

# Configuration Manual

MSc Research Project  
Cloud Computing

Tenzin Tsephel  
Student ID: x21176574

School of Computing  
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Tenzin Tsephel
<b>Student ID:</b>	x21176574
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2022
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Vikas Sahni
<b>Submission Due Date:</b>	14/08/2023
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	1330
<b>Page Count:</b>	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	Tenzin Tsephel
<b>Date:</b>	7th August 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Tenzin Tsephel  
x21176574

## 1 Introduction

The kube-hunter implementation, containerization, and deployment are all covered in the configuration manual, which focuses on enhancements made to probes and checks utilizing the Role-Based Access Control (RBAC) algorithm. The project was created in Python, packaged as a container image using a Dockerfile, and intended for deployment within a Kubernetes pod.

Step-by-step instructions are provided in the manual for setting up the development environment, cloning the repository, submitting contributions to the project, and other crucial activities. Additionally, it contains extensive information about the unusual technology stack employed, which consists of Docker for containerization, Kubernetes for orchestration, and Visual Studio Code as the development editor. The goal of this manual is to ensure that the setup procedure is well understood and to help users and developers replicate the development and deployment environment. This makes it possible for the kube-hunter project to be consistently tested, developed, and used, guaranteeing that all stakeholders can interact with the system successfully. The main source code can be found here: [Kube-hunter Repository](#)

## 2 Cloning the Github Repo

To download the project's code to the local machine, it must first clone the repository. The terminal would be used to accomplish the following: Go to the directory where the project is to be kept by navigating there. Using the supplied URL, clone the repository: `git@github.com:aquasecurity/kube-hunter.git` is cloned using `$git`

```
$ git clone git@github.com:aquasecurity/kube-hunter.git
$ cd kube-hunter
```

Enter the kube-hunter directory from the forked repo just made: The procedures stated in the last letter can be done now and should be in the kube-hunter directory. The cloning was made with a forked repository rather than the original one as it is a smart idea with the intent to contribute to the project. So that it can send a pull request to the original repository after pushing the changes of the fork.

## 3 Adding the RBAC Algorithm into the code

The `kube_hunter` directory houses all of the vulnerability checks that are currently in use (collectively referred to as "hunters") as well as the fundamental functionality of kube-hunter.

A subdirectory called `modules` can be found inside the `kube_hunter` directory, as shown in the Figure 1 and it has further subdirectories that group the hunters into categories based on the Kubernetes components they are intended to examine. There are directories for things like finding, hunting, reporting, etc. The modules in the `discovery` directory are in charge of finding the various parts of the Kubernetes cluster. There are several modules that carry out vulnerability scanning in the `hunting` directory. A new Python module in the `kube_hunter/modules/hunting` directory is created since the RBAC Policy algorithm is directly tied to the Misconfiguration Check.

```
tsephel24@Tenzins-MacBook-Air modules % cd hunting
tsephel24@Tenzins-MacBook-Air hunting % ls -l
total 272
-rw-r--r--  1 tsephel24  staff   183 25 Jun 11:26 __init__.py
-rw-r--r--  1 tsephel24  staff  5378 25 Jun 11:26 aks.py
-rw-r--r--  1 tsephel24  staff 24381 25 Jun 11:26 apiserver.py
-rw-r--r--  1 tsephel24  staff  1635 25 Jun 11:26 capabilities.py
-rw-r--r--  1 tsephel24  staff  1974 25 Jun 11:26 certificates.py
-rw-r--r--  1 tsephel24  staff  9280 25 Jun 11:26 cves.py
-rw-r--r--  1 tsephel24  staff  1537 25 Jun 11:26 dashboard.py
-rw-r--r--  1 tsephel24  staff  6163 25 Jun 11:26 etcd.py
-rw-r--r--  1 tsephel24  staff 45914 25 Jun 11:26 kubelet.py
-rw-r--r--  1 tsephel24  staff  5484 25 Jun 11:26 mounts.py
-rw-r--r--  1 tsephel24  staff  4156 25 Jun 11:26 proxy.py
-rw-r--r--  1 tsephel24  staff  2146 25 Jun 11:26 secrets.py
tsephel24@Tenzins-MacBook-Air hunting % █
```

Figure 1: Algorithm structure Flow

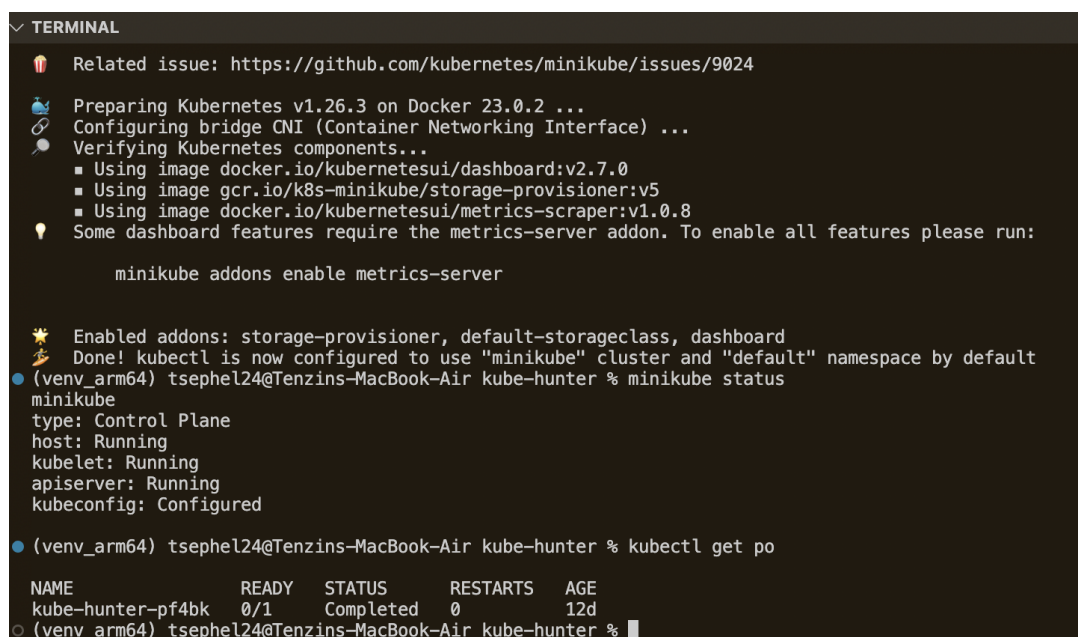
A Kubernetes cluster's Role-Based Access Control (RBAC) setup can be examined and evaluated using the supplied algorithm. The script goal is several tests on role bindings across namespaces using the Kubernetes client library. It first recognizes extremely permissive roles, which are those that use wildcards ("\*") in resources or verbs. A related vulnerability event is recorded if certain overly permissive roles are found. If not, it examines the role-binding subjects further to see whether they are of the types "ServiceAccount" or "User." The script then uses the `check_access` method to verify access for each resource and verb in the rule, recording any violations of the least privilege principle. This thorough analysis provides information on possible RBAC configuration errors, strengthening cluster security.

## 4 Kubernetes Cluster Setup

Using a local Kubernetes cluster is strongly advised for the effective execution of the `kube-hunter` modification. One popular option, `Minikube`, offers a single-node Kubernetes cluster inside a Virtual Machine (VM) on the local system, making it appropriate for individuals looking for straightforward development and testing. Install `kubectl`, the command-line interface for running tasks against Kubernetes clusters, first (`$brew install kubectl`), after updating Homebrew (`$brew update`) to ensure that it has the most recent package information. To make it easier to create a local cluster, continue by installing `Minikube` (`$brew install minikube`) and starting it (`$minikube start`). Combining `Minikube` and `Kubectl` makes it possible to manage pods, services, and deployment ef-

fectively. The given Figure 2 which makes it possible to test the Kube-Hunter script and its RBAC modification.

```
$ brew install kubectl
$ brew update
$ brew install minikube
$ minikube start
```



```
TERMINAL
Related issue: https://github.com/kubernetes/minikube/issues/9024
Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
  ■ Using image docker.io/kubernetesui/dashboard:v2.7.0
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ■ Using image docker.io/kubernetesui/metrics-scrafer:v1.0.8
Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

Enabled addons: storage-provisioner, default-storageclass, dashboard
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
(venv_arm64) tsephel24@Tenzins-MacBook-Air kube-hunter % minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

(venv_arm64) tsephel24@Tenzins-MacBook-Air kube-hunter % kubectl get po
NAME                READY   STATUS    RESTARTS   AGE
kube-hunter-pf4bk   0/1     Completed 0           12d
(venv_arm64) tsephel24@Tenzins-MacBook-Air kube-hunter %
```

Figure 2: Minikube status after the start

The cluster can be accessed via a number of commands, including `$minikube status`, `$minikube stop`, and `$minikube delete`, which can be used to eliminate the cluster if necessary. Using a local environment like Minikube makes testing quick and affordable, and it's the best option for inspecting the kube-hunter modification to make sure the RBAC configuration checks are implemented properly.

## 5 Push the code to DockerHub

The next step is to create a Docker image with the required kube-hunter change and upload it to Docker Hub after the local Kubernetes cluster has been set up using Minikube. Building the image and uploading it to the repository may be done in this way `$docker build` and `$docker push` to the DockerHub Account as can be found in the Figure 3. The Role-Based Access Control (RBAC) configuration of the cluster, including ClusterRoles, ClusterRoleBindings, and Jobs, may then be managed. Using commands like `$kubectl apply clusterrole rbac-check-role` and `$kubectl apply -f rbac-config.yaml`, it is feasible to remove existing setups. These configurations can be defined and used by using specialized YAML files, such as `ClusterRoleBinding.yaml`, `rbac-deployment.yaml`, and `rbac-job.yaml`.

```
$ docker build -t tsephel24/kube-hunter-rbac.
```

```

$ docker push tsephel24/kube-hunter-rbac
$ kubectl apply clusterrole rbac-check-role
$ kubectl apply -f rbac-config.yaml
$ kubectl apply -f rbac-job.yaml

```

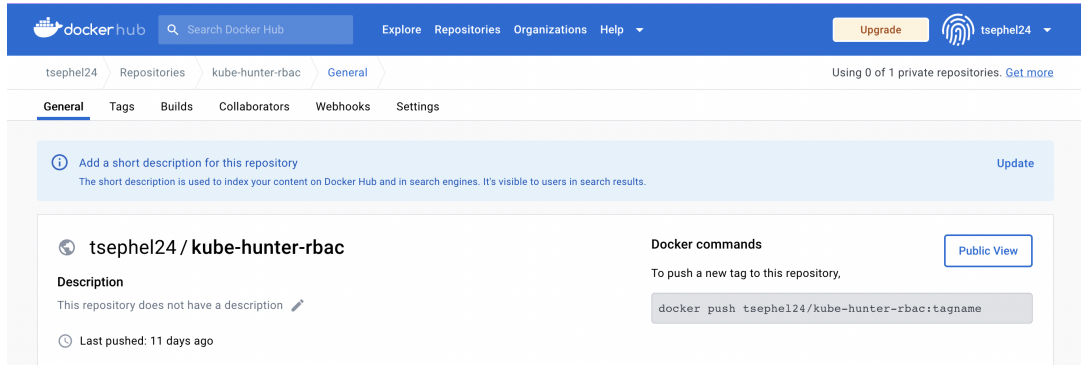


Figure 3: Image in DockerHub, latest change captured

Using `kubectl logs kube-hunter-id` and `kubectl describe pod pod-id`, the cluster may be monitored and debugged. This procedure makes sure that the RBAC setup is looked at and any weaknesses are found. The `kube_hunter` library is apparently functioning as expected, and the RBAC configuration appears to be secure, according to the logs. Examining the final report or warnings/errors is crucial to figuring out whether the scan was indeed successful and served the intended purpose.

```

$ kubectl logs {pod-id}
$ kubectl describe {pod-id}

```

The whole procedure required to build, push, and run the modifications made within the code, as well as manage RBAC in the cluster, is summarized in this paragraph. Please feel free to modify as necessary!

## 6 Running it as a POD within the cluster

Following Minikube creation of the local Kubernetes cluster, generate and manage pods to test the functionality of the RBAC configuration. It is also good to ensure that containers are running as expected as the Figure 4 illustrates. Apply the ClusterRole and ClusterRoleBinding first using the `kubectl apply -f` command and the relevant YAML files, such as `ClusterRoleBinding.yaml` and `rbac-config.yaml`. Another way to build the RBAC-related Job is to save the script into a YAML file, such as `rbac-job.yaml`, and then apply it. The `kube-hunter` pod will be deployed to test the RBAC algorithm within the cluster once these roles and bindings have been established. This might have pod restarts throughout this process as a result of unclean exits, which can be brought on by problems like unhandled exceptions or memory conditions.

Commands like `kubectl logs pod-id` and `kubectl describe pod pod-id` can be used to identify these crashes where the Figure 5 provides insight into it. These will shed light on the pod's condition and any underlying problems that contributed to the crash.

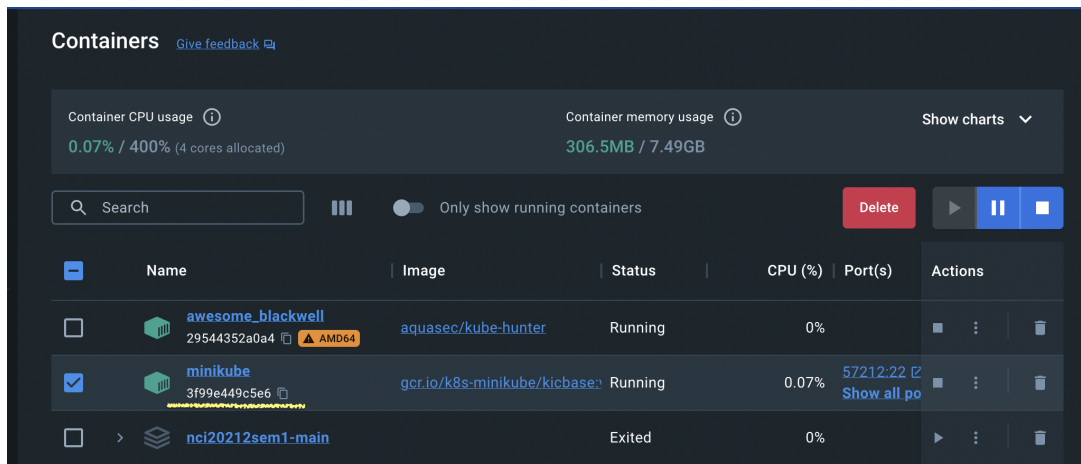


Figure 4: Minikube status in Docker

To deploy the script, examine the logs, and get the ClusterRole and ClusterRoleBinding details, it can be run into a variety of kubectl commands. By following these steps, it can be easily concluded that the local environment is set up to investigate the kube-hunter modification and validate the RBAC configuration checks, providing a rapid and affordable option to carry out the necessary testing.

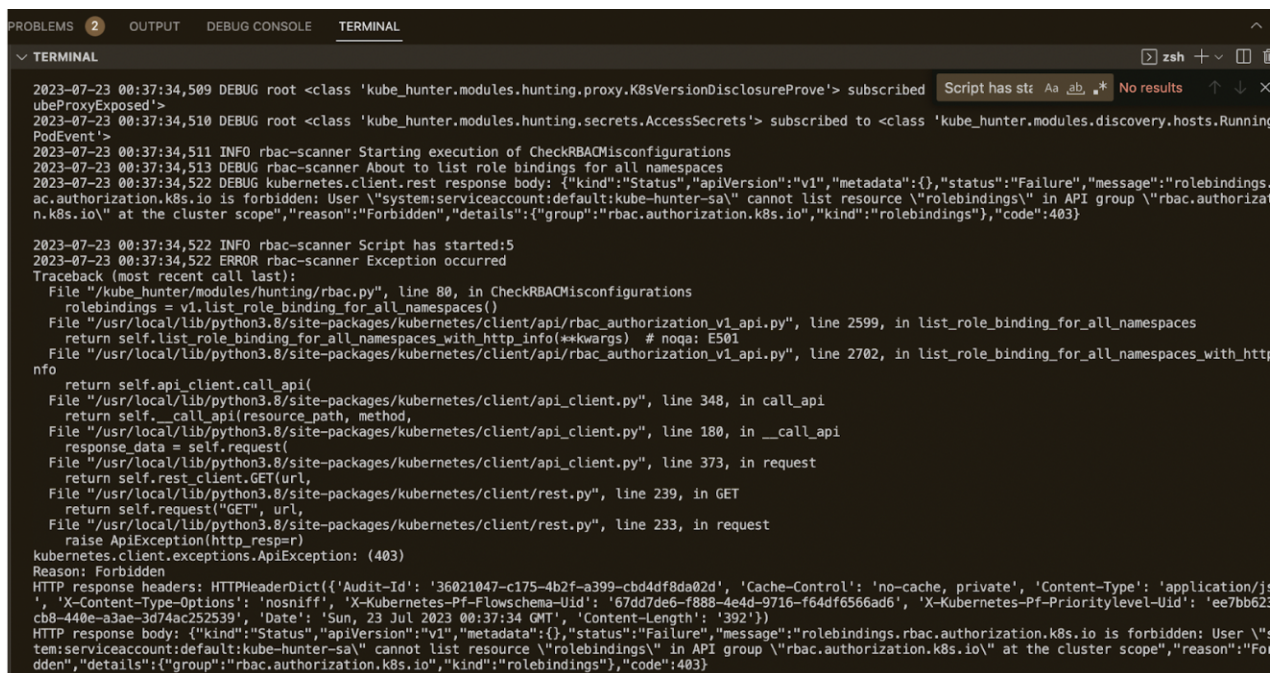


Figure 5: Output of the Kubectl commands