая удобная документация, но есть на Github
- Нет большого количества примеров сложных констрейнтов, примеры – вы о чем? – пишите сами!
- Комьюнити меньше, чем у Kyverno

Демо – офтоп
Закрепление в OS Linux с использованием docker, tar и linux capabilities

K8s (in) security

БЕКОН

Наше сообщество

@k8security

@k8security – канал о (не)безопасности Kubernetes + микросервисных, контейнеризированных приложений. Ценим и любим reliability и security, а также observability.