Online appendix

I. Development

Pomegranate-suite was implemented using bash script for scanning tools and Python (version 3.10) for analysis. We created a repository on GitHub for the project to store and track changes. Our repository consists of scanning and analysis scripts, essential scanner files under the tools folder, the source code of the tested microservice under the apps folder and template files for outputs. After performing scans, Pomegranate-suite creates reports under the report folder, summary table file and categorized vulnerabilities under the output folder.

In the scanner module, a bash script is encapsulating all commands to run scanners. Thanks to the installed kubectl which provides communication with the Kubernetes cluster through the command line interface, tools can be deployed in the cluster and interact with the microservice. The analysis module is designed to be able to parse generated reports in different formats like JSON or XML besides classifying results based on our category approach. The summary table is generated in Markdown file format and a template is provided in the repository. Categorized files are generated in JSON format to make it possible for further processing like monitoring and analysis easier.

In the analysis module, a mapping strategy is necessary to be able to classify results from tool reports. Therefore, we went through the policies belong to used open-source tools in their documentation. After studying policy, we mapped and addressed policies in tools with our classification approach in this thesis. The policies [1][2][3][4][5][6][7] might change in different versions of scanning tools but major changes are rarely introduced in tools which have high community reputation. When a new Kubernetes or Docker version is introduced, it is important to ensure that classification mappings are aligned with the latest policies.

II. Benchmarking and Testing Environment

This section explains how we set up our benchmarking and testing environment that is utilized in this thesis work. For a selection of the most convenient security scanning tools, we not only checked their documentation and community reputation but also performed benchmarking by deploying them in our environment. Each scanning tool was installed and used to scan candidate microservice based on its known security facts. After the implementation of the Pomegranate, we used our testing environment to run through prepared infrastructure with exposed microservice on the Kubernetes cluster.

To have an isolated, portable and easy to a setup test environment, we utilized Vagrant and VirtualBox. Vagrant is a tool to build, configure and manage virtual machine environments easily by automation. VirtualBox is a virtualization software that provides to run Operating Systems (OS) in another OS. Our Vagrantfile creates a Linux Virtual Machine (VM) (an Ubuntu distribution ubuntu-focal-20.04) according to the configuration mentioned in the file. Vagrant also has a provisioning feature that enables installing software in VM and altering configuration easily by using multiple provisioners like Puppet agent, Ansible playbooks and Chef. This capability provides installing tools like Kubernetes, Docker and Helm in VM automatically. Vagrantfile also allows configuring resource utilization like CPU and memory management besides network settings.

The test environment can be moved and run in any host machine where Vagrant and VirtualBox software thanks to its isolated and portable features are installed. Since vulnerable docker images and insecure network settings for microservices are used for benchmarking tools, virtualized test environment also helped to protect the host machine from any unintended behaviors and security issues.

Since it is aimed to test containerized microservices deployed on Kubernetes cluster, VM is provisioned to install Docker as containerization technology, Kubernetes command line tool (kubectl), minikube to create a local Kubernetes cluster and helm package manager to deploy tested microservice. The list of software installed in the environment and their version information are given in Table II.

TABLE I
Software Version Information in Test Environment

Software	Version
Kubectl	v1.24.4
Minikube	v1.26.1
Docker	v20.10.18
Helm	v3.10.1
Vagrant	v2.3.0
VirtualBox	v6.1.38

The list of used open-source scanning tools in this research and version information are given in Table II. Open-source security scanning tools are highly diverse in terms of how they can be installed and run. While one tool can be installed only as binary to scan, another tool might be containerized and run in the cluster or directly on the host machine. Ideally, each tool is supposed to be containerized and scanned in a container without having any dependency. For the scanners which aim to detect Kubernetes weaknesses and misconfigurations, it is preferred to run the containerized image as a pod in Kubernetes minikube cluster. Because deploying the tool as a pod in the Kubernetes cluster with a tested microservice gives

the opportunity to achieve more findings. It is clearly stated in kube-hunter [8] documentation that this way of deployment may reveal significantly more vulnerabilities.

TABLE II
OPEN-SOURCE TOOLS VERSION INFORMATION

Tool	Version
Docker Bench Security	v1.3.6
trivy	v0.34.0
kube-hunter	v0.6.8
terrascan	v1.17.0
kube-bench	v0.6.10
kubesec	v2.12.0
kubescape	v2.0.175
kubeaudit	v0.20.0
popeye	v0.10.1
datree	v1.7.1
OWASP ZAP	v2.12.0
nmap	v7.80

REFERENCES

[1] CIS Docker Benchmarks, https://www.cisecurity.org/benchmark/docker, 2022/11/23
[2] CIS Kubernetes Benchmarks, https://www.cisecurity.org/benchmark/kubernetes, 2022/11/23
[3] Terrascan policy, https://github.com/tenable/terrascan/tree/master/docs/policies, 2022/11/14.
[4] Docker-bench-for-security policy, https://www.cisecurity.org/benchmark/docker, 2022/11/14.
[5] Kube-hunter policy, https://aquasecurity.github.io/kube-hunter/kbindex.html, 2022/11/14.
[6] ZAP policy, https://www.zaproxy.org/docs/alerts/, 2022/11/14.
[7] Trivy policy, https://avd.aquasec.com/, 2022/11/14.
[8] Kube-hunter tool https://github.com/aquasecurity/kube-hunter, 2022/05/14.