5 июня 2024 Москва, LOFT HALL#2

БЕКОН'24

Конференция по БЕзопасности
КОНтейнеров и контейнерных сред

Linux user namespace в чертогах Kubernetes

Дмитрий Евдокимов

Основатель Luntry

WhoAmI

БЕКОН

- Основатель и технический директор Luntry
- Опыт в ИБ более 15 лет
- Специализация безопасность контейнеров и Kubernetes
- CFP DevOpsConf, HighLoad++
- Бывший автор статей и редактор рубрик в журнале “ХАКЕР”
- Автор Telegram-канала “k8s (in)security”
- Автор курса “Cloud Native безопасность в Kubernetes”
- Не верит, что систему можно сделать надежной и безопасной, не понимая ее
- Организатор конференции по безопасности контейнеров - БеКон
- Докладчик: BlackHat, HITB, ZeroNights, HackInParis, Confidence, SAS, OFFZONE, PHDays, Kazhackstan, DevOpsConf, DevOops, KuberConf, VK Kubernetes Conference, HighLoad++, БеКон и др.

План доклада

БЕКОН

- Теория Linux user namespace
- Проблематика
- Контейнеры и K8s встречают UserNS
- Заключение

БЕКОН

Теория Linux user namespace

Linux namespaces

БЕКОН

Namespace types

The following table shows the namespace types available on Linux. The second column of the table shows the flag value that is used to specify the namespace type in various APIs. The third column identifies the manual page that provides details on the namespace type. The last column is a summary of the resources that are isolated by the namespace type.

Namespace	Flag	Page	Isolates
Cgroup	CLONE_NEWCGROUP	cgroup_namespaces(7)	Cgroup root directory
IPC	CLONE_NEWIPC	ipc_namespaces(7)	System V IPC, POSIX message queues
Network	CLONE_NEWNET	network_namespaces(7)	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	mount_namespaces(7)	Mount points
PID	CLONE_NEWPID	pid_namespaces(7)	Process IDs
Time	CLONE_NEWTIME	time_namespaces(7)	Boot and monotonic clocks
User	CLONE_NEWUSER	user_namespaces(7)	User and group IDs
UTS	CLONE_NEWUTS	uts_namespaces(7)	Hostname and NIS domain name

The *proc*/pid/*ns*/ directory

Each process has a */proc/pid/ns/* subdirectory containing one entry for each namespace that supports being manipulated by setns(2):

```
$ ls -l /proc/$$/ns | awk '{print $1, $9, $10, $11}'
total 0
lrwxrwxrwx. cgroup -> cgroup:[4026531835]
lrwxrwxrwx. ipc -> ipc:[4026531839]
lrwxrwxrwx. mnt -> mnt:[4026531840]
lrwxrwxrwx. net -> net:[4026531969]
lrwxrwxrwx. pid -> pid:[4026531836]
lrwxrwxrwx. pid_for_children -> pid:[4026531834]
lrwxrwxrwx. time -> time:[4026531834]
lrwxrwxrwx. time_for_children -> time:[4026531834]
lrwxrwxrwx. user -> user:[4026531837]
lrwxrwxrwx. uts -> uts:[4026531838]
```

История root пользователя и capability

БЕКОН

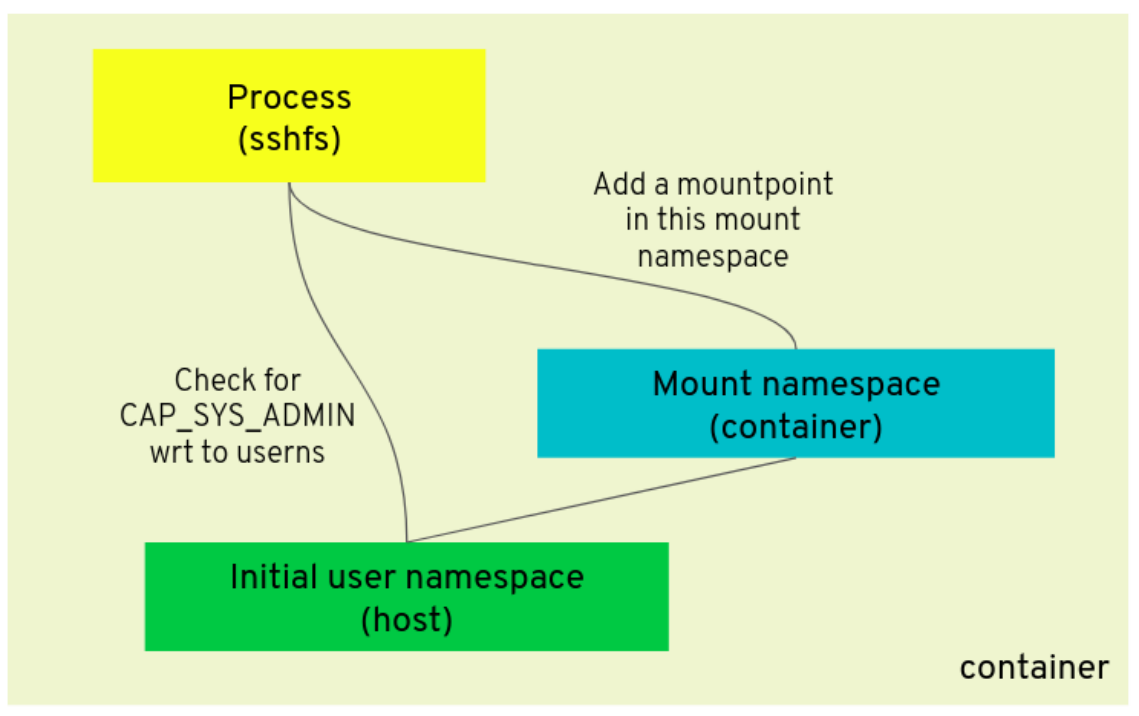
Вехи развития

- До версии Linux 2.2
  - До 1999 года
  - Пользователь root (UID 0) со всеми привилегиями
  - Остальные пользователи (UID не 0) без привилегий
- С версии Linux 2.2 до 3.8
  - Привилегии пользователя root разбиваются на capability
    - На пример, CAP_NET_ADMIN, CAP_SYS_ADMIN, CAP_NET_BIND_SERVICE и т.д.
  - Capability можно забирать и выдавать
  - За capability по сути скрывается системный вызов с определёнными аргументами
- С версии Linux 3.8
  - С 2013 год
  - Появление user namespaces (UserNS) и деление на initial и unprivileged user namespaces
  - Capability перестали носить глобальный характер, а стали относится к user namespaces
  - Появился Global root это root в initial user namespace

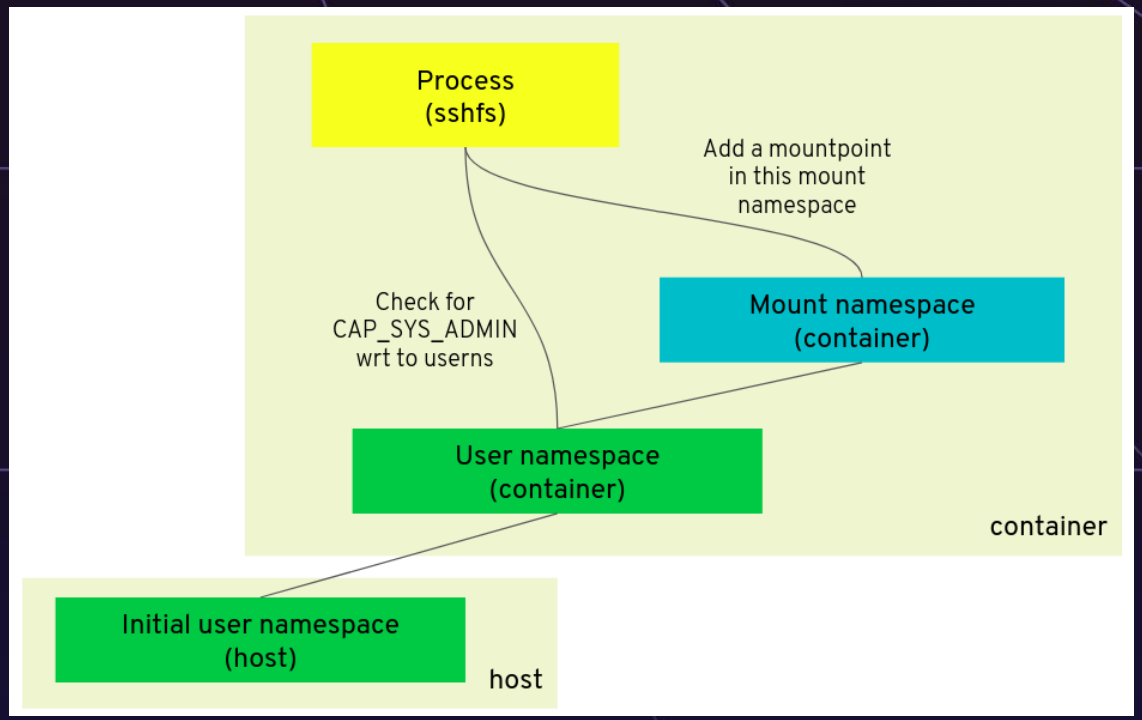
Проверка привилегий в UserNS

БЕКОН

Было



Стало



"Improving Kubernetes and container security with user namespaces", Alban Crequy

Маппинг пользователей

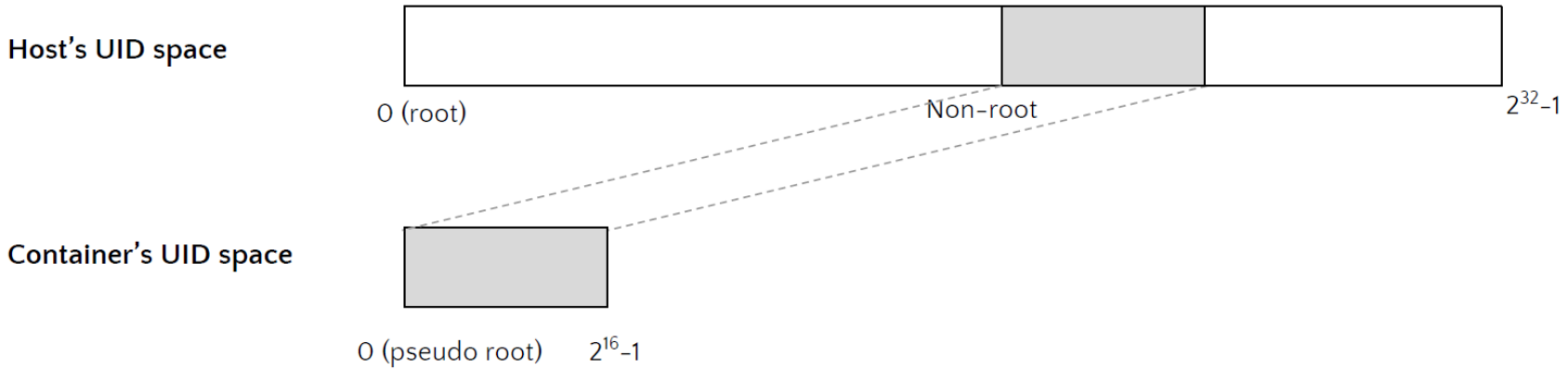
БЕКОН

(Optional) User namespaces

Don't confuse this with "User space" (the antonym of "kernel space")

NTT

- Maps a non-root user to the pseudo "root" in a container
- Pretends to be the root in the container (`apt-get`, `dnf`, ...)
- Just a non-root user outside the container
- Mitigates potential container breakout attacks



User namespaces were merged in Linux v2.6.23 (2007), enhanced in Linux v3.8 (2013)

24

"The internals and the latest trends of container runtimes", Akihiro Suda (NTT)

Создание unprivileged user namespaces

БЕКОН

Creation of new namespaces using clone(2) and unshare(2) in most cases requires the **CAP_SYS_ADMIN** capability, since, in the new namespace, the creator will have the power to change global resources that are visible to other processes that are subsequently created in, or join the namespace. User namespaces are the exception: since Linux 3.8, no privilege is required to create a user namespace.

unshare(2)

The unshare(2) system call moves the calling process to a new namespace. If the *flags* argument of the call specifies one or more of the **CLONE_NEW*** flags listed above, then new namespaces are created for each flag, and the calling process is made a member of those namespaces. (This system call also implements a number of features unrelated to namespaces.)

```
root@nginx-deployment-fc57c95c8-cvw5l:/# su tester
tester@nginx-deployment-fc57c95c8-cvw5l:/$ id
uid=1000(tester) gid=1000(tester) groups=1000(tester),100(users)
tester@nginx-deployment-fc57c95c8-cvw5l:/$ unshare -r
root@nginx-deployment-fc57c95c8-cvw5l:/# id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
root@nginx-deployment-fc57c95c8-cvw5l:/#
```

БЕКОН

Проблематика

Unprivileged user namespaces добро или зло?

БЕКОН

INFORMATION SECURITY

Home
Questions
Tags
Users
Jobs NEW

user namespaces: do they increase security, or introduce new attack surface?

Asked 1 year, 4 months ago Modified 1 year, 4 months ago Viewed 768 times

0

user namespaces in Linux are presented as a security feature, which should increase security. But is this really true?

Is it possible that while user namespaces fix one kind of problem, they introduce another,

The Overflow Blog

Developers get by with AI: Stack Overflow Knov

Linux user namespaces might not be secure enough? a.k.a. subverting POSIX capabilities

Erica Windisch · Follow

5 min read · Nov 3, 2015

UserNamespacesWhySecurityProblems

One reason why user namespaces keep enabling Linux kernel security issues

December 7, 2016

Q: How disabling `CONFIG_USER_NS` cuts the attack surface? It's needed for containers!

A: Yes, the `CONFIG_USER_NS` option provides some isolation between the userspace programs, but the tool recommends disabling it to cut the attack surface **of the kernel**.

Ссылки: 1,2,3,4

Oooops!

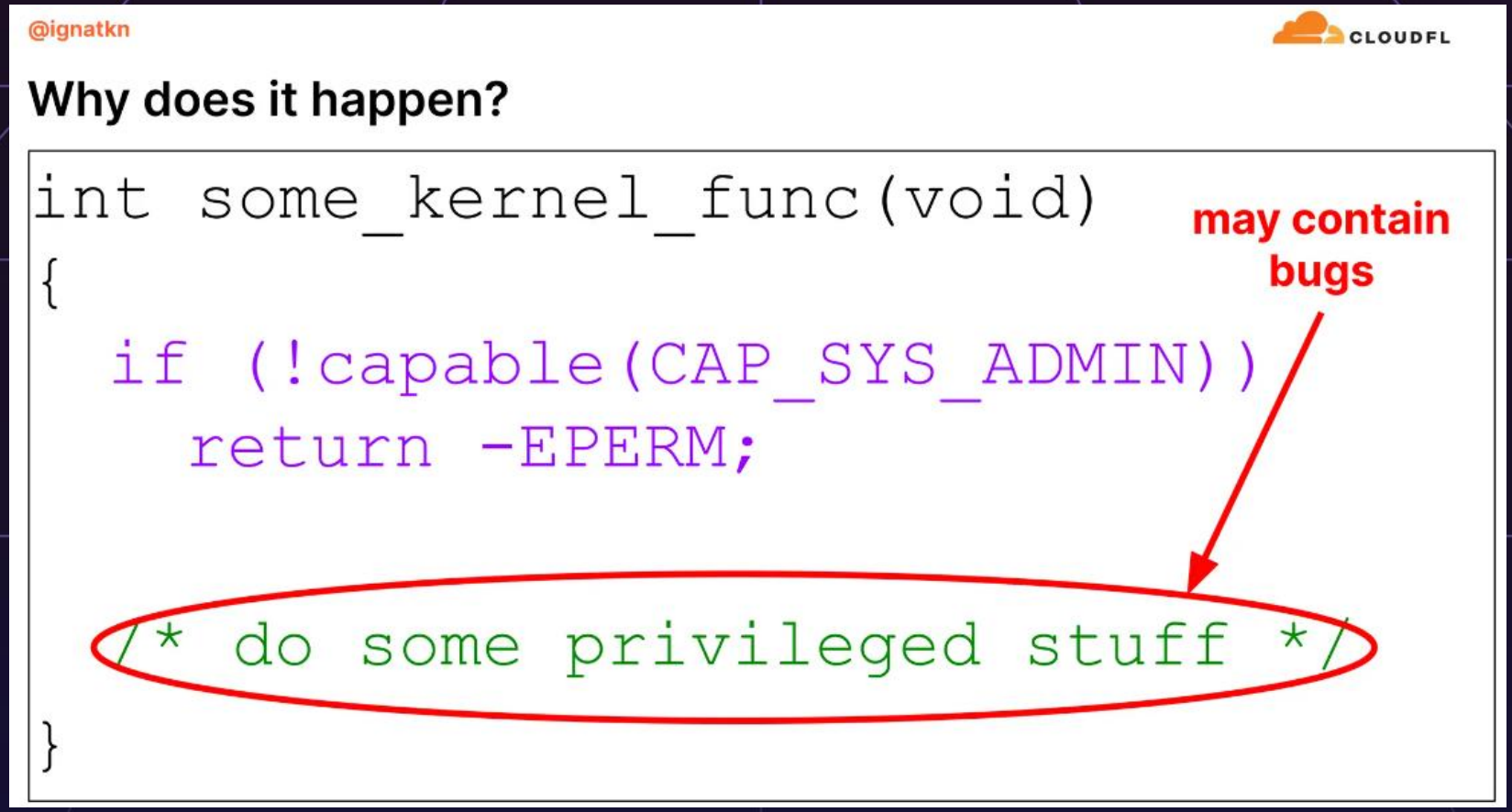
БЕКОН

Creation of new namespaces using clone(2) and unshare(2) in most cases requires the **CAP_SYS_ADMIN** capability, since, in the new namespace, the creator will have the power to change global resources that are visible to other processes that are subsequently created in, or join the namespace. User namespaces are the exception: since Linux 3.8, no privilege is required to create a user namespace.

Увеличиваем поверхность атаки на ядро

БЕКОН

Раньше это мог сделать только привилегированный пользователь, а теперь и обычный



"Linux user namespaces: a blessing and a curse", Ignat Korchagin

Уязвимости ядра

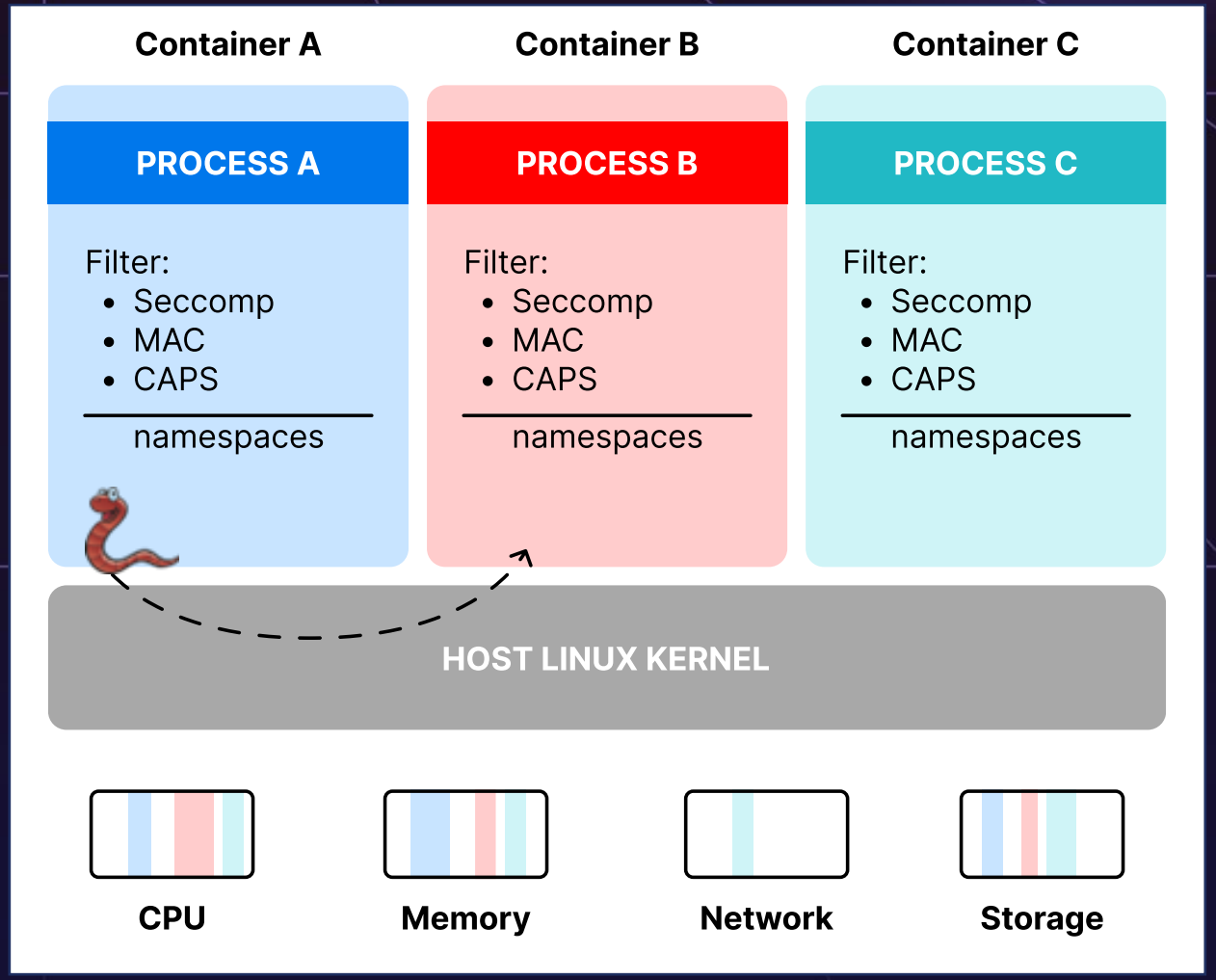
БЕКОН

Все контейнеры разделяют между собой ядро хостовой ОС.

Примеры уязвимостей для побега:

- CVE-2023-35001
- CVE-2023-31248
- CVE-2023-25809
- CVE-2022-47929
- CVE-2022-2327
- CVE-2022-0492
- CVE-2022-0185
- CVE-2021-22555
- ...

https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=user+namespace



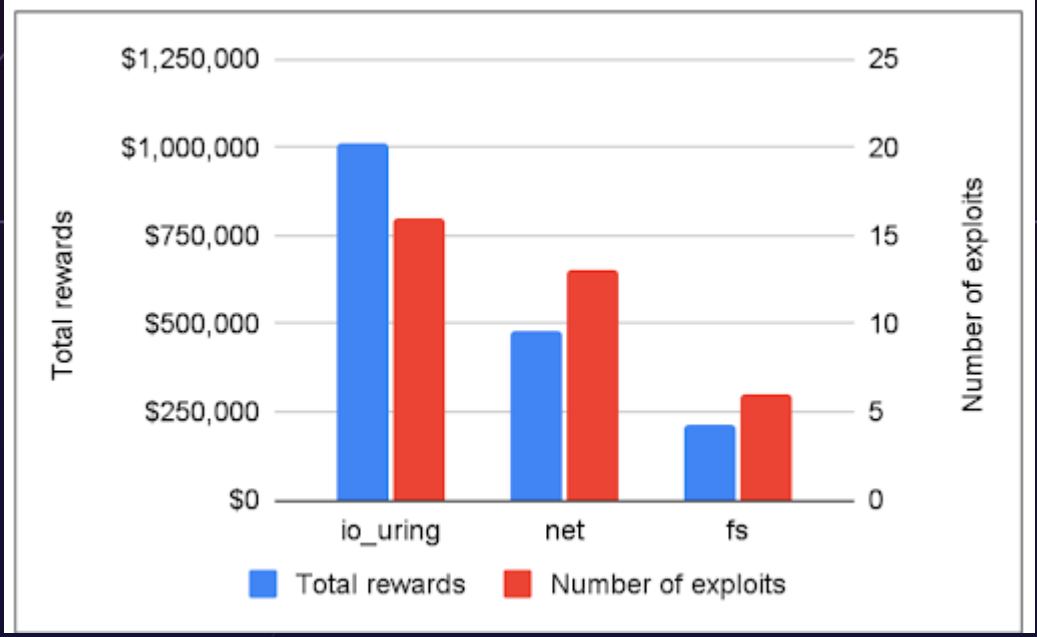
Google Bug Bounty

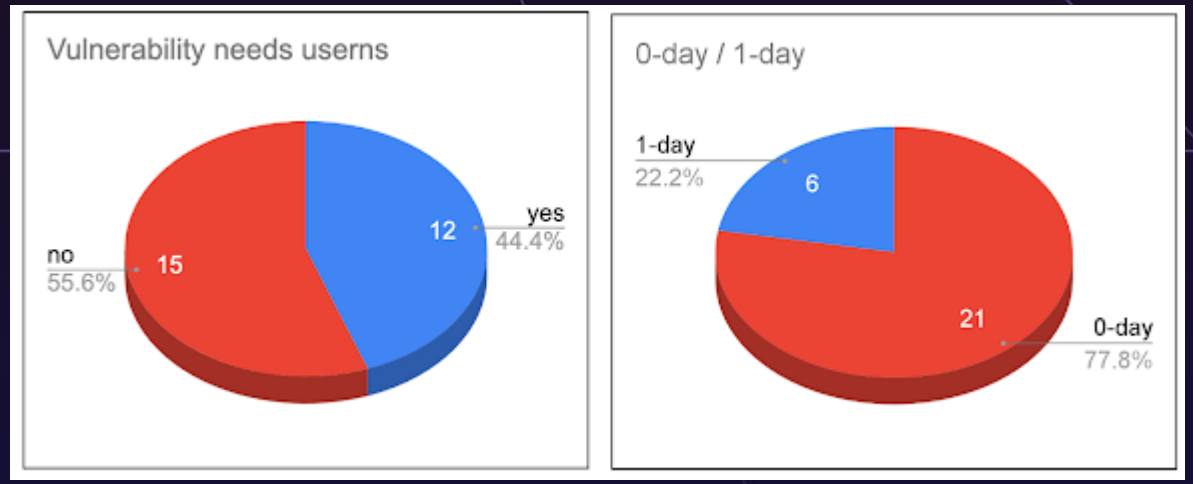
БЕКОН

Google Vulnerability Rewards Program (VRP):

- kCTF – программа вознаграждений за уязвимости к Kubernetes
- kernelCTF – программа вознаграждений за уязвимости в ядре Linux

Public kCTF VRP / kernelCTF responses





kernelCTF

БЕКОН

Уменьшение выплат при использовании unprivileged user namespaces

Reward

- $21.000 if the exploit does not use user namespaces and io_uring
- $10.500 if the exploit uses user namespaces or io_uring
  - This reward is based on whether the exploit works on GKE AutoPilot or not. AutoPilot currently does not enable unprivileged user namespaces and they are also considering disabling io_uring.

- Reduced attack surface bonus (+$20.000)
  - Criteria: Exploit works without using unprivileged user namespaces.

kernelCTF rules

Контроль UserNS в разных ОС

БЕКОН

- Ubuntu
  - kernel.unprivileged_userns_clone
  - С Ubuntu 23.10 новый sysctl kernel.apparmor_restrict_unprivileged_userns
    - Специальный AppArmor профиль /etc/apparmor.d/usr.bin.<FOO> для создания UserNS
- Debian
  - user.max_user_namespaces
- ALT Linux
  - kernel.userns_restrict
- RHEL
  - user.max_user_namespaces

- В Linux v6.1 (2022) появился LSM hook: userns_create
  - Можно поймать через eBPF LSM и реализовать свою логику вынесения вердикта
  - Более гибкий способ контроля за созданием новых UserNS

БЕКОН

Контейнеры и K8s встречают UserNS

Проблема root пользователя в контейнерах

БЕКОН

По умолчанию, root внутри контейнера это root на хосте.

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

Rootless контейнеры

БЕКОН

Построены на базе UserNS

Rootless containers

NTT

- Puts container runtimes (as well as containers) in a user namespace
  - UserNS: Linux kernel's feature that maps a non-root user to a fake root (the root privilege is limited inside the namespace)
- Can mitigate potential vulnerabilities of the runtimes
  - No access to read/write other users' files
  - No access to modify the kernel
  - No access to modify the firmware
  - No ARP spoofing
  - No DNS spoofing

e.g., runc breakout
CVE-2024-21626
(2024-01-31)

- Also useful for shared hosts (High-performance Computing, etc.)
  - Works with GPU too

2

"Rootless Containers", Akihiro Suda (NTT)

Взгляд CIS Docker Benchmark

БЕКОН

2.9 Enable user namespace support (Manual)

Profile Applicability:

- Level 2 - Docker - Linux

Description:

You should enable user namespace support in Docker daemon to utilize container user to host user re-mapping. This recommendation is beneficial where the containers you are using do not have an explicit container user defined in the container image. If the container images that you are using have a pre-defined non-root user, this recommendation may be skipped as this feature is still in its infancy, and might result in unpredictable issues or difficulty in configuration.

Rationale:

The Linux kernel "user namespace" support within the Docker daemon provides additional security for the Docker host system. It allows a container to have a unique range of user and group IDs which are outside the traditional user and group range utilized by the host system.

For example, the root user can have the expected administrative privileges inside the container but can effectively be mapped to an unprivileged UID on the host system.

Impact:

User namespace remapping is incompatible with a number of Docker features and also currently breaks some of its functionalities. Reference the Docker documentation and included links for details.

Фичи Kubernetes

БЕКОН

- KubeletInUserNamespace
- UserNamespacesSupport
- UserNamespacesPodSecurityStandards

```
// owner: @AkihiroSuda
// alpha: v1.22
//
// Enables support for running kubelet in a user namespace.
// The user namespace has to be created before running kubelet.
// All the node components such as CRI need to be running in the same user namespace.
KubeletInUserNamespace featuregate.Feature = "KubeletInUserNamespace"
```

```
// owner: @rata, @giuseppe
// kep: https://kep.k8s.io/127
// alpha: v1.25
// beta: v1.30
//
// Enables user namespace support for stateless pods.
UserNamespacesSupport featuregate.Feature = "UserNamespacesSupport"
```

```
// owner: @saschagrunert
// alpha: v1.28
//
// Enables user namespace support for Pod Security Standards. Enabling this
// feature will modify all Pod Security Standard rules to allow setting:
// spec[.*].securityContext.[runAsNonRoot,runAsUser]
// This feature gate should only be enabled if all nodes in the cluster
// support the user namespace feature and have it enabled. The feature gate
// will not graduate or be enabled by default in future Kubernetes
// releases.
UserNamespacesPodSecurityStandards featuregate.Feature = "UserNamespacesPodSecurityStandards"
```

KEP-2033: Kubelet-in-UserNS (aka Rootless mode)

БЕКОН

This KEP allows running the entire Kubernetes components (`kubelet` , CRI, OCI, CNI, and all `kube-*`) as a non-root user on the host, by running them in a user namespace. See Notes/Constraints/Caveats for the caveats.

Goals

- Allow `kubelet` and `kube-proxy` to be executed inside user namespaces create by a non-root user. See "Required changes to Kubernetes".

Motivation

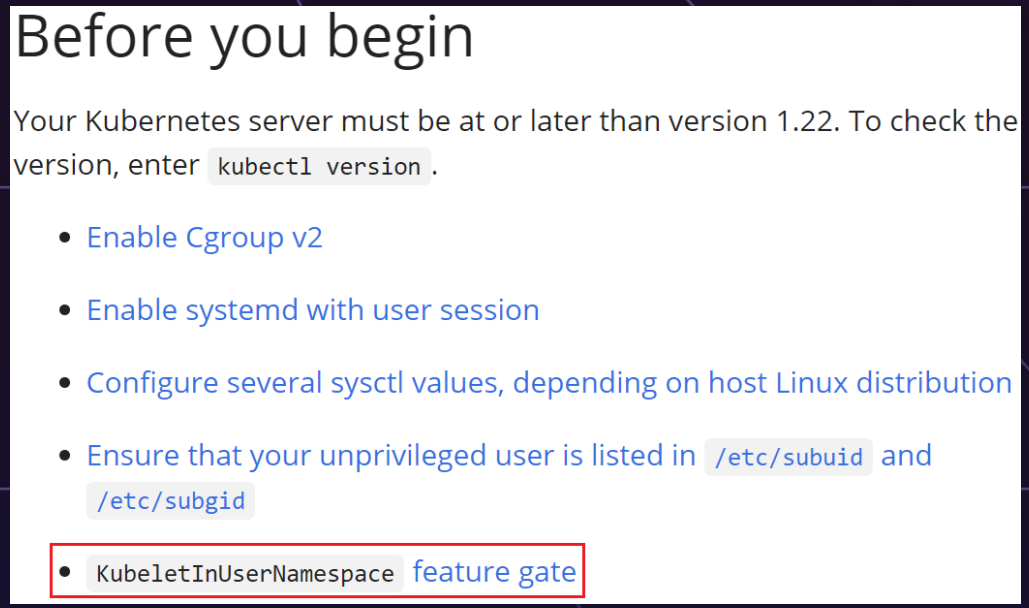
- Protect the host from potential container-breakout vulnerabilities. This is the main motivation.
- Allow users of shared machines (especially HPC) to run Kubernetes without the risk of accidentally breaking their colleagues' environments. Not recommended for real multi-tenancy where the users cannot be trusted.
  - Safe `kind` : Kubernetes inside Rootless Docker/Podman.
  - Safe Kubernetes-on-Kubernetes, to isolate workloads more strictly than Kubernetes API namespaces.

- CVE-2017-1002102: kubelet could delete files on the host during syncing secret/configMap/downwardAPI volumes
- CVE-2019-11245: Dockerfile USER instruction was ignored by kubelet
- CVE-2018-11235: kubelet could execute an arbitrary command as the root via gitRepo volumes
- Potential image extraction zip-slip vulnerabilities in CRI runtimes. Both containerd and CRI-O are working on implementing supports for new archive formats like zstd, imgcrypt, and stargz. Potentially these implementations have such vulnerabilities.
- And lots of CRI/OCI vulnerabilities in the past.

KubeltInUserNamespace

БЕКОН

- Все Kubernetes компоненты и контейнеры могут работать от non-root пользователей
  - Работа идет с 2016 года
  - В статусе Alpha с 1.22
  - По умолчанию выключен



Before you begin

Your Kubernetes server must be at or later than version 1.22. To check the version, enter `kubectl version`.

- Enable Cgroup v2
- Enable systemd with user session
- Configure several sysctl values, depending on host Linux distribution
- Ensure that your unprivileged user is listed in `/etc/subuid` and `/etc/subgid`
- `KubeletInUserNamespace` feature gate

Usernetes

БЕКОН

Usernetes - Kubernetes не требующий никаких root привилегий.

Usernetes: Kubernetes without the root privileges (Generation 2)

Usernetes (Gen2) deploys a Kubernetes cluster inside Rootless Docker, so as to mitigate potential container-breakout vulnerabilities.

Usernetes (Gen2) is similar to Rootless `kind` and Rootless minikube, but Usernetes (Gen 2) supports creating a cluster with multiple hosts.

```
sadm      5094      \_ /sbin/init
sadm      5220          \_ /lib/systemd/systemd-journald
sadm      5233          \_ /lib/systemd/systemd-udevd
sadm      5241          \_ /usr/local/bin/containerd
sadm      5589          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id 574bd1081ee947120ab1fdb19d68da8c5d3c9ba95e224ae291bfac6f01c6bb4c -address /run/containerd/containerd.sock
165534    5701          |   \_ /pause
sadm      5779          |   \_ kube-apiserver --advertise-address=172.20.218.132 --allow-privileged=true --authorization-mode=Node,RBAC --client-ca-file=/etc/kubernetes/pki/ca.crt --cloud-provider=external --enable-admission-plugins=Nod
sadm      5590          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id 28a32cb2e51cb36dbe250dbe94ff8c79281fea0d862ad7c935a154a9dfbc4f8b -address /run/containerd/containerd.sock
165534    5708          |   \_ /pause
sadm      5867          |   \_ kube-scheduler --authentication-kubeconfig=/etc/kubernetes/scheduler.conf --authorization-kubeconfig=/etc/kubernetes/scheduler.conf --bind-address=127.0.0.1 --kubeconfig=/etc/kubernetes/scheduler.conf --le
sadm      5629          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id b0b94fb7535fdcc5b9a21380b702924591f260e344e04554691364e62cc2d3c8 -address /run/containerd/containerd.sock
165534    5692          |   \_ /pause
sadm      5966          |   \_ etcd --advertise-client-urls=https://10.100.238.100:2379 --cert-file=/etc/kubernetes/pki/etcd/server.crt --client-cert-auth=true --data-dir=/var/lib/etcd --experimental-initial-corrupt-check=true --experim
sadm      5636          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id c562eb325392db23f73d92851a027b7242991a3762b447d95d0400139f9b4172 -address /run/containerd/containerd.sock
165534    5720          |   \_ /pause
sadm      5830          |   \_ kube-controller-manager --allocate-node-cidrs=true --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf --authorization-kubeconfig=/etc/kubernetes/controller-manager.conf --bind-address=127.
sadm      6024          \_ /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --container-runtime-endpoint=unix:///var/run/conta
sadm      6141          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id dc09426dc408b73d84b173b70aa6f63e7fd068e1c942b5d7e4d8872c64677ec0 -address /run/containerd/containerd.sock
165534    6165          |   \_ /pause
sadm      6199          |   \_ /usr/local/bin/kube-proxy --config=/var/lib/kube-proxy/config.conf --hostname-override=u7s-rootles-k8s
sadm      6391          \_ /usr/local/bin/containerd-shim-runc-v2 -namespace k8s.io -id 54c0505f039cde83f3a03ae64dd01a6dd59924269b8729286c32a5ff95183f27 -address /run/containerd/containerd.sock
165534    6413          |   \_ /pause
```

KEP-127: Support User Namespaces

БЕКОН

The goal of supporting user namespaces in Kubernetes is to be able to run processes in pods with a different user and group IDs than in the host. Specifically, a privileged process in the pod runs as an unprivileged process in the host. If such a process is able to break out of the container to the host, it'll have limited impact as it'll be running as an unprivileged user there.

Goals

Here we use UIDs, but the same applies for GIDs.

- Increase node to pod isolation by mapping user and group IDs inside the container to different IDs in the host. In particular, mapping root inside the container to unprivileged user and group IDs in the node.
- Increase pod to pod isolation by allowing to use non-overlapping mappings (UIDs/GIDs) whenever possible. In other words: if two containers runs as user X, they run as different UIDs in the node and therefore are more isolated than today.
- Allow pods to have capabilities (e.g. `CAP_SYS_ADMIN`) that are only valid in the pod (not valid in the host).
- Benefit from the security hardening that user namespaces provide against some of the future unknown runtime and kernel vulnerabilities.

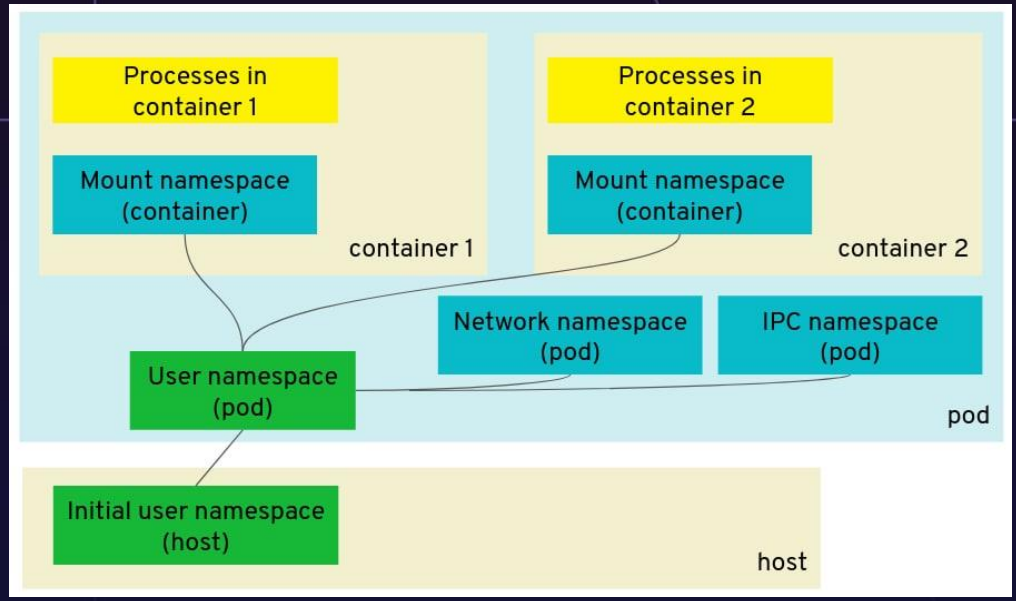
- CVE-2019-5736: Host runc binary can be overwritten from container. Completely mitigated with userns.
  - Score: 8.6 (HIGH)
  - https://github.com/opencontainers/runc/commit/0a8e4117e7f715d5fbeef398405813ce8e88558b
- Azurescape: Completely mitigated with userns, at least as it was found (needs CVE-2019-5736). This is the **first cross-account container takeover in the public cloud**.
- CVE-2021-25741: Mitigated as root in the container is not root in the host
  - Score: 8.1 (HIGH) / 8.8 (HIGH)
- CVE-2017-1002101: mitigated, idem
  - Score: 9.6 (CRITICAL) / 8.8 (HIGH)
- CVE-2021-30465: mitigated, idem
  - Score: 8.5 (HIGH)
- CVE-2016-8867: Privilege escalation inside containers
  - Score: 7.5 (HIGH)
  - moby/moby#27590
- CVE-2018-15664: TOCTOU race attack that allows to read/write files in the host
  - Score: 7.5 (HIGH)
  - moby/moby#39252

UserNamespacesSupport

БЕКОН

- Позволяет контейнерным процессам работать в отдельном пространстве (user namespace), отличном от хостового
  - В статусе Alpha с 1.25, в Beta с 1.30
  - До 1.28 назывался UserNamespacesStatelessPodsSupport
  - По умолчанию выключен
- Ядро старше 6.3, зависимости от CRI-O и crun …
- Директива hostUsers в манифесте
  - hostUsers: false не совместима с
    - hostNetwork: true
    - hostIPC: true
    - hostPID: true

```
apiVersion: v1
kind: Pod
spec:
  hostUsers: false
  containers:
  - name: nginx
    image: docker.io/nginx
```



Не путаем KEP-127 и KEP-2033

БЕКОН

Non-Goals

- Provide a way to run the kubelet process or container runtimes as an unprivileged process. Although initiatives like kubelet in user namespaces and this KEP both make use of user namespaces, it is a different implementation for a different purpose.

Non-Goals

The Node-level UserNS KEP is similar to this KEP, but out of scope for this KEP.

While Node-level UserNS executes only containers inside UserNS. this KEP executes all the node components inside UserNS to mitigate vulnerabilities of all components,

Node-level UserNS and this KEP do not conflict and can be stacked together. (Node-level UserNS inside Kubelet's UserNS.)

** - создать кластер, где одновременно включены эти фичи на сегодняшний день не удалось ...*

UserNamespacesPodSecurityStandards

БЕКОН

- Позволяет смягчить политику Pod Security Standards для Pods запущенных в UserNS
  - В статусе Alpha с 1.29
  - По умолчанию выключен

If you enable the associated feature gate and create a Pod that uses user namespaces, the following fields won't be constrained even in contexts that enforce the *Baseline* or *Restricted* pod security standard. This behavior does not present a security concern because `root` inside a Pod with user namespaces actually refers to the user inside the container, that is never mapped to a privileged user on the host. Here's the list of fields that are **not** checks for Pods in those circumstances:

- `spec.securityContext.runAsNonRoot`
- `spec.containers[*].securityContext.runAsNonRoot`
- `spec.initContainers[*].securityContext.runAsNonRoot`
- `spec.ephemeralContainers[*].securityContext.runAsNonRoot`
- `spec.securityContext.runAsUser`
- `spec.containers[*].securityContext.runAsUser`
- `spec.initContainers[*].securityContext.runAsUser`
- `spec.ephemeralContainers[*].securityContext.runAsUser`

Default seccomp

БЕКОН

- Можно встретить под названиями
  - Default seccomp profile
  - SeccompDefault
  - RuntimeDefault
- Развитие в Kubernetes версиях
  - 1.22 - alpha
  - 1.25 - beta
  - 1.27 - GA
- Настройка флага kubelet или его конфигурационный файл
  - --seccomp-default
  - seccompDefault: true
- Отключает/запрещаем многострадальный вызов unshare внутри контейнера

```
securityContext:
  seccompProfile:
    type: RuntimeDefault
```

```
// DefaultProfile is used to allow mutations from the DefaultProfile from the seccomp library.
// Specifically, it is used to filter `unshare` from the default profile, as it is a risky syscall for unprivileged containers
// to have access to.
```

БЕКОН

Заключение

Выводы

БЕКОН

- Неуместное, неконтролируемое использование unprivileged user namespaces приводит к увеличению поверхности атаки на ядро
  - Где данная функциональная не требуется - отключайте

- Уместное, правильное использование unprivileged user namespaces позволяет реализовать sandbox концепцию, rootless контейнеры и уменьшить поверхность атаки у ядра из контейнера
  - Следуйте принципу наименьших привилегий
    - SecurityContext, seccomp и т.д.
  - Для контейнерных окружений, Kubernetes используйте специализированное окружение , без доступа туда человека
    - Container-specific OS

5 июня 2024 Москва, LOFT HALL#2

Конференция по БЕзопасности
КОНтейнеров и контейнерных сред

БЕКОН

LUNTRY

Email: de@luntry.ru

Twitter: @evdokimovds

@Qu3b3c

Channel: @k8security

Site: www.luntry.ru

Спасибо за помощь в подготовке доклада
DevOps команде и особенно Никите Родкину!