A NOVEL CONTAINER ATTACKS DATA SET FOR INTRUSION DETECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HALE BERA OĞUR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2022

Approval of the thesis:

A NOVEL CONTAINER ATTACKS DATA SET FOR INTRUSION DETECTION

submitted by **HALE BERA OĞUR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil KALIPÇILAR
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit OĞUZTÜZÜN
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Pelin ANGIN
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Ali Hikmet DOĞRU
Computer Engineering, METU _____

Assoc. Prof. Dr. Ahmet Burak CAN
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Pelin ANGIN
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Hale Bera OĞUR

Signature :

ABSTRACT

A NOVEL CONTAINER ATTACKS DATA SET FOR INTRUSION DETECTION

OĞUR, Hale Bera
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Pelin ANGIN

May 2022, 53 pages

Recent years have witnessed a rapid increase in the use of the cloud, and especially the container technology, which is very convenient to use in the cloud environment due to its ability to deploy microservices quickly and easily. A number of studies have been carried out on the security of this technology since the day it started to be used. However, ensuring inclusive security is still a critical need. As containers are a relatively new technology, it is essential to discover their security vulnerabilities by testing them with continuous and up-to-date attacks to develop effective defense systems.

Today, machine learning-based intrusion detection and prevention systems are an effective option for securing many platforms including containers. The major issue with these approaches is the need for appropriate and comprehensive labelled data sets, which is a common problem in any machine learning-based study.

In this thesis, we describe a novel public container attacks data set we have created for machine-learning based intrusion detection, which focuses on container attacks extracted from the Common Vulnerabilities and Exposures (CVE) platform for the

period 2019-2022. The data set comprises attacks simulated on vulnerable container images deployed in a Kubernetes orchestration environment. We believe the data set will be instrumental for advancing intrusion detection research and practice for containers, which will be increasingly widespread in the years to come.

Keywords: Container security, cyber attack dataset, Kubernetes

ÖZ

SIZMA TESPİTİ İÇİN YENİ BİR KONTEYNER SALDIRILARI VERİ KÜMESİ

OĞUR, Hale Bera
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Pelin ANGIN

Mayıs 2022 , 53 sayfa

Son yıllarda bulutun, ve özellikle mikroservisleri hızlı ve kolay bir şekilde dağıtabilme özelliğinden dolayı bulut ortamında kullanımı oldukça uygun olan konteyner teknolojisinin kullanımında hızlı bir artış yaşanmıştır. Bu teknolojinin kullanılmaya başlandığı günden itibaren güvenliği konusunda bir takım çalışmalar yapılmıştır. Bununla birlikte, konteynerler için kapsayıcı güvenliğin sağlanması hala kritik bir ihtiyaçtır. Konteynerler nispeten yeni bir teknoloji olduğundan, etkili savunma sistemleri geliştirmek için sürekli ve güncel saldırılarla test edilerek güvenlik açıklarının keşfedilmesi önemlidir.

Günümüzde makine öğrenme tabanlı saldırı tespit ve önleme sistemleri, konteynerler de dahil olmak üzere birçok platformun güvenliğini sağlamak için etkili bir seçenektir. Bu yaklaşımlarla ilgili en büyük sorun, herhangi bir makine öğrenme tabanlı çalışmada yaygın bir sorun olan uygun ve kapsamlı etiketlenmiş veri kümelerine duyulan ihtiyaçtır.

Bu tezde, 2019-2022 dönemi için Ortak Güvenlik Açıkları ve Etkilenmeler (CVE)

platformundan çıkarılan konteyner saldırılarına odaklanan, makine öğrenme tabanlı sızma tespiti için oluşturduğumuz yeni bir genel konteyner saldırı veri kümesi sunulmuştur. Veri kümesi, bir Kubernetes düzenleme ortamında kurulu, güvenlik açıkları bulunan kapsayıcı görüntüleri üzerinde simüle edilen saldırıları içermektedir. Oluşturulan veri kümesinin, önümüzdeki yıllarda giderek yaygınlaşacak olan konteynerler için sızma tespit araştırmalarını ve uygulamalarını geliştirmede etkili olması beklenmektedir.

Anahtar Kelimeler: Konteyner Güvenliği, siber saldırı veri kümesi, Kubernetes

To My Family

ACKNOWLEDGMENTS

I would like to express my appreciation to my thesis supervisor Pelin Angın for her encouragement, support and guidance during the thesis period. I would like to thank my thesis committee members Ali Hikmet Doğru and Ahmet Burak Can for their invaluable feedback, which has helped me improve the thesis a lot.

I would like to thank my labmates Yiğit Sever, İlter Taha Aktolga, Said Gürbüz, Meriç Karadayı, Vahab Jabrayilov, Adnan Harun Doğan, Buğra Alparslan, Batuhan Dilek, Göktuğ Ekinci, Şerif Can Tekin, Ali Kömürcü and Mert Atay for their great help and support. I would also like to thank my friends and co-workers for their patience and support.

Lastly, I would like to thank to my family.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

NIST	National Institute of Standards and Technology
CVE	Common Vulnerabilities and Exposures
NVD	National Vulnerability Database
CISA	Cybersecurity and Infrastructure Security Agency
DHS	Department of Homeland Security
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
YAML	Yet Another Markup Language
DNS	Domain Name Server
PE	Privilege Escalation

CHAPTER 1

INTRODUCTION

Advances in cloud computing systems in recent years have led to the development of applications based on the microservices architecture to meet the high performance requirements of systems using the cloud. This architecture includes a complex application structure that consists of services that exist as independent entities and interact with each other through specific APIs. Microservice architecture frequently uses a structure called container, which is much lighter and faster than virtual machines in the cloud. The rapid digital transformation experienced in recent years has led to the widespread use of container-based application structures [7]. The proliferation of container networks has revealed the potential of exposing these networks to many cyber attacks caused by the malicious capture of endpoints such as IoT devices. Recent surveys on container adoption in enterprises show that security is still the top concern standing in the way of more widespread adoption of the technology [8].

A number of studies have been carried out on the security of the container technology since its introduction. However, ensuring inclusive security is still a critical need. As a new technology, it is essential to discover the vulnerabilities that threaten containers by continuously testing them with up-to-date attacks to develop effective defense systems. In recent years, machine learning (ML) based studies have been carried out on the development of intrusion detection systems for containers. The need for comprehensive datasets, a common problem in machine learning-based studies, is an important issue to be addressed for developing effective intrusion detection models.

1.1 Contributions

Studies carried out in the field of cyber security primarily refer to Common Vulnerabilities and Exposures (CVE) [9], which contain the security vulnerabilities reported worldwide. Within the scope of the information we have obtained from the literature research conducted in this context, it has been determined that there are very few studies related to attacks against containers, most being limited to the vulnerabilities published in 2017 and 2018. Still, there is no machine learning data set study that addresses the vulnerabilities used in attacks against containers thus far.

In this thesis we aim to make a contribution to the field of intrusion detection for container-based application systems by:

- Describing a systematic process for extracting the vulnerabilities affecting containers
- Describing the implementation of attacks for forming a container attacks data set in the Kubernetes environment
- Creating a publicly available data set to be used in studies on machine learning-based intrusion detection systems for containers

To the best of our knowledge, this is the first systematic study for forming a machine learning data set for intrusion detection, which focuses on the top vulnerabilities for containers in the period 2019-2022.

1.2 Thesis Outline

The remainder of this thesis is organized as follows:

In Chapter 2, we provide background information on cloud agility challenges, containers, microservices and the Kubernetes orchestration platform. We also discuss attack vectors against container-based applications, and provide an overview of databases containing container-specific vulnerabilities as well as other security vulnerabilities.

In Chapter 3, we provide an overview of related works in the field of container security and a detailed discussion of previously created data sets for attacks against containers.

In Chapter 4, we present our methodology for forming the container attacks data set. We also describe our simulation environment for attack implementation and provide the details of our data set.

In Chapter 5, we conclude the thesis.

CHAPTER 2

PRELIMINARIES

2.1 Cloud Agility Challenges

The increasing adoption of cloud computing for a variety of applications in recent years has made agile cloud services a must. Providing high agility in the cloud faces several challenges, among which are the following:

- Virtual machine based cloud solutions are resource-demanding.
- Lightweight, portable application components are needed for quick deployment.
- Geographically-distributed mission-critical applications may suffer from limitations in network quality of service (QoS).
- Provisioning networks to deliver optimal paths between new applications and users can take hours or days.
- Network configuration is highly manual, requiring device-by-device configuration in data center networks.
- Enforcement of policies involve a complex architecture.
- Enterprises have limited control over cloud operations.

A number of solutions for these challenges have been introduced, with some resulting in paradigm changes in their respective fields. Among the major technologies that support an agile cloud are microservices, containers and software-defined networking

(SDN)/network functions virtualization (NFV). Figure 2.1 provides a summary of the enabling technologies for cloud agility.

Enabling Technology	Agility Component
Containers	Rapid service deployment
	Lightweight, high-performance components
	Easy service migration
	High service uptime / Rapid update integration
Microservices	Flexible service composition
	Interoperability
SDN & NFV	Rapid network resource provisioning
	Easy network policy definition / enforcement
	Rapid network maintenance
	Quick network failure recovery

Figure 2.1: Cloud agility solutions

While these enabling technologies have been instrumental in increasing cloud agility, they have their peculiar security vulnerabilities that need to be addressed. In this work, we focus on the first two technologies, i.e. microservices and their enabling software unit, containers, which are described in the following subsections.

2.2 Microservices

Microservices are a realization of the service-oriented architecture model for designing applications composed of small services that can be distributed and individually scaled by completely integrated delivery machines, with limited centralized administration [10]. Microservices are designed around different functionalities of the organization. Each microservice operates in its process and communicates through mechanisms of lightweightness, often using application programming interfaces (API). Figure 2.2 demonstrates the difference between monolithic applications and microservices.

Microservices tackle the drawbacks of monolithic applications. The major differences of microservices from monolithic applications can be summarized as follows:

- Microservices are small and can reboot faster when upgrading or recovering from an unsuccessful state.
- Microservices are loosely coupled, so the failure of one microservice would have little effect on other microservices.
- The microservices architectural style's fine granularity makes scaling more flexible and more efficient, as each microservice can evolve at its own pace.
- In the microservices architecture, services are distributed across servers, instead of replicating whole applications in different servers for scalability.
- Microservices provide the opportunity for continuous delivery, while monolithic applications take longer for delivery of updates.

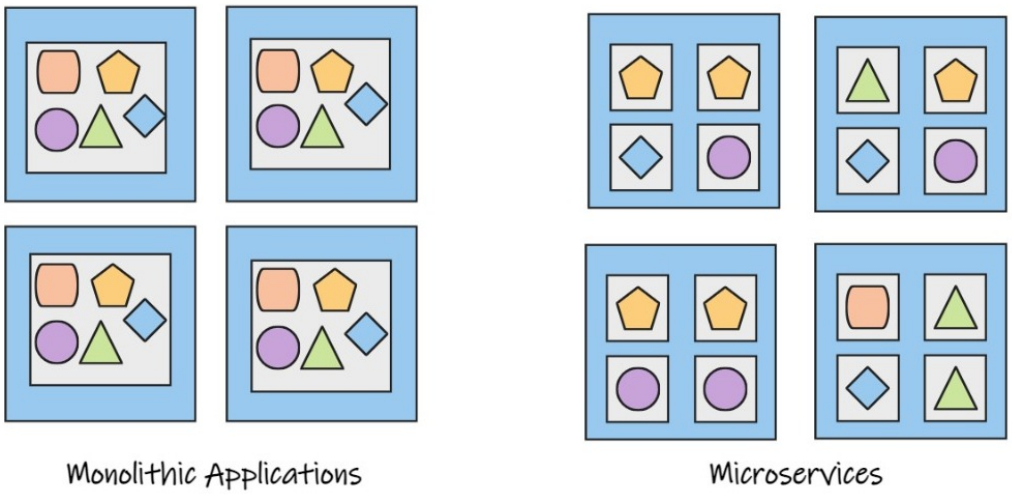

Figure 2.2: Microservices [2]

2.3 Containers

A container is a lightweight package in which a software application can be stored, carried and run on a compatible OS. It encapsulates application logic components provisioned with minimal resources (its own operating environment) required. Containers involve virtualization at operating system (OS) level rather than hardware level

and they can be run on any host without special configuration, i.e. there is no need for an embedded OS. While the requirements for running a container are are a compatible OS and a compatible runtime environment for the containers, the required components to build a microservices-based application based on containers is a platform to manage the applications and a tool for container orchestration, management, networking and security.

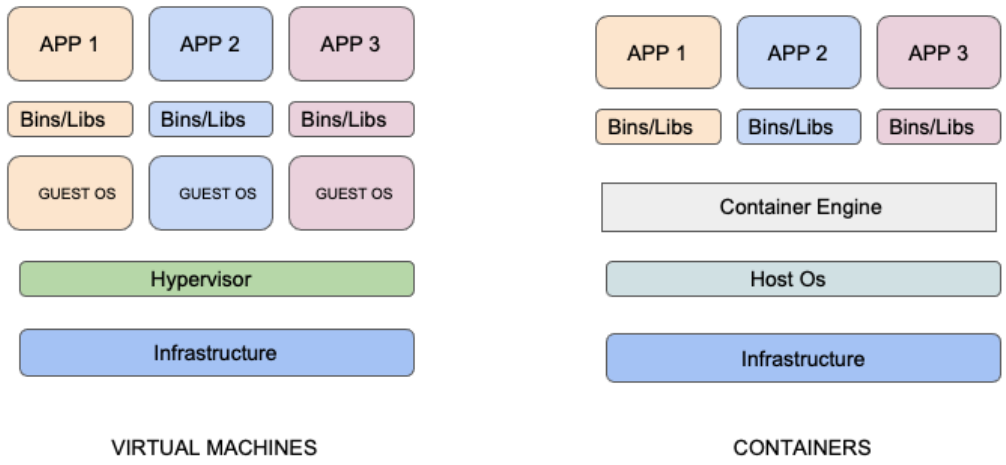

Figure 2.3: Containers vs virtual machines

The main advantages of containers over virtual machines can be summarized as follows [11]:

- Although both technologies are based on virtualization, containers use OS-level virtualization rather than hardware level virtualization, which achieves improved performance.
- In containers, application instructions do not go through a guest OS and hypervisor to reach CPU, while in virtual machines they do.
- Containers have a smaller size compared to virtual machines.
- Containers provide faster application startup as compared to virtual machines.
- Containers provide reduced footprint compared to virtual machines.

Containers provide enhanced agility as compared to virtual machines by sharing the same OS with many other containers, which achieves reduced resource consumption and increased efficiency. They also provide fast application upload and no downtime during application updates, as well as increased portability features with applications separated from the underlying infrastructure.

2.3.1 Docker

Docker [3] has been one of the most popular container technologies since the inception of containers. Docker images have an onion-like layered structure that uses union file systems to form a single coherent file system. Image build changes yield the creation of a new layer rather than replacing the whole image. This makes the distribution faster as it only sends the updated images built from bases (e.g. Ubuntu image) using instructions.

A Docker container is composed of an operating system, user-added files, and metadata. When a container runs, a new container is created with a read-write layer added on top of it. A network interface is allocated, and an IP address is assigned to the container. Figure 2.4 provides an overview of the components of Docker.

The Docker architecture consists of the following elements:

- Docker daemon: This component handles the main functionalities such as building, running, and distributing Docker containers. It listens to and communicates with clients.
- Docker client: Binary user interface that accepts commands from the user and can communicate with multiple daemons in parallel.
- Docker images (build): Template container images.
- Docker registries (distribution): Public or private stores for uploading/downloading container images.
- Docker containers (run): Hold everything needed to run applications created from Docker images.

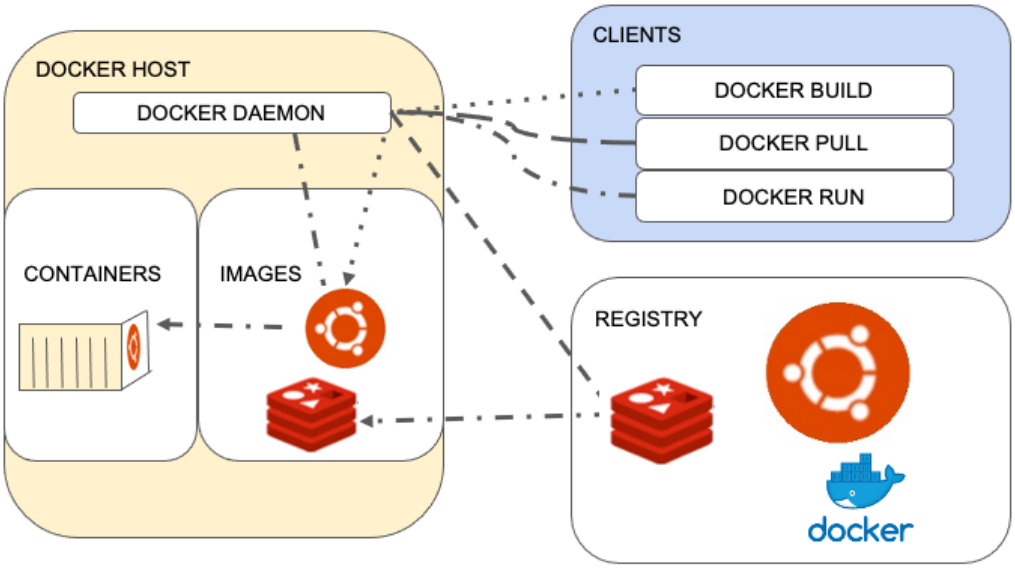

Figure 2.4: Docker overview [3]

2.4 Kubernetes

Since containers are isolated, they do not know each other. Therefore, an orchestration framework is required to handle container deployment in a way that they can communicate with each other when part of a microservices-based application. Kubernetes is a Google open source project that provides a framework for managing containerized distributed systems that support portable configuration, automation, easy deployment and creation of containers, and scalability [12]. It has become one of the most popular container orchestration platforms in recent years.

In this work, we also use the Kubernetes Environment due to its being an open-source and widely used project by many organizations, ranging from Red Hat to VMware Tanzu [13].

Figure 2.5 demonstrates Kubernetes architecture. Each node, which hosts part of the distributed application, does so by leveraging a container technology. The nodes also run two kube-proxy, which gives access to the running application, and kubelet, which receives commands from the control plane. The control plane (master node) runs the API server, the scheduler, the controller manager and etcd, which is a highly available key-value store for service discovery and shared configuration.

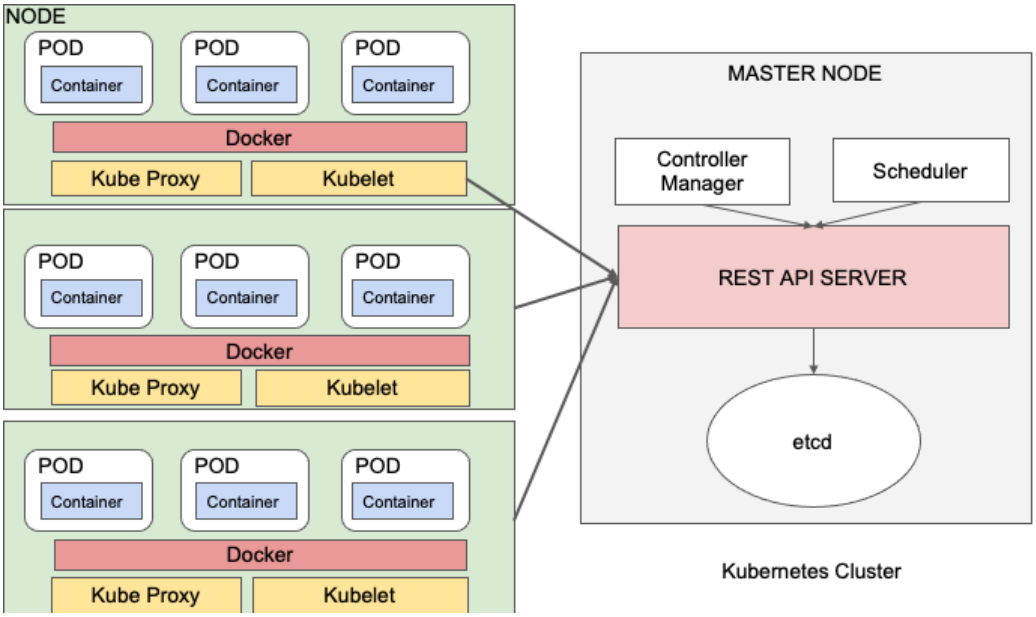

Figure 2.5: Kubernetes architecture

2.5 Container Security

This section will give the simple definitions of vulnerability, exploit, and attack terminologies with an overview of container attack vectors.

2.5.1 Definitions

Vulnerabilities: Bugs are defined as mistakes that happen during the building or coding process that create weaknesses in systems. Many of these bugs have the potential to be exploited for malicious actions and are described as vulnerabilities [14].

Exploits/Attacks: An exploit is simply defined as the behavior of the utilization of a vulnerability by using code pieces or commands, which is the attack itself [14].

Remote Code Execution (RCE): RCE is a remote host with a downloaded malicious malware exploitation attack, that can be exploited by injecting an input from the user-side, and can result in the compromising of the entire server. This attack type is commonly used for remote shell, coin mining, and mass-scanning.

Privilege Escalation (PE): PE attacks aim to gain unauthorized access to an orga-

nization's systems by finding weak points in defense systems. Mostly attackers start with a lower level of access than they need. After that point, attackers perform further attempts to elevate their current privileges and try to gain more permissions for more critical systems [15].

This thesis focuses primarily on RCE and PE attacks on containers. The reasons for selecting these two will be elaborated on in the following sections.

2.5.2 Container Attack Vectors

Figure 2.6 provides an overview of attack surfaces for a container. While some of these attacks such as insecure networking are also valid for other application structures, some such as container image related attacks are specific to containers. In any case, the isolation between containers running on the same physical machine is not as strong as the isolation between virtual machines running on the same physical machine. This creates additional security issues when containers are preferred for application deployment.

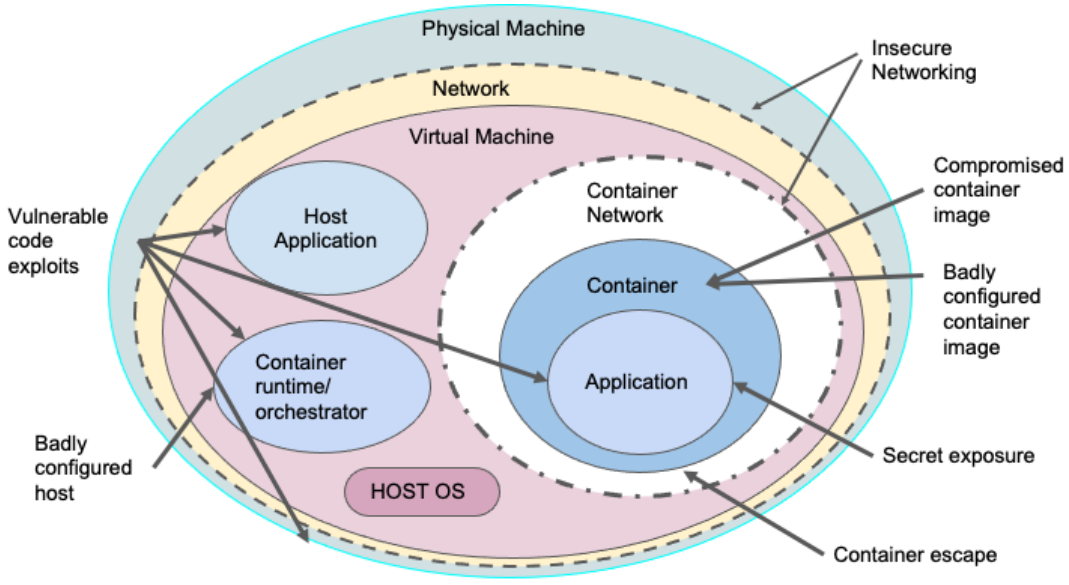

Figure 2.6: Container attack vectors [4]

The major attack vectors against containers can be summarized as follows:

- **Vulnerable application code**: The software code itself or its third-party dependencies built in an image can threaten the system's security.
- **Badly configured container images**: Shortsightedly configured container images may realize vulnerabilities during further periods of their lifecycles. We need to avoid giving over privileges to this type of image in the configuration step.
- **Build machine attacks**: If an unauthorized actor finds a way to modify the building process on an image, malicious code can be inserted in it, which can bypass the production environment.
- **Supply chain attacks**: A malicious actor can upload specific code inserted images into publicly known and trusted image repositories, where anybody can pull it without knowing what is inside.
- **Badly configured containers**: These results from running a container with unexpected privileges with an unknown configuration file.
- **Vulnerable hosts**: The host machine the container is deployed on may be vulnerable, which can result in sensitive data thefts.
- **Exposed secrets**: In a containerized environment, tokens, credentials, and passwords used in an application could be exposed if not securely transferred between the destination and source without violating their security from other components.
- **Insecure networking:** Container-to container or container-to-outside communications could lead to vulnerabilities if not properly secured.
- **Container escape vulnerabilities:** This type of threat can occur at the weak points of the container runtime by escaping from its defined locations in the environment.

MITRE ATT&CK publishes a matrix [16] including tactics and techniques for container security. This container attacks matrix is grouped under 8 tactics, which are Initial Access, Execution, Persistence, Privilege Escalation, Defense, Evasion, Credential Access, Discovery and Impact. MITRE gives detailed techniques and tactics

for each attack vector category. Following this matrix, we decided to base our data set on the two tactics, Execution and Privilege Escalation. This decision was determined by our result from OWASP's Most Hazardous Top 25 Weaknesses list analysis. Detailed information will be given under the attack selection strategies section.

Interruption of an attack without getting the last access or the sensitive information is not the natural behavior of an attacker. For that reason, from the initial discovery to gaining the highest accessing capability should be considered aggregately. Thus, in addition to these two base tactics, we also examined a few other tactics and techniques that are supportive of RCE and PE; we consider Initial Access, Persistence, Credential Access, and Discovery tactics in our attack selection process.

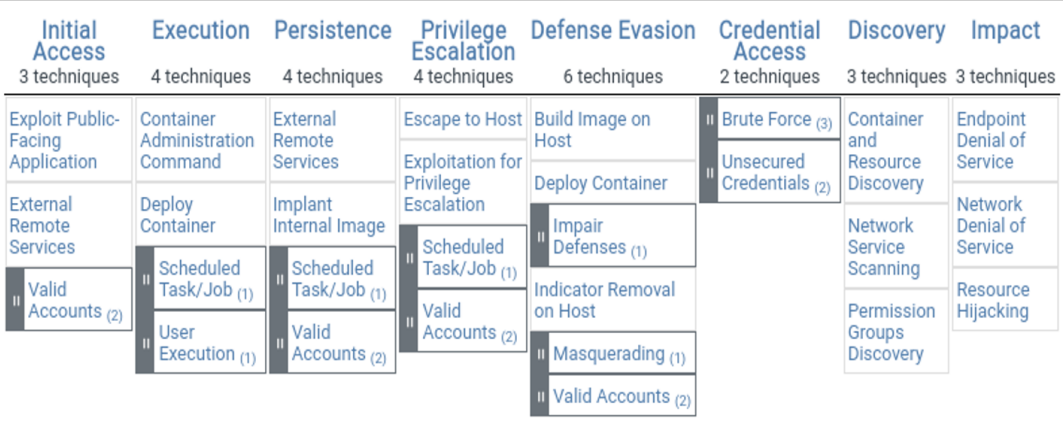

Figure 2.7: MITRE container attack matrix [5]

2.5.3 Container Security Databases

2.5.3.1 CVE

Common Vulnerabilities and Exposures (CVE) program is a benchmark for the security community and is owned, starting in 1999 by The MITRE Corporation, and it is sponsored by the U.S. DHS from CISA.

It maintains publicly exposed vulnerabilities collected by partners from all over the world. This program identifies, defines, and catalogues the collected vulnerabilities and publishes each vulnerability with a unique identifier named CVE code or number, which creates a common language among information technology and cyber security

experts.

There are 176,170 CVE Records in its database collected from 1999 to the date of retrieval of this information, which is May 14, 2022 [9].

2.5.3.2 CVE DETAILS

CVE only maintains the database containing the CVE Records containing the vulnerability details. Thus, to fill the need for additional interpretations using these records, there are additional databases.

CVE Details is a personal database maintained by Serkan ÖZKAN who is also the project owner of Open Vulnerability and Assessment Language (OVAL). The site maintains different statistics extracted from CVE Records published by CVE[6].

2.5.3.3 NVD

NIST launched a program named U.S. National Vulnerability Database (NVD) in 2005, which is a different program than CVE. It is also sponsored by the U.S. DHS from CISA.

NVD serves as a vulnerability database that is strictly synchronized with the CVE Records and immediately reflects any updates on CVE. NVD proposes enhanced information per record by giving suggestions for fixing, assigning severity level scores and impact ratings, which also enables advanced searching opportunities [17].

2.5.3.4 Other Databases

Other than the three databases mentioned above there are some large companies like Sysdig, RedHat, Tenable etc. that maintain studies and publish references, guides and white papers about container security. Among these, Sysdig is a company that provides cloud, container and Kubernetes related security tools. In 2021, the company published a guide including best practices for container security [18]. In this guide it is listed that web application security is mostly in the responsibility of the user-

side instead of the provider, which means that it is more risky relative to the others that are in the responsibility of providers and needs to be defended by comprehensive intrusion detection systems.

CHAPTER 3

RELATED WORK

3.1 Container Security Studies

Since the rise of containers as a new technology, organizations have been working on establishing security standards for them. Standards specify the common properties for a product or service in order to work by themselves and with other product or service systems in a compatible and proper [19] manner. Besides standards, there is also comprehensive documentation like policies, procedures, whitepapers, and guidelines [20].

The first standardization for this novel technology was published by the National Institute of Standards and Technology (NIST) as a Special Publication named Application Container Security Guide (NIST 800-190) at the end of 2017 [21]. In the NIST 880-190, major risks and countermeasures are handled for container technology's core components which include "Image", "Registry", "Orchestrator", "Container" and "Host OS". Although there is also a variety of reports, guides, and whitepapers published by large cyber security organizations like Sysdig, Red Hat, Tenable, etc. NIST 800-190 is considered a milestone for container security.

On the other hand, a comprehensive survey on container security done by [22] in 2019 matches the countermeasures retrieved from around 30 container security studies with the risk subtitles listed/specified in NIST 800-190. The study shows that some threats remain that NIST uncovered but are covered by other container security studies and lays emphasis on the need for additional research studies to achieve much more comprehensive standardization.

In addition to NIST 800-190 and the other studies mentioned above, MITRE has published a release (v9), last updated in April 2021, that discusses its techniques for attack tactics against container platforms. The release has based the threats against containers on a more detailed systematic than previous studies.

Our literature research focused on the range of each study, reference points of the study in terms of the standards, guides, reports, if they exist, categorization of container threats, handled vulnerabilities and attacks, and especially the attack data sets they used or created. Here, we aim to comprehensively look at the literature research on container security data sets, examine how the studies deal with container security, and determine which area the data set is needed more primarily. For this reason, we have included predominantly ML-based and also some recent image-based studies in our research.

3.2 Container Security Datasets

Among previously released container security data sets, we have seen static analysis on container images and dynamic analysis on container runtime for container security purposes under the argument of the container engines' sufficiency at protecting the application within the container and the insufficiency at container image protection from unauthorized access [23]. They take general security requirements under the division of six as a base: system integrity, system protection, service acquisition, maintenance, authentication and identification, and access control mechanism. They try to predict access control mechanisms through four use cases considering the safety of container, inter-container, and host. For this purpose, they perform vulnerability scanning analysis on popular ten docker images in 2021, resulting in the most frequent vulnerabilities being related to memory corruption/buffer over-read and DoS attack, which will lead to information leakage and file modification.

Due to having a large number of images, the DockerHub has been studied widely. Another research that focuses on container image protection [24] performed a wider analysis to present the vulnerability landscape on 2500 Docker images collected at the beginning of 2020. They extracted several statistics, such as 329 being the average

occurrence rate of each vulnerability per image from the 5,554 unique vulnerabilities having an occurrence rate of 14,031 in total. Besides, they showed the most frequently occurring ten vulnerabilities having the CVE code starting from 2010 to 2019. The types of the ten most frequent vulnerabilities are listed as related to improper input validation, code execution, restriction bypass, and overflow. The research concludes with the statement that the execution of code and overflow-related vulnerabilities are the most frequent and critical ones. This study did not present an ML-ready data set though.

The study by Cavalcanti et al. [25] focused on the performance comparison of container-level anomaly-based intrusion detection on application containers containing four specific MYSQL vulnerabilities in the CVE code range of 2012 to 2017, considering the classes, DoS and integer overflows, PE, and authentication bypassing. They constructed 200000 system call records in total. For this, they created malicious and benign containers of a specific version of a MySQL image. Half of the resulting records belong to benign, and others belong to malicious behavior, including an equal number of samples from each of the four vulnerabilities (25,000 records). They used 60 percent for training and the rest for test data to compare several different intrusion detection classifiers.

Another study was performed by Pope et al. [26]. In their research, they collected a data set of the publicly released 374 kernel memory corruption vulnerabilities for their strong relation with PE and container escape attacks, which are the two most severe threats for cloud container services. The vulnerabilities were collected from the NVD database between 2008 to 2018. In addition to that, essential metric data were extracted against the attacker on the PE and container escape attacks. As a conclusive statement, even by obtaining ROOT privilege in a container, it is still hard to escape from a public cloud container.

A previous study from Lin et al. [27] created an exploit data set for the Linux container platform and a defense mechanism for privilege escalation. They collected 400 exploits published in 2016 and 2017 related to web application, remote control, PE, and DoS according to the Exploit-DB division of the exploits into four categories. They found 274 CVEs used by these exploits (Only 24 CVEs were published between

2013 and 2015.). 235 exploits were found capable of affecting container platforms, and to fasten the evaluation process, 88 typical exploits were filtered out of 233, and 148 vulnerabilities related to these exploits were collected. However, due to some conceivable reasons, i.e., failure of exploits, in the end, they focused on the 11 PE-related exploits.

As a result of our research, we observed that studies were carried out on image files, vulnerabilities, and exploitation in order to contribute to the development of defense mechanisms for container security. We have seen that there are studies that reveal the state of the existing landscape, such as what the type of vulnerabilities can be found on the top 10 popular images of that year, as in [23] or 2500 images collected from different years independently from the time, [24], in order to shed light on the future defense mechanisms, instead of aiming to create any defense mechanism.

When we examined the scope of the studies, we saw that many reference databases such as CVE, NVD and ExploitDB were used in them. However, when we look at the inspiration points of the studies, it is seen that some of them carried out their studies based on different points such as container images [24], [23], some of them based on Linux container mechanisms [27], while some others proceeded quite differently from these, by a more specific selection of an application type, for example a MySQL application container in [25].

In summary, the types of attacks addressed in these studies, all of which have been carried out on containers, are:

- [23]: Memory Corruption / Buffer Over-Read and DoS
- [24]: Improper Input Validation, Code Execution, Restriction Bypass, and Overflow
- [25]: DoS and Integer Overflows
- [26]: Memory Corruption vulnerabilities, Privilege Escalation and Container Escape Attacks
- [27]: Web Application, Remote Control, Privilege Escalation and DoS

Here, it can be clearly seen that the vulnerabilities of relations with code execution and overflow, which are the outputs of study [24], are also emphasized in other studies. For this reason, we expected to accelerate and contribute to the literature studies on container intrusion detection systems in the coming years by a novel data set for training and testing, created with the data to be obtained from the attacks using the vulnerabilities related to two scenarios in 2021, which have the highest number of vulnerabilities of all time with 20,141.

Among container security solutions is KubAnomaly [28], which made a contribution by presenting a neural network-based detection system that will detect anomalies on the system and network part of containers, has carried out its studies based on the sub-headings of the NIST SP-800-190 Standard on orchestration and runtime risks. They compared the detection model which they proposed with many large monitoring systems such as AppArmor, Sysgig, Aqua, which offer features for container runtime risks too. For the evaluation, they carried out container runtime attacks in four categories, mainly orchestrator, network-based policy-based, and anomaly-based. Systems that offer solutions against orchestrator and network-based risks do not provide a solution for anomaly detection. For this reason, they enabled their anomaly-based solution to meet orchestrator and network-based risks at the same time. For their training and testing purposes, they used three different data sets, private, public, and real-world. For these data sets, instead of listing all the events related to container behavior, they worked on a data set with a total of 17 system calls, among which they selected the 14 root directory access features they added on top of them, and four categories: file input-output, network input-output, memory, and scheduler. The first data set is called the private data set, which consists of a simple and a complex data set that they created. Both of the data sets were created from normal user and abnormal hacker samples in order to be suitable for use for training and testing. They used JMeter, the Apache performance measurement tool, to simulate normal behavior in the simple data set, and hacker attempts that perform path traversal and DoS attacks that will affect web service containers were simulated with OWASP Zap, an open-source penetration testing tool for abnormal behaviors. A more comprehensive version of the simple data set was obtained by creating fourteen different user behaviors in the complex data set. Normal user behaviors were created from randomly

selected user events to simulate fourteen different user behaviors. In addition, to Zap attacks for abnormal behavior, SQL injection and CVE-2017-5638 vulnerability for command injection attacks were added. On the other hand, a total of 26,000 abnormal behavior samples were collected, more than half of which were SQL injection, whereas about 9,000 exhibited normal behavior for the complex data set and about 27,000 abnormal behavior. By combining these two created data sets, using the result data set, approximately sixty percent of which consists of abnormal behaviors, performance comparison was carried out primarily with only the system call logs in the data set and then with all the collected events.

Here, an important point that shows how their work was hindered due to a data set-related restrictions is that the classification model they produced according to the result of this first evaluation did not show good performance during anomaly detection for Zap attacks, probably because Zap attacks are not uniform, and there are different types of brute force, and XSS attacks are inside it. It was stated that this was caused by the presence of various types of web attacks. Since the attack data was not presented in a separate form before, it was a process that would require labor and time. It was seen that those who carried out the study preferred to perform the separation process, so the attack scope that was mastered during the training and testing stages was limited to the existing one.

They used the selection of synthetic insider test data collected by the Cert Division, a public data set, to perform secondary validation with a different data set that their models were not trained on. There is no system call data in this data set. Apart from that, there are various types, such as device and email data. Log data of malicious behavior of a user in the data set has been used. It is stated here that the categorization of abnormal behavior is done by performing the feature extraction process with unsupervised learning since the data set does not contain any label.

This article, among other machine learning-based intrusion detection systems, covers in detail a complete life cycle from start to finish, unlike the others, in the process of creating the model itself and collecting the train and test data sets in order to achieve the best performance of the system. It is a good example for those to create a data set, as it offers three different methods, such as creating a data set from scratch, using a

ready-made data set, and hunting live data with the local term.

CHAPTER 4

METHODOLOGY

4.1 Attack Selection Strategy

In this section, we describe our attack selection strategy used in forming the data set.

Before creating the dataset, we need to select vulnerabilities that could contribute to the container security literature. In order to do so, we made an investigation based on the NVD and CVE Details to get some general information related to CVEs worldwide. We preferred using the NVD because in addition to the description information, it provides further insights on the severity level of a vulnerability. Moreover, NVD includes references for proof-of-concept and exploitation codes for most of the vulnerabilities with a competent search criterion. Finally, the CVE Details maintains complementary statistics on the CVEs published by the NVD.

4.1.1 CVE Details Review

The chart in Figure 4.1 was taken from CVE Details. It shows the distribution of CVE codes published from 1999 to January 2022 by year. Especially in 2017 and later, except for 2022, we can observe that the number of CVE codes per year has been multiplied by 2 to 3 times compared to previous years. The reason for this is stated in [29]: CVE has made an improvement on the assignment process of CVE numbers in 2017, which decreased the processing time dramatically compared to the previous years. In addition to this, the rise in cloud, mobile, and IoT platforms after 2017 led to an enlarged attack surface causing a rise in vulnerabilities. The number of published CVE codes does not necessarily reflect the number of vulnerabilities

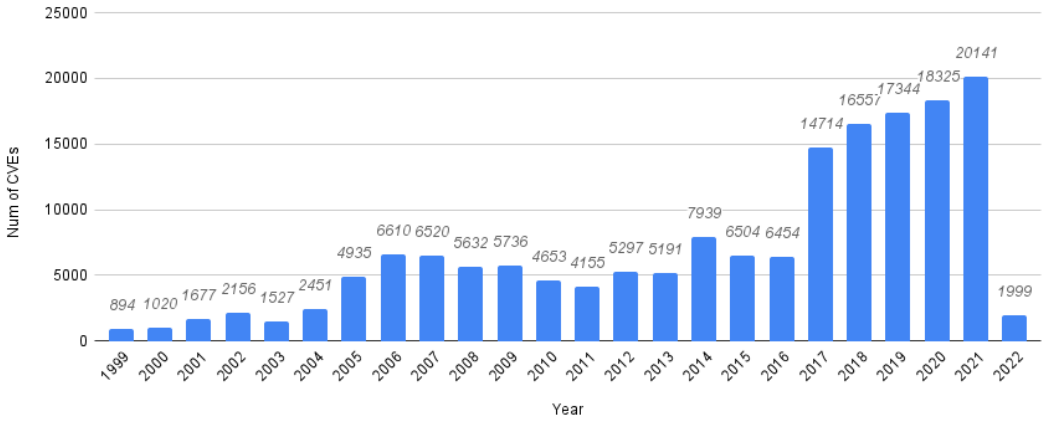

Figure 4.1: Number of CVE codes per year retrieved form CVE Details in January 2022 [6].

discovered in that year either [29], as there are some vulnerabilities published in a year with a CVE code from another year. However, it still reflects the number of vulnerabilities per year that have been made public. It can be deduced from the graph that there is still a significant number of vulnerabilities that have potentially not been studied yet, especially in 2021.

4.1.2 NVD Review

On the NVD database, we followed a straightforward methodology to roughly extract the vulnerabilities related to containers. Most of the other vulnerabilities related to container security possibly may not contain the keyword “container”. However, as our purpose is to get a general impression for a starting point, we searched the vulnerabilities that contain the word “container”. For the search criteria, where the search keyword is set as “container” without selecting any published date range which covers the entries containing the “container” keyword for all time, the search returned 627 records up to 01/01/2022. After that, we added a date range as an additional search criterion. All in all, we selected the published date range between 01/01/2019 and 31/12/2021 which results in 353 entries in total.

We eliminated vulnerabilities with CVE codes that do not reflect its publishing year by hand, which left us with 337 vulnerabilities. We mentioned that NVD immediately

Table 4.1: NVD Container Security-Related Data Statistics

Access Date	Year	Total
01/01/2022	2019	104
01/01/2022	2020	140
01/01/2022	2021	109
-	2011-2018	16
-	Total	353
-	NET	337

reflects changes in the CVE Database. Thus, in terms of the above search criteria, for this thesis, we can roughly assume that by just handling the vulnerabilities with a CVE code, including any of 2019, 2020, and 2021, we could make our selection/election/-filtering among more than half of the records that can be obtained from the data set for the vulnerabilities including the container keyword, which is a very inclusive rate.

4.1.3 Nuclei Review

For implementation, we analyzed the open-source vulnerability scanner Nuclei Engine [30], which we chose to automate attacks, to discover its capabilities in terms of vulnerabilities that it can cover. In Nuclei's dedicated repository, we performed further analysis on the ready-to-use vulnerability template files, which used the special YAML format, to form our attack data set. For this purpose, we retrieved the Nuclei templates from its GitHub repository on 01 January 2022. Table 4.2 summarizes our findings. We sought the CVE code and description, if they exist, by using bash commands and converting them into Excel format.

In order to see how many of the NVD vulnerabilities we filtered for the "container" keyword and the dates between 2019 and 2021 the extracted Nuclei templates can cover, we matched the CVE codes we obtained from NVD with the Nuclei template YAML files. This resulted in only six matched Nuclei YAML files, which are CVE-2020-11854, CVE-2020-11853, CVE-2020-17496, CVE-2020-9757, CVE-

Table 4.2: Number of YAML files having a proper description text

Year	Total Number of YAML Files	Nuber of YAML Files with Proper Description	Total
2019		106	
2020	951	173	485
2021		206	
2000-2018		Not examined	

2021-37573, and CVE-2021-21978. The result was well below our expectations as, in the end, the number of implementable vulnerabilities was not sufficient to create a rich data set. Half of these were eliminated because of their relation to commercial software (CVE-2020-11853, CVE-2020-11854, CVE-2020-9757). Thus, we decided to follow a different methodology in the vulnerability selection process.

4.1.4 OWASP Review

Due to the reasons listed under the Nuclei Engine Review section, as an alternative solution. Instead of directly using CVE codes, we decided to look at one level higher, and based our research on weaknesses to increase the coverage of our Nuclei templates set on the NVD vulnerabilities set. For this, in addition to our investigation on the NVD database, we examined the OWASP's top twenty-five hazardous weaknesses listed for 2021. The top ten weaknesses among these are listed in Table 4.3.

OWASP has published top 25 CWE lists for several years [31]. For the 2021 list, OWASP conducted research on around 32,500 CVE codes by referencing CVE, NVD, and CVSS records retrieved in March 2021 to find the most hazardous weaknesses observed in 2019 and 2020. OWASP noted that the main difference in the 2021 list compared to previous years is that there are more specific weaknesses rather than class-level weaknesses that are more informative. The score column data in the table

Table 4.3: Top CWE codes for 2021 retrieved from CVE Details in January 2022 [1].

Rank	**ID**	**Name**	**Score**	**2020 Rank Change**
[1]	CWE-787	Out-of-bounds Write	65.93	1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	5

is calculated as an average value out of 100, using the information of how many of the vulnerabilities collected from the NVD database are related to the relevant CWE code and the average CVSS score information generated per CWE as a result of this matching. The next column shows the change in the ranking of the related weakness compared to the previous report [1].

We detailed our examination by aggregating the possible technical impacts and adequate scope information of each listed weakness shown in (Only the first 10 CWEs are shown) Table 4.3. We filtered the technical impact information of each CWE with the keywords “remote”, “exec”, “code,” and “command” to find the remote code/-command execution related weaknesses. From this filtration process, we observed that 13 out of 25 of the listed weaknesses are directly related to RCE for 2019 and 2020.

In addition to the description information, to find out which of these YAML files contain one of the 25 most hazardous vulnerabilities OWASP has released, we retrieved the CWE code information from our NUCLEI YAML set. The set size decreased from 485 to 430. The change is shown in Table 4.4.

Table 4.4: Number of YAML files including a CWE-ID information.

Year	**Number of Unique CWEs per YAML Files**				**Total Number of YAML Files Including CWE Information**
	0	**1**	**2**	**Net**	
2019	12	94	0	94	
2020	20	143	10	153	430
2021	23	181	2	183	

Here it can be seen that 55 patch files were not associated with any CWE. However, 12 YAML files were also associated with two weaknesses. Then, we matched the CWE codes we obtained from OWASP top 25 list with the CWE codes in each Nuclei YAML file. Selecting CWE codes instead of CVE codes gives more remarkable results and covers our set of YAML files better than the NVD set does.

Table 4.5: NUCLEI Yaml Files (2019-2021) Related with OWASP Top 10 CWEs

Rank	**ID**	**Name**	**Unique Number of Yaml Files**	**Repeating Yaml Files**
[1]	CWE-787	Out-of-bounds Write	1	0
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	119	1
[4]	CWE-20	Improper Input Validation	6	1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	32	1
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	26	4
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	59	0
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	2	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	12	0

Table 4.5 shows the distribution of NUCLEI YAMLs by CWE IDs included in the OWASP Top 10. Here, the last column shows the number of YAML files that simultaneously contain another weakness in this list, among the YAML files where the weakness is seen. It is noticeable that nearly one-third of the total number of YAML files, including CWE code information, are related to weakness CWE-79, which is listed as the second most hazardous weakness between the years 2019 and 2020, according to OWASP.

We can argue that the raw data set consisting of NUCLEI YAMLs can cover 19 of the weaknesses in the OWASP 2021 top 25 list. Also, it can be seen that our raw set can cover 10 RCE-related weaknesses. We know that 6 of these nine are in the top 10. Besides these, the rest of the CWEs in the rows which do not have zero number YAML files are the weaknesses that can possibly be used to exploit PE-related attacks. Thus, there are 9 weaknesses that can be associated with PE attacks where 3 of these CWEs are listed in the top 10.

When we include only the rows associated with RCE we see that our vulnerability set, which we have reduced from 485 to 430 by choosing YAMLs containing CWE code information, is reduced to 238, i.e., almost half these vulnerabilities are related with the RCE weaknesses listed in OWASP Top 25. The rest of the vulnerabilities, which is again nearly half of the total, are related with PE weaknesses.

After this step, we combined NUCLEI YAML CWE and CVE descriptions with the related NVD description and CVSS core information. Thus, on the resulting version, we can perform a much more effective search to find vulnerable container images. This process also allowed us to be capable of applying some specific keyword filtration to narrow our attack set selection processes. From this, to ease the automation for simulating the attacks, we filtered with the keyword “unauth” to find the unauthenticated and unauthorized entry vulnerabilities which are more suitable for automation.

There were a significantly high number of weaknesses and vulnerabilities. To focus on a narrowed set of these vulnerabilities, we applied several strategies. If a significant number of vulnerabilities belonging to one type of weakness could not be implemented, then this weakness set is automatically eliminated from our search scope.

4.1.5 Vulnerability Elimination

Most of the weaknesses were eliminated after the process mentioned above. After that, by taking the OWASP rank values into account, we chose the first five weaknesses for which we could find vulnerable container images, successfully exploited by the related Nuclei Engine vulnerability YAML files.

As a result of this process, we ended up with a total of seven vulnerable images, where two belong CWE-22, two belong to CWE-78, and the remaining belong to CWE-78, CWE-79, and CWE-434. The resulting weakness and vulnerabilities are listed in Table 4.6

Table 4.6: List of Selected Attacks for Implementation

OWASP Rank	CWE ID	CWE Name	CVE ID
2	CWE-79	Cross-Site Scripting (XSS)	CVE-2019-7543
5	CWE-78	OS Command Injection	CVE-2019-15107, CVE-2019-16662
6	CWE-89	SQL Injection	CVE-2020-9483
8	CWE-22	Path Traversal	CVE-2020-17518, CVE-2021-26086
10	CWE-434	Unrestricted Upload of File with Dangerous Type	CVE-2019-25213

4.1.6 Selected Weakness Descriptions

The first weakness is named "SQL Injection" [32] with code CWE-89. Some unfiltered SQL queries created by user inputs may be interpreted as SQL commands, which can bypass security checks and modify the back-end database, together with the execution of system commands. SQL injection attacks commonly target database-

driven web sites. They are mostly related with databases and the data inside, while obtaining shells in order to run system commands and take control over a target or network can also be performed by this injection type. Attackers initially perform a database enumeration process to determine the number and data types of columns in use. Finding a web server's root directory path may yield to opening a reverse shell allowing attackers to run OS commands and gain wider capabilities on systems reaching to get higher privileges and critical information [33].

The second weakness is named "Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')" [34] with code CWE-22. File operations occur outside the restricted directories by using special characters like ".." and "/" separators. This can cause relative (accessing root directory) or absolute (using direct pathnames) path traversal to access unexpected system files or directories. Path Traversal can also be used to get reverse shells. By attempting several directory traversals, an attacker can determine whether a target is vulnerable to a local file inclusion (LIF), which allows the attacker to read local system files, perform XSS, and can even lead to code execution. After finding an LFI, the attacker uses several injection attacks to find where to inject the script for the reverse shell [35].

The third weakness is named "Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')" [36] with code CWE-78. An attacker can execute commands on the operating system. This can lead to the attacker having direct control on the operating system, or, it could allow the attacker to invoke commands with privileges. Thus, it can be harmful in the manner of the code execution, privilege escalation and persistence attack vector listed in the MITRE Container Matrix. This can result in executing arbitrary command execution, sensitive data disclosure and denial of service. This can be potentially hazardous for a container environment.

The fourth weakness is named "Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting' or 'XSS')" with code CWE-79. If a user-controllable input is not neutralized (controlled), untrusted data enters through a web request into the web application. There are three types of XSS: Reflected XSS (or Non-Persistent), Stored XSS (or Persistent), and DOM-Based XSS. In Reflected XSS, in the HTTP re-

quest, the data is read directly by the server and it is reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. In stored XSS, dangerous data is stored in the application database or other trusted data store. After a period of time, the application reads back the stored data, and an attacker can get elevated privileges or reach sensitive data. In DOM-Based XSS, a trusted script is sent to the client, which makes client perform injections to the page or, the injection is performed by the server. Once the malicious script is injected, the attacker can perform a variety of malicious activities. The private information, such as cookies including the session information, can be transferred, or a malicious request is sent by the attacker that may result in taking over the victim machine [37].

The last weakness is named "Unrestricted Upload of File with Dangerous Type (UUF)" with code CWE-434. Attackers can upload dangerous type files that are automatically processed within the target system. The uploaded file can yield to arbitrary code execution, especially automatically executable .asp and .php extensions. An non-properly validated file is allowed as executable code, which can result in a reverse shell, or overwrite critical files simply by replacing them with a file having the same name and get the control over the server [38] .

The weaknesses different from the CWE-78 are very commonly used categories and they are used for the purposes of "Initial Access" and "Discovery" attack vectors defined in the MITRE container Matrix, thus, detection of any of these attacks potentially warn us for further and much hazardous attacks that can deeply affect a container environment.

4.2 Simulation Environment

In this section, we explain our attacker machine and the testing environment for the monitoring processes.

Table 4.7: NUCLEI Template Statistics

Access Date	Year	Total
01/01/2022	2019	108
01/01/2022	2020	176
01/01/2022	2021	206
-	2000-2018	468
-	Total	958
-	NET	490

4.2.1 Nuclei Engine

NUCLEI Engine is an open-source web application vulnerability scanner. The tool enables researchers to create custom YAML templates for security research and bug discovery [30] [39]. NUCLEI uses YAML templates to launch traffic from the client and test the server's response against signs of vulnerabilities. It supports various protocols, including HTTP, TCP, DNS, FILE. Furthermore, NUCLEI maintains a collection of YAML templates that has been continuously updated since 2000. As of 2022, a total of 958 are present in this collection. We decided to use NUCLEI due to its easy deployment and fast configuration features for our attack simulations.

4.2.2 Testing Environment

There are two main actors in our testing environment. The attacker machine which runs the Nuclei Engine and the vulnerable container environment which is a Kubernetes cluster installed on an Ubuntu 20.04 machine. We deployed different containers on the Kubernetes cluster which spans each for separate nodes. Within the scope of five weaknesses (CWE codes) that we have previously determined on this environment, seven different attacks have been studied using the Nuclei Engine, with the number of vulnerabilities (CVE codes) per weakness are up to two. The constraints that determine the weakness selections and the number of nodes are discussed in detail in the following sections.

4.3 Data Set

In this section, we describe our experimental strategy for the data set creation processes.

4.3.1 Experiments

In the previous sections, we described five specific weaknesses we selected and a total of seven unique vulnerabilities. In order to form the attack simulation environment, we deployed each of the found vulnerable containers on our Kubernetes environment to a total of seven nodes, where each container runs on a single node. We established the attacker machines with the Nuclei Engine outside of the cluster on the same machine.

For this study, we used the Nuclei templates related with each of the seven vulnerable images. In addition to these default attack templates files, we produced more Nuclei template files by simply changing the payloads inside the default templates. We treated each different payload template as a different attacker/behavior. We defined each attacker by creating Python scripts that run the specified Nuclei template from the Nuclei Engine.

For the testing, we created more than one Vagrant [40] (a tool for building and managing virtual machines), virtual machines and installed Nuclei to each. We performed our attacks from these virtual machines. By this method, attackers can have IP addresses different from the host that they are attacking and the cluster running the vulnerable images.

Also, we defined randomly generated timestamps based on UNIX Time and assigned them to the each of the attacks. By this, we avoid attackers running at exactly the same time. The timestamps were generated between a period of time differing from one to eight seconds.

When we performed the attacks with the random timestamp in parallel, our feature extractor named CICFlowMeter generated very few flows, and none for most. Thus we decided to perform the attacks from the same CVE category in random serial

order. This is defined as at each time, we assume that the image related with CVE-X has six attackers that can exploit the vulnerability CVE-X inside the image, the six of the attackers are ordered randomly in another script, and perform their attacks once per each execution of that script. To collect enough network traffic, we repeated the execution of these scripts between four and eight times, that can differ for each CWE category to make the resulting flow numbers nearly equal to each other.

4.3.2 Monitoring

For monitoring, we used the latest version of `tcpdump`, a command-line packet analyzer. The tool captures network traffic in "pcap" format generated using a portable C/C++ library, libpcap [41].

Inside the Kubernetes environment, we monitored each CWE category one-by-one and saved their traffic during the attack period.

4.3.3 Feature Extraction

For feature extraction, we used CICFlowMeter [42], an open-source network traffic flow generator that extracts the features from the given Packet Capture (.pcap) files into a Comma Separated Values (.csv) file.

A flow can be defined as all packets, in forward and backward direction between two connection endpoints during a connection period. If another connection between these endpoints starts after one ends, then it is named as another flow.

CICFlowMeter can generate bidirectional flows, i.e. in the forward (source to destination) and backward (destination to source) directions. This allows the calculation of time-related feature statistics both in the forward and backward directions.

In [43] the authors examined and troubleshooted another CICFlowMeter generated data set. The study found errors in the old version of CICFlowMeter and released a fixed version of it. In this study we used the fixed version [44].

The feature extraction tool can extract 76 network flow features by default. The list

of features is listed in Table 4.8.

Table 4.8: List of extracted features and descriptions

Feature Name	Description
Flow duration	Duration of the flow in Microsecond
total Fwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	Number of flow bytes per second
Flow Packets/s	Number of flow packets per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow

Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)

Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Packet Length Min	Minimum length of a packet
Packet Length Max	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWR
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Fwd Segment Size Avg	Average size observed in the forward direction
Bwd Segment Size Avg	Average size observed in the backward direction

Fwd Bytes/Bulk Avg	Average number of bytes bulk rate in the forward direction
Fwd Packet/Bulk Avg	Average number of packets bulk rate in the forward direction
Fwd Bulk Rate Avg	Average number of bulk rate in the forward direction
Bwd Bytes/Bulk Avg	Average number of bytes bulk rate in the backward direction
Bwd Packet/Bulk Avg	Average number of packets bulk rate in the backward direction
Bwd Bulk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Fwd Init Win bytes	The total number of bytes sent in initial window in the forward direction
Bwd Init Win bytes	The total number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle

Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

4.3.4 Resulting Data Set

To prepare the raw CICFlowMeter features data set for ML testing and validation purposes, we labeled each row with its related vulnerability category in an additional column on the raw dataset. As we had already specified each vulnerability's weakness category, we also added another column to the existing data set to label each row's weakness category. The resulting data set consists of flows, the group of packets having the same source IP, destination IP, source port, destination port, and the protocol type [45]. There is also a third column that contains the OWASP CWE category unique name for readability.

As mentioned, we monitored and saved the traffic of each CWE category one-by-one. First we specified a single test file and a single train file per CWE category by combining two-thirds of the collected flow data into a single .csv file as the training data and the rest as the testing data. Then, we combined each weakness category's test data and created our finalized test data set. We applied the same process for the training data to gather our finalized training data set.

Our project codes and the data set can be found at https://github.com/HaleBera/A-NOVEL-CONTAINER-ATTACKS-DATASET-FOR-INTRUSION-DETECTION.

We performed some initial experiments with different classification algorithms on the formed data set in order to evaluate their ability to distinguish between different classes of attacks on the containers. Specifically, we performed multi-class classification with the Support Vector Machines (SVM) and Random Forest (RF) algorithms, which have been frequently utilized in the intrusion detection domain. The training and test sets consisted of traffic flow instances from implementations of the five different weakness classes in the data set. Table 4.9 summarizes the results of the experiments with the SVM algorithm and Table 4.10 summarizes the results of the experiments with the RF algorithm. We observe that while some of the attack classes have been captured accurately by both algorithms, there is room for improvement for others. This calls for development of algorithms that will achieve improved results for accurately detecting all attack types.

Table 4.9: Classification Results for SVM algorithm.

##### SVM #####	precision	recall	f1-score
'Path Traversal'	0.89	0.93	0.91
'Cross-site Scripting'	1.00	1.00	1.00
'OS Command Injection'	1.00	0.95	0.98
'SQL Injection'	1.00	1.00	1.00
'Unrestricted Upload of File '	1.00	1.00	1.00
'Normal Traffic'	1.00	1.00	1.00

Table 4.10: Classification Results for Random Forest Algorithm.

##### Random Forest #####	precision	recall	f1-score
'Path Traversal'	1.00	0.89	0.94
'Cross-site Scripting'	1.00	1.00	1.00
'OS Command Injection'	1.00	1.00	1.00
'SQL Injection'	1.00	1.00	1.00
'Unrestricted Upload of File '	0.86	1.00	0.92
'Normal Traffic'	1.00	1.00	1.00

CHAPTER 5

CONCLUSIONS

In recent years, machine learning-based intrusion detection systems for containers have been on a tremendous rise. Thus, the need for appropriate and convenient training and test data sets is still an existing problem in machine learning-based studies for container security.

In this thesis, we presented a novel container attack traffic data set for intrusion detection system research in container-based application environments. The data set was constructed in a systematic manner considering the top hazardous vulnerabilities used in recent attacks against containers. For the data set, seven remote code execution and privilege escalation related vulnerabilities that have a CVE code belonging to the years 2019, 2020, and 2021 were implemented in the Kubernetes environment. Using CICFlowMeter Tool, seventy-six bidirectional features were extracted from the attack traffic. We have made this data set publicly available for the benefit of container security researchers utilizing machine learning techniques for intrusion detection.

To ensure inclusive security, it is crucial to discover the behavior of attacks exploiting the vulnerabilities that are malicious for containers. We believe that testing containers with data sets based on up-to-date attacks continuously will help the community to develop defense systems much more effectively.

REFERENCES

[1] MITRE, "2021 cwe top 25 most dangerous software weaknesses." https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html, 2022. Last accessed May 14 2022.

[2] J. Lewis, "Microservices: a definition of this new architectural term." https://martinfowler.com/articles/microservices.html, 2014. Last accessed May 14 2022.

[3] "Docker overview." https://docs.docker.com/get-started/overview/. Last accessed May 14 2022.

[4] L. Rice, "Container security." https://www.oreilly.com/library/view/container-security/9781492056690/ch01.html. Last accessed May 15 2022.

[5] MITRE, "Containers matrix." https://attack.mitre.org/matrices/enterprise/containers/. Last accessed May 15 2022.

[6] "Current cvss score distribution for all vulnerabilities." https://www.cvedetails.com/. Last accessed June 11 2022.

[7] L. S. Vailshery, "Containerization in organizations worldwide 2021." https://www.statista.com/statistics/1223916/it-container-use-organizations/#:~:text=In%202021%2C%2019%20percent%20of,of%20an%20entire%20runtime%20environment., Feb 2022. Last accessed June 11 2022.

[8] Aquasec, "Portworx annual container adoption survey shows container adoption accelerates while security and data management concerns remain top of mind." https://www.aquasec.com/news/portworx-container-adoption-survey/, 2019. Last accessed May 14 2022.

[9] MITRE, "Common vulnerabilities and exposures." https://cve.mitre.org/. Last accessed June 11 2022.

[10] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, 2015.

[11] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[12] "What is kubernetes?." https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/, Jul 2021. Last accessed May 14 2022.

[13] S. M. Kerner, "Top kubernetes management platforms." https://www.datamation.com/cloud/top-kubernetes-management-platforms/, January 2021. Last accessed May 14 2022.

[14] "Vulnerabilities, exploits, and threats at a glance." https://www.rapid7.com/fundamentals/vulnerabilities-exploits-threats/. Last accessed June 10 2022.

[15] Cynet, "Understanding privilege escalation and 5 common attack techniques." https://www.cynet.com/network-attacks/privilege-escalation/#:~:text=Privilege%20escalation%20is%20a%20type,gaining%20access%20to%20a%20system., Apr 2022. Last accessed June 11 2022.

[16] MITRE|ATT&CK, "Containers matrix." https://attack.mitre.org/matrices/enterprise/containers/#. Last accessed June 11 2022.

[17] NIST, "National vulnerability database." https://nvd.nist.gov/. Last accessed June 11 2022.

[18] A. Iradier, "Container security best practices: Comprehensive guide." https://sysdig.com/blog/container-security-best-practices/, October 2021. Last accessed June 10 2022.

[19] G. Tassey, "Standardization in technology-based markets," *Research Policy*, vol. 29, no. 4, pp. 587–602, 2000.

[20] C. Spoden, "Security policies, standards, procedures, and guidelines." https://frsecure.com/blog/differentiating-between-policies-standards-procedures-and-guidelines/, August 2017. Last accessed June 11 2022.

[21] M. P. Souppaya, J. Morello, and K. Scarfone, "Application container security guide," *NIST Special Publication 800-190*, 2017.

[22] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," *IEEE Access*, vol. 7, p. 52976–52996, 2019.

[23] S. Subramanian, P. B. Honnavalli, and S. S. Shylaja, "Container security: An extensive roadmap," in *Proceedings of the 3rd International Conference on Integrated Intelligent Computing Communication Security (ICIIC 2021)*, pp. 427–436, 2021.

[24] K. Wist, M. Helsem, and D. Gligoroski, "Vulnerability analysis of 2500 docker hub images," in *Advances in Security, Networks, and Internet of Things* (K. Daimi, H. R. Arabnia, L. Deligiannidis, M.-S. Hwang, and F. G. Tinetti, eds.), (Cham), pp. 307–327, Springer International Publishing, 2021.

[25] M. Cavalcanti, P. Inacio, and M. Freire, "Performance evaluation of container-level anomaly-based intrusion detection systems for multi-tenant applications using machine learning algorithms," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES 2021, pp. 1–9, 2021.

[26] J. Pope, F. Raimondo, V. Kumar, R. McConville, R. Piechocki, G. Oikonomou, T. Pasquier, B. Luo, D. Howarth, I. Mavromatis, P. Carnelli, A. Sanchez-Mompo, T. Spyridopoulos, and A. Khan, "Container escape detection for edge devices," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, p. 532–536, 2021.

[27] X. Lin, L. Lei, Y. Wang, J. Jing, K. Sun, and Q. Zhou, "A measurement study on linux container security: Attacks and countermeasures," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, p. 418–429, 2018.

[28] C.-W. Tien, T.-Y. Huang, C.-W. Tien, T.-C. Huang, and S.-Y. Kuo, "Kubanomaly: Anomaly detection for the docker orchestration platform with neural network approaches," *Engineering Reports*, vol. 1, no. 5, p. e12080, 2019.

[29] Y. Wu, L. Lei, Y. Wang, K. Sun, and J. Meng, "Evaluation on the security of commercial cloud container services," in *Proceedings of the 23rd International Conference on Information Security (ISC 2020)*, p. 160–177, 2020.

[30] Dimitriverhoeven, "Hacker tools: Nuclei, a yaml based vulnerability scanner." `https://blog.intigriti.com/2021/05/10/hacker-tools-nuclei/`, May 2021. Last accessed June 11 2022.

[31] MITRE, "Cwe top 25 archive." `https://cwe.mitre.org/top25/archive/`. Last accessed June 11 2022.

[32] W. G. Halfond, J. Viegas, A. Orso, *et al.*, "A classification of sql-injection attacks and countermeasures," in *Proceedings of the IEEE international Symposium on Secure Software Engineering*, pp. 13–15, IEEE, 2006.

[33] "Use sql injection to run os commands & get a shell." `https://null-byte.wonderhowto.com/how-to/use-sql-injection-run-os-commands-get-shell-0191405/#:~:text=SQL%20injection%20is%20typically%20only,reverse%20shell%20on%20the%20server`. Last accessed June 11 2022.

[34] A. A. Almutairi, S. Mishra, and M. AlShehri, "Web security: Emerging threats and defense," *Computer Systems Science and Engineering*, vol. 40, no. 3, pp. 1233–1248, 2022.

[35] "From local file inclusion to reverse shell." `https://a3h1nt.medium.com/from-local-file-inclusion-to-reverse-shell-774fe61b7e1e`, April 2020. Last accessed June 11 2022.

[36] I. Security, "Os command injection vulnerability: Cwe-78 weakness: Exploitation and remediation." `https://www.immuniweb.com/vulnerability/os-command-injection.html`. Last accessed June 11 2022.

[37] O. Foundation, "Cwe-79: Improper neutralization of input during web page generation ('cross-site scripting')." https://cwe.mitre.org/data/definitions/79.html. Last accessed June 11 2022.

[38] Portswigger, "File uploads: Web security academy." https://portswigger.net/web-security/file-upload. Last accessed June 11 2022.

[39] OWASP, "Vulnerability scanning tools." https://owasp.org/www-community/Vulnerability_Scanning_Tools. Last accessed June 11 2022.

[40] "Vagrant." https://github.com/hashicorp/vagrant. Last accessed June 11 2022.

[41] "Tcpdump & libpcap." https://www.tcpdump.org/. Last accessed June 11 2022.

[42] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, p. 357–370, 2022.

[43] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW)*, pp. 7–12, IEEE, 2021.

[44] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Cicflowmeter." https://github.com/ahlashkari/CICFlowMeter. Last accessed June 11 2022.

[45] A. Habibi Lashkari., G. Draper Gil., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*, pp. 253–262, 2017.