5 июня 2024 Москва, LOFT HALL#2

БЕКОН'24

Конференция по БЕзопасности
КОНтейнеров и контейнерных сред

Латаем огрехи в образах приложений с помощью Kubernetes

Анатолий Карпенко

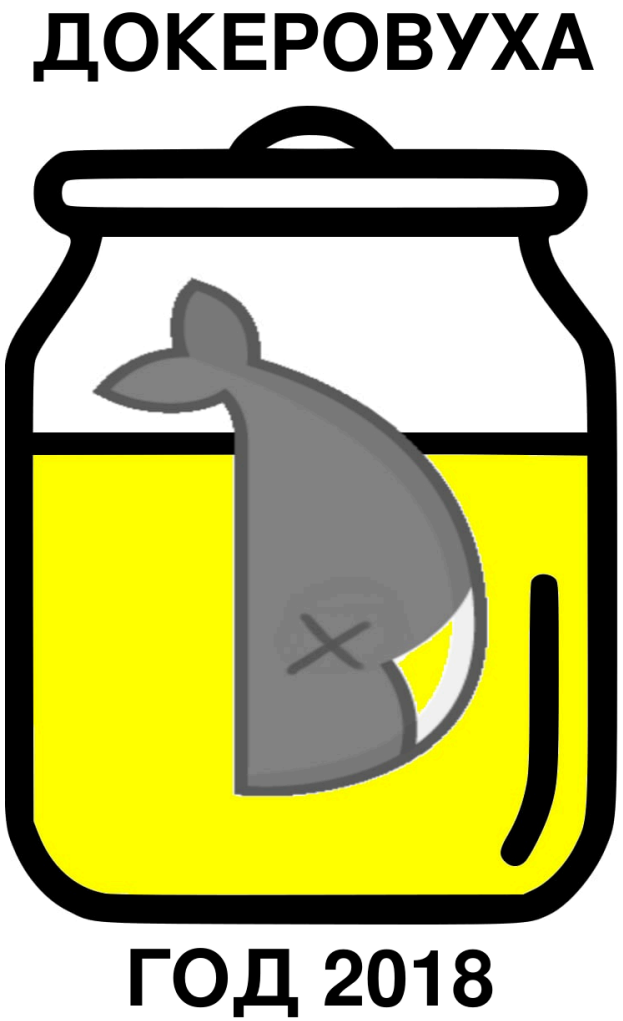
Luntry

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"



Что в образе твоём?

"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,...

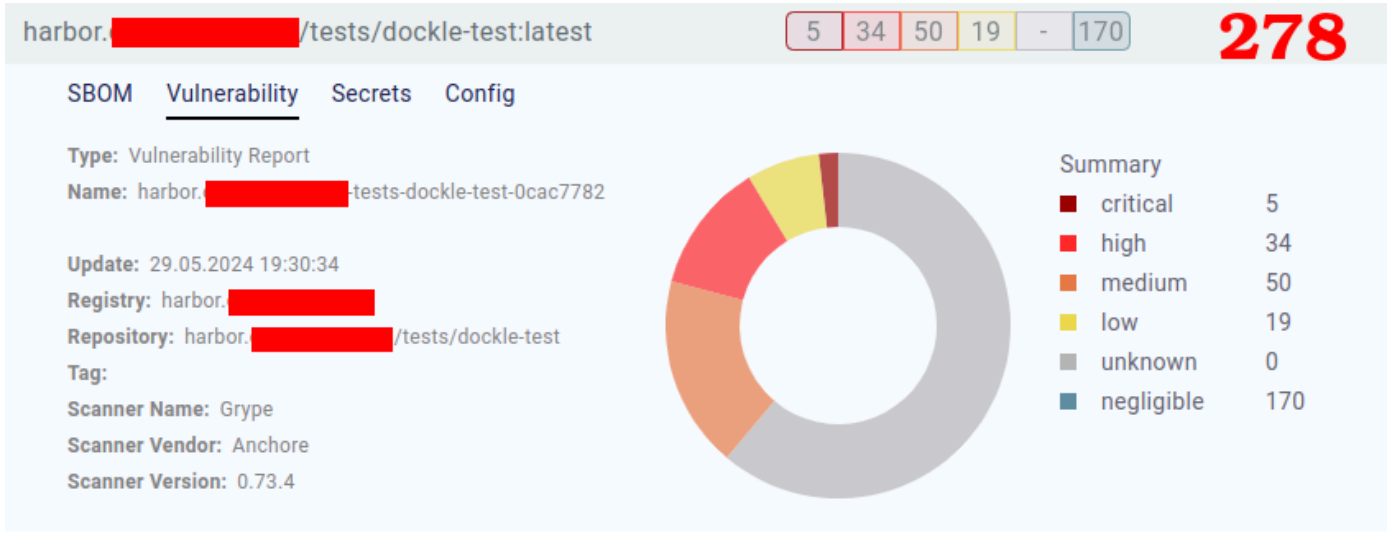
```
$ curl --version
. . .
Release-Date: 2023-02-20, security patched: 7.88.1-10+deb12u5

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

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

... "зашитыми" кредами,...

```
$ env | grep TOKEN
ENV JWT_TOKEN="eyJhbGciOiJ. . .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 используется легитимные программы и функции для выполнения вредоносных действий в целевой систем

LotL-атака

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

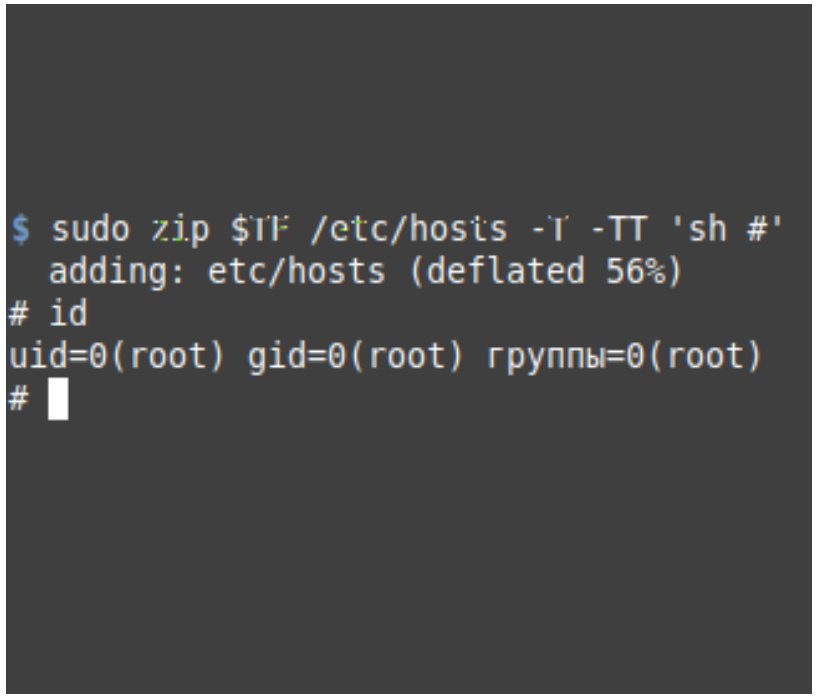
LotL-атака && GTFOBins && nano

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

https://gtfobins.github.io/gtfobins/nano/

LotL-атака && GTFOBins && zip

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



https://gtfobins.github.io/gtfobins/zip/

Известные уязвимости

```
$ trivy image $IMAGE_NAME
. . .
Total: 321 (UNKNOWN: 0, LOW: 199, MEDIUM: 70,
                    HIGH: 49, CRITICAL: 3)
```

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: Pulumi (pulumi-api-token)

Pulumi API token
 /a.txt:1

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

Лирическое отступление №1: "зацепка"

afdesk commented on Mar 14, 2023 — Contributor

> shouldn't the file `test.txt` containing the `AWS_ACCESS_KEY_ID` still be detected by `trivy image IMAGE --debug --scanners secret`?

yes, sure. you're right. Trivy has to detect secrets inside files, and Trivy does it)
but Trivy skips test data...

if you change a file name (eg. to `tst.txt`), Trivy will catch it:

```
FROM nginx:alpine
```

https://github.com/aquasecurity/trivy/issues/3778#issuecomment-1467988262

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"),
},
```

https://github.com/aquasecurity/trivy/blob/main/pkg/fanal/secret/builtin-allow-rules.go

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)

```
# DKL-DI-0006
FROM debian:latest

RUN mkdir /tmp/sys
# DKL-DI-0002
VOLUME /sys /tmp/sys
VOLUME /usr
# CIS-DI-0007
# DKL-DI-0005
RUN apt-get update
# DKL-DI-0001
# DKL-DI-0005
RUN apt-get install -y git sudo curl gosu ncat default-jre
# DKL-LI-0001
RUN useradd nopasswd -p ""
RUN chmod u+s /etc/shadow
RUN chmod g+s /etc/passwd
# CIS-DI-0009
# CIS-DI-0010
ADD credentials.json /app/credentials.json
COPY ./test.txt test.txt
COPY file.txt file.txt
RUN chmod u+s file.txt
RUN chmod g+s file.txt
```

Мисконфиги (2)

```
RUN curl -LO https://. . .xmrig-6.21.3-linux-static-x64.tar.gz \
    && tar xvf xmrig-6.21.3-linux-static-x64.tar.gz \
    && rm -f xmrig-6.21.3-linux-static-x64.tar.gz
# CIS-DI-0010
ENV MYSQL_PASSWD password
RUN chmod g-s file.txt
# DKL-DI-0001
RUN sudo cat /etc/passwd
# DKL-DI-0003
RUN apt-get dist-upgrade -s
# CIS-DI-0010
# DKL-LI-0003:
COPY .aws/ .aws/
# CIS-DI-0010
ENV JWT_TOKEN="eyJh. . .sw5c"
# DKL-LI-0002
COPY passwd /etc/passwd
COPY ./web-service-app.jar .
COPY ./run.sh .
COPY id_rsa /root/.ssh/
RUN mkdir /tmp/test/ \
    && rm -rf /root/.ssh/id_rsa

CMD ["/run.sh"]
```

Мисконфиги (3)

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_PASSWD on ENV MYSQL_PASSWD=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",
        }
}
```

https://github.com/goodwithtech/dockle/blob/master/pkg/assessor/credential/credential.go

Мисконфиги (3)

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
. . .
```

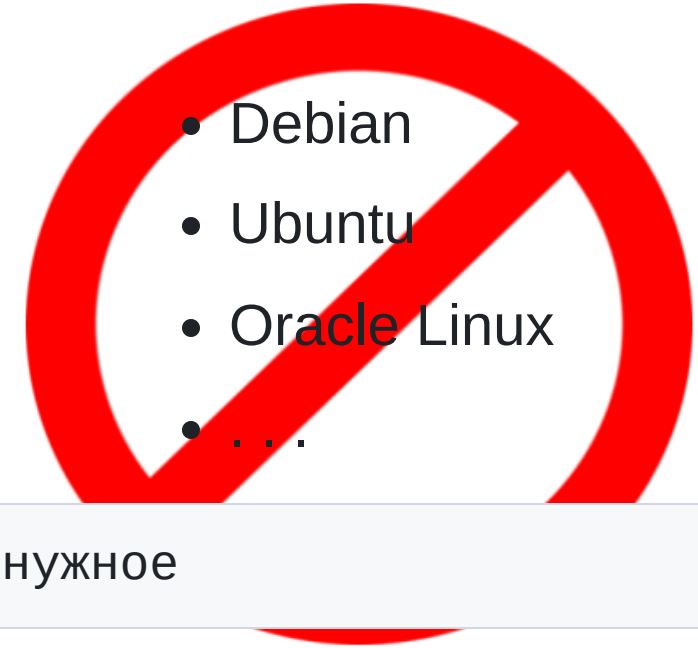
Всё плохо

- На уровне Dockerfile
  - Мисконфигурации
- 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
- . . .



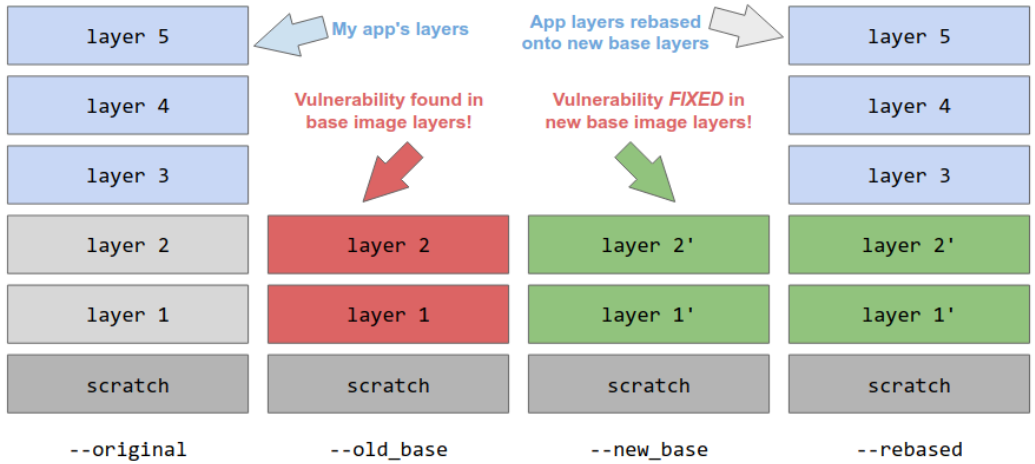
- Debian
- Ubuntu
- Oracle Linux
- . . .

```
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)"
```



https://github.com/google/go-containerregistry/blob/main/cmd/crane/rebase.md

Удаляем промежуточное

docker-squash

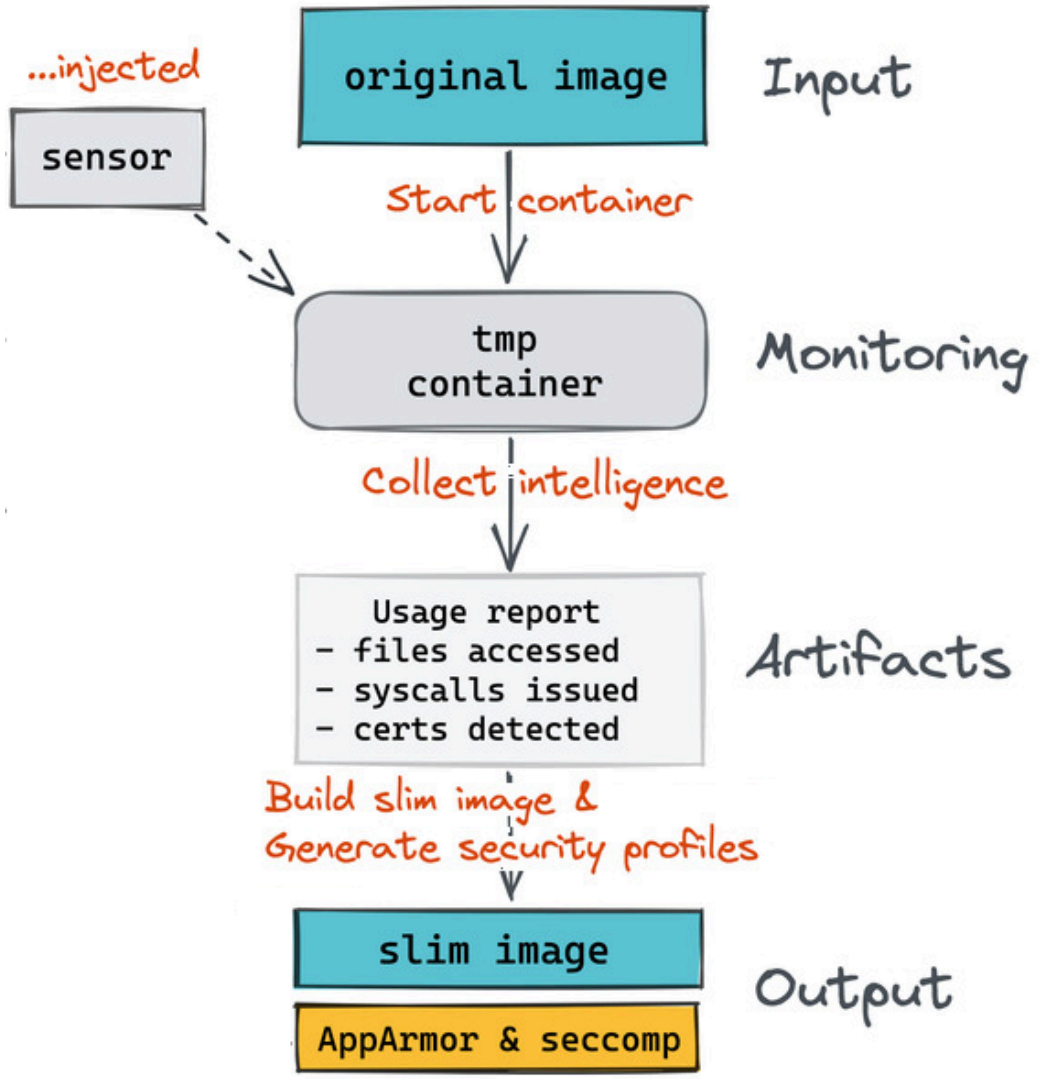
```
$ docker history $IMAGE_NAME
IMAGE          CREATED          CREATED BY
c274f07a418f   20 minutes ago   CMD ["/run.sh"]
<missing>      20 minutes ago   RUN ...    && rm -rf…
<missing>      20 minutes ago   COPY id_rsa /root/.ssh/ # buildkit
. . .
```

```
$ 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
```

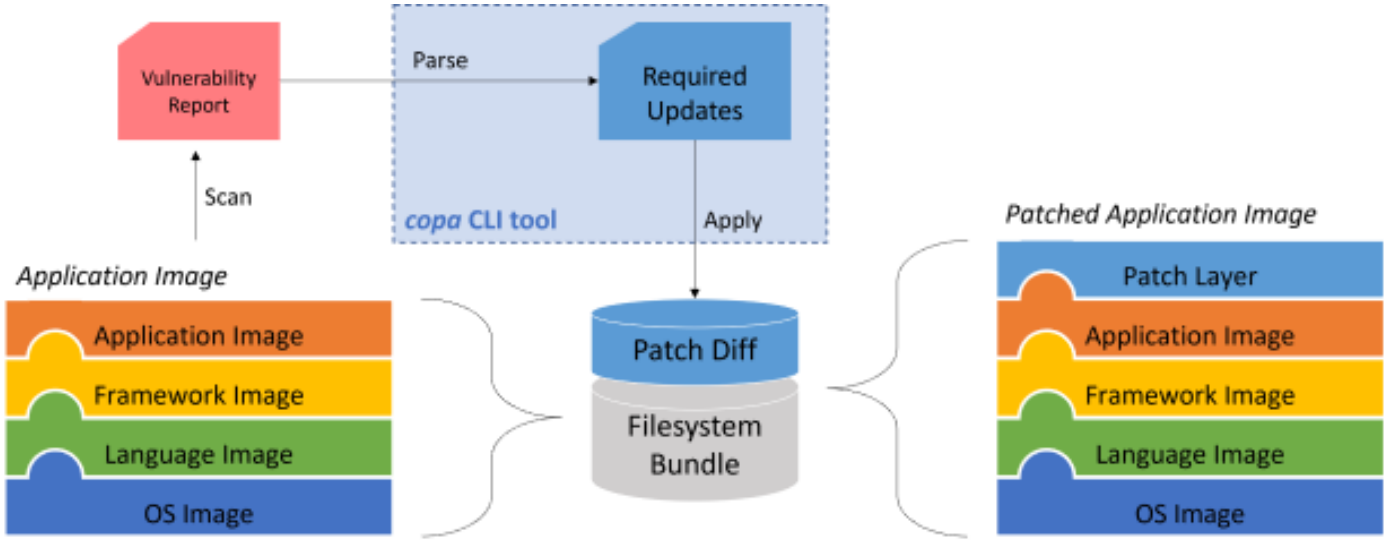


https://github.com/mintoolkit/mint

Модифицируем слои

copacetic

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



https://github.com/project-copacetic/copacetic

Модифицируем слои

copacetic

```
$ 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 CVE-2024-33601 CVE-2024-33602	MEDIUM		
libc6	CVE-2024-33599	HIGH		
	CVE-2024-33600 CVE-2024-33601 CVE-2024-33602	MEDIUM		

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 Copa

- 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 libx11-dev libcairo-gobject2 libncurses6 libpixman-1-0 libsqlite3-0 libssl-dev libexpat1-dev libwebp-dev libx11-6 libxslt1.1 libcairo-script-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 libfribidi0 libglib2.0-dev-bin libgnutlsxx28 libkadm5srv-mit11 libopenexr23 libcairo2 libfreetype6 unzip zlib1g libnettle6 openssl libaprutil1 libcairo2-dev libgnutls-dane0 libkrb5-3 libksba8 libmagickcore-6-headers libmaxminddb0 libp11-kit-dev libsasl2-2 icu-devtools libdjvulibre21 libsqlite3-dev imagemagick libgmpxx4ldbl libp11-kit0 libpython3.7-minimal libxslt1-dev gpgsm imagemagick-6.q16 libglib2.0-0 liblzma-dev libmagickcore-6.q16-6 libperl5.28 gpg-agent ncurses-bin libmariadb-dev-compat libss2 git libcurl4 libde265-0 libdpkg-perl libicu63 libk5crypto3 libssl1.1 libudev1 libwebp6 libx11-data libxml2-dev tzdata libmagickcore-6.q16-6-extra libmagickwand-dev libtasn1-6 libgssapi-krb5-2 libmagickcore-6.q16-dev libbz2-dev libc6-dev libcurl3-gnutls libext2fs2 libglib2.0-bin libgnutls30 libpcre2-8-0 libmagickcore-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 libpixman-1-dev librsvg2-common sudo gpgconf libdjvulibre-dev libexif12 libgcrypt20 libglib2.0-dev libncurses-dev xz-utils libgnutls28-dev libsvn1 libtiffxx5 libtinfo6 libwebpmux3 linux-libc-dev libdjvulibre-text libtiff-dev libunbound8 curl gnupg-l10n gpg-wks-server libglib2.0-data libhogweed4 libtasn1-6-dev perl-modules-5.28 krb5-multidev libgssrpc4 libsystemd0 nettle-dev libidn2-dev zlib1g-dev liblz4-1 libmaxminddb-dev libncursesw6 libwebpdemux2 mariadb-common subversion gir1.2-rsvg-2.0 gpg-wks-client libicu-dev libjpeg62-turbo-dev libmariadb3 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

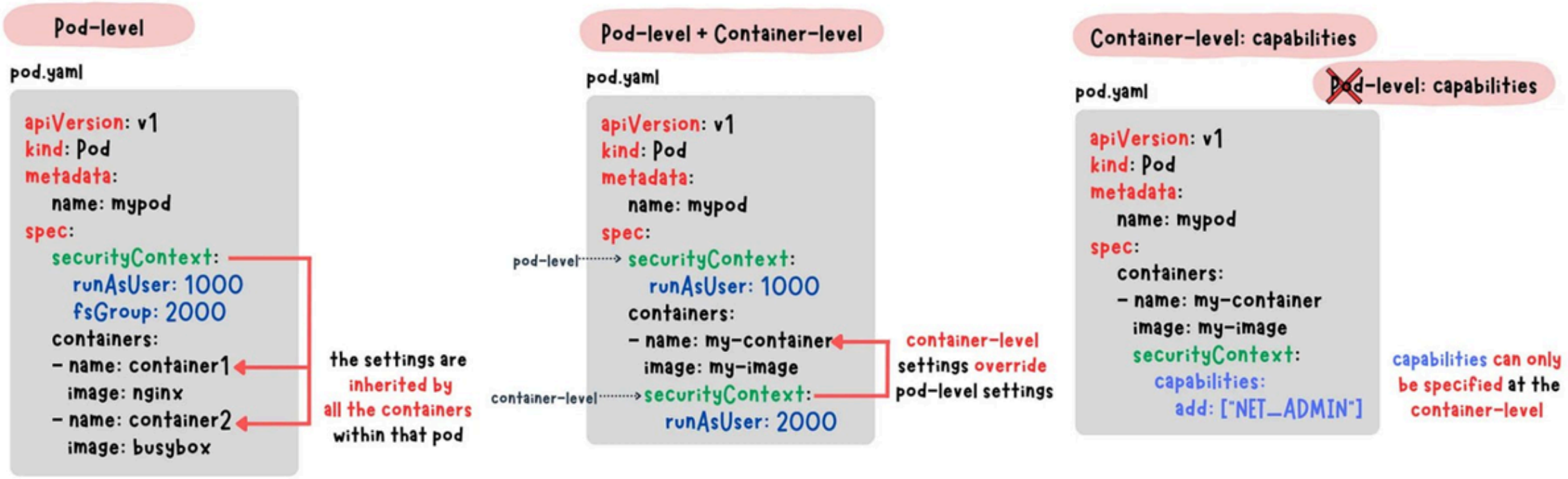
На уровне Kubernetes

> 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



https://yuminlee2.medium.com/kubernetes-security-contexts-e54624e29d52

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
```

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**

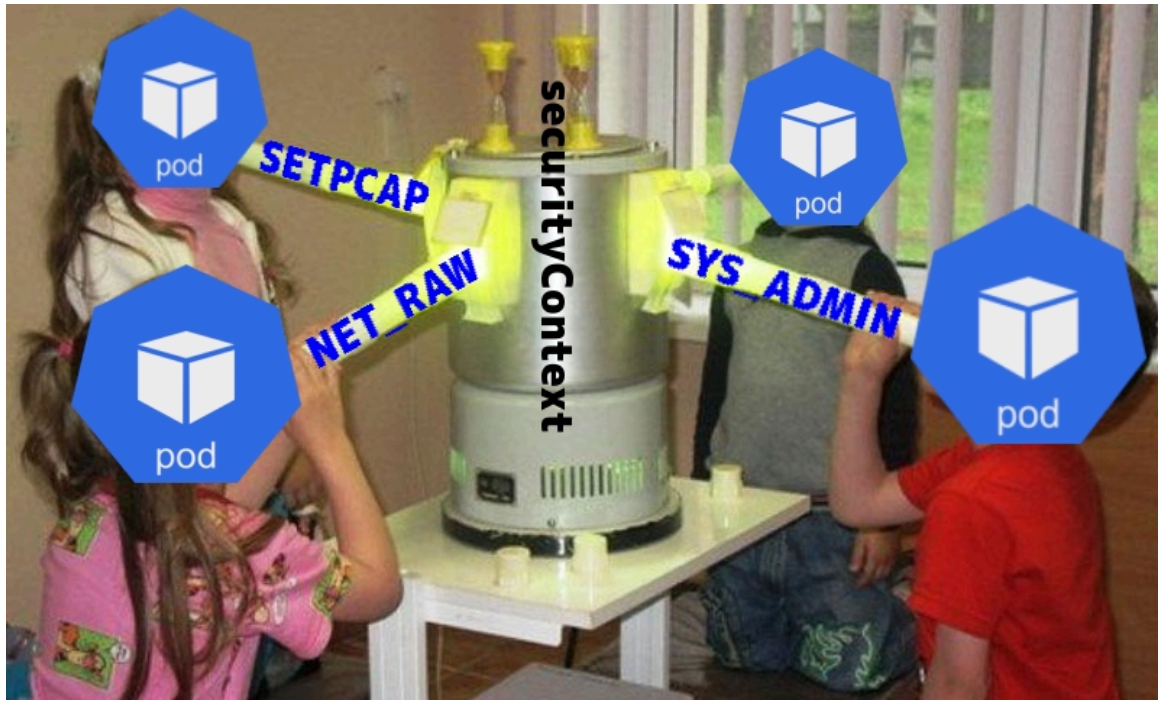
`allowPrivilegeEscalation` выставляем в `false`, что бы запретить процессам повышать привелегии через `sudo`, `setuid`

```
      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
- . . .
```

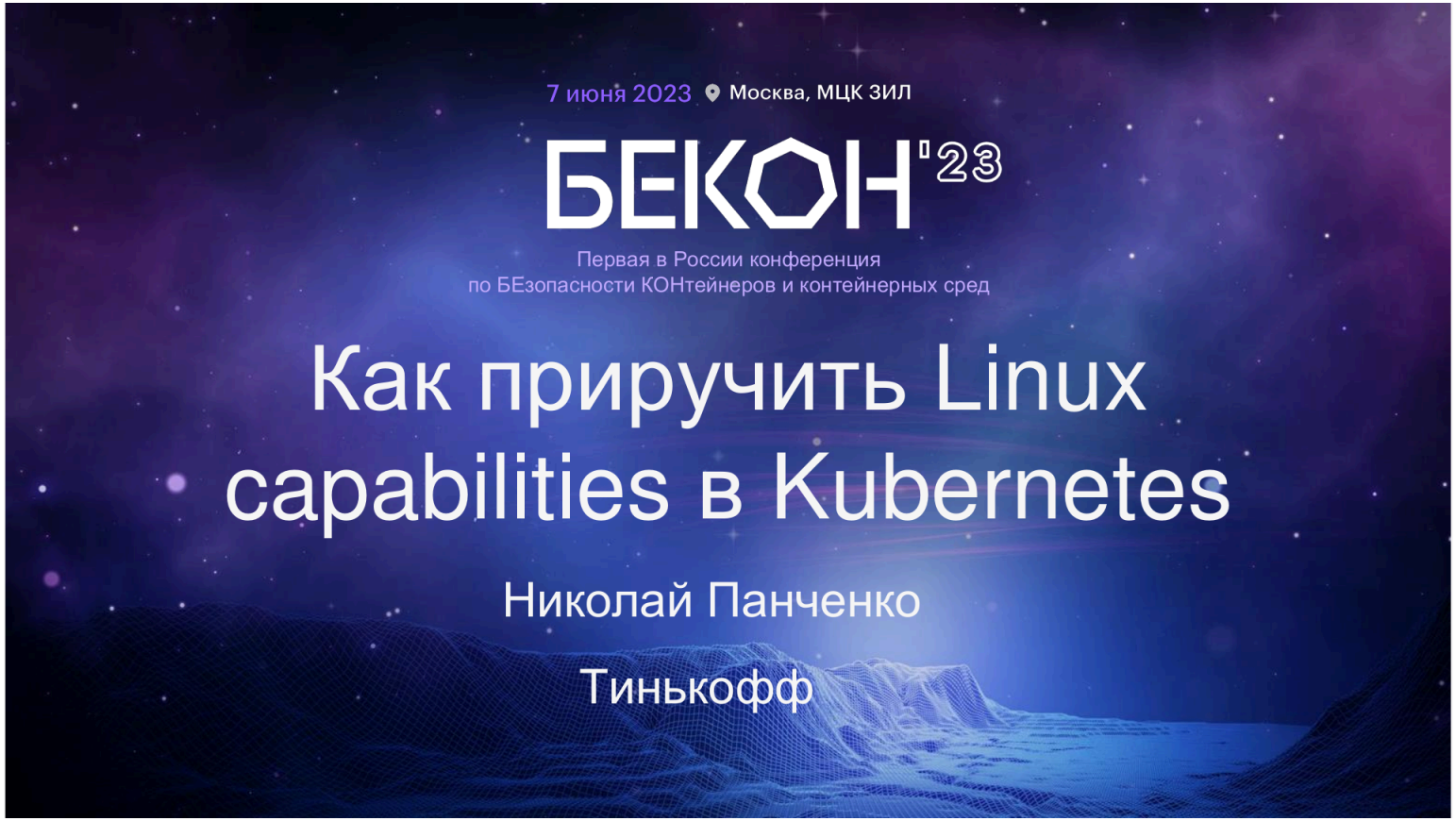


https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.30/#capabilities-v1-core

capabilities practice

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

Linux capabilities и Kubernetes



https://www.youtube.com/watch?v=maCIprVnBcQ

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

https://gitlab.com/apparmor/apparmor/-/wikis/About

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
```

https://kubernetes.io/docs/tutorials/security/apparmor/#example

AppArmor и Kubernetes

https://www.youtube.com/watch?v=IvVVUajYUAg

seccompProfile

> SeccompProfile defines a pod/container's seccomp profile settings

```yaml
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"
        }
    ]
}
```

https://kubernetes.io/docs/tutorials/security/seccomp/

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

https://medium.com/@LachlanEvenson/how-to-enable-kubernetes-container-runtimedefault-seccomp-profile-for-all-workloads-6795624fcbcc

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

https://www.youtube.com/watch?v=uouH9fsWVIE

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-userns.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cve-pod-userns
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-userns.yaml
pod/cve-pod-userns created
$ # Let's wait until the pod is Running
$ kubectl get pod cve-pod-userns
NAME             READY   STATUS    RESTARTS   AGE
cve-pod-userns   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
```

https://www.youtube.com/watch?v=M4a2b4KkXN8

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

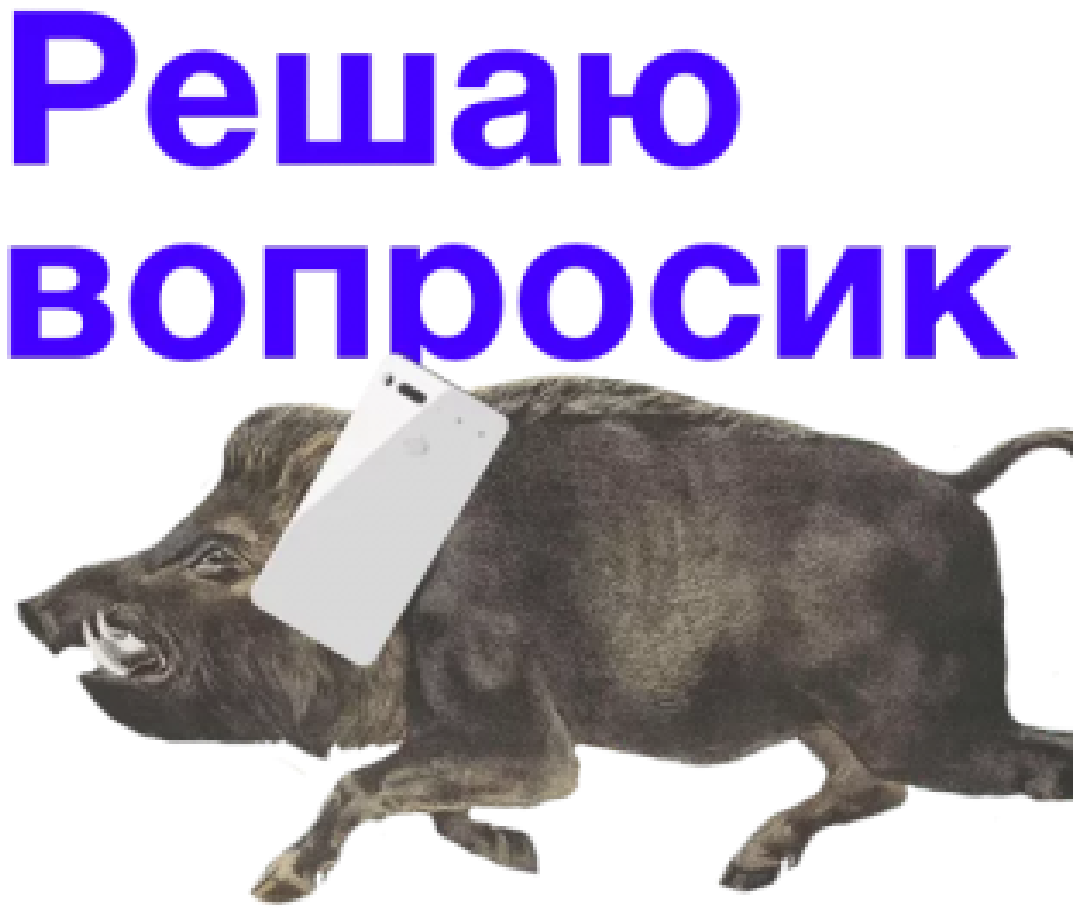
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: pod-two
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
      - podSelector:
          matchLabels:
            app: pod-one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

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/setguid-файлы)
- На уровне кода
  - Известные уязвимости (CVE)
    - 🛡️ privileged, alloPrivelegeEscalation, capabilities, appArmorProfile, seccompProfile, readOnlyRootFilesystem, NetworkPolicy

Резюмируя



Dockle: update, upgrade...

Примеры

- `/bin/sh -c apt-get update`
- `/bin/sh -c apt-get install -y git sudo curl gosu ncat`
- `/bin/sh -c apt-get dist-upgrade -s`

Что делаем?

Dockle: update, upgrade...

Примеры

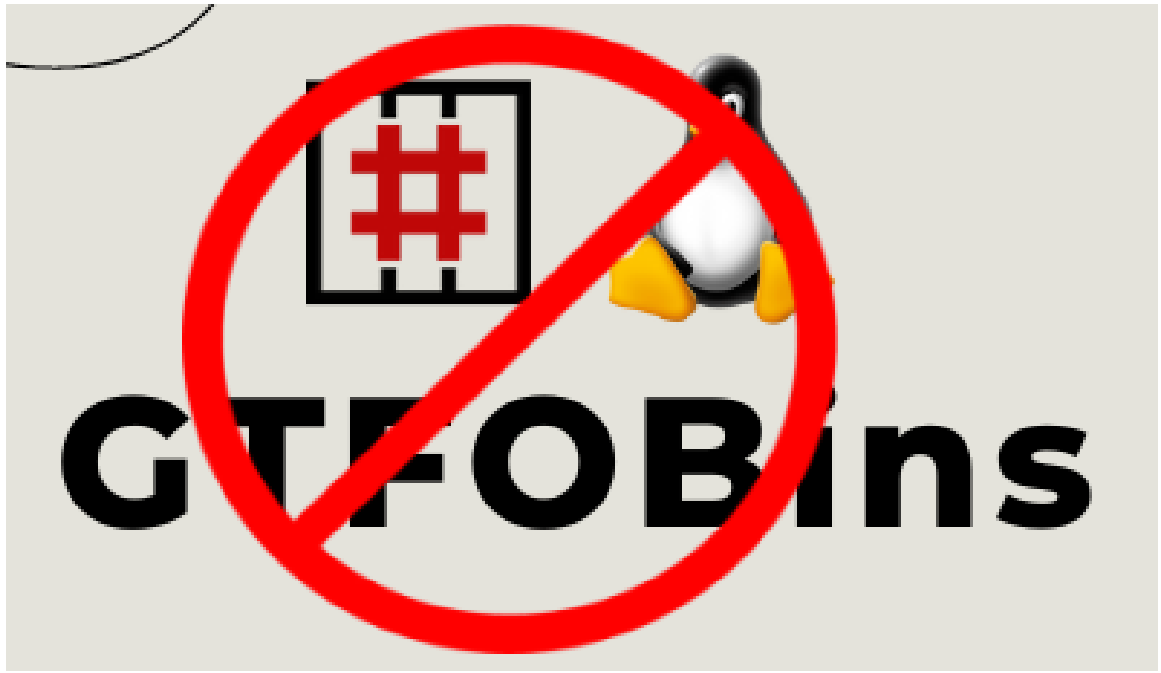
- `/bin/sh -c apt-get update`
- `/bin/sh -c apt-get install -y git sudo curl gosu ncat`
- `/bin/sh -c apt-get dist-upgrade -s`

Что делаем?

Принимаем тяжесть бытия

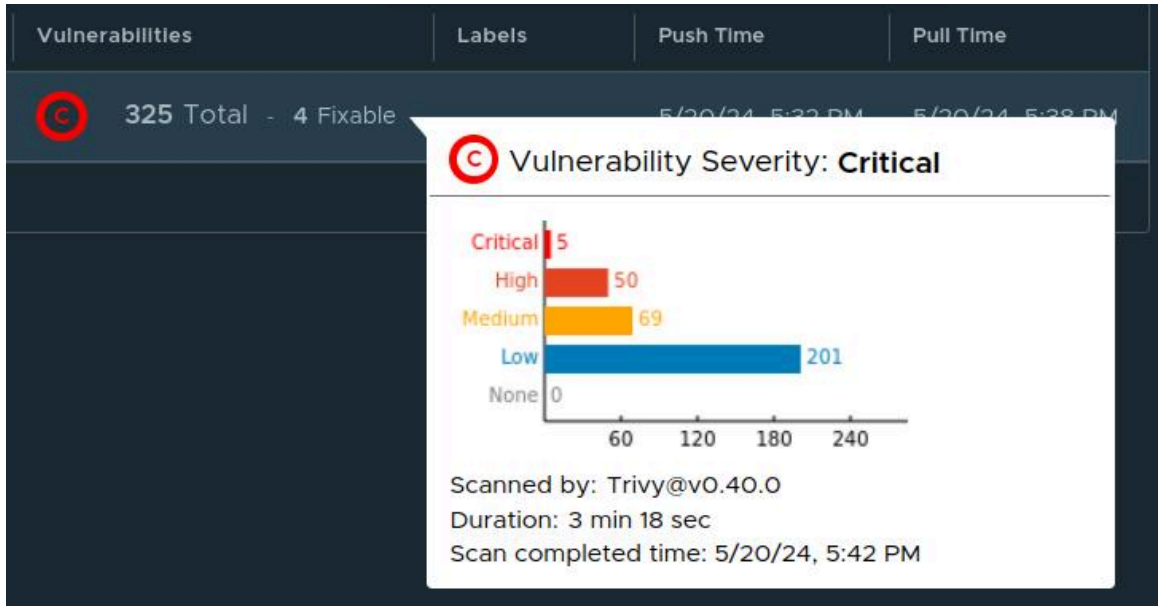
Миллионы исполняемых файлов (LotL-атаки)

- 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

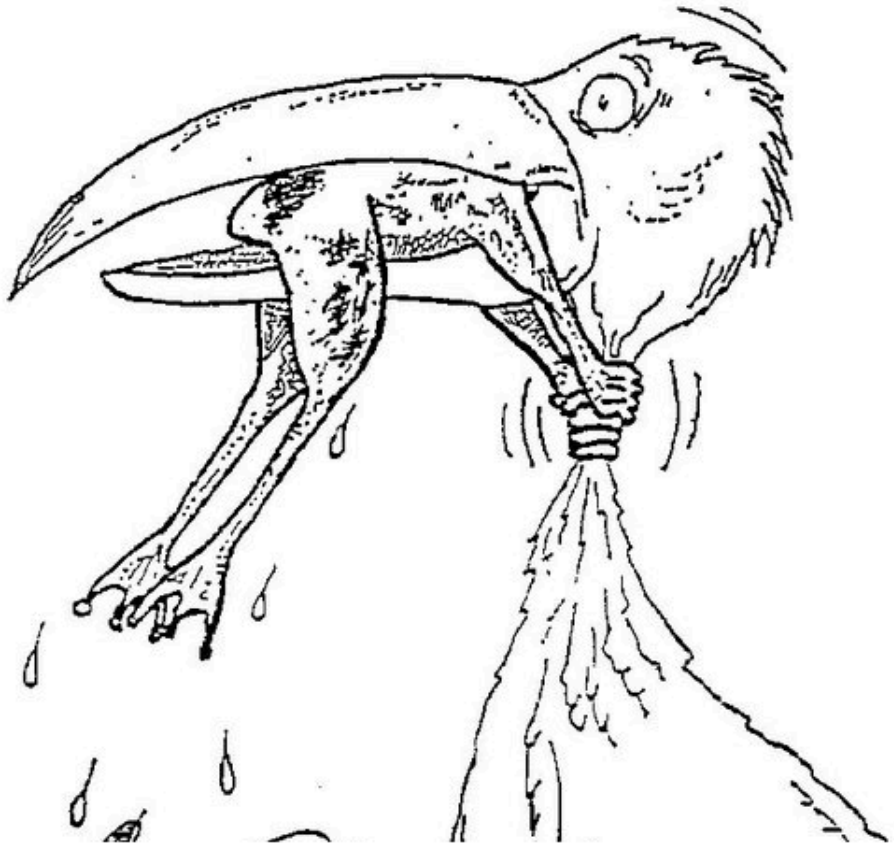
- Модифицируем слои
  - docker-squash
  - crane
  - mint

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
- 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

5 июня 2024 Москва, LOFT HALL#2

БЕКОН'24

Конференция по БЕзопасности
КОНтейнеров и контейнерных сред

Спасибо за внимание!

site: luntry.ru

tg: @rusdacent

channel: @tech_b0lt_Genona