Help I want to attack Kubernetes

Finn (f3rn0s)
3rd of May, 2023

Volkis

Who am I?

Big nerd. Youngest member @Volkis.

What I am currently passionate about:

- Kubernetes.
- Active Directory and internal testing.
- Red Teaming
- Finding jank vulnerabilities.

What is this talk?

- Learn some basic Kubernetes.
- Take a look at a real-world scenario of going from nothing to Domain Admin using Kubernetes.

What is this talk?

- Learn some basic Kubernetes.
- Take a look at a real-world scenario of going from nothing to Domain Admin using Kubernetes.

Sorry not sorry to people who haven't done Active Directory stuff, just nod along :').

What is Kubernetes?

I do not want to start here... instead lets look at the motivations for this project to exist.

What do we want to do?

- Manage services running on computers

What do we want to do?

- Manage services running on computers
- Manage shared resources between those services

What do we want to do?

- Manage services running on computers
- Manage shared resources between those services
- Manage the networking/firewall rules of those services

What do we want to do?

- Manage services running on computers
- Manage shared resources between those services
- Manage the networking/firewall rules of those services
- Containerise those services so they are reproducible across machines

What do we want to do?

- Manage services running on computers
- Manage shared resources between those services
- Manage the networking/firewall rules of those services
- Containerise those services so they are reproducible across machines
- I want create role based access controls over all these actions

What do we want to do?

- Manage services running on computers
- Manage shared resources between those services
- Manage the networking/firewall rules of those services
- Containerise those services so they are reproducible across machines
- I want create role based access controls over all these actions
- I want to define resource limits for services
- I want to dynamically scale services according to resource usage
- I want to log all services and make those logs accessible
- I want to define custom configurations for the scheduling of these services
- I want to pass secrets into machines
- ~~I want to~~ OH GOD PLEASE MAKE IT STOP

Introducing... Kubernetes!!!

You might have head of:

- One tool, one task
- Do one thing and do it well.
- Less is more.

Now Kubernetes does not necessarily disregard every piece of advice about simplicity.

We need to think of Kubernetes as a stack of different applications. (Cause it is).

We have a few components:

- The Kubernetes API Server
- The Scheduler
- Kubelet
- ETCd
- Controllers
- Kube Proxy
- The high-level container runtime
- The low-level container runtime
- Networking Solution (This is bring-your-own)

And these components all individually support a large amount of complex behaviours.

All of which can be configured *incorrectly*.

What should I (as a penetration tester) look out for?

An Exhaustive List

- Unauthorized access to the Kubernetes API server
- Misconfigured RBAC permissions
- Insecure communication between components
- Vulnerable containers
- Insufficient monitoring and logging
- Lack of proper network segmentation
- Insecure image storage and management
- Unsecured secrets and configuration data
- Inadequate security policies and procedures
- Lack of encryption for sensitive data in transit and at rest
- Unsecured container runtime environments
- Lack of proper patch management and software updates
- Inadequate threat intelligence and intrusion detection
- Insufficient data protection and privacy controls

An Exhaustive List (cont.)

- Lack of segregation of duties and least privilege principles
- Insufficient security awareness and training
- Insecure storage and data management
- Unsecured container image registries
- Insufficient container runtime security
- Lack of container isolation and resource controls
- Insecure Kubernetes cluster configuration
- Unsecured Kubernetes secrets and keys
- Insufficient Kubernetes pod security
- Lack of Kubernetes network security and isolation
- Unsecured Kubernetes deployment and update processes
- Lack of Kubernetes auditing and logging
- Unsecured Kubernetes admission controllers and webhooks

The end

Kidding...

I do not want to make you Kubernetes *Auditors*, I want to give you the knowledge and tools to go into a cluster, escalate privileges, grab secrets and run.

Think of it more as smash-and-grab training :).

Why Kubernetes?

Kubernetes allows administrators to create configurations that state the way a cluster should be configured. These configurations can be used to reproduce parts of the cluster, and can easily be modified.

Kubernetes makes it easy for administrators to manage:

- Secrets
- Shared Storage
- Exposed Services
- Role Based Access Controls (RBAC)*

Why Kubernetes?

Kubernetes allows administrators to create configurations that state the way a cluster should be configured. These configurations can be used to reproduce parts of the cluster, and can easily be modified.

Kubernetes makes it easy for administrators to manage:

- Secrets
- Shared Storage
- Exposed Services
- Role Based Access Controls (RBAC)*

Because all of this is represented in easy to read** and modify text files.

* RBAC is hard to get right

** Can actually be hard to read if you don't know all of Kubernetes.

Nodes

A node is just a computer that is joined into our cluster.

- **Master nodes:** These nodes manage the cluster and are considered the Kubernetes control plane. It consists of the following components:
  - API server
  - etcd
  - Scheduler
  - Controller manager
  - Kubelet
- **Worker nodes:** These nodes run our workloads
  - Container runtimes
  - Kubelet

Resources

A resource is an object that represents a piece of the cluster's state.

Here are the most common ones:

- Pod
- Service
- Deployment
- Secrets
- Namespace

Secrets stored in Kubernetes

Secrets

In order to safely use Secrets, take at least the following steps:

- Enable Encryption at Rest for Secrets.
- Enable or configure RBAC rules with least-privilege access to Secrets.
- Restrict Secret access to specific containers.
- Consider using external Secret store providers.

Pods

A pod is a container in kube-land.

Think a small, self-contained virtual machine that uses the host computers resources as much as possible, but still tries to seperate itself.

Deployments

Deployments just define a set of pods.

i.e. One deployment might contain 100 pods across 20 nodes.

Service

This is a definition on how a service running on a pod should be exposed to the cluster, or to the rest of the network.

i.e. A Service might be created for port 22 on the pod `ssh-server` that is running on `node01`.

Namespace

Namespaces are a way to group different resources, and can be targeted by RBAC and network controls to split these resources into logical segments of our cluster.

Note: Network segmentation is not enabled by default.

So what do we want

These are the permissions that Systems Administrators might have that we **want**.

- Permission to create pods (can allow us to pivot)
- Permission to retrieve secrets (the goods)

So what do we want

These are the permissions that Systems Administrators might have that we **want**.

- Permission to create pods (can allow us to pivot)
- Permission to retrieve secrets (the goods)
- Lists of services inside the cluster (helps us find what to hack)
- Environment variables used to launch applications (the goods, but like, on disk)

Quick commands to remember

Get a list of secrets:

```
kubectl get secrets
kubectl get secrets --all-namespaces
```

Retrieve the contents of a secret:

```
kubectl get secret secretname -n namespace jsonpath='{.data}'
```

Get a list of services:

```
kubectl get services --all-namespaces
```

Important on-disk things to remember

`/proc/self/environ` usually contains a bunch of environment variables that define all the services in the cluster.

`/var/run/secrets/kubernetes.io/serviceaccount/token` is the default service account mounted, by default, inside all pods.

Quick summary

What we need:

- Kubernetes-cli configs
- RCE inside a Pod

What we want to get:

- Secrets
- RCE on a node

Okay let's tell a real-world story

We have been contracted to do an internal penetration test, on-site at *ACME*[1].

We've performed initial enumeration and discovered the following:

- Network is quite small with a few **laptops**, most of these laptops are for developers.
- SMB Signing is enabled
- LDAP Signing is **not** enabled
- MAQ is set to the default (10)
- No Coerced Authentication

[1] ACME is a fake company name

First attack: RBCD

Every morning, developers arrive and plug their laptops into the network.

Idea

Relay the authentication of the laptops machine account to:

- Add a new machine to the network
- Give this machine delegation privileges over the laptop
- Tell the laptop I'm Domain Admin and shell in
- Profit?

See: https://chryzsh.github.io/relaying-delegation/

Uh oh

Well that worked, but now we are stuck.

Turns out the local user on the workstation doesn't have Active Directory priviledges.

The Active Directory domain is well configured and doesn't have many issues (easy since it's really small).

We are mostly stuck on developers workstations.

Look around

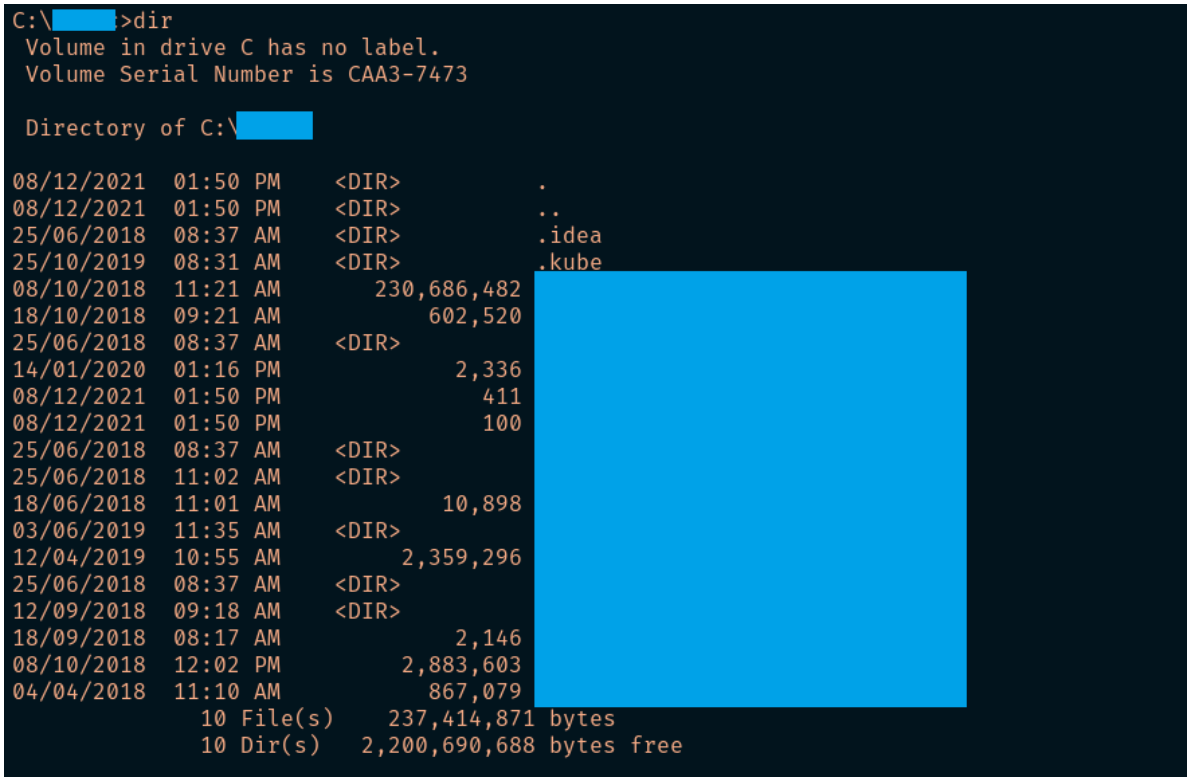
Let's look around this machine and see what juicy stuff we can find.

We *know* that this developer probably has access to something right?

Look around

Let's look around this machine and see what juicy stuff we can find.

We *know* that this developer probably has access to something right?



Enumerate

We can use https://github.com/rajatjindal/kubectl-whoami to find out information about our user. But let's just check our permissions first:

Can we list namespaces? (**yes**)

```
kubectl get namespaces
```

Can we access secrets? (**yes**)

```
kubectl get secrets -n cattle-system
```

Can we deploy pods? (**yes**)

```
kubectl auth can-i create pods --all-namespaces
```

Okay but what can we get with this? (Also what is `cattle-system`)

Rancher

After looking around, we discover that we are currently inside a cluster *hosting* Rancher.

Unfortunately, not many articles will tell you “here’s how you break Rancher” or “here’s where Rancher stores it’s configurations for clusters”

You kind of have to go it alone...

Some hints

Okay so we see a `cattle-system` namespace, and searching the code we find:

```
const (
    statusAnn                  = "cattle.io/status"
    System                     = "cattle-system"
    GlobalNamespace            = "cattle-global-data"
    NodeTemplateGlobalNamespace = "cattle-global-nt"
)
```

What do you think this namespace is used for?

More discovery

Alright, let's grab secrets in the cattle system:

```
kubectl get secrets -n cattle-system
```

We find some secrets

```
NAME                  TYPE                  DATA    AGE
c-c-iyjut             Opaque                1       2y97d
c-c-ulqmn             Opaque                1       2y201d
c-c-yzkat             Opaque                1       2y242d
... excerpt ...
```

But what are they?

Some hints

Looks like `c-iyjut` etc. is some sort of identifier for a cluster?

```
// isLegacyCluster returns true if the cluster name for a
clusters.provisioning.cattle.io/v1 or
// clusters.management.cattle.io/v3 cluster name matches the
regex for a legacy cluster (c-XXXXX|local) where XXXXX is a
// random string of characters.
```

Alright let's grab the secret

Enough beating around the bush, let's just get the contents of these secrets:

```
kubectl get secret -n cattle-system c-c-iyjut jsonpath='{.data}'
```

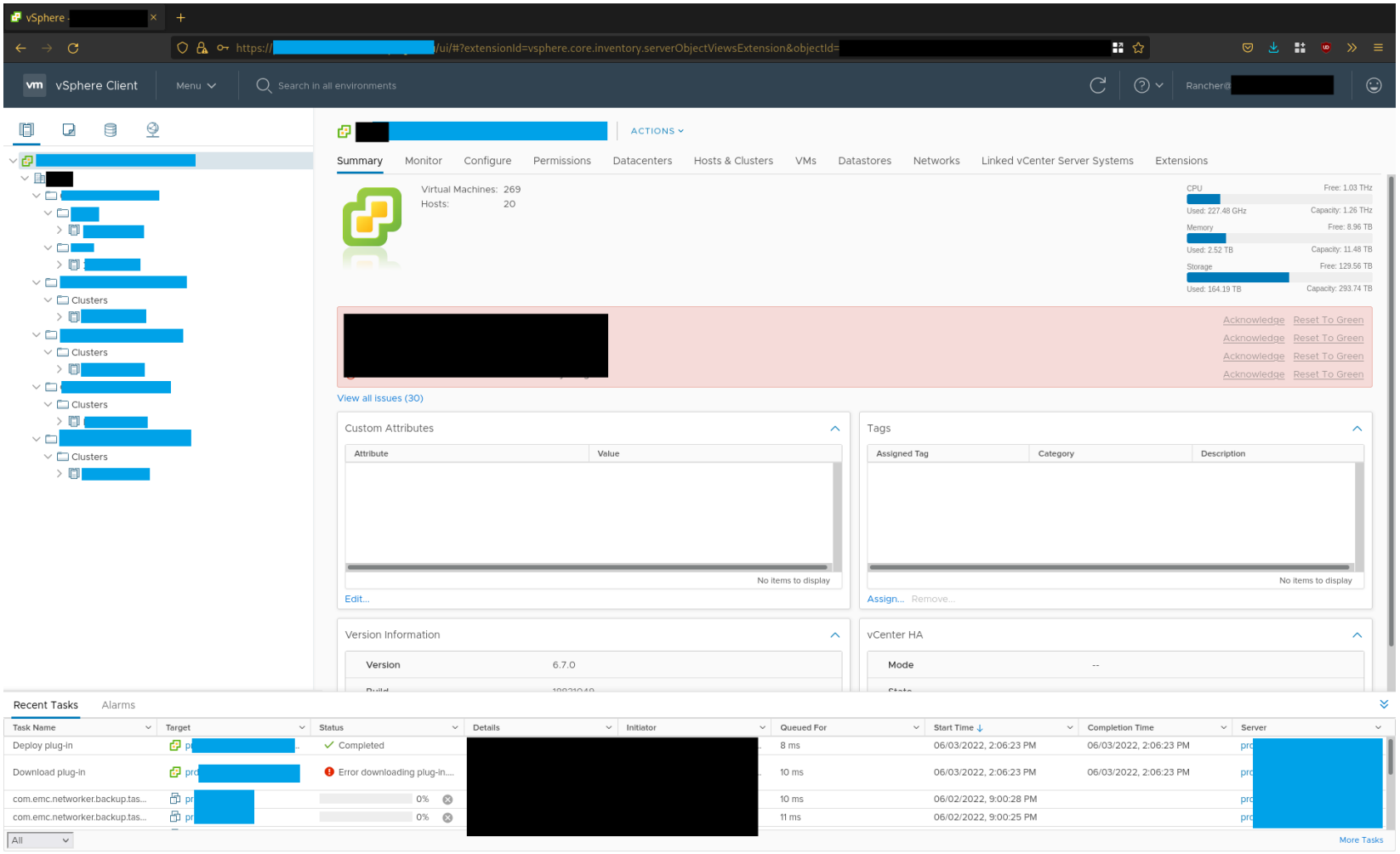
This returns a *massive* base64 encoded blob:

```
┌──[~]
└─$ kubectl get secrets -n cattle-system c-c-iyjut
{
    "apiVersion": "v1",
    "data": {
        "cluster": "eyJrZXkiOiAiaGV0d3NmbXBlcXVlcmd4eGFsZnNwcG1rbm1yZWl3aGJ0Z3llZGN1ZHhnaG5meXpr
aW5pcmZta2J1Z3JocHBuZ2xyY25meHd0dXFkdmttcml5ZnJ1ZWh2YXF2d3pkZXFjb3JlZHVveWtheXl3cmxkZG1md3l6bWxn
bXhsZnFxaWNwbmxldG94cWt5dHB6Z2RpZXp4dGxhb2hxc2JkeGNibHp2aHN0ZmttcG5uY3lqcmVwaGt2bWVrb3B4YWxybnlr
dnBybGZqYnVieGV2a2Zka2JhbXFkaHpvZGRxaWdybW93ZW11dmlpaG5qcXFmamlqYnJhenpnaGF3endwZWdwYXFvaXFvYnpi
eWtpYWZtZndiaHVoY3dmbXZiZHdkbWxzZW1rZ3hoZGt4aXZtb3JydGlmc3F5em1reG1raHhvcHloaWx3dGV3aXp1emptcWt5
ZnBxZ2ppYWhrZ2luY3B4eXV5ZXNodGNycGRtc3dxa25xcGp0cGtpZ3ZwYXFzdXRvZGNhaXVtemJmaHNmcWVlZGp0d2dwa3Fv
ZnhqZnZ3b2RhZmR4amdhd2ZzanR2bHR6bHRsZnhzcnNnYmNzb3hxeGlnd2RqanprdG90cHVvcHFwZ2VkdWF1cXN4cHZ4ZHNr
bnlmZ2lqbWt6enB4aGh5aXlhZWh4Zmx4aWxzZHZheWJmdHZrZXphdWtmYm5vdW1xeGp1Y3ZlcGVjcnhoeXBudmV0c3hpcWJh
cnVseXRnemNma3JyYnlybG5naWJ6aHFiZXVzZXhmcXJmeXdoYXVsbXp0ZGZwa21xa3hhY293cW1heWxpYmxhdXlzY2dhdHNq
YmtwdWl2Z2ViY2ttcmlheWlseHJhYW5rZGl5a2xxdGd3aXB0cG9pam9qdXRlcW93amFjaWtwd2N0cXN3dm51bHN5bGd1c2tm
YmFteHp0Ym1oaXFhbGtianZ4ZGplZ2lqa3NhY2R5aHp2YnpscnBxemVxb2Z5bHpyYml0ZGh2ZGJmZHJ4Y29oYXJ1em5oZm1q
YXNyeWFqd2xocnF6a3V4a3BreXdvYWJiamFyZ3l1YWR3d3pseWtjYWR2Z2xzeHRucGN5bWhwYXh6cGZza21na3Nxc251d2l2
bWV4YWV5cmFvdWJxbXNxa2JqZXF6aW52Z3hqYXV1ZnFrbGN0Y2xwaHJyZm9yeHN0b2NlcWt6ZHFzZnFib2V4am9xaW14d2Jy
Z25rZWdkZ3Vta2xmamZxbm9qbmhnZWpld3J3bXp0Z3NkYndtbGpocW50bGp1YXNxb3B4bGFobHl2ZWJsbmJubHFvbWRidWRn
Y3F0bmlpc3drcGt0Z2FlYWl4aG5jZWhqcHh5bXNid2hxb2NucGZpemZubHh6amZ1dmN4dnpzcGpwZmxjenhlZG1teWt4aGpn
```

What's inside?

```
cloud_provider:
    name: vsphere
    customCloudProvider: |-
        [Global]
        user = ""
        password = ""
        insecure-flag = "true"
        soap-roundtrip-count = "0"
        [VirtualCenter ACMECluster]
        user = "Rancher@VSPHERE.LOCAL"
        password = "ea3aet6oohah5Kei7ahva7Po"
        port = "443"
        datacenters = "/ACME"
        ...
```

Juicy



Note: The machine being shown, but is redacted is the Domain Controller.

Any ideas?

Any ideas?

- Take a memory dump of the Domain Controller

Any ideas?

- Take a memory dump of the Domain Controller
- Dump LSASS from the memory dump

Any ideas?

- Take a memory dump of the Domain Controller
- Dump LSASS from the memory dump
- LSASS contains the NT hash of the Domain Controller

Any ideas?

- Take a memory dump of the Domain Controller
- Dump LSASS from the memory dump
- LSASS contains the NT hash of the Domain Controller
- Domain Controllers can perform a DCSync

Any ideas?

- Take a memory dump of the Domain Controller
- Dump LSASS from the memory dump
- LSASS contains the NT hash of the Domain Controller
- Domain Controllers can perform a DCSync
- Profit?

Summary

- Kubernetes is large, but just focus on grabbing secrets for quick wins
- Don’t ignore kubernetes configs on workstations
- Developers often store credentials in cluster secrets
- Kubernetes is fun[1]!

[1]Still to be decided

I want to learn more

Kubernetes Goat is a playground with writeups about Kubernetes vulnerabilities.

CISA's Kubernetes Hardening Guide is a good entry-way to understanding how to protect a cluster.

I have a couple posts (some still aren't complete sorry) on my blog: f3rn0s.xyz

Q/A