

Латаем огрехи в образах приложений до рантайма, во время и после

Анатолий Карпенко

Luntry

[@rusdacent](#)

whoami

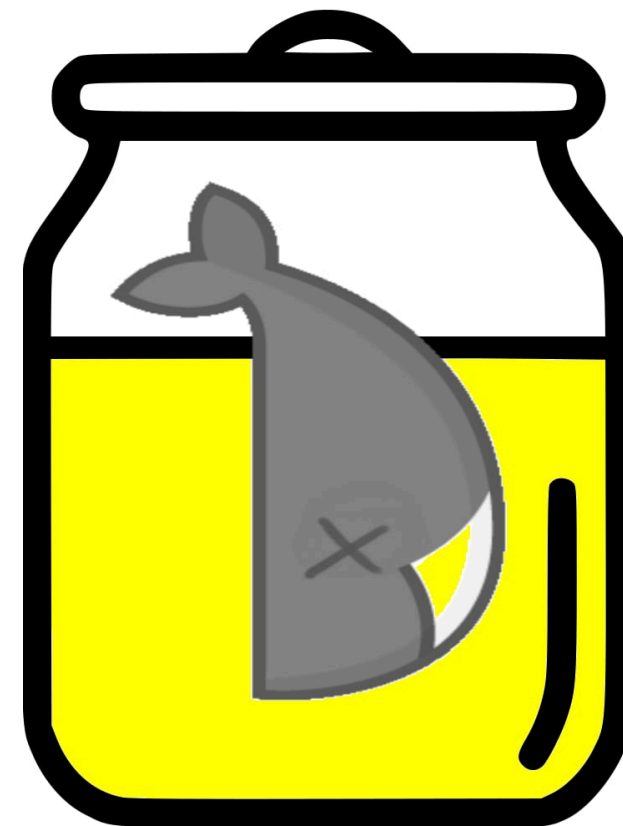
- Автоматизатор автоматизации в [Luntry](#)
- Любитель митапошных форматов: [SPb Reliability Meetup](#), [ITGM](#), [TechTrain](#), [DevOops](#), [DEFCON'ы](#), [B4CKSP4CE](#), [DevOps40](#)
- Веду канал «Технологический Болт Генона»
- Рисую несмешные мемы



Грустная предыстория

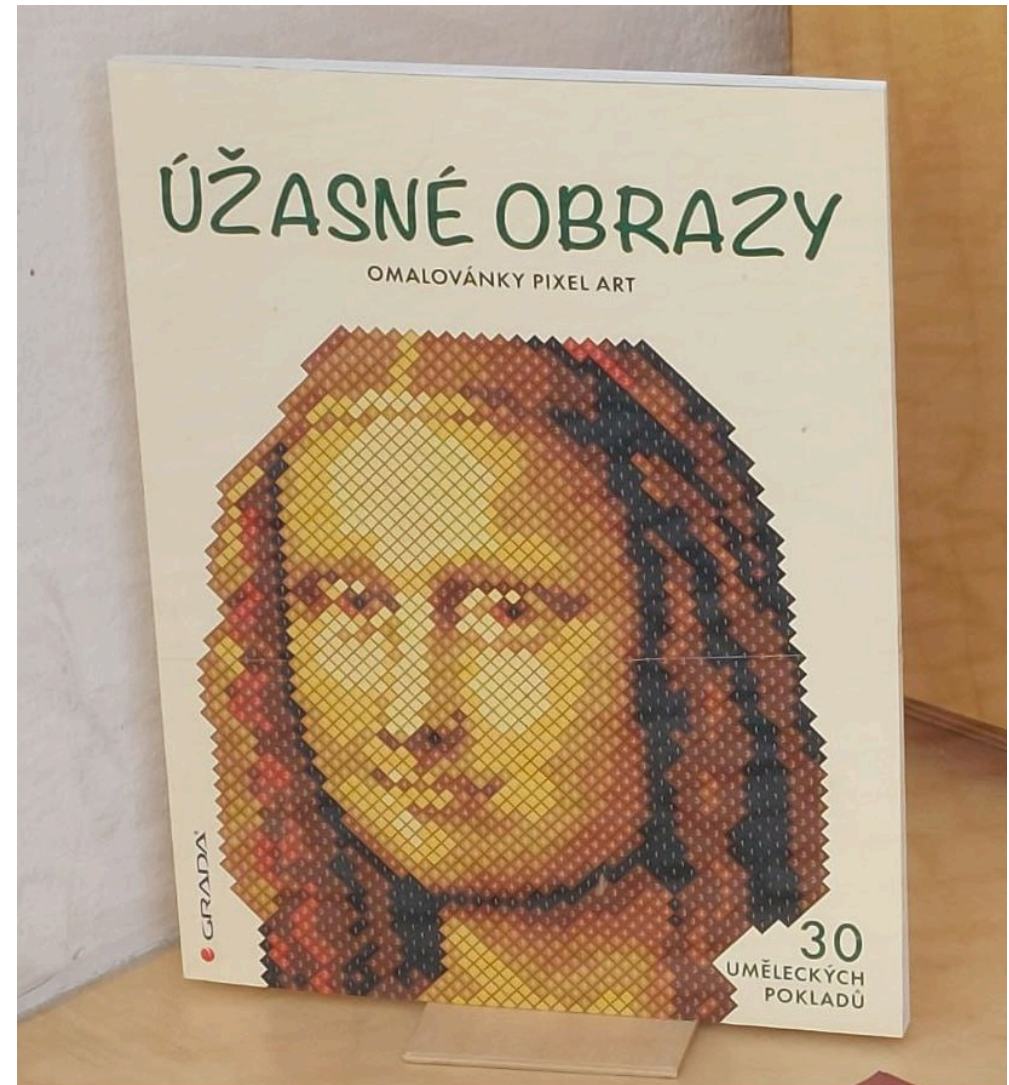
- Старый проект, кодовое название «Планета Железяка»
 - Поддержки нет
 - Исходных кодов нет
 - Кого спросить тоже нет (заказная разработка)
- Горят сроки и ждать исправлений нет времени
- Другая "жиза"

ДОКЕРОВУХА



ГОД 2018

Что в образе твоём?



**Про твои образы Docker
написали книгу**

"Жирный" базовый образ...

```
$ cat /etc/os-release  
  
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"  
NAME="Debian GNU/Linux"  
VERSION_ID="11"  
VERSION="11 (bullseye)"  
VERSION_CODENAME=bullseye  
. . .
```

... с кучей ненужных исполняемых файлов,...

```
$ curl --version
. . .
Release-Date: 2023-02-20, security patched: 7.88.1-10+deb12u5

$ gosu --help
. . .
gosu version: 1.14 (go1.19.8 on linux/amd64; gc)

$ nc --version
Ncat: Version 7.93 ( https://nmap.org/ncat )
```

... "зашитыми" кредитами,...

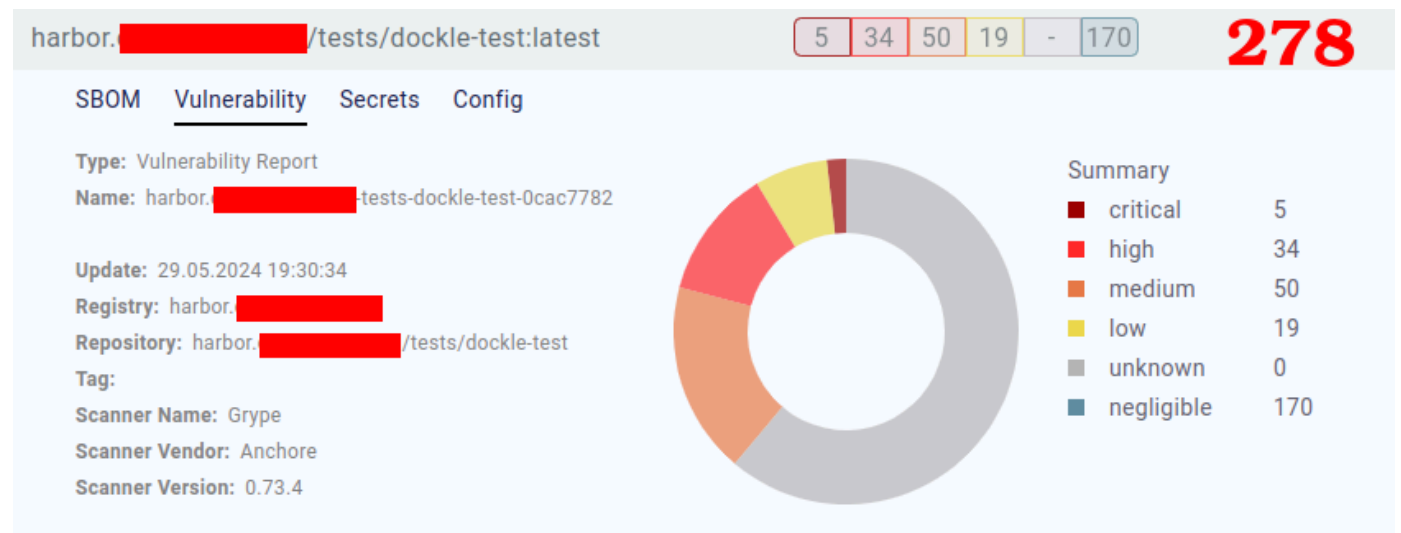
```
$ env | grep TOKEN
ENV JWT_TOKEN="eyJhbGciOiJI...V_adQssw5c"

$ grep "AWS" run.sh
export AWS_SECRET_ACCESS_KEY="Qk6dDZ3Ev6x...5CNwYmM1s9+vf0E="
export AWS_ACCESS_KEY_ID="AKIAATC...F27LK"

$ grep "TOKEN" test.txt
PULUMI_TOKEN_TEST="pul-afde034feeebc098...deadbeef"
```


... log4shell и 100500 CVE,...

```
$ ./log4dhell.sh web-service-app.jar
. . .
Log4J version : 2.14.1
[!] Inspection finished - Class found!
```



... и вообще всё неправильно

```
$ dockle $IMAGE_NAME | grep "FATAL\|WARN"
```

```
FATAL - CIS-DI-0007: Do not use update instructions alone in the Dockerfile
FATAL - CIS-DI-0009: Use COPY instead of ADD in Dockerfile
FATAL - CIS-DI-0010: Do not store credential in environment variables/files
FATAL - DKL-DI-0001: Avoid sudo command
FATAL - DKL-DI-0002: Avoid sensitive directory mounting
FATAL - DKL-DI-0005: Clear apt-get caches
FATAL - DKL-LI-0001: Avoid empty password
FATAL - DKL-LI-0002: Be unique UID/GROUP
WARN - CIS-DI-0001: Create a user for the container
WARN - DKL-DI-0003: Avoid "apt-get dist-upgrade"
WARN - DKL-DI-0006: Avoid latest tag
```

К чему это ведёт?

LotL-атака (Living off the Land)

Атака, при которой используются легитимные программы и функции для выполнения вредоносных действий в целевой системе



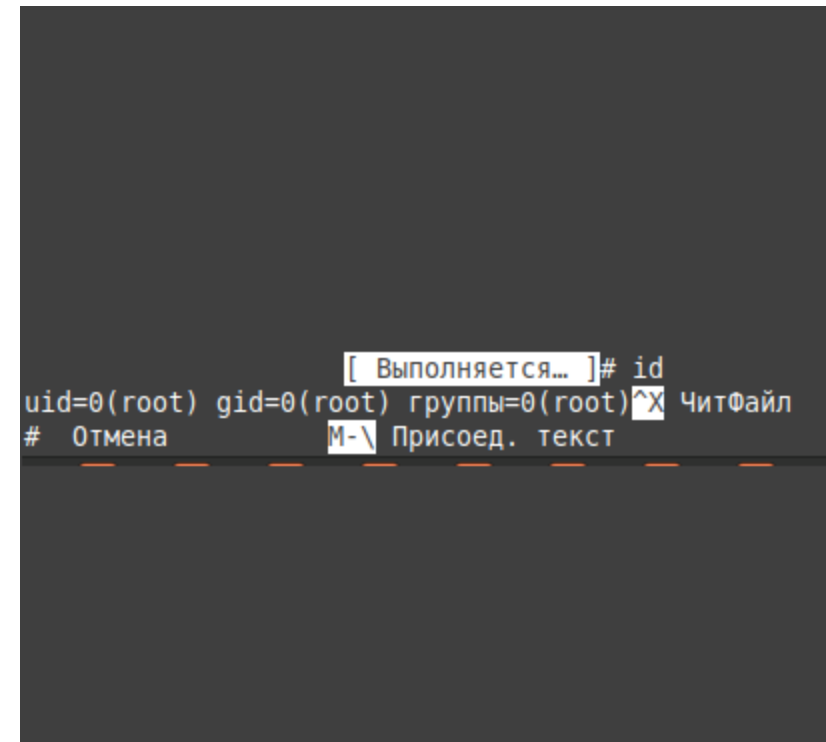
LotL-атака

- Сигнатуры не помогут
- Вредоносных файлов нет
- Shell может быть легитимным
- Расследование и атрибуция атак сложны



LotL-атака && GTFOBins && nano

```
sudo nano  
^R^X  
reset; sh 1>&0 2>&0
```



```
[ Выполняется... ]# id  
uid=0(root) gid=0(root) группы=0(root)^X ЧитФайл  
# Отмена M-\ Присоед. текст
```

LotL-атака && GTFOBins && zip

```
TF=$(mktemp -u)
sudo zip $TF /etc/hosts -T -TT 'sh #'
sudo rm $TF
```

```
$ sudo zip $TF /etc/hosts -T -TT 'sh #'
  adding: etc/hosts (deflated 56%)
# id
uid=0(root) gid=0(root) группы=0(root)
# █
```

Известные уязвимости

```
$ trivy image $IMAGE_NAME
```

```
┌───┐  
Total: 321 (UNKNOWN: 0, LOW: 199, MEDIUM: 70,  
HIGH: 49, CRITICAL: 3)
```

CAPTAIN
OBVIOUS



Секреты, токены, ключи...

```
$ trivy image --scanners secret $IMAGE_NAME

/file.txt (secrets)
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 1, CRITICAL: 0)
HIGH: Pulumi (pulumi-api-token)

/run.sh (secrets)
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 1)
CRITICAL: AWS (aws-access-key-id)

$IMAGE_NAME (secrets)
Total: 2 (UNKNOWN: 0, LOW: 0, MEDIUM: 2, HIGH: 0, CRITICAL: 0)
MEDIUM: JWT (jwt-token)

. . .
```



Лирическое отступление №1: спрятанный секрет

```
$ cat test.txt
PULUMI_TOKEN_TEST="pul-afde034feeebc098ebc4000036e00000deadbeef"

$ trivy fs test.txt
$ cp test.txt a.txt
$ trivy fs a.txt

/a.txt (secrets)

Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 1, CRITICAL: 0)
HIGH: Pulumì (pulumì-api-token)

Pulumì API token
/a.txt:1

1 [ PULUMI_TOKEN_TEST="*****" ]
```

Лирическое отступление №1: разгадка спрятанного секрета

```
{
  ID:          "tests",
  Description: "Avoid test files and paths",
  Path:        MustCompile(`(^test|\/test|-test|_test|\.test)`),
},
{
  ID:          "examples",
  Description: "Avoid example files and paths",
  Path:        MustCompile(`example`),
  Regex:       MustCompile("(?i)example"),
},
```

Оставшийся ключ (1)

```
. . .  
COPY id_rsa /root/.ssh/  
RUN . . . && rm -rf /root/.ssh/id_rsa  
CMD ["/run.sh"]  
. . .
```

```
$ docker history $IMAGE_NAME  
IMAGE          CREATED          CREATED BY  
c274f07a418f   20 minutes ago  CMD ["/run.sh"]  
<missing>      20 minutes ago  RUN ... && rm -rf...  
<missing>      20 minutes ago  COPY id_rsa /root/.ssh/ # buildkit  
. . .
```

Оставшийся ключ (2)

```
$ docker save $IMAGE_NAME | tar -x -C .

$ find blobs/sha256/ -type f -exec file {} \; | grep "tar"
blobs/sha256/9b47fc. . .fde2bd: POSIX tar archive
. . .
blobs/sha256/6e9c5e. . .98398c: POSIX tar archive

$ tar xvf 9b47fc. . .fde2bd
root/
root/.ssh/
root/.ssh/id_rsa

$ cat root/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
. . .
-----END OPENSSH PRIVATE KEY-----
```

Мисконфиги (1)

FATAL - **CIS-DI-0009**: Use COPY instead of ADD in Dockerfile

```
* Use COPY : ADD credentials.json /app/credentials.json # buildkit
```

FATAL - **CIS-DI-0010**: Do not store credential in environment variables/files

```
* Suspicious filename found : app/credentials.json
* Suspicious filename found : .aws/credentials
* Suspicious ENV key found : MYSQL_PASSWORD on ENV MYSQL_PASSWORD=password
* Suspicious ENV key found : JWT_TOKEN on ENV
JWT_TOKEN=eyJhbGc. . .Qssw5c
```

WARN - **DKL-DI-0006**: Avoid latest tag

```
* Avoid 'latest' tag
```

Лирическое отступление №2

```
func (a CredentialAssessor) RequiredFiles() []string {
    return []string{
        "credentials.json",
        "credential.json",
        "credentials",
        "credential",
        "password.txt",
        "id_rsa",
        "id_dsa",
        "id_ecdsa",
        "id_ed25519",
        "secret_token.rb",
        "carrierwave.rb",
        "omniauth.rb",
        "settings.py",
        "database.yml",
        "credentials.xml",
    }
}
```

Мисконфиги (2)

FATAL - **DKL-DI-0001**: Avoid sudo command

```
* Avoid sudo in container :  
RUN /bin/sh -c apt-get install -y git sudo curl gosu ncat # buildkit  
RUN /bin/sh -c sudo cat /etc/passwd # buildkit
```

WARN - **CIS-DI-0001**: Create a user for the container

```
* Last user should not be root
```

INFO - **CIS-DI-0008**: Confirm safety of setuid/setgid files

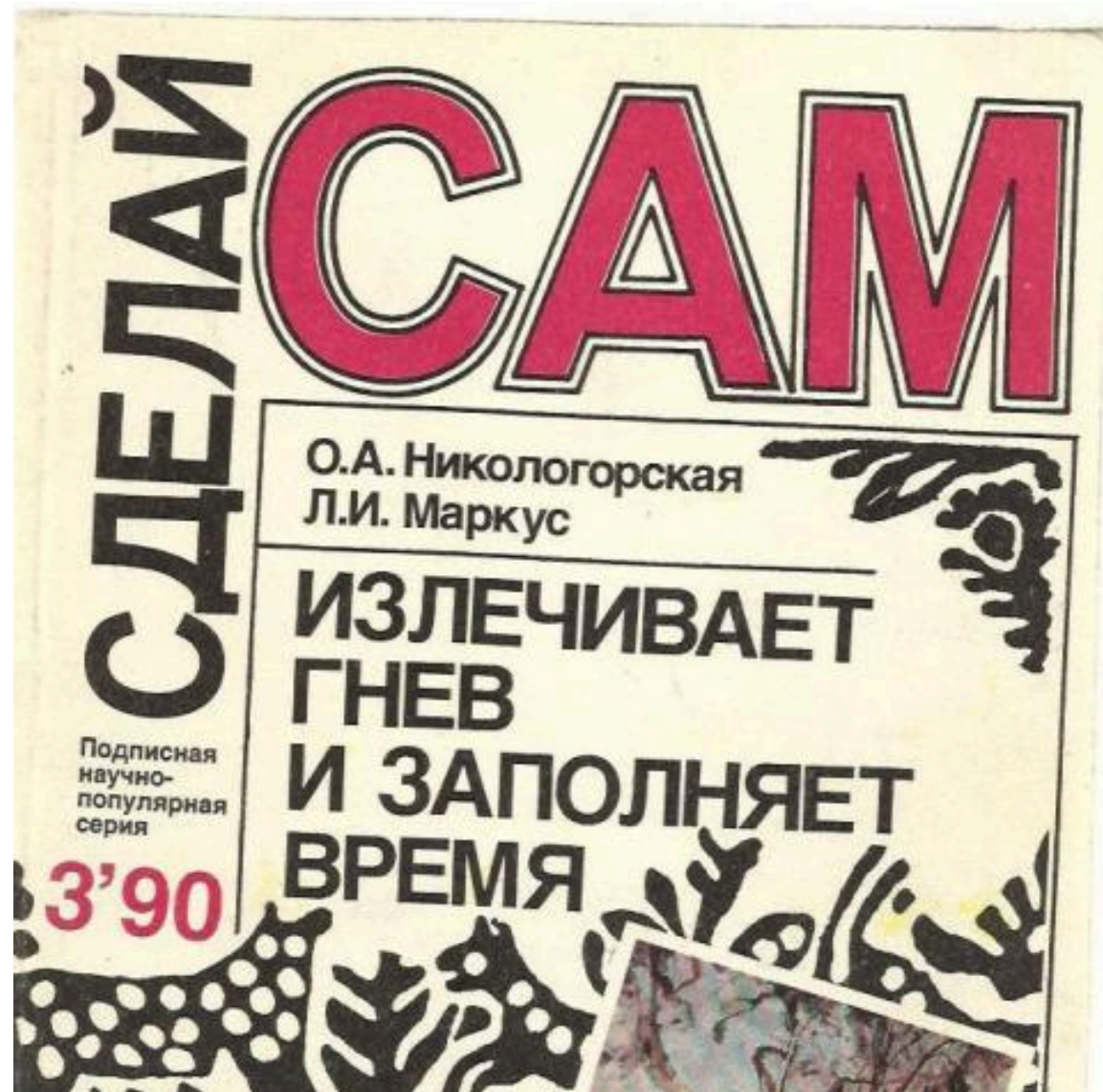
```
* setgid file: grwxr-xr-x usr/bin/expiry  
* setuid file: urwxr-xr-x usr/bin/sudo  
. . .
```


Всё плохо

- На уровне Dockerfile
 - Мiskonфигурации
- На уровне слоев образа
 - Неисполняемые файлы
 - Ключи, токены, пароли, сертификаты, ...
 - Исполняемые файлы
 - LotL-атака, GTFOBins
 - Вредоносные программы
 - setuid/setgid файлы
- На уровне кода
 - Известные уязвимости (CVE)



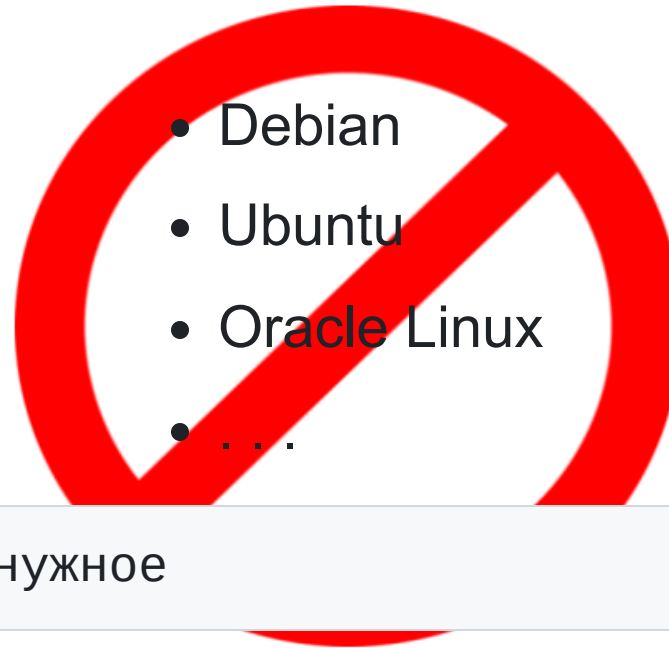
Исправляем
ситуацию



Замена базового образа

- gcr.io ([distroless](#))
- [Chainguard](#)
- [SIGHUP](#)
- ...

- Debian
- Ubuntu
- Oracle Linux
- ...



Выбрасываем ненужное

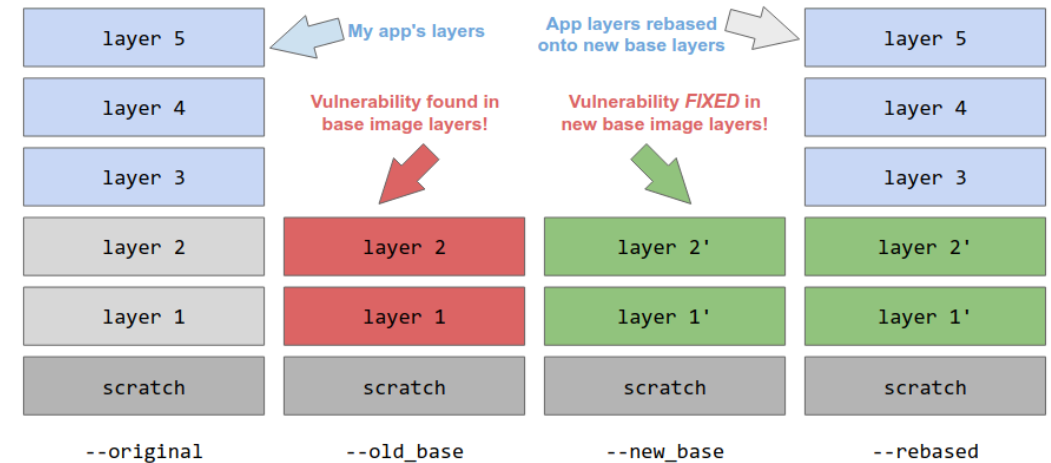
Меняем слои

crane

```
$ docker run -it --rm $IMG_NAME:latest \
    head -1 /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"

$ ./crane rebase $IMG_NAME:latest \
    --old_base=debian:latest \
    --new_base=debian:12 \
    --tag=$IMG_NAME:rebased

$ docker run -it --rm $IMG_NAME:rebased \
    head -1 /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
```



Удаляем промежуточное

docker-squash

```
$ docker history $IMAGE_NAME
IMAGE          CREATED          CREATED BY
c274f07a418f   20 minutes ago  CMD ["/run.sh"]
<missing>     20 minutes ago  RUN ...      && rm -rf...
<missing>     20 minutes ago  COPY id_rsa /root/.ssh/ # buildkit
. . .
```

```
$ docker-squash -f 2 -t $IMAGE_NAME:squashed $IMAGE_NAME

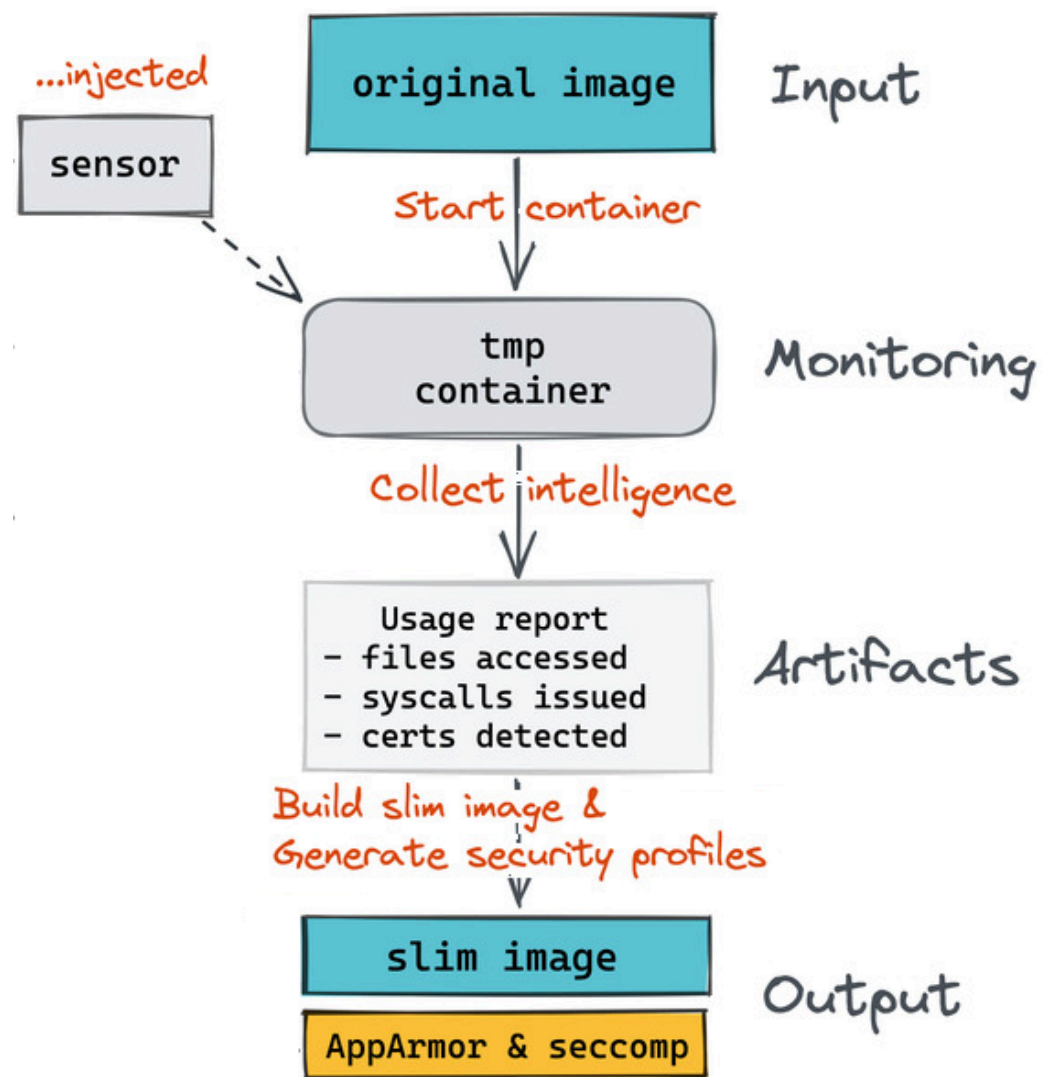
$ docker save $IMAGE_NAME:squashed | tar -x -C .

$ tar xvf f4152d. . .750c19 | grep "id_rsa"
```

Минифицируем образ

mint (ex-DockerSlim)

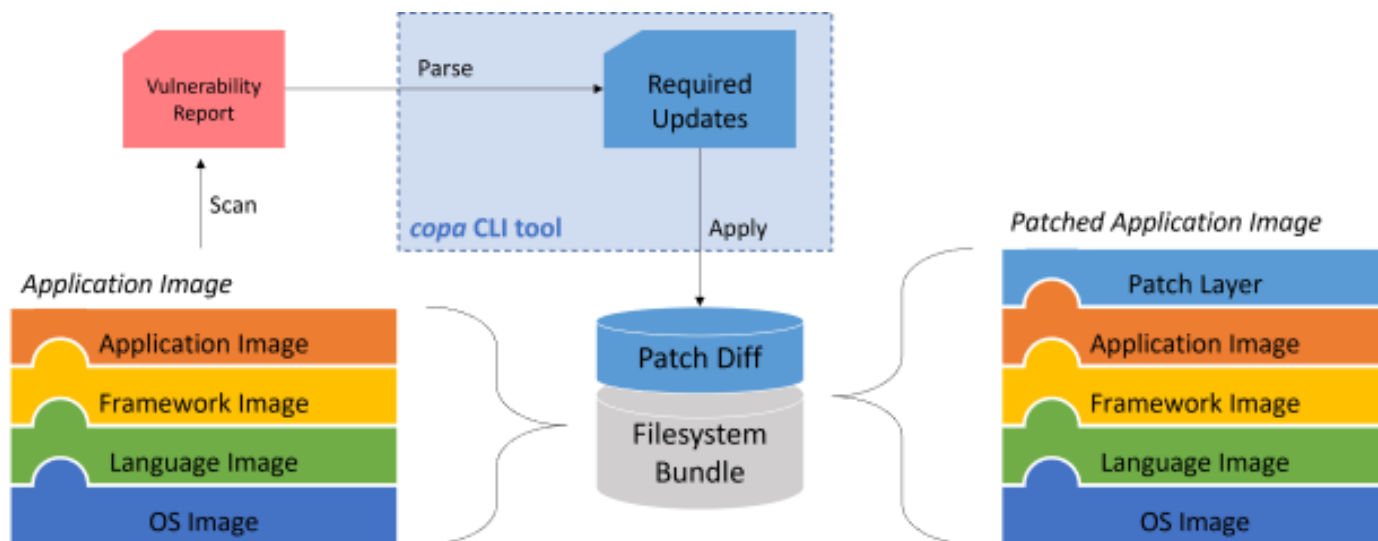
```
$ docker images -a
REPOSITORY                                SIZE
harbor.server/tests/bad-image.slim        169MB
harbor.server/tests/bad-image             832MB
```



Модифицируем слои

copacetic

CLI tool for directly patching container images using reports from vulnerability scanners



Модифицируем слои

corasctic

```
$ trivy image --vuln-type os --ignore-unfixed --scanners vuln $IMAGE_NAME  
Total: 8 (UNKNOWN: 0, LOW: 0, MEDIUM: 6, HIGH: 2, CRITICAL: 0)
```

Library	Vulnerability	Severity	Installed Version	Fixed Version
libc-bin	CVE-2024-33599	HIGH	2.36-9+deb12u6	2.36-9+deb12u7
	CVE-2024-33600	MEDIUM		
	CVE-2024-33601			
CVE-2024-33602				
libc6	CVE-2024-33599	HIGH		
	CVE-2024-33600	MEDIUM		
	CVE-2024-33601			
	CVE-2024-33602			

Модифицируем слои

copacetic

```
$ trivy image --vuln-type os --ignore-unfixed -f json  
-o bad-image.latest.json $IMAGE_NAME  
  
$ sudo ./copa patch -i $IMAGE_NAME -r bad-image.latest.json  
-t bad-image-patched  
  
$ trivy image --vuln-type os --ignore-unfixed --scanners vuln  
$IMAGE_NAME:bad-image-patched  
  
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```






Недостатки Сора

- Новый образ больше
- Только от trivy
- Далеко не всё патчит
- Поддерживает только

apk и apt

```
mount / from exec sh -c apt install --no-install-recommends --allow-change-held-packages -y libxml2 python3.7 gzip imagemagick-6-  
common libc-bin libgmp-dev libkrb5support0 libtiff5 dpkg-dev libbsd0 libcurl4-openssl-dev libgnutls-openssl27 libkadm5clnt-mit11 li  
bx11-dev libcairo-gobject2 libncurses6 libpixmap-1-0 libsqlite3-0 libssl-dev libexpat1-dev libwebp-dev libx11-6 libxslt1.1 libcairo-scri  
pt-interpreter2 libexif-dev libjpeg62-turbo liblzma5 libpq-dev librsvg2-dev perl gnupg-utils libldap-common libmagickwand-6-headers  
libncurses5-dev libzstd1 libopenjp2-7 git-man gnupg gpg libc6 libldap-2.4-2 libopenexr-dev libpq5 perl-base libc-dev-bin libfridi0 lib  
glib2.0-dev-bin libgnutlsxx28 libkadm5srv-mit11 libopenexr23 libcairo2 libfreetype6 unzip zlib1g libnettle6 openssl libaprutil1 libcair  
o2-dev libgnutls-dane0 libkrb5-3 libksba8 libmagickcore-6-headers libmaxminddb0 libp11-kit-dev libsasl2-2 icu-devtools libdjvulibre2  
1 libsqlite3-dev imagemagick libgmpxx4ldbl libp11-kit0 libpython3.7-minimal libxslt1-dev gpgsm imagemagick-6.q16 libglib2.0-0 libl  
zma-dev libmagickcore-6.q16-6 libperl5.28 gpg-agent ncurses-bin libmariadb-dev-compat libss2 git libcurl4 libde265-0 libdpkg-perl li  
bicu63 libk5crypto3 libssl1.1 libudev1 libwebp6 libx11-data libxml2-dev tzdata libmagickcore-6.q16-6-extra libmagickwand-dev libtas  
n1-6 libgssapi-krb5-2 libmagickcore-6.q16-dev libbz2-dev libc6-dev libcurl3-gnutls libext2fs2 libglib2.0-bin libgnutls30 libpcre2-8-0 li  
bmagickcore-dev libmagickwand-6.q16-6 libmagickwand-6.q16-dev ncurses-base libmariadb-dev python3.7-minimal apt dpkg libapt-  
pkg5.0 libexpat1 libidn2-0 libjpeg-dev bzip2 libcom-err2 libfreetype6-dev librsvg2-2 libsasl2-modules-db comerr-dev dirmngr libkrb5-  
dev libmagickcore-6-arch-config libncursesw5-dev libpython3.7-stdlib e2fsprogs gpgv libbz2-1.0 libgmp10 libkdb5-9 libpixmap-1-dev  
librsvg2-common sudo gpgconf libdjvulibre-dev libexif12 libgrypt20 libglib2.0-dev libncurses-dev xz-utils libgnutls28-dev libsvn1 lib  
tiffxx5 libtinfo6 libwebpmux3 linux-libc-dev libdjvulibre-text libtiff-dev libunbound8 curl gnupg-1.0n gpg-wks-server libglib2.0-data lib  
hogweed4 libtasn1-6-dev perl-modules-5.28 krb5-multidev libgssrpc4 libsystemd0 nettle-dev libidn2-dev zlib1g-dev liblz4-1 libmaxmi  
nddb-dev libncursesw6 libwebpdemux2 mariadb-common subversion gir1.2-rsvg-2.0 gpg-wks-client libicu-dev libjpeg62-turbo-dev lib  
mariadb3 libopenjp2-7-dev && apt clean -y
```

Закрываем проблемы в "статике"

- На уровне Dockerfile
 -  Модификация слоёв
- На уровне слоев образа
 - Неисполняемые файлы
 -  docker-squash
 - Исполняемые файлы
 -  Минималистичный базовый образ
- На уровне кода
 - Известные уязвимости (CVE)
 -  Минималистичный базовый образ
 -  Последние версии библиотек

На уровне Kubernetes

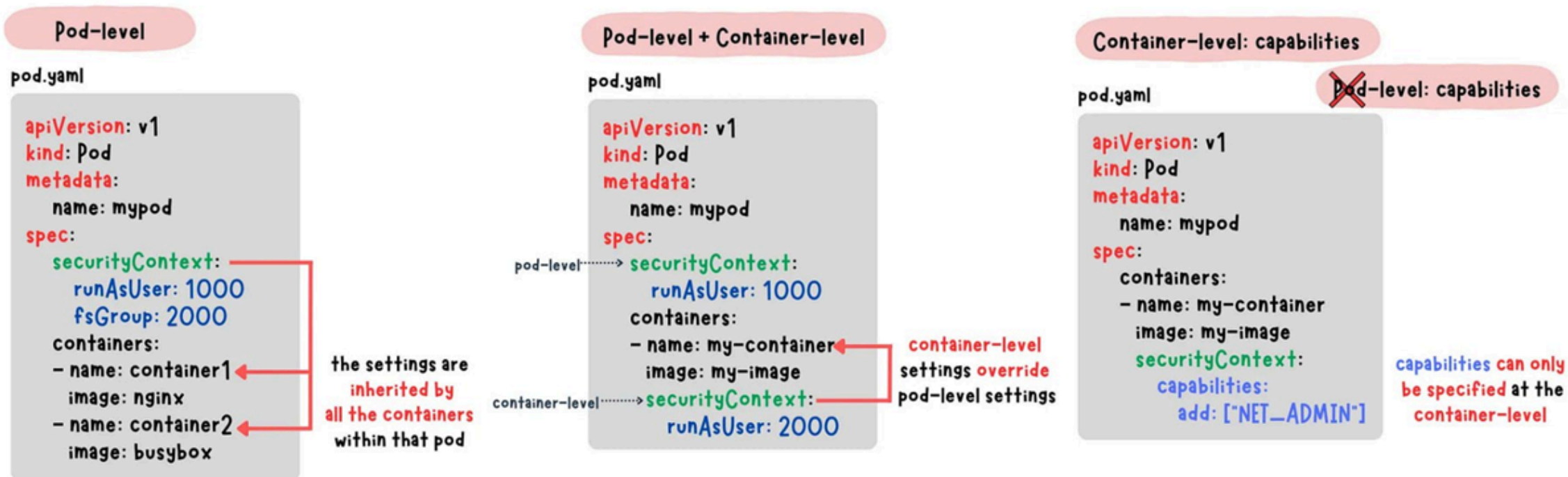
SecurityContext holds security configuration that will be applied to a container

PodSecurityContext holds pod-level security attributes and common container settings.

<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.30/#securitycontext-v1-core>



SecurityContext vs PodSecurityContext



privileged

Run container in privileged mode.
Processes in privileged containers are essentially equivalent to root on the host.
Defaults to **false**.

```
containers:
```

- name: my-best-app
image: my-best-container
securityContext:
privileged: false

CAPTAIN
OBVIOUS



privileged

```
$ docker run -it --rm $IMAGE_NAME sh -c 'apt install -y libcap2-bin;  
                                         capsh --print | grep Current:'
```

```
. . .  
Current: 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_audit_write,cap_setfcap=ep
```

```
$ docker run -it --rm --privileged $IMAGE_NAME sh -c 'apt install -y libcap2-bin;  
                                         capsh --print | grep Current:'
```

```
. . .  
Current: =ep
```

If the capability set show =ep, that means it has all the capabilities from the bounding set

allowPrivilegeEscalation

AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. Default: **true**

`allowPrivilegeEscalation` выставляем в `false`, что бы запретить процессам повышать привелегии через `sudo`, `setuid`

```
containers:  
  - name: my-best-container  
    securityContext:  
      allowPrivilegeEscalation: false
```


capabilities

Adds and removes POSIX capabilities from running containers

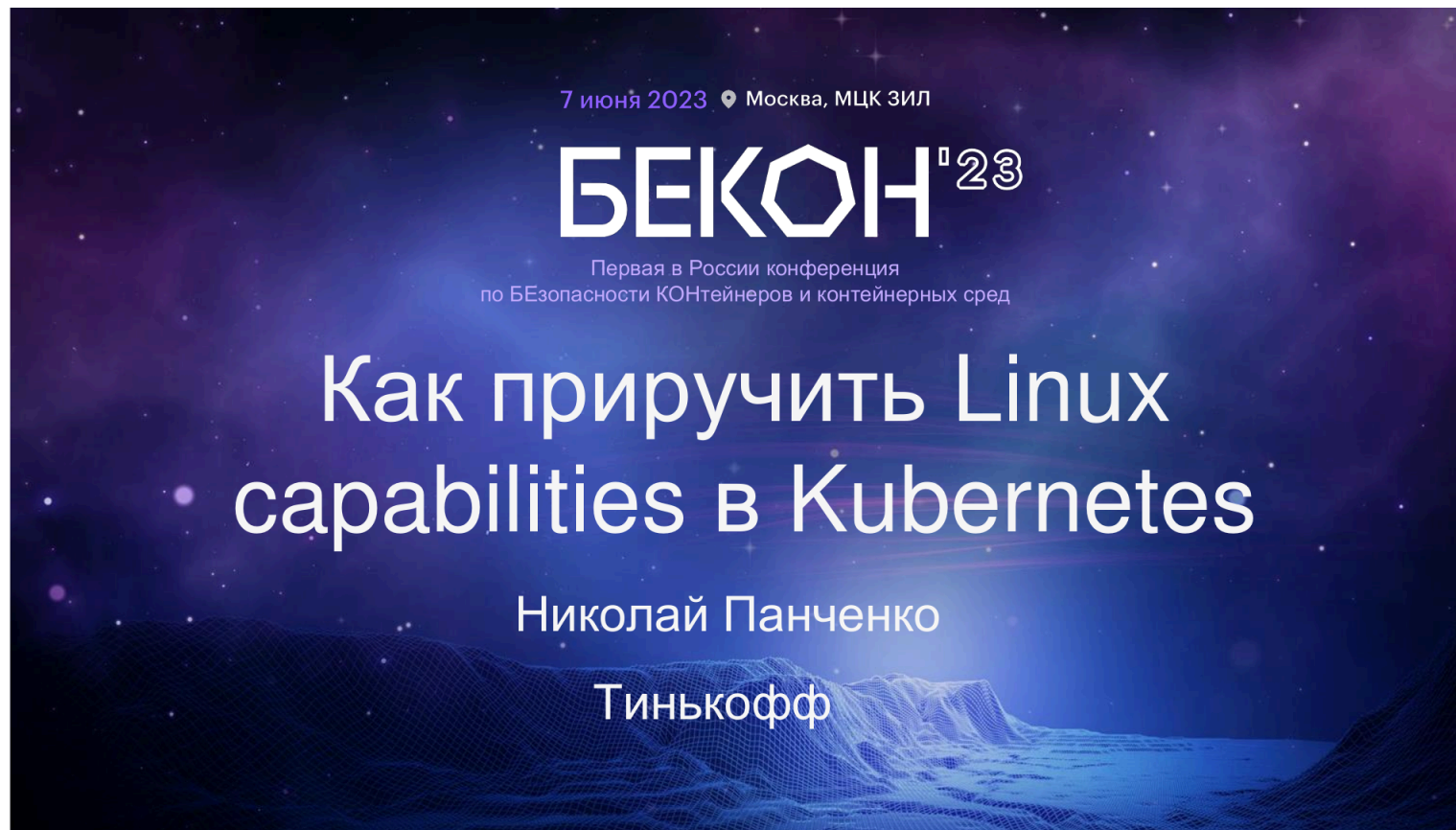
- CAP_CHOWN
- CAP_DAC_OVERRIDE
- CAP_DAC_READ_SEARCH
- CAP_SETUID
- CAP_SETGID
- CAP_NET_RAW
- CAP_SYS_ADMIN
- CAP_SYS_PTRACE
- CAP_SYS_MODULE
- CAP_FOWNER
- CAP_SETFCAP
- . . .



capabilities practice

```
containers:  
- name: payment  
  image: nginx  
  securityContext:  
    capabilities:  
      drop:  
        - all  
      add:  
        - NET_BIND_SERVICE
```

Linux capabilities и Kubernetes



seccompProfile

SeccompProfile defines a pod/container's seccomp profile settings

```
apiVersion: v1
kind: Pod
. . .
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: file.json
  containers:
  - name: my-best-app
    image: my-best-container:latest
    securityContext:
      allowPrivilegeEscalation: false
```

```
{
  "defaultAction": "SCMP_ACT_ERRNO",
  "syscalls": [
    {
      "names": [
        "read",
        "write",
        "exit",
        . . .
      ],
      "action": "SCMP_ACT_ALLOW"
    }
  ]
}
```

Default Seccomp

SeccompDefault enables the use of RuntimeDefault as the default seccomp profile for all workloads. Default: **false**

RuntimeDefault отличаются

- Docker/Moby
- cri-o
- containerd

strace

```
$ strace -c ping
ping: usage error: Требуется указать адрес назначения
% time      seconds  usecs/call   calls   errors syscall
-----
 39,06      0,000382      13         28        mmap
 12,17      0,000119      10         11        1 openat
 11,86      0,000116      14          8        mprotect
  5,01      0,000049       4         10        fstat
  4,60      0,000045       3         12        close
  4,50      0,000044       6          7        read
  4,40      0,000043       5          8        4 prctl
  4,29      0,000042      42          1        munmap
  3,17      0,000031       5          6        pread64
  2,25      0,000022       7          3        3 access
  2,04      0,000020       5          4        write
  1,64      0,000016       5          3        brk
  1,53      0,000015       5          3        capget
  1,02      0,000010       5          2        getuid
  1,02      0,000010      10          1        capset
  0,72      0,000007       7          1        setuid
  0,51      0,000005       2          2        1 arch_prctl
  0,20      0,000002       2          1        geteuid
  0,00      0,000000       0          1        execve
  0,00      0,000000       0          3        fcntl
-----
100.00      0,000978                115        9 total
```

appArmorProfile

AppArmor is a Mandatory Access Control (MAC) security system. Its aim is to provide an easy-to-use security system targeted at securing individual applications by defining and restricting what resources and application has access to and can share

appArmorProfile is the AppArmor options to use by this container

```
apiVersion: v1
kind: Pod
metadata:
  name: test-apparmor
spec:
  securityContext:
    appArmorProfile:
      type: Localhost
      localhostProfile: my-app
  containers:
    . . .
```

appArmorProfile practice

```
NODES=$(kubectl get nodes -o name)

for NODE in ${NODES[*]}; do ssh $NODE 'sudo apparmor_parser -q <<EOF
#include <tunables/global>

profile my-app flags=(attach_disconnected) {
  #include <abstractions/base>

  /{,usr/}bin/myapp mixr,
  /etc/modules.conf r,
}
EOF'
done
```


AppArmor и Kubernetes



Summary of features

Security feature	Description	How to use	Example
seccomp	Restrict individual kernel calls in the userspace. Reduces the likelihood that a vulnerability that uses a restricted syscall would compromise the system.	Specify a loaded seccomp profile in the Pod or container specification to apply its constraints to the processes in the Pod.	Reject the <code>unshare</code> syscall, which was used in CVE-2022-0185 .
AppArmor	Restrict program access to specific resources. Reduces the attack surface of the program. Improves audit logging.	Specify a loaded AppArmor profile in the container specification.	Restrict a read-only program from writing to any file path in the system.
SELinux	Restrict access to resources such as files, applications, ports, and processes using labels and security policies.	Specify access restrictions for specific labels. Tag processes with those labels to enforce the access restrictions related to the label.	Restrict a container from accessing files outside its own filesystem.

readOnlyRootFilesystem

Whether this container has a read-only root filesystem. Default is **false**.

```
containers:  
  image: my-best-container:latest  
  name: my-best-app  
  securityContext:  
    readOnlyRootFilesystem: true
```



runAsNonRoot

Indicates that the container must run as a non-root user

```
$ docker ps -a -q | xargs docker inspect -f  
  '{{.Id}}: User {{if .Config.User}}{{.Config.User}}{{else}}root{{end}}'  
  
be8e6c. . .04e12e: User root
```

```
containers:  
  - name: my-best-app  
    securityContext:  
      runAsNonRoot: true
```

```
Error: container has runAsNonRoot and image will run as root  
(pod: "my-best-pod. . .", . . .)
```

runAsUser/runAsGroup

The UID to run the entrypoint of the container process

Defaults to user specified in image metadata if unspecified

The GID to run the entrypoint of the container process

Uses runtime default if unset

spec:

securityContext:

runAsUser: 1000

runAsGroup: 1000

Exception in thread "main" java.lang.IllegalStateException:

Unable to create the directory [/tomcat.8888] to use as the base directory

at org.apache.catalina.startup.Tomcat.initBaseDir(Tomcat.java:847)

at org.apache.catalina.startup.Tomcat.getServer(Tomcat.java:613)

at org.apache.catalina.startup.Tomcat.getEngine(Tomcat.java:586)

at org.apache.catalina.startup.Tomcat.getHost(Tomcat.java:570)

at launch.Main.main(Main.java:20)

fix runAsUser

Возможно иногда решить
через:

- `initContainer`
- `supplementalGroups`



hostUsers

Use the host's user namespace.

Defaults to **true**.

Setting false is useful for mitigating container breakout vulnerabilities even allowing users to run their containers as root without actually having root privileges on the host

UserNamespacesSupport practice

```
$ #
$ # Let's create a pod with user namespaces enabled now
$ # And run the exact same steps to see if we can exploit the bug
$ cat ./cve-pod-usersn.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cve-pod-usersn
spec:
  hostUsers: false
  terminationGracePeriodSeconds: 1
  containers:
  - name: container1
    image: debian
    command: ["sleep", "infinity"]
$ # It is the same pod as before, but with the hostUsers=false field
$ # This enables user namespaces
$ sleep 10
$ kubectl apply -f ./cve-pod-usersn.yaml
pod/cve-pod-usersn created
$ # Let's wait until the pod is Running
$ kubectl get pod cve-pod-usersn
NAME          READY   STATUS    RESTARTS   AGE
cve-pod-usersn 1/1     Running   0           7s
$ # With user namespaces, we map root in the container to a different (unprivileged) user on the host
$ # Let's see from the host as which user it is running
$ ps faux | grep "sleep infinity" | grep -v grep
3937140+  77550  0.1  0.0  2484  1152 ?        Ss   15:41   0:00  \_ /usr/bin/sleep infinity
```


UserNamespacesSupport (deep dive)



NetworkPolicy

NetworkPolicies are an application-centric construct which allow you to specify how a pod is allowed to communicate with various network "entities" over the network.



NetworkPolicy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: pod-two
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
      - podSelector:
          matchLabels:
            app: pod-one
  egress:
    - to:
      - podSelector:
          matchLabels:
            app: pod-one
```

```
kubectl exec -it my-best-pod -- curl -I -L evil.site
curl: (6) Couldn't resolve host 'evil.site'
```



Дополнительные возможности (Network)

- Calico
- Cilium
- Flannel
- Kube-router
- Antrea
- Multus
- Weave 
- . . .


Дополнительные возможности (PolicyEngine)

- Kyverno
- OPA Gatekeeper
- Kubewarden
- jsPolicy
- KubeLatte
- . . .

Нестандартное применение Kyverno (SafeCode 2024 Spring)



Закрываем проблемы в "динамике"

- На уровне Dockerfile
 - ✗
- На уровне слоев образа
 - Неисполняемые файлы
 - ✗
 - Исполняемые файлы
 -  privileged, alloPrivelegeEscalation, capabilities, appArmorProfile, seccompProfile, readOnlyRootFilesystem, NetworkPolicy
 -  runAsUser/runAsGroup, runAsNonRoot, hostUsers (setuid/setgid-файлы)
- На уровне кода
 - Известные уязвимости (CVE)
 -  privileged, alloPrivelegeEscalation, capabilities, appArmorProfile, seccompProfile, readOnlyRootFilesystem, NetworkPolicy

Чо, всё?!

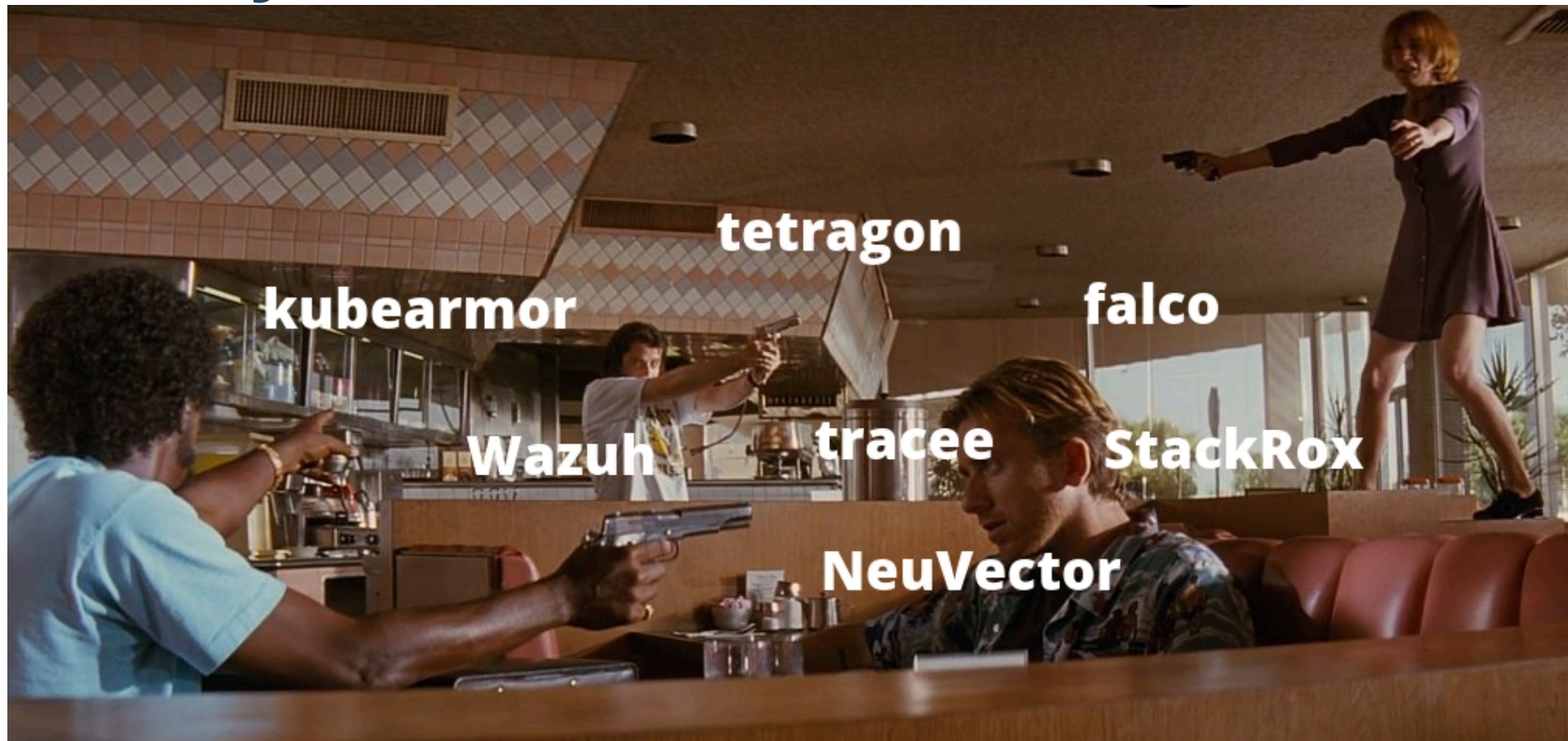
Чо, всё?!

Нет, не всё



Надо за всем этим смотреть

Community <3



CNCF Landscape

Security & Compliance

 CNCf GRADUATED	 CNCf GRADUATED	 CNCf GRADUATED	 CNCf GRADUATED	 CNCf INCUBATING	 CNCf INCUBATING	 CNCf INCUBATING	 CNCf INCUBATING														
																					
																					
																					
																					
																					

Falco (example)

```
- rule: Contact K8S API Server From Container
  desc: >
  Detect attempts to communicate with the K8S
  API Server from container by non-profiled users
  . . .
  condition: >
  evt.type=connect and evt.dir=<
  and (fd.typechar=4 or fd.typechar=6)
  and container
  and k8s_api_server
  and not k8s_containers
  and not user_known_contact_k8s_api_server_activities
  output: Unexpected connection to K8s
  API Server from container (connection=%fd.name)
  . . .
  priority: NOTICE
  tags: [maturity_stable, container, network,
  k8s, mitre_discovery, T1565]
```

DETAILS

JSON

Time

2024-09-23T22:34:45.792270199Z

Source

syscall

Hostname

gimme-b5799d85-2

Priority

Notice

Rule

Terminal shell in container

Output

22:34:45.792270199: Notice A shell was spawned in a container
proc_exepath=/usr/bin/bash parent=containerd-shim command=k
container_image_tag=latest container_name=[REDACTED] k8s_ns=defa

Fields

container.id	07aefd692ef8	container.image.repository	[REDACTED]	contain		
k8s.pod.name	[REDACTED]	proc.cmdline	bash	proc.exepath	/usr/bin/bash	pro

Tags

T1059

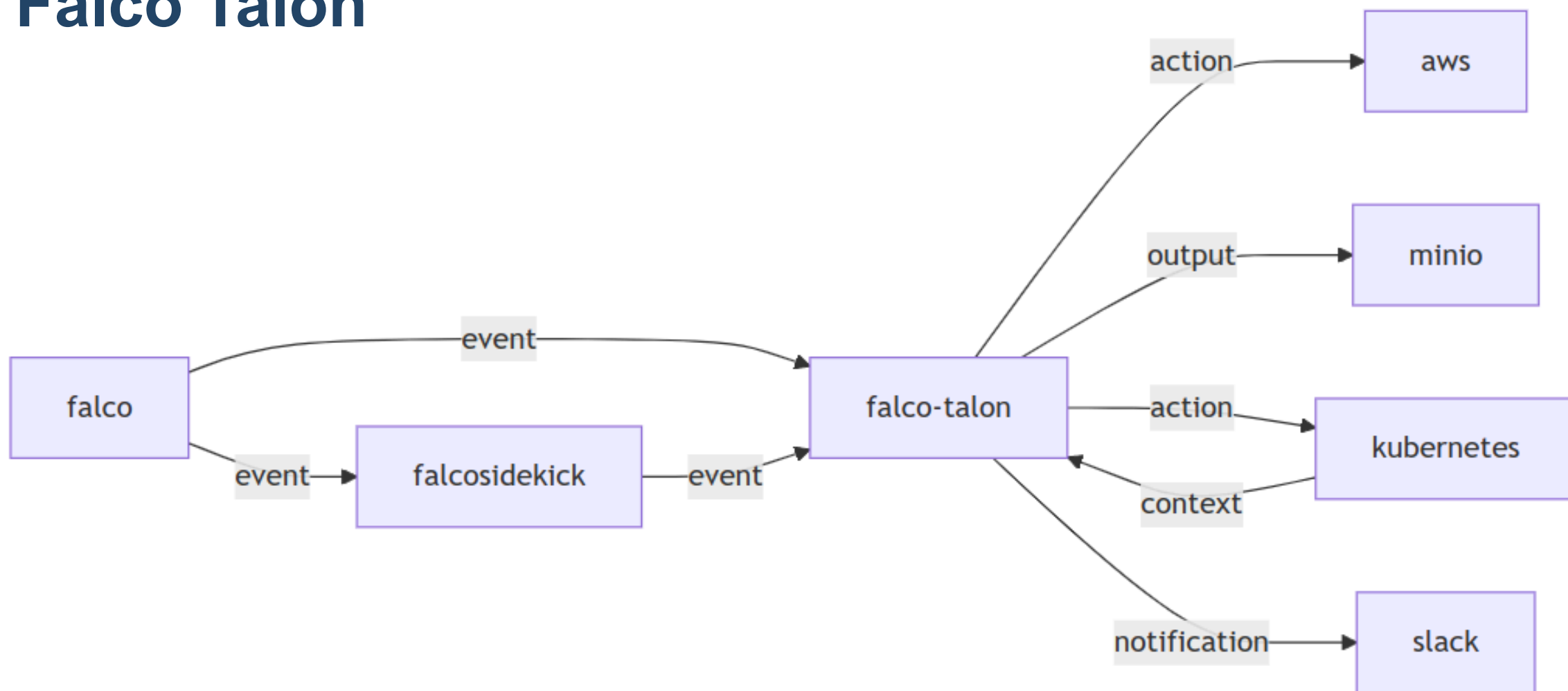
container

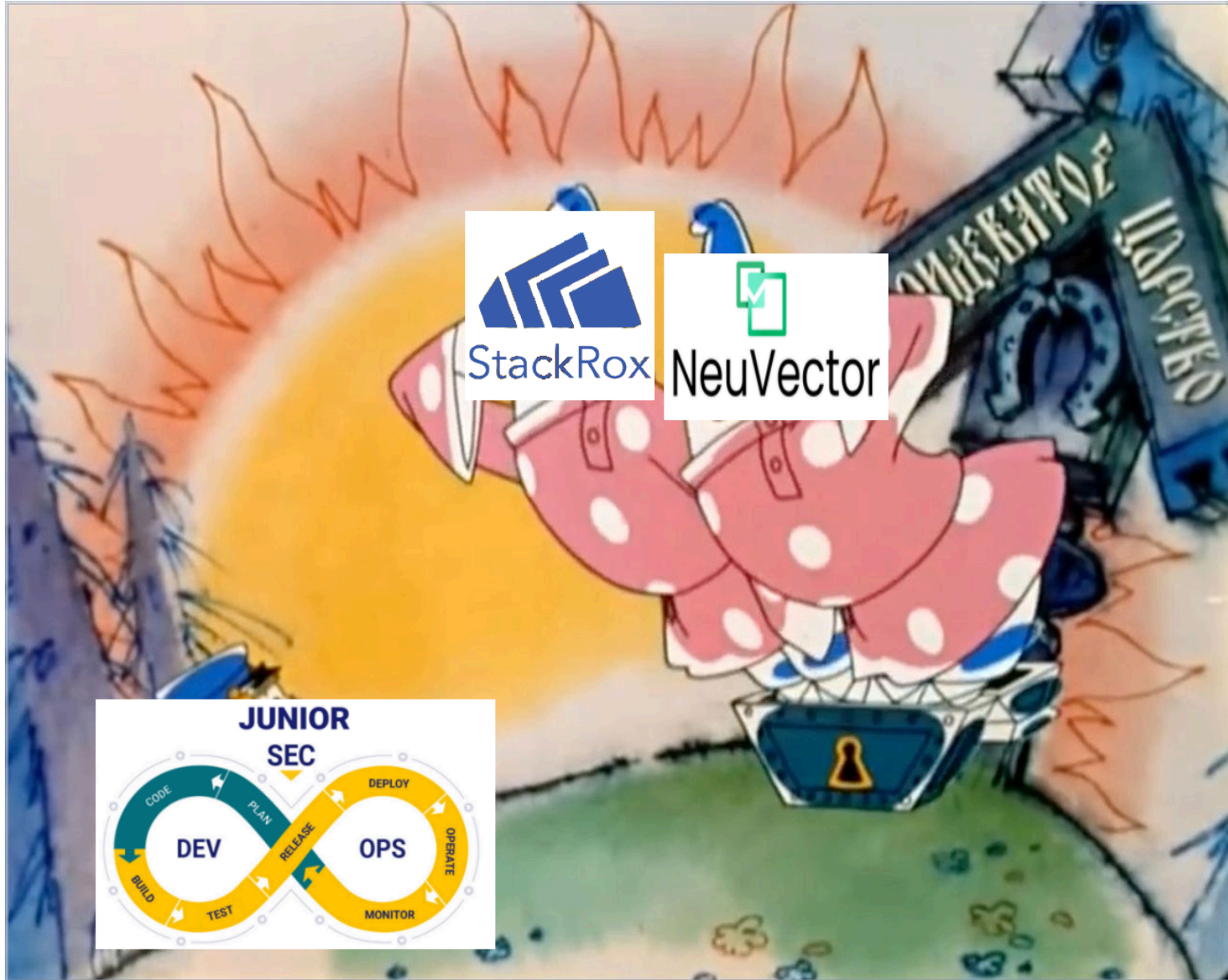
maturity_stable

mitre_execution

shell

Falco Talon





StackRox

- В 2021 году был поглощён RedHat (Red Hat Advanced Cluster Security for Kubernetes)
- Open source
- Features
 - Vulnerability
 - Network Graph
 - Compliance (⚠ устарел)
 - Events
 - . . .



DevOps

**Защищаем Kubernetes
при помощи StackRox – дешево,
сердито, эффективно?**

StackRox (examples)

CVEs by severity

🔴 2 Critical
🟡 15 Moderate
🟠 8 Important
🟢 48 Low


CVEs by status

🔧 2 vulnerabilities with avail
— 71 vulnerabilities without




73 results found

<input type="checkbox"/>	CVE ↑	CVE severity ↓	CVE status
>	<input type="checkbox"/> CVE-2023-45853	🔴 Critical	— Not fixable
>	<input type="checkbox"/> CVE-2023-6879	🔴 Critical	— Not fixable
>	<input type="checkbox"/> CVE-2023-2953	🟠 Important	— Not fixable
>	<input type="checkbox"/> CVE-2023-31484	🟠 Important	— Not fixable

Event Timeline

1 Policy Violation	27 Process Activities	0 Restarts / Terminations	 View Graph
-----------------------	--------------------------	------------------------------	---

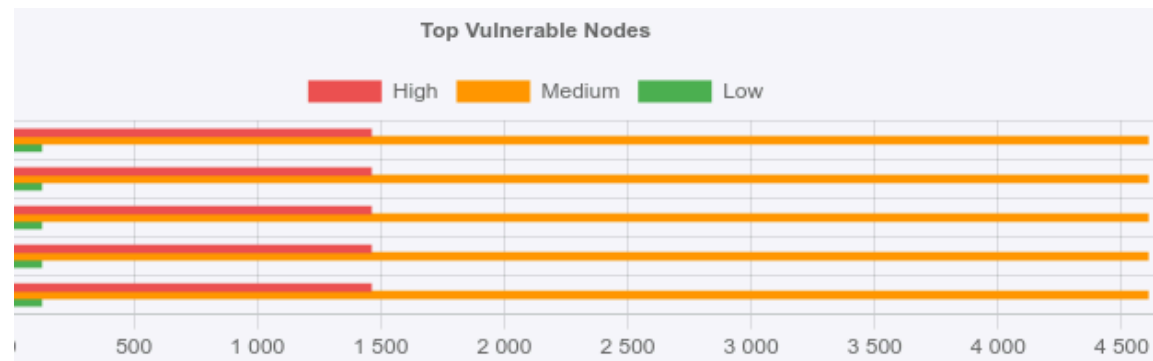
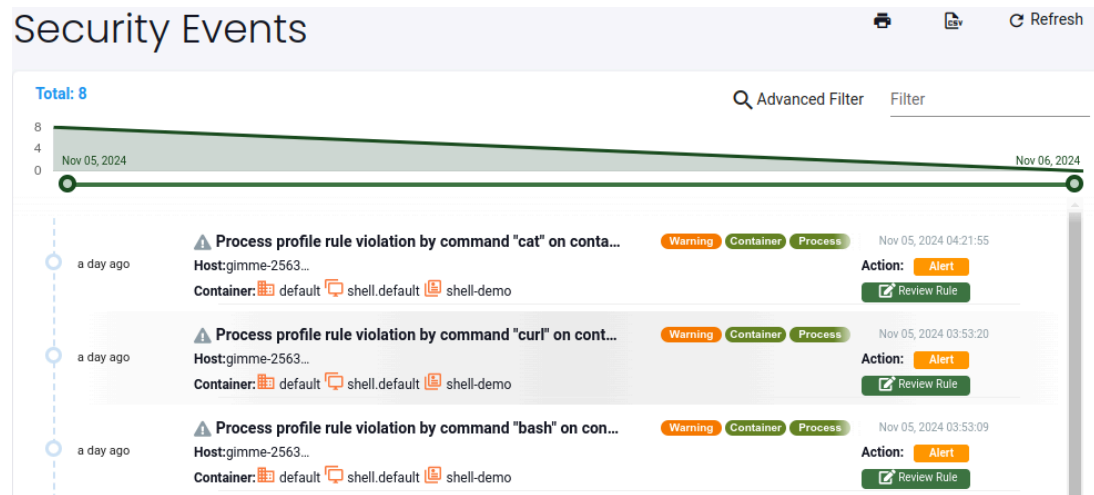
History of Running Processes

/nc64 in container nginx		
/usr/bin/chmod in container nginx		
/usr/bin/ls in container nginx		
/bin/bash in container nginx		

NeuVector

- В 2021 году был поглощён SUSE (NeuVector Prime)
- Open source
- Features
 - Vulnerability
 - Network Graph
 - Compliance (⚠ устарел)
 - Events
 - . . .

NeuVector (examples)



Score(V3) [dropdown] [chart icon] [Advanced Filter]

Node	Score(V3)	Published at	Impact
344	7.6	May 23, 2024	1 5
269	8.8	May 22, 2024	1 5
203	5.9	May 22, 2024	1 5
163	5.5	May 22, 2024	1 5
722	8.8	May 07, 2024	1 5
547	6.5	Feb 01, 2024	1 5
377	5.3	Dec 19, 2023	1 5
112	4.7	Nov 14, 2023	1 5
455	7.8	Oct 05, 2023	1 5

```
#!/bin/bash
```

```
while true;  
do echo -e "HTTP/1.1 200 OK\n\n{ \"status\": { \"code\": 200, \"status\": \"OK\" }}" \  
  | nc -l -k -p 8082 -q 1;  
done
```



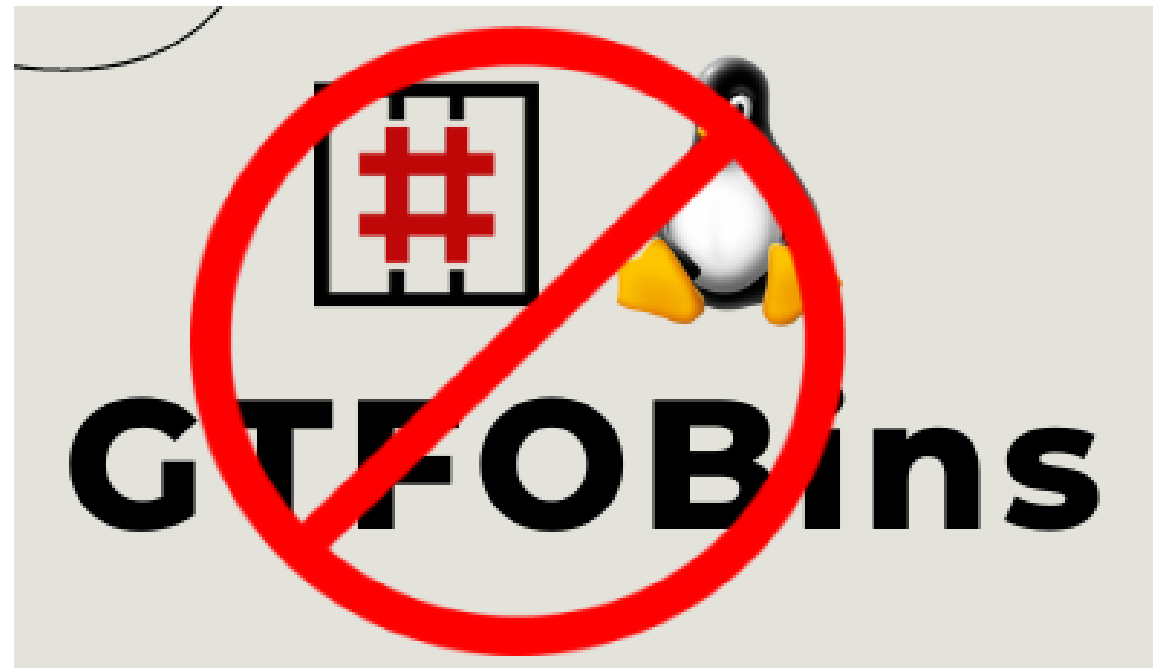
Резюмируя

**Решаю
вопросик**



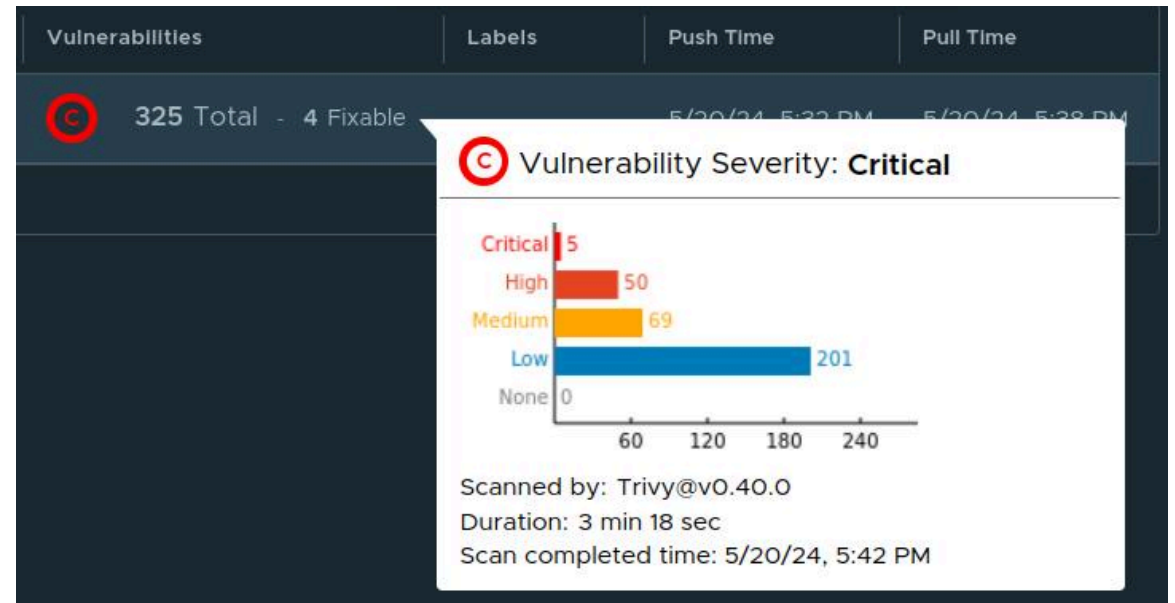
Миллионы исполняемых файлов (LotL-атаки)

- Модифицируем слои
- Выбрасываем ненужное
- Патчим
- Сетевые политики
- Параметры Kubernetes
- Делаем ReadOnly файловую систему
- Привилегии, capabilities,...
- ... ВСЁ, что бы решить проблемы в будущем



Миллионы CVE

- Модифицируем слои
- Выбрасываем ненужное
- Патчим
- Сетевые политики
- Параметры Kubernetes
- Делаем ReadOnly файловую систему
- Привилегии, capabilities,...
- ... ВСЁ, что бы решить проблемы в будущем



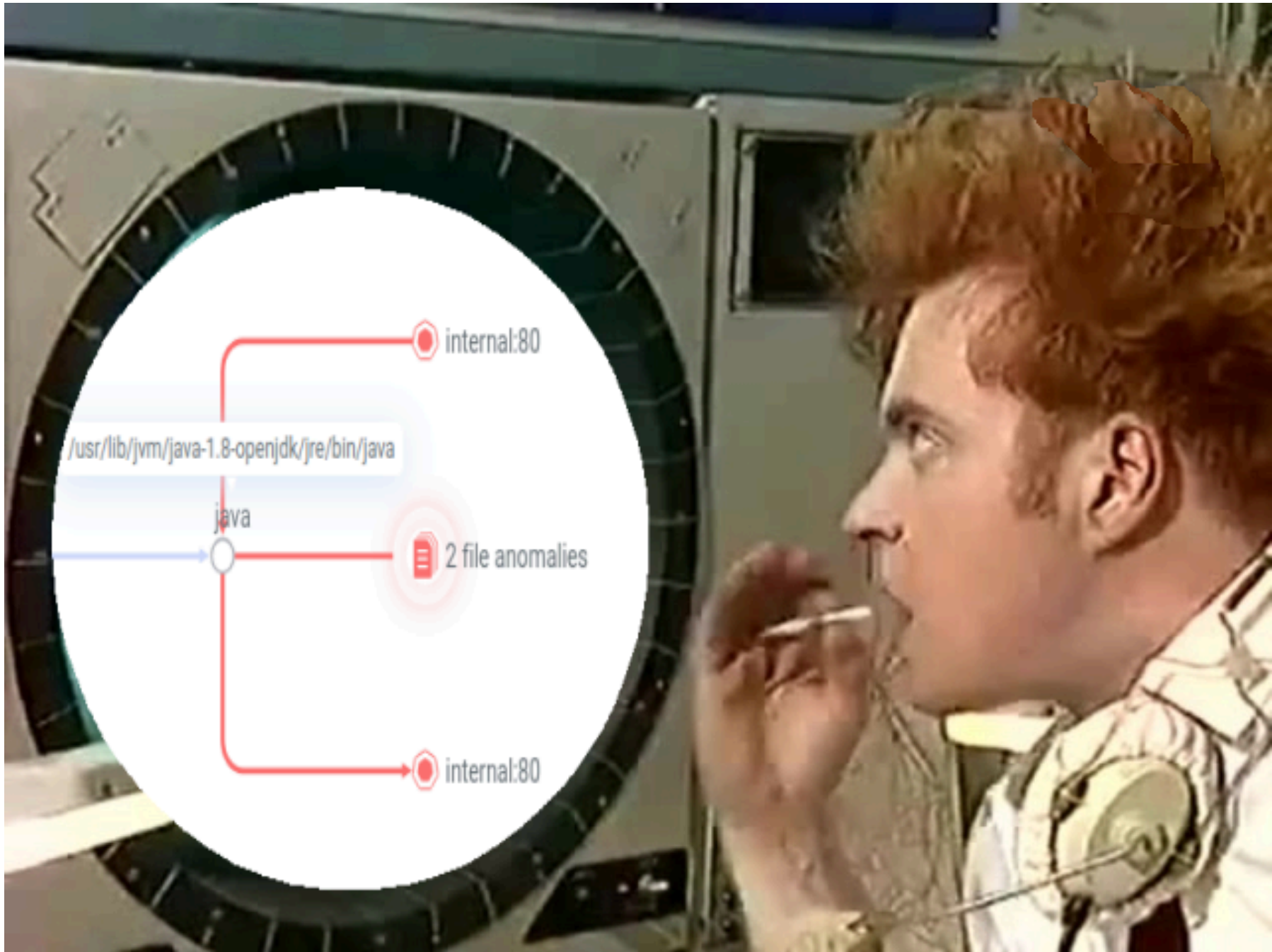
Чувствительная информация

- Модифицируем слои
 - docker-squash
 - crane
 - mint



Пользователи, SUID...

- "Статика"
 - Модифицируем слои
 - Выбрасываем ненужное
 - Патчим
- Параметры накладывающие ограничения на пользователя
 - runAsNonRoot
 - runAsUser/runAsGroup
 - allowPrivilegeEscalation
 - UserNamespacesSupport



Вместо заключения

- Контейнеры и Kubernetes позволяют митигировать разные проблемы и хорошо харденить среду выполнения
- Всегда можно настроить мониторинг "под себя"
- Никогда не сдавайся!



Полезные ссылки

- [Docker security best practices](#)
- [CIS Docker Benchmark](#)
- [Configure a Security Context for a Pod or Container](#)
- [Как собрать контейнер и не вооружить хакера](#)
- [10 Kubernetes Security Context, которые необходимо понимать](#)

Спасибо за внимание!

site: luntry.ru

tg: [@rusdacent](https://t.me/@rusdacent)

channel: [@tech_b0lt_Genona](https://t.me/@tech_b0lt_Genona)