Aalto University
School of Science
Master's Programme in Security and Cloud Computing

Svitlana Chaplinska

A Purple Team Approach to Attack Automation in the Cloud Native Environment

Master's Thesis
Espoo, July 29, 2022

Supervisors Prof. Pontus Johnson, KTH Royal Institute of Technology
Prof. Mario Di Francesco, Aalto University
Advisors Professor Mathias Ekstedt, KTH Royal Institute of Technology
Tuomo Makkonen, M.Sc. (Tech.), Fraktal Oy

Aalto University
School of Science
Master's Programme in Security and Cloud Computing

ABSTRACT OF
MASTER'S THESIS

Author: Svitlana Chaplinska

Title:
A Purple Team Approach to Attack Automation in the Cloud Native Environment

Date: July 29, 2022 **Pages:** 68

Major: Security and Cloud Computing **Code:** SCI3113

Supervisors: Professor Pontus Johnson
Professor Mario Di Francesco

Advisors: Professor Mathias Ekstedt
Tuomo Makkonen, M.Sc. (Tech.)

The threat landscape is changing with the increased popularity of cloud native systems. Adversaries are adopting new ways to attack systems. Therefore, security specialists have to adopt new approaches to their security practices. This thesis explores a purple team approach to attack automation in a cloud native environment. There are two thesis goals. The first goal is to investigate cyber threats encountered in cloud native environments. The second goal is build an attack automation tool to improve a purple team evaluation of the cloud native environments. As a result, we create a more comprehensive resource of cloud native threats that we refer to as the *Cloud Native Threat Matrix*. Based on this matrix, we build a tool for attack automation. The tool follows the assume breach approach, providing defense-in-depth security testing. As a final step, we propose an improvement to the purple team evaluation of the cloud native environments, that combines created *Cloud Native Threat Matrix* with an automated attack techniques execution and active collaboration as a fundamental concept of purple team evaluations.

A *Cloud Native Threat Matrix* solves the problem of scattered threat data, providing a coherent and easy-to-use platform. In addition, the automation provides a possibility for rerunning security evaluations and making sure that security weaknesses are not re-introduced during major changes. A purple team approach allows improving system defense and response capabilities.

Keywords: Automation, cloud native, security, threat

Language: English

Aalto-universitetet
Högskolan för teknikvetenskaper
Magisterprogrammet i data-, kommunikations- och informationsteknik

SAMMANDRAG AV DIPLOMARBETET

Utfört av: Svitlana Chaplinska

Arbetets namn:
Integritetsförbättrande datarapporteringssystem för deltagande avkänning

Datum:	Den 29 juli 2022	**Sidantal:**	68
Huvudämne:	Säkerhet och Cloud Computing	**Kod:**	SCI3113
Övervakare:	Professor Pontus Johnson Professor Mario Di Francesco		
Handledare:	Professor Mathias Ekstedt Diplomingenjör Tuomo Makkonen		

Angriparna förändrar sina attacker för dessa system. Därför måste även säkerhetsspecialister ta till nya metoder i sitt säkerhetsutövande. Den här avhandlingen utforskar en purple team-strategi för automatiserade attacker i en molnbaserad miljö. Avhandlingen har två mål. Det första är att utforska de cyberhot som molnbaserade miljöer möts av. Det andra är att skapa ett ramverk som kan användas för purple team-övningar med stöd av de automatiserade attackerna, som i sin tur är baserade på den hotmatrisen för molnbaserade miljöer. Resultatet av detta är att vi har skapat en hotmatris - en mer omfattande resurs av molnbaserade hot. Baserat på den här matrisen har vi byggt ett verktyg för automatiserade attacker. Verktyget bygger på antagandet att intrång redan har skett och bidrar med djupförsvarsbaserad säkerhetstestning av en molnbaserad miljö. Slutligen så föreslår vi även en förbättring till hur utvärderingen av ppurple teams"i en molnbaserad miljö kan utföras. Denna utvärdering kombinerar vår molnbaserade matris med en automatiserad attack exekvering samt aktivt samarbete som ett fundementalt koncept i utvärdering av purple teaming".

Löser en molnbaserad matris problemet med oorganiserad data angående hot genom att tillhandahålla en sammanhängande och lättanvänd plattform. Automatiseringen tillåter en möjlighet att utföra säkerhetsutvärderingarna regressivt och säkerställer att kända svagether inte blir återintroducerade under större systemförändringar. Ett purple teamtillvägagångsätt möjliggör förbättring av systemförsvar och incidenthantering.

Nyckelord:	Automation, molnbaserat, säkerhet, hot
Språk:	Engelska

Acknowledgements

To **Ukrainian people** for showing the whole world what it means to be brave!

I want to thank you, SECCLO consortium, for giving me this opportunity and **Eija Kujanpää**, **Kiviharju Anne** for the administrative support.

I want to express my appreciation to my examiner **Pontus Johnson** and my supervisors **Mario Di Francesco** and **Mathias Ekstedt** for sharing feedback, comments and ideas to improve the quality of this thesis work.

I want to express my gratitude to Fraktal Oy and its employees. **Tuomo Makkonen** for supervising this thesis work and always bringing bright ideas to the table, **Jarno Virtanen** for knowledge sharing and proofreading this thesis work and **Mikhael Weckstén** for an excellent translation of the abstract.

I also want to thank you **Joseph Attieh** and **Juha Kai Mattila** for providing insightful feedback and ideas.

A special thank you to my boyfriend and to my friends all over the globe!

My biggest thank you to my mother and grandmother. Without them, I would not be writing this thesis.

Слава Україні!

Дякую! Kiitos! Tack så mycket! Thank you!

Espoo, July 29, 2022

Svitlana Chaplinska

With the support of the Erasmus+ Programme of the European Union

Contents

List of Acronyms and Abbreviations

Amazon EC2 Amazon Elastic Compute Cloud.

API Application Programming Interface.

AWS Amazon Web Services.

C2 Command Control.

CI/CD Continuous Integration and Continuous Delivery/Deployment.

CLI Command-Line Interface.

CPU Central Processing Unit.

DoS Denial-of-Service.

ECR Elastic Container Registry.

GCP Google Cloud Platform.

IaaS Infrastructure-As-a-Service.

ICS Industrial Control System.

JSON JavaScript Object Notation.

MITRE ATT&CK MITRE Adversarial Tactics, Techniques and Common Knowledge.

nmap Network Mapper.

OS Operating System.

RBAC Role-Based Access Control.

SaaS Software-As-a-Service.

SIEM Security Information and Event Management.

SOC Security Operation Center.

TTPs Tactics, Techniques, Procedures.

UI User Interface.

VM Virtual Machine.

Chapter 1

Introduction

The rate of technological change is increasing. Therefore, the methods used to produce software have changed. The need to have more efficient and more advanced systems requires a major shift in the approaches used to create these systems. This shift can be seen in development processes that switched from waterfall to Agile and DevOps. In application architecture it changed from monolithic to microservices. Deployment and packaging switched from physical servers to virtual ones and to container-based deployments. Application infrastructures have migrated from on-premise data centers to the cloud. Combining all these methods in one approach introduced us to the cloud native mindset. Cloud native are technologies that help businesses to create scalable applications in dynamic environments [1]. Some examples of these technologies are containers, microservices and declarative Application Programming Interface (API). Together with automation, cloud native helps engineers make significant changes in a modular, scalable and distributed way.

However, technological progress also brings new attack vectors and vulnerabilities. The number of cyber-attacks is rising every year. According to the survey conducted in [2], 93% of the more than 300 DevOps, security and engineering professionals had experienced a security incident in 2021. Now that the industry has been adopting the cloud native approach, adversaries have to adapt their attacks to new environments and exploit cloud native applications via cloud, containers, Continuous Integration and Continuous Delivery/Deployment (CI/CD) and other.

1.1 Problem Statement

As cloud native solutions are gaining more popularity, the number of adversaries targeting these is also rising. Therefore, there is a need for tools that perform security testing in such environments. The top common incidents are disabled encryption of SQL databases, firewall rules allowing all traffic to the Kubernetes cluster and instances directly exposed to the internet [3]. In these rapidly changing conditions, companies have to adopt new approaches to their security practices.

When communicating attacks, the industry standard is to use the MITRE Adversarial Tactics, Techniques and Common Knowledge (MITRE ATT&CK) as a structured and coherent adversary behavior data source. MITRE ATT&CK is a knowledge base of adversary tactics and techniques based on real-world observations in various environments [4]. The knowledge base is used for building tools and methodologies in both private and governmental sectors for the red, blue and purple teams. Security teams are conventionally divided to offensive (red), defensive/response (blue) and mixed (purple). Security teams can benefit from introducing the purple team by improving the efficacy of threat detection and mitigation. Thus, a tool for executing attack techniques that combines purple team approaches and MITRE-like matrix can be built to improve security operations.

This thesis work is focused on building a more comprehensive table that consists of attack techniques relevant to the cloud native environments, referred to as a *Cloud Native Threat Matrix*. These attack techniques target distinct phases of the attack lifecycle. Another vector of focus is to provide an automatic execution framework of attack techniques as part of the purple team cloud native security evaluation tool. The focus area of this attack automation is Amazon Web Services (AWS) and Kubernetes.

This research assignment has been conducted as part of the industrial collaboration with Fraktal Oy [5].

1.2 Goals

This thesis has two goals. The first one is to create a more comprehensive *Cloud Native Threat Matrix*. The matrix has to include various attack tactics and cover techniques related to cloud, containers and CI/CD pipeline. The second goal of the thesis is to develop a framework for automating the execution of the attacks using the aforementioned matrix. This framework should be customizable to allow the easy addition of new techniques.

1.3 Research Questions

- What type of new cyber threats do cloud native environments face?
- How to effectively utilize the automated purple team approach for attack automation in cloud native environments?

1.4 Sustainability and Ethics

This work has a limited direct influence on sustainability issues. However, utilizing created framework helps increase the security and availability of systems. Furthermore, defining security gaps helps mitigate risks before adversaries exploit the systems. Thus, the framework increases the operational sustainability of systems.

From the ethical perspective, we tested the framework in a restricted laboratory environment, thus, not violating access limitations of the real systems. However, for future usage, the framework can only be used after the prior agreement with the system owner. The work does not infringe any copyrights or disclose confidential information of the company that assigned the research.

We gathered all attack information from public resources. Thus, the adversary can access the same information directly from the openly available sources. On the other hand, the automation framework implementation is not public. Therefore, it cannot be used with a wrong intention by the adversary.

1.5 Structure of the Report

The rest of this thesis is organized as follows. Chapter 2 presents the background. Then chapter 3 contains an overview of the created *Cloud Native Threat Matrix*. Chapter 4 provides an overview of the attack automation. Chapter 5 discusses the research process, created artifacts, improvement to the purple team evaluation approach, limitations, possible improvements and future work. Lastly, chapter 6 gives a conclusion of the thesis work.

Chapter 2

Background

In this chapter, we present an overview of technologies relevant to the thesis work. First, we define cloud native technologies and its building blocks. Next, we discuss approaches to attack analysis. Then, we define different security operation teams. Finally, we discuss automation as part of security testing.

2.1 Cloud Native

Cloud native is a new mindset adopted to build and deploy applications that are easier to maintain in a cloud infrastructure, allowing these applications to scale effectively and remain highly available. Building cloud native applications usually require the use of a plethora of technologies. Thus, a cloud native application can be defined as a combination of different technologies, such as virtualization, containerization, microservices, CI/CD. Cloud native gained popularity amongst development teams for its ability to reduce operational overhead, automate deployment, reduce the time to deploy, scale and update software.

Virtual Machines and Containers

One of the main technologies in cloud native environments is virtualization. Virtualization builds a layer of abstraction over the hardware of the computer. It allows dividing processors, memory, storage and other hardware elements into several virtual instances. Virtual computer is called Virtual Machine (VM) and is the main component of the virtualization. A VM is a computational resource that utilizes software instead of hardware to deploy and run application [6]. Multiple VMs can run on a single physical server,

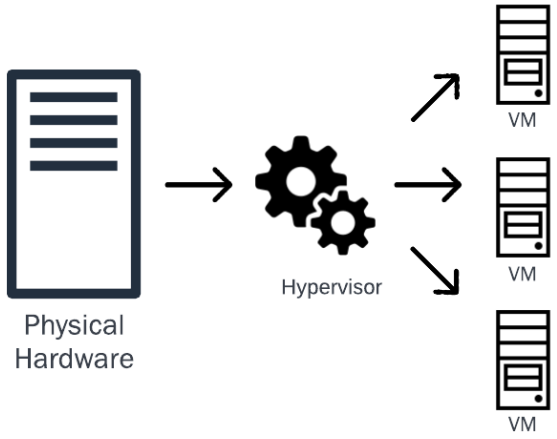


Figure 2.1: Division of physical infrastructure into the virtual machines.

also called a host machine. A "host" machine provides all resources to a VM, such as Central Processing Unit (CPU), memory and storage [7]. Virtual machines running on the same server can be isolated from each other, where each VM can have different functions and run different Operating System (OS).

A critical component of systems that are using VMs is a hypervisor. A hypervisor, also called VM Monitor, is a layer between a physical server and a virtual machine responsible for creating, modifying and destroying VMs [8]. For example, Figure 2.1 shows how virtualization divides a physical infrastructure into virtual machines and a hypervisor as a layer in between.

Another unit of virtualization is a container. Containers are more lightweight, less resource and time-consuming compared to virtual machines. A container encapsulates code and all of its dependencies. It allows applications to be transferred between other computing environments quickly and conscientiously [9]. One of the benefits of containerization is that a container has everything required for running the application. Thus, running the application in different server environments does not depend on the configurations of the physical machine. The number of containers on one machine is only limited by the computing resources of this physical machine [10].

Both containers and virtual machines run virtualized isolated environments, but they have important differences for building cloud native applications. Figure 2.2 shows the main differences in the architecture between VMs and containers. In the case of virtual machines, the hypervisor vir-

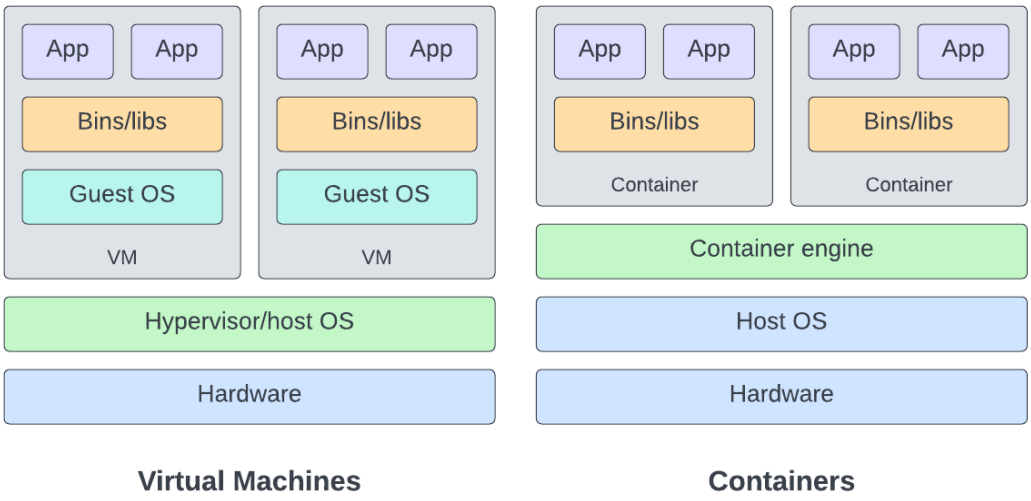


Figure 2.2: Comparison of the virtual machines and containers architecture.

tualizes physical hardware. Each virtual machine has a guest OS. A guest OS is a virtual replica of the hardware with libraries and dependencies that the operating system requires to function [11]. Different OSs can be run on a single physical machine. Unlike VM, containers virtualize the operating system instead of virtualizing the hardware. They use the host machine operating system, kernel and have some lightweight operating system API in user mode. Each container has applications, libraries and dependencies. Containers provide lightweight isolation of virtualized environments but do not provide as strong security boundaries as VM do [11].

Containerization Platform – Docker

Container platforms are software solutions for managing containerized applications. Docker is one of the open-source containerization platforms used for running, developing and shipping containers. Docker defines the specification of container images and run time. A container image is a static file of the executable code that can be executed in isolated environments, such as a container [12].

The docker architecture follows the client-server model. Figure 2.3 shows a Docker architecture and its main components:

- **Docker Client** is used for providing communication of users with Docker. The daemon and the Docker client can both run on the same

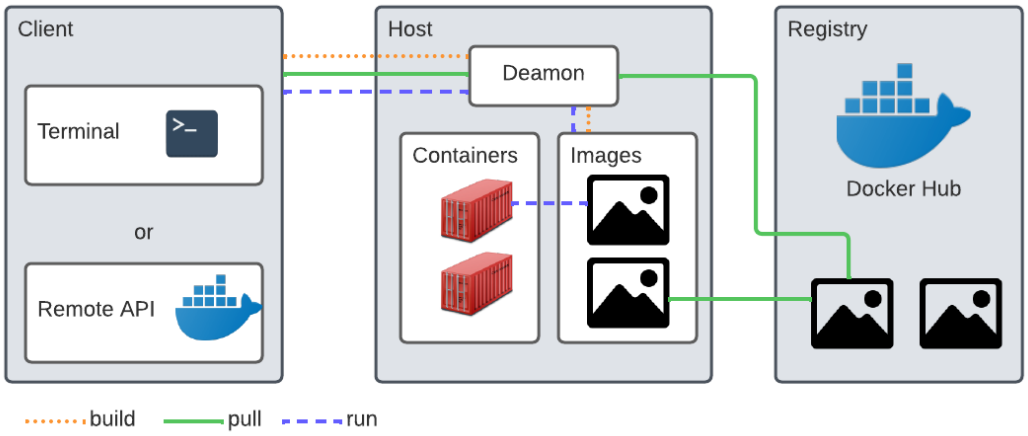


Figure 2.3: Docker architecture.

computer or connect to one running on a remote [13]. Communication with Docker is done via commands to the Docker daemon. Typical commands are *docker run, docker build, docker pull.*

- **Docker Host** contains an environment for running containers. It contains Images, Containers, Networks, Storage and Docker daemon. The Docker daemon is responsible for all the activities related to the container, and it accepts commands from the REST API or the Host Command-Line Interface (CLI) [13].
- **Docker Registry / Hub** is a storage registry for images. The central repository is the Docker Hub, which is used to publicly share Docker images [12].

Container Orchestration System – Kubernetes

Automation is at the core of cloud native systems. Thus, the orchestration of containerized systems is needed. Orchestration is the process of scheduling and controlling the work of containers in clusters. Orchestration involves ensuring that all containers running different workloads are scheduled to operate physical or virtual machines. One of the most popular container orchestration platforms is Kubernetes. Kubernetes is an open-source framework for orchestrating containerized workloads [14].

Kubernetes is responsible for maintaining the system in the state defined by the Kubernetes manifests. Kubernetes uses manifest specification files

in the YAML or JavaScript Object Notation (JSON) format that describes the required state of running Pods, ReplicaSets and their connection to other objects. Thus, Kubernetes monitors all running containers and replaces those that are dead, unresponsive, or otherwise unhealthy. If the current state of the Pod or ReplicaSet is different from the defined in the Kubernetes manifests, Kubernetes will try to change a state to match the desired.

The core element of Kubernetes is the control plane. It is responsible for making global choices about the cluster and managing cluster events, such as scheduling Pods creation, Pods removal and Pods restart. The main parts of the control plane are the following [15]:

- **kube-apiserver** is the frontend module that exposes the Kubernetes API.
- **etcd** is a Kubernetes cluster data key-value store.
- **Kube scheduler** is a process that monitors for recently created Pods with no allocated Node and assigns them one.
- **Kube controller manager** is a runner of controller processes. Controller processes include a node controller, a replication controller, a service account and the token controller. The node controller is responsible for responding to changes in the Pod state, while the replication controller maintains a correct number of Pods. The endpoint controller joins endpoint objects, and the token controller generates default accounts and access tokens.
- **Cloud controller manager (optional)** embeds cloud-specific control logic by linking clusters to the API of a cloud provider and then separating components that communicate with the cluster from components that interact with that cloud platform. The cloud controller manager does not exist in the control plane if the Kubernetes cluster is run locally.

Pod

Pods are the smallest deployable computing units in Kubernetes that a user can build and control. A Pod is a set of one or more containers. Pods can have shared specifications for running containers, network and storage. To run a workload, Kubernetes places containers to the Pods in order to run them on Nodes [16].

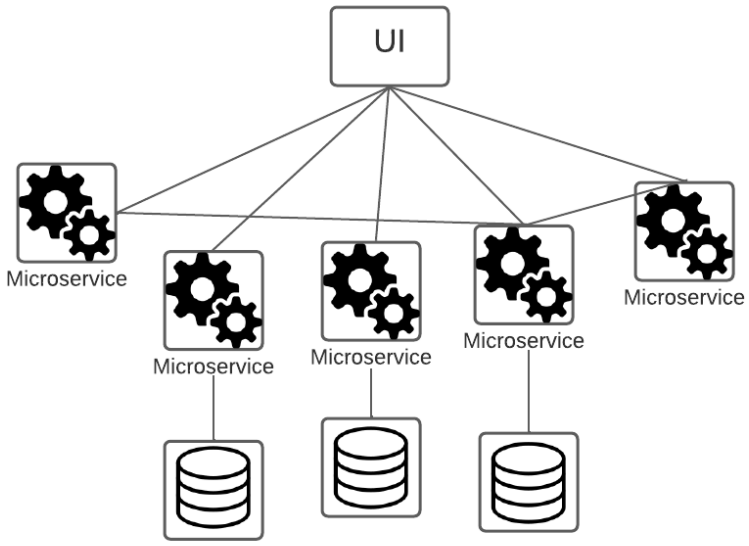


Figure 2.4: Microservices architecture.

Node

A Node is a worker machine that contains everything needed to run a Pod. It can be a physical or virtual machine. Nodes are managed by the control plane. There are several components on the Node: kube-proxy, container runtime, kubelet [17]. The set of Nodes that run a containerized application is referred to as a Kubernetes cluster.

Deployment

A Deployment offers declarative updates for ReplicaSets and Pods. ReplicaSet maintains a stable number of replica Pods in a given time. A user provides the desired state in the Deployment. The desired state is managed by a deployment controller. The deployment controller gradually converts the actual state to the desired state. Deployments can be utilized to create new ReplicaSets or delete current Deployments and replace them with new Deployments [18].

Microservices

The microservice architecture is a paradigm for building applications. Microservices are tiny services that run their processes and communicate via

lightweight procedures. They decompose massive systems into a group of discrete services, allowing the development of individual microservices independently. This approach provides modularity, scalability and distributed development [19]. Figure 2.4 shows the modules of the application, where each module is a separate microservice.

DevOps and CI/CD

A CI/CD is an approach to frequent software delivery to customers through automation development, testing and deployment of applications. The main concepts of CI/CD are continuous integration, delivery and deployment. CI/CD allows applying quick fixes without breaking core system elements [20]. The main objectives are to shorten release cycles, increase productivity and enhance early fault detection [21].

The CI/CD is a fundamental part of DevOps. DevOps is a method of developing software that optimizes the systems development life cycle by combining software development (Dev) with IT operations (Ops) [22]. The main benefit of DevOps is to efficiently coordinate the production and upgrading of software products and services. It is based on the following processes: plan, code, build, test, release, deploy and operate.

2.2 Contemporary Attack Environment – MITRE ATT&CK

Companies use several approaches to analyze possible threats to the systems, such as Cyber Kill Chain, Diamond Model, MITRE ATT&CK [23]. These approaches help structure analysis of attacks and adversarial behavior.

However, the framework that is mostly adopted in the security industry is the MITRE ATT&CK. For example, the National Cyber Security Centre uses the MITRE ATT&CK structure for their threats analysis [24]. One of MITRE ATT&CK's main benefits is that it is based on observations of real-world attacks. Thus, it helps build possible attack scenarios for the specific system.

MITRE ATT&CK

MITRE is a non-profit organization that focuses on solving challenges related to safety in the public interest. MITRE works on various research projects including MITRE ATT&CK. Published in 2015, MITRE ATT&CK provided companies with a coherent framework of known tactics used by adversaries

based on information about previously conducted attacks. The ATT&CK stands for Adversarial Tactics, Techniques and Common Knowledge. The framework contains a profound list of possible Tactics, Techniques, Procedures (TTPs) related to different environments. It is publicly accessible and can be used for building thread models, methodologies and security defense systems in different organizations. In addition, the framework provides a classification for both the defensive and offensive security teams. Nowadays, it is commonly used in the industry for security-related assignments [25].

When discussing threats and attacks in different environments, a typical problem is a lack of common vocabulary between technical and non-technical people. In fact, the MITRE ATT&CK framework tackles this problem by giving people a common vocabulary to use while discussing threats and possible TTPs related to different environments.

The main components of the framework are the following:

- Tactics – "why"/goals of the adversary during the attack.
- Techniques – how the adversary achieves a goal.
- Sub-techniques – more specific means of achieving a goal on the lower level than techniques.
- Other metadata that includes: documented adversarial techniques usage, mitigations and detection.

The framework consists of three matrices such as enterprise, mobile, and Industrial Control System (ICS). The enterprise matrix contains information related to platforms, mainly Windows, macOS, Linux, cloud, network and containers. The mobile matrix includes Android and iOS platforms. It covers device access and network-based techniques without adversary access to the device. Finally, the ICS matrix covers information within an ICS network.

There are multiple benefits of using the MITRE ATT&CK framework in organizations. Employing this will allow support for adversary emulation, red/purple team activities, defensive gap analysis, Security Operation Center (SOC) maturity analysis.

2.3 Security Operation Teams

Security operations are traditionally organized into three teams: offensive (red), defensive/response (blue) and mixed (purple).

Red Team

A red team deals with the offensive side of security operations. Red teams emulate potential attacks of the adversary or exploitation capabilities in the target environments to identify vulnerabilities in systems and applications. The main goals of the red team are to showcase the consequences of successfully conducted attacks and to help the defense team define what activities they can detect in a real-world setting [26].

Blue Team

A blue team covers the defensive side of security operations. Blue teams typically work in the SOC. They monitor system activities to detect any subnormal ones that can be a sign of attack. Another responsibility of a blue team is responding to attacks, if any, and building prevention mechanisms based on faced attack or open-source attacks data. Thus, blue teams are responsible for defending systems from attacks by working on threat prevention, detection and response [27].

Purple Team

Companies utilize red and blue teams for building system defenses, detecting threads effectively and finding gaps in security systems. Both red and blue teams are usually organized as separate entities with separate goals. Usually, there is not much knowledge sharing between teams or frequent feedback exchanges. Thus, a purple team was introduced to improve the effectiveness of utilizing red and blue team activities.

A purple team is a collaborative approach to system testing with active knowledge and feedback sharing. It enhances the effectiveness of threat hunting, network monitoring and vulnerability detection [28].

The purple team security evaluation consists of the following steps:

1. The red team executes different attack techniques in the target environment.

2. The red team actively collaborates with the blue team to test the defense and response.

3. Based on the results, defense and response capabilities are improved.

Automated Purple Team

A purple team is typically a person-to-person collaboration. Person-to-person collaboration is very laborious and resource-intensive. Thus, it cannot provide a constant view on the protection of the organization. However, it can be automated. Security teams can provide automated purple team capability as a purple team security evaluation tool by simulating possible techniques of attacks utilized by adversaries (the red team) and giving defensive and mitigation procedures (the blue team). Therefore, organizations can utilize an automated purple team evaluation tool to have an ongoing security evaluation process [29].

2.4 Test/Attack Automation

Increasing threat vectors result in an increasing need for a security testing of systems. Therefore, the system needs to be prepared for real attacks by testing its vulnerabilities and preventing them by setting up detection and defense. Security testing techniques consist of penetration testing and red/purple team testing. Penetration testing focuses on exploiting the system and network vulnerabilities. On the other hand, red/purple team evaluation focuses on checking the resilience of the system against attacks that are likely to be performed by real adversaries [30].

We distinguish two approaches to penetration testing or red/purple team testing. The first approach is manual testing is more detailed and well customized to the specific system. Manual testing is a time-consuming process that requires the advanced knowledge of security experts. Typically a group of experts is required to perform a comprehensive manual security testing of the system, especially in a limited time frame. Another approach is automated testing. Automated testing is a highly adopted approach to decreasing time spent on system testing, as it allows to repeat the tests easily. Automation tools are usually built based on threat research and combined into easy-to-package and simple-to-use solutions [31].

Due to the multiple benefits offered by automated approaches, we choose to focus on this set of methods in this thesis work. The current state-of-art development of automation tools for security testing of a cloud native environment consists of several solutions. We can summarise the landscape of the existing attack automation solutions as follows:

- **Atomic Red Team** is a set of tests mapped to the MITRE ATT&CK framework for testing environments in a fast, portable and consistent

manner. The tool is not built specifically for the cloud native environments and has a limited number of TTPs for cloud native environments. Attack techniques are formulated in a text markup format [32].

- **Stratus Red Team** is a tool for emulating offensive attack techniques that are mapped to the MITRE ATT&CK framework. The tool is similar to the Atomic Red Team but created explicitly for cloud environments. The Stratus Red Team cannot simulate attacks against user environments and only simulates attacks in the pre-created for the attack needs test environments [33].

- **Leonidas** is a framework for executing attacks in the cloud. It is a web application that can be deployed in the target AWS environment using Terraform. Leonidas describes attack techniques using the YAML format [34].

- **Pacu** is an AWS exploitation framework for penetration testing against cloud environments. It is implemented in Python. It has numerous attack techniques to exploit and enumerate AWS environments [35].

- **MITRE CALDERA** is a framework for automated adversary emulation, the assistance of manual red teams and automated incident response. It is developed by MITRE and is based on the MITRE ATT&CK framework. The framework has a client-server system, where the server is responsible for setting up client and operations initiation. It emulates the typical Command Control (C2) pattern found in real-world attacks and employs defense evasion techniques, such as jittering and obfuscation of C2 communication [36].

Chapter 3

Cloud Native Threat Matrix

This chapter contains an overview of the created cloud native threat matrix. Firstly, we discuss the methodology of creating it. Then we overview the threats in cloud native environments. Next, we describe the *Cloud Native Threat Matrix* structure and its implementation. Finally, we discuss background work.

3.1 Methodology

The matrix is a result of a systematic study of sources that contain possible threats to the cloud native environments and security analysis of such environments. The systematic mapping study is adapted from [37] and modified to the needs of creating the *Cloud Native Threat Matrix*. We decided to base the structure on the MITRE ATT&CK matrix for the *Cloud Native Threat Matrix*. As MITRE ATT&CK is a widely used framework in the security industry. For example, *Microsoft Defender for Cloud* uses MITRE ATT&CK framework for mapping its security recommendations [38].

Figure 3.1 shows the steps of the systematic mapping study. We present the methodology used in this study:

1. We defined a research question that established a review scope, as described at chapter 1.

2. We conducted a search and gathered resources.

3. We completed the screening of resources and defined the relevant ones. The relevant sources of information are described at the section 3.5.

4. We performed the data extraction and mapping process. There are several steps in the data extraction and mapping process. The first

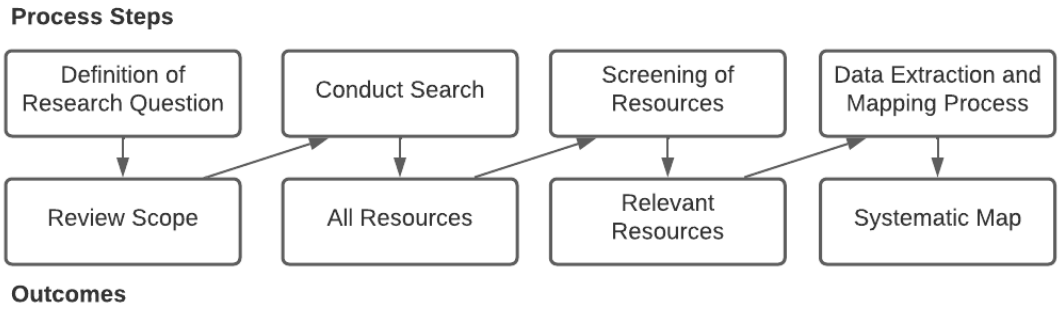


Figure 3.1: A systematic mapping study process steps.

step is to gather techniques after the screening of resources. This step also requires a decision on whether the technique is in the scope of cloud native environments. This thesis looks into cloud native environments as a combination of containerization, cloud and CI/CD.

5. We collect the techniques from the sources to a systematic map. Since there are multiple redundant techniques, we map similar techniques into one technique while mentioning the source and specific details in the description.

The following paragraph contains an example of the same and similar techniques mapped into one. We define a technique called "Valid Accounts", present in the MITRE ATT&CK Cloud matrix and MITRE ATT&CK Containers matrix. It has multiple sub-techniques (such as default accounts, local accounts and cloud accounts). We make sure to include information about these sub-techniques in the description of the technique as shown in Figure 3.3. After thorough analysis of other techniques, we discover that "Using cloud credentials" from the Microsoft Threat matrix for Kubernetes is very similar in the context to the "Valid Accounts". Therefore, we map this technique to "Valid Accounts" and proceed by adding information and source in the "Valid Accounts" technique description. Analyzing even further, we notice that Common Threat Matrix for CI/CD Pipeline contains similar techniques, such as "Valid Account of Git Repository", "Valid Account of CI/CD Service", "Valid Admin account of Server hosting Git Repository". We map it and perform the same steps as with "Using cloud credentials" technique. The mapping is shown in Figure 3.2. Performing these steps makes a matrix more coherent and deletes collisions.

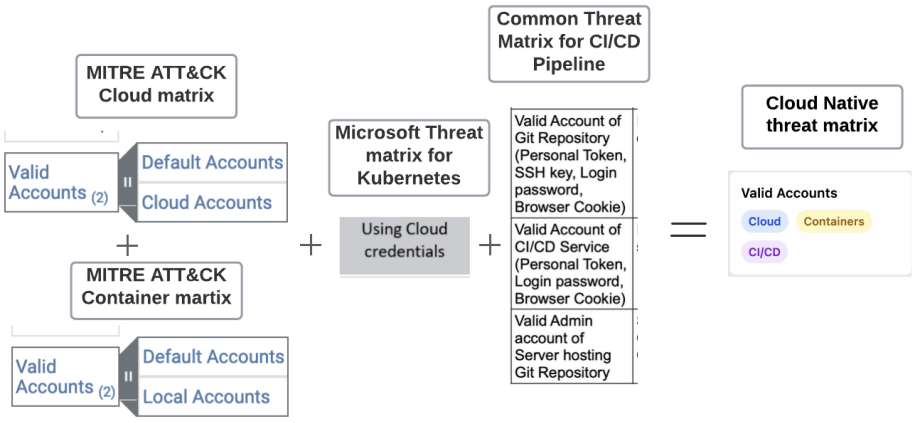


Figure 3.2: Mapping "Valid Accounts" technique in the process of cloud native matrix creation.

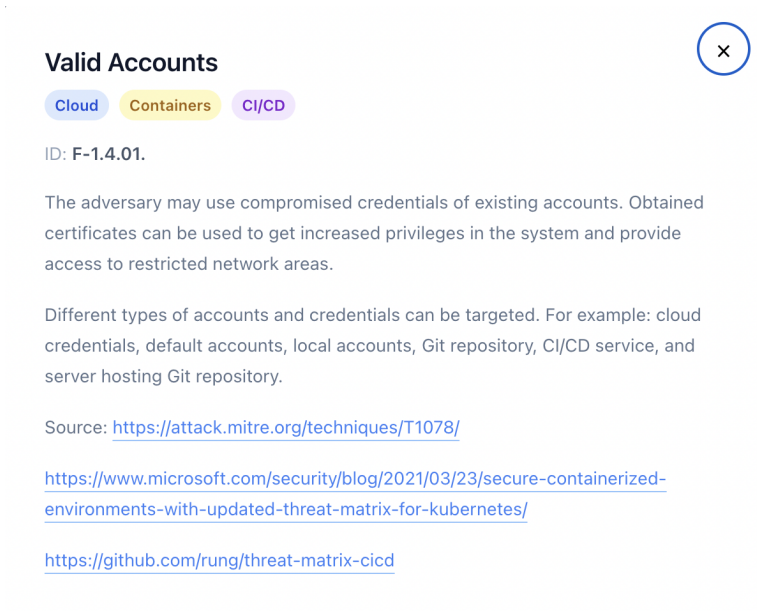


Figure 3.3: View of the "Valid Accounts" technique, after mapping with similar techniques.

The next step is to rename some of the techniques in the matrix for consistency. We adopt the same technique used by the MITRE ATT&CK framework that creates a simple technique naming and includes specific details in the description. For example, a source document "Common Threat Matrix for CI/CD Pipeline" and a technique "Modify the configuration of Production environment" can be rephrased as "Production Environment Configuration".

At this stage, we already have a matrix with techniques divided between different tactics. Each technique has a description. During this step, we update the description and create an ID for each technique. The technique ID facilitates the matrix navigation.

The final step is to develop a view board for the matrix. Firstly the view board for the matrix was an Excel table, where the matrix was created. However, having an Excel table is not the most effective solution in this case. Moreover, there is no good functionality for listing a description of techniques in a readable form with easy navigation. Therefore, we created a view board for the matrix. It is described in more detail in the section 3.4.

3.2 Cloud Native Threats

Adversaries are adapting their attack techniques to target both the software supply chain and the infrastructure of the cloud native applications. In fact, according to a study conducted [39], software supply chain attacks increased by more than 300% in 2021 compared to 2020. For example, the SolarWinds attack [40] and the Codecov breach affected the software supply chains of many companies, remaining undetected for months. This directed the attention of organizations to build their defense capabilities against software supply attacks. Adversaries are continuously searching for new techniques to gain persistence, leverage privilege-escalation and perform data exfiltration to increase possible damage to the infrastructure and businesses.

Considering that we are looking into cloud native environments as a combination of containerization, cloud, CI/CD, we divided threats into three categories: container threats, cloud threats and CI/CD threats.

Container Threats

A containerized environments have several possible routes to be attacked. The most common ones are badly configured containers, build machine attacks, vulnerable application code and exposed secrets [41]. As companies adopt container technologies like Docker and Kubernetes and microservice design patterns, security teams have to advance container security solutions

that allow such infrastructure migrations [42]. In this thesis work, we look into container threats from both container environment and container orchestration perspectives.

Cloud Threats

Cloud security is the practice of protecting data, programs and cloud-based infrastructures from cyber attacks. In 2021 the number one security weakness of cloud environments is misconfigurations (23%), followed by exposed data by users (15%) and account compromise (15%) [3]. When communicating cloud-related threats in this thesis work, we define them as threats related to Infrastructure-As-a-Service (IaaS) and Software-As-a-Service (SaaS) platforms.

CI/CD Threats

CI/CD pipeline is a significant attack vector for adversaries, as it has access to the code, databases, credentials, secrets, development and production environments. Additionally, as CI/CD pipeline attacks are less researched, companies are less prepared to detect or prevent this type of attacks. In this thesis work, we define CI/CD threats as threats related to the development process, including the CI/CD pipeline.

3.3 Cloud Native Threat Matrix

The *Cloud Native Threat Matrix* contains 125 techniques divided among 11 tactics. Each technique has an ID, description, source and category tag. Figure 3.4 includes the structure of each technique in the matrix, where 1 is the name of the technique, 2 is a category tag, 3 is the ID, 4 is the description of the technique, 5 is the source of the technique.

Our matrix consists of 11 tactics, where each tactic entails the goal of the adversary action. Table 3.1 contains all tactics with a description.

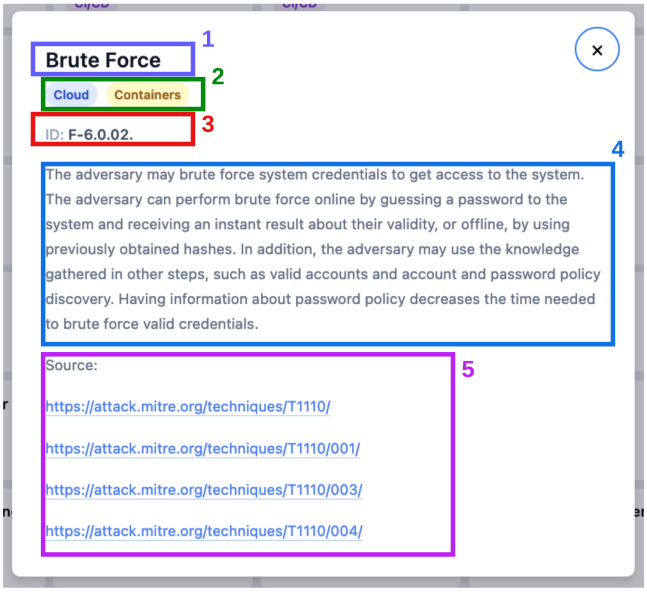


Figure 3.4: View of the single technique in the matrix.

Table 3.1: Adversarial tactics in the *Cloud Native Threat Matrix*.

Adversarial tactics		
ID	Name	Description
1	Initial Access	Techniques that the adversary uses to get initial access to the system. It consists of various entry approaches to get a foothold in a network.
2	Execution	Techniques that the adversary uses to run a malicious code in the system. It entails using ways to execute an adversary-controlled code on the remote or local systems.
3	Persistence	Techniques that the adversary uses to persist the malicious code in the system so that it is not removed by the defender. Adversaries utilize access, activity, or configuration modifications to stay on systems. These strategies are used to keep systems accessible even after restarts, changes in credentials and other disruptions that could prohibit them from re-accessing the system.

Continuation of Table 3.1		
ID	Name	Description
4	Privilege Escalation	Techniques that the adversary uses to get higher privileges in the system. Typically the adversary will access a system and explore a network with unprivileged access. Then they will perform techniques to escalate privileges to reach other goals.
5	Defense Evasion	Techniques that the adversary uses to avoid being detected. Used techniques are usually removing/-turning off security software or encrypting/obscuring data and scripts. Adversaries also utilize and misuse trusted processes to hide the malware.
6	Credential Access	Techniques that the adversary uses to steal accounts credentials. Possible techniques include keylogging or credential dumping to steal account login and password.
7	Discovery	Techniques that the adversary uses to learn more about the environment. The knowledge about the system and network helps the adversary decide how to perform the next attack steps.
8	Lateral Movement	Techniques that the adversary uses to move through the system. The majority of techniques are focused on gaining access to and controlling remote systems on a network. It frequently involves exploring the network for the target and getting access to it.
9	Collection	Techniques that the adversary uses to collect information that matches their objectives from the system. Collection of the data is a preparation step for data stealing. Techniques typically include capturing screenshots and keyboard input.
10	Exfiltration	Techniques that the adversary uses to steal data from the environment. After collecting the data, adversaries often package it using compression and encryption to avoid detection. Exfiltration is typically performed by transferring the data over an adversary command and control channel.

Continuation of Table 3.1		
ID	Name	Description
11	Impact	Techniques that the adversary uses to alter, disrupt, or destroy data and systems. The adversary is attempting to alter, disrupt, or destroy the data and systems. Adversaries utilize impact approaches to modify operational and business processes in order to disrupt availability or compromise the integrity of systems.

Next, we present a more detailed description of the technique ID. As its name says, the technique ID is a unique identifier that contains information about the technique. It has a standard template F-x1.x2.x3. and is formed as follows:

- The first digit (x1) is a tactic. The numeration follows Table 3.1 content.
- The second digit (x2) is category, which can take the following values: 0 – cloud, containers and CI/CD; 1 – cloud; 2 – containers; 3 – CI/CD; 4 – containers and CI/CD.
- The third digit (x3) is a counter of a technique in each tactic.

There are 125 techniques in the *Cloud Native Threat Matrix* from both cloud, container and CI/CD categories.

In this chapter, we present a selection of a few sample techniques from each tactic to provide a general understanding of what each tactic looks like and how they align with a tactic goal.

Initial Access

Valid Accounts

F-1.4.01.

The adversary may use compromised credentials of existing accounts. Obtained certificates can be used to get increased privileges in the system and provide access to restricted network areas. Different types of accounts and credentials can be targeted, such as cloud credentials, default accounts, local accounts, Git repository, CI/CD service and server hosting Git repository [43–45].

Supply Chain Compromise

F-1.0.03.

The adversary may exploit the weakest/less secure link in the CI/CD pipeline or application. Mainly targeted areas are exploits on widely used open-source packages, open-source vulnerabilities, compromised CI/CD tools and changes in the build process as part of an unpatched or malicious supply chain [39].

Compromised Images in Registry

F-1.2.05.

The adversary may attempt to add a compromised image to the private registry given its access. Additionally, they can add compromised images to a public registry, like Docker Hub and expect the user to use this compromised image [46].

Execution

Deploy Container

ID: F-2.2.03.

The adversary may deploy a container via Kubernetes dashboard, Kubeflow, or via Kubernetes API server. The deployed container can be used to execute malicious commands or download malware. Another important aspect of malicious containers is that it allows using a wider range of techniques to be used and achieve persistence [47].

Scheduled Task/Job

ID: F-2.2.04.

The adversary may compromise job scheduling in containerized environments. It can be used to schedule the deployment of a container that executes malicious code. Container job scheduling works similarly to the Cron job in Linux. Thus, it performs defined tasks in a designated time [47].

CronJob in Kubernetes can be used to schedule a specific Job (e.g., a Job that executes a malicious code in the Pod) [46].

Container Lifecycle Hooks

ID: F-2.2.08.

The adversary may add container lifecycle hooks to a Pod by editing a Pod YAML file and adding, for example, postStart and preStop events. Those events can contain a malicious script [48].

Persistence

Account Manipulation

ID: F-3.1.03.

The adversary may modify cloud accounts by adding a new set of credentials. Having adversary-controlled credentials in the cloud accounts allows adversaries to have persistent access to the targeted system. There are several ways to add SSH key in the cloud environment, such as using ImportKeyPair or gcloud compute os-login ssh-keys add command in Google Cloud Platform (GCP), or CreateKeyPair API in AWS [49].

Host Mount

ID: F-2.2.07.

The adversary may mount a file or folder to a target container using hostPath volume. It is used to mount a file/folder from a host machine to a container and can be exploited by adversaries to get persistent access to the host machine [46].

CI/CD Configuration

F-2.3.16.

The adversary may try to modify CI/CD configurations. When CI/CD configurations on the Git repository are allowed without review, Git allows pushing unsigned commits. When there is no signature to CI/CD configurations, the adversary can try to change configurations. Weak audit login can help the adversary to remain undetected [45].

Privilege Escalation

Privileged Container

ID: F-4.2.05.

The adversary may get access to or deploy a privileged container. A privileged container has permission to do a wide variety of actions on the host. Thus, by having access to it, the adversary may modify the host and get access to the admin information and resources located on the same host [46, 50].

Cluster-Admin Binding

ID: F-4.2.06.

The adversary may create a binding to the privileged role or to an admin role. A role binding allows giving permissions specific to the role to the user or group of users. It is an initial part of Role-Based Access Control (RBAC) in Kubernetes [46, 51].

Defense Evasion

Impair Defenses

ID: F-5.0.02.

The adversary may turn off, exhaust, or block the defensive mechanisms of a target system. Defensive mechanisms include security tools, cloud firewalls, or cloud logs. Having these mechanisms disabled allows adversaries to avoid detecting malicious activities and system modification [52].

In cloud environments, modifying cloud firewall rules allow bypassing limitations on accessing cloud resources [53].

Masquerading

ID: F-4.2.08.

The adversary may change/manipulate the name or metadata of a file/object. Masquerading is used to evade detection. It takes place when malicious files/objects are modified to look legitimate by changing the name and defining a different file type so that the user will misidentify it [54]. Legitimate location can also be matched when applicable [55].

In containerized environments, the adversary may exploit Pod/container name similarity. Kubernetes system Pods names are appended by a random suffix. Thus, the adversary can create a Pod that will emulate the look of

the legitimate Pod by using a known Pod name and adding a random suffix [43].

Credential Access

Brute Force

ID: F-6.0.02.

The adversary may brute force system credentials to get access to the system. The adversary can perform brute force online by guessing a password to the system and receiving an instant result about their validity, or offline, by using previously obtained hashes. In addition, the adversary may use the knowledge gathered in other steps, such as valid accounts and password policy discovery. Having information about password policy decreases the time needed to brute force valid credentials [56].

Network Sniffing

ID: F-5.1.10.

The adversary may perform network sniffing of the traffic to get information about the system, users and authentication details. Network sniffing is monitoring wireless or wired networks to record data. User credentials are one of the biggest targets, especially when transmitted using unencrypted protocols. The adversary may also discover system configurations that are helpful for the consequent attack steps. In the cloud environments, the adversary may utilize GCP Packet Mirroring, Azure vTap to record traffic and AWS Traffic Mirroring [57].

Discovery

Network Service Discovery

ID: F-7.0.01.

The adversary may scan the network to discover running services or remote hosts. Some of the discovered services may be vulnerable, which can be used for exploitation. The adversary may discover services running on the non-target machine. Service discovery from the cloud host may allow the adversary to discover services of the underlying host. In the case of containerized systems, the adversary with access to one container might be able to map all networks, as, by default, there are no limitations on Pods communication [46, 58].

Cloud Infrastructure Discovery

ID: F-7.1.04.

The adversary may try to discover components of IaaS environments using cloud API and CLI. System components include cloud instances, snapshots, virtual machines and cloud services. API and CLI commands have the capability of requesting information about the infrastructure. The adversary can use the gathered information for planning the next attack steps [59].

Lateral Movement

Use Alternate Authentication Material

ID: F-8.0.01.

The adversary may use alternate authentication material to access the target system. For example, web session cookies are commonly used in cloud web applications. Session cookies keep the user logged in to the application, even when the application is not actively used. Thus, the adversary may try to steal a cookie to bypass the log in and access the data available to the victim user [60]. In containerized environments, secrets, including application credentials, can be stored in the configuration files. Thus, adversaries may attempt to retrieve the secrets to escalate privileges [46].

Writable Volume Mounts on the Host

ID: F-8.2.04.

The adversary may escape from the exploited container to the underlying system by creating a container with a writable hostPath volume. Escaping the container will give the adversary access to the host system (underlying system in the containerized environments) [46].

Collection

Data from Cloud Storage Object

ID: F-9.1.01.

The adversary may collect information from the cloud data storage that does not have proper protection. For example, retrieving data from cloud storage directly via vulnerable to attacks API. Commonly the end-users misconfigure such storage, leaving them vulnerable to attacks. Credentials gathered in the previous steps can be used for accessing storage with access permissions controls [61].

Automated Collection

ID: F-9.1.05.

The adversary may collect internal data using automatic data gathering techniques. The adversary may use cloud API, CLI, or transform services to collect information in the cloud environments automatically [62].

Exfiltration

Transfer Data to Cloud Account

ID: F-10.1.01.

The adversary may exfiltrate data and backups from one cloud account to another within the same cloud provider to avoid detection. This transfer within the same cloud provider might be more difficult to be monitored, unlike data transfers over external network interfaces [63].

Impact

Resource Hijacking

ID: F-11.0.03.

The adversary may use the system to solve resource-intensive problems, likely affecting the system's availability. The common use case is cryptojacking [64]. Cryptojacking is an act of mining cryptocurrencies. Cloud-based systems are a common target for this attack, as they have a high potential of available resources. Containerized environments are also commonly targeted because of exposed API and the possibility for the resource scaling [65].

Denial of Service

ID: F-11.4.02.

The adversary may perform a Denial-of-Service (DoS) attack to make the services unavailable to users. A DoS is an attack that is intended to cause a machine or network to be inaccessible to the intended users. There are many possible DoS attack types: Endpoint DoS, Network DoS, Application DoS, Node scheduling DoS, Service discovery DoS, SOC/Security Information and Event Management (SIEM) DoS [45, 46, 66, 67].

3.4 Implementation

As mentioned earlier, there is a need for a more efficient way to display and navigate the matrix. Therefore, we created a web application that displays a dashboard. Figure 3.5 shows a dashboard view. The application builds a dashboard view based on the data in the Excel spreadsheet. The application parses the .xlsx file during build time to a JSON file. The application uses React and NextJS for displaying the page. Styling is implemented using Tailwind library. It is written in TypeScript. The category label is determined by the technique ID. The dashboard view of the matrix contains all techniques and can be filtered by category. Having a dashboard application provides us with a better look and usability for a future building on the other components of the framework. The source code of the matrix is available on its GitHub repository `https://github.com/fraktalcyber/threat-matrix-CN`.

3.5 Background work

The *Cloud Native Threat Matrix* is based on the information collected from multiple resources. The primary resources that the threat matrix is based on are listed below.

MITRE ATT&CK Cloud Matrix

As a part of the MITRE ATT&CK framework, the Cloud matrix belongs to the Enterprise section, as discussed in the section 2.2. It covers cloud-related techniques based on five platforms, such as Office 365, Google Workspace, Azure AD, SaaS and IaaS [68]. For the *Cloud Native Threat Matrix*, we used techniques related to the IaaS and SaaS platforms. These techniques are divided into 11 tactics.

MITRE ATT&CK Containers Matrix

The MITRE ATT&CK containers matrix covers techniques targeted to the containerized technologies. It is a part of MITRE ATT&CK Enterprise, as discussed in the section 2.2. It has 9 tactics, such as Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement and Impact [69].

Microsoft Threat Matrix for Kubernetes

Initially, MITRE ATT&CK only provided coverage for Windows and Linux systems. The Azure Security Center noticed that Windows and Linux systems share plenty of techniques with Kubernetes [70]. Thus, the Microsoft Threat matrix for Kubernetes was created as an ATT&CK-like matrix of techniques related to the container orchestration security focusing on Kubernetes [43, 46]. It was built on and influenced by the MITRE ATT&CK framework. It has 9 tactics and 40 techniques.

Hacking Kubernetes

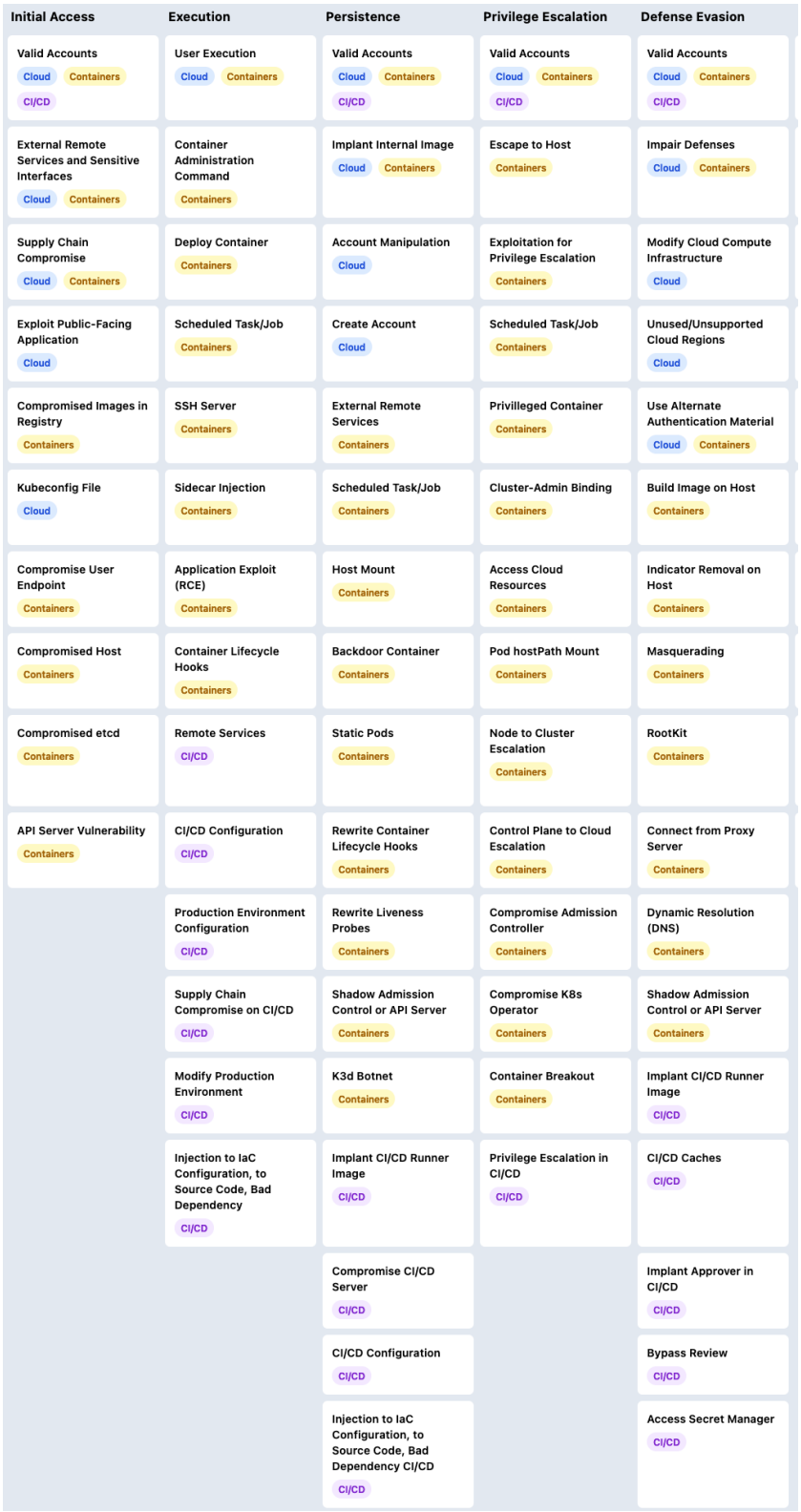
Hacking Kubernetes is a book authored by Andrew Martin and Michael Hausenblas, that contains a threat-based guide to the Kubernetes security [48]. The authors provide practical advice on how to configure a Kubernetes cluster from the perspective of the adversary. Authors reference a Microsoft Threat matrix for Kubernetes and add 69 new techniques shared within 9 tactics to it.

Common Threat Matrix for CI/CD Pipeline

It is an ATT&CK-like matrix that focuses on CI/CD pipeline-specific threats [45]. Its main goal is to educate the cybersecurity community on how to secure CI/CD pipelines. It uses the same structure as the MITRE ATT&CK framework. It has 29 techniques shared between 9 tactics.

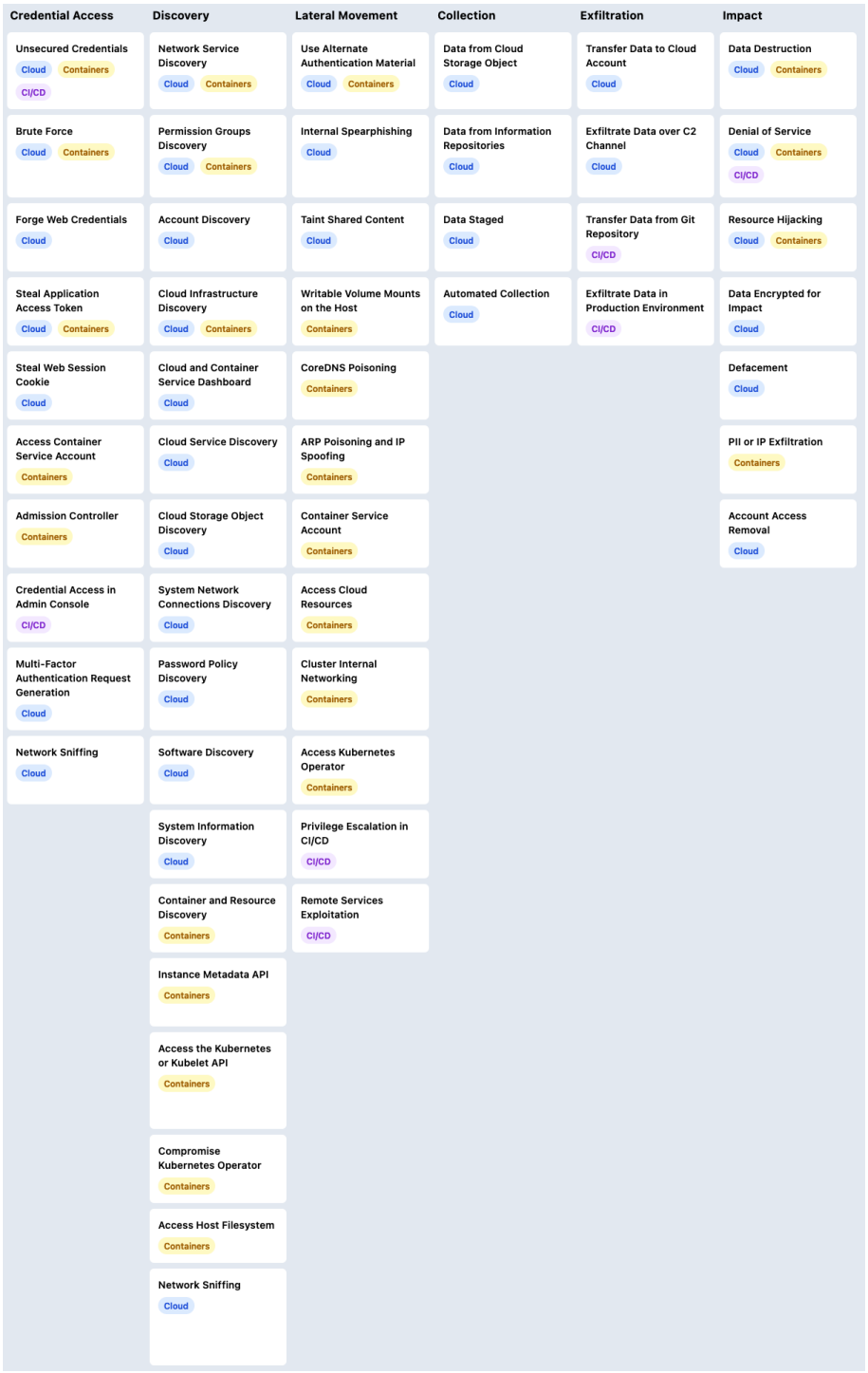
Other public research reports (Aqua Security)

Aqua Security is a cloud native security company that conducts a plethora of security research projects. The most relevant to this thesis work is "Cloud Native Threat Report: Attacks in the Wild on the Container Supply Chain and Infrastructure" [71]. This project was conducted in 2020 and 2021. The 2021 report contains the data collected from honeypots for the duration of six months, during which the Aqua Security team observed more than 17 thousand attacks. The results were mapped to the MITRE ATT&CK Containers matrix and added 19 new techniques split over 9 tactics.



(a) Part 1 of the *Cloud Native Threat Matrix*.

Figure 3.5: The *Cloud Native Threat Matrix* dashboard view.



(b) Part 2 of the *Cloud Native Threat Matrix*.

Figure 3.5: The *Cloud Native Threat Matrix* dashboard view.

Chapter 4

Attack automation

In this chapter, we present an overview of attack automation. Firstly, we provide a methodology. Next, we describe a tool for techniques automation. We present a laboratory environment for tool testing. Finally, we describe combining techniques into attacks.

4.1 Methodology

The overall methodology follows the design science approach that focuses on problem-solving [72] and designing a tool based on predefined needs. We picked design science as it bridges the gap between academic research and the requirements of the organization. Furthermore, this thesis work is based on an industrial collaboration. Thus, the aim is to provide a feasible tool for attack techniques automation to improve purple team evaluations of cloud native environments.

We followed the principles of a problem-solving cycle based on the design research for the automation tool development. Figure 4.1 shows steps of the problem-solving cycle [73]. The first step of this approach requires a clear definition of the problem. We defined a problem at the very beginning of the thesis work in the section 1.3. The next step is the analysis and diagnosis of the problem. During this step, we accomplished a literature review and defined existing solutions for the problem, discussed in the section 2.4. We identified that we need a simple static solution, and current tools do not cover this requirement. Thus, we started a solution design. We identified that Fraktal Oy has an internal automation tool that fits the requirements of a simple and static tool. We designed our tool inspired by Fraktal's internal automation tool. As an intervention, we implemented a solution in the organization. As a final step, we evaluated the solution by implementing

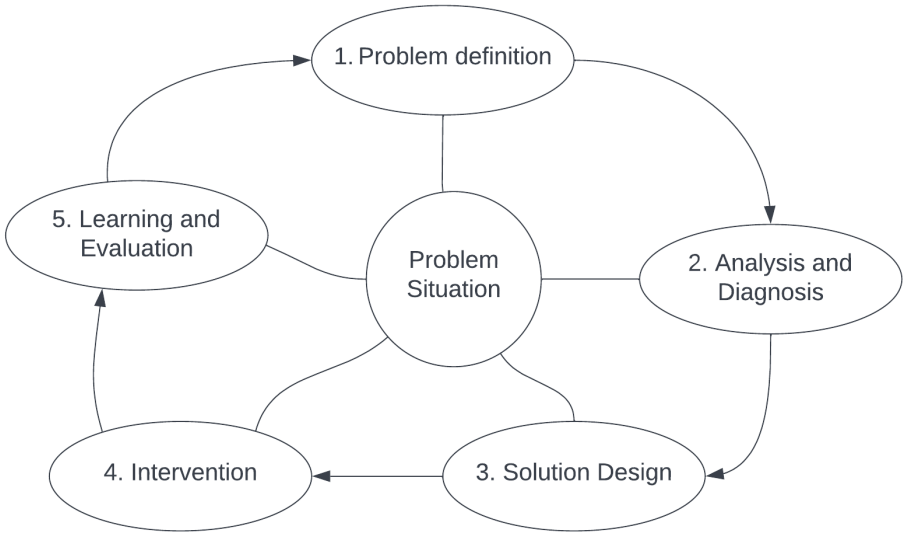


Figure 4.1: Problem-solving cycle [73].

automation of attack techniques and testing them in the laboratory environment, as discussed in section 4.4.

4.2 Tool for the Techniques Automation

Our tool is inspired by Fraktal's internal automation tool, which is a simple container-based tool for running automated security tests. Individual tests are modules that can be executed sequentially. The internal automation tool has a set of simple commands automated (e.g., Network Mapper (nmap) port scan). However, automated commands are not exclusive to cloud native environments.

The automation tool created for the thesis work is a closed source software built to perform the cloud native attack techniques automation. It is written in Go and has a structured way of adding new techniques description and commands for execution. After the proper framework and the process are both in place, adding new techniques is relatively easy. The tool is lightweight and can be containerized for easy deployment in the target environment.

In our automation tool, we want to control when and which techniques are being executed as it is undesirable for the attack container to automatically run the automation every time it is installed or executed in the target

environment. We also aim to have a standardized way to trigger the deployment of the attack container and its deletion after the testing is completed. Therefore, we developed a script runner. Script runner is a bash code that triggers several actions. The script runner deploys a Pod in the target cluster, executes all or selected attack techniques, cleans up after techniques execution and deletes a Pod from the cluster. The user selects and triggers commands for the execution from the list of commands. We show the CLI tool for a script runner below.

```
$ ./scriptRunner.sh
    Commands: :
    0 - deploy an attack automation container in the
        cluster
    1 - list all techniques
    2 - execute all techniques
    3 - clean up after executing all techniques
    4 - pick a technique for the execution
    5 - delete the attack automation container from the
        cluster
    exit - terminate
    Please select a command:
    1
    Listing all commands...
    help        - Prints command help.
    execute     - Runs all attack simulations with default
                  arguments.
    cleanUp     - Clean up after simulation of all techniques.
    dud         - Simulates dud attack i.e. does nothing.
    dpcn        - Simulates deploying container.
    lfhk        - Simulates creating a pod with life-cycle
                  hooks.
    scjb        - Simulates scheduling a job.
    dprc        - Simulates deploying a privileged container.
    clab        - Simulates a cluster admin binding.
    ascr        - Simulates accessing cloud resources.
```

Assume Breach Approach

Our automation tool is built based on the assume breach principle. When securing systems and applications, we always understand that there is no complete security, which means that security incidents are inevitable. Thus, when building security for systems, we have to ensure that the blast radius of the breach is limited, meaning that we are able to detect and respond to the attack in a timely manner [74].

The traditional approach for building security controls and defenses is a prevent breach approach. A prevent breach focuses on the security of the system's outer layer to avoid security breaches. It contains preventive controls that try to secure a system from the initial access of the adversary. However, preventative security controls are not enough in the modern landscape of new threats. For instance, the log4j incident [75] or Dependency Confusion attack [76] could not be predicted, and once they happen, something other than the outer perimeter of defense is required.

That is mainly why an assume breach approach has gained popularity in recent years. Building security controls based on the assume breach approach accepts that the adversary already has access to the system. The testing is focused on checking if protections inside the system can prevent the adversaries from lateral movement/privilege escalation to minimize the final impact on the system. An assume breach approach helps to build the defense-in-depth system capabilities. The defense-in-depth is an approach to building layered defense mechanisms to protect a system in situations where a security control fails or a vulnerability is exploited [77]. The defense-in-depth is needed because the outer layer of security cannot be relied on.

4.3 Laboratory Environment

For testing the automation tool, we set up a restricted laboratory environment. Figure 4.2 shows the architecture of the laboratory environment. It is hosted on the AWS cloud. The main building blocks of the laboratory environment are:

- **Elastic Container Registry (ECR)** is a fully managed container registry, that allows to deploy, share and store application images. In our laboratory environment, this module stores an attack container image that is used for deployment in the target system. The attack container image is built and pushed to the ECR from the bastion host.

- **Bastion host** is a server that provides access to the private network

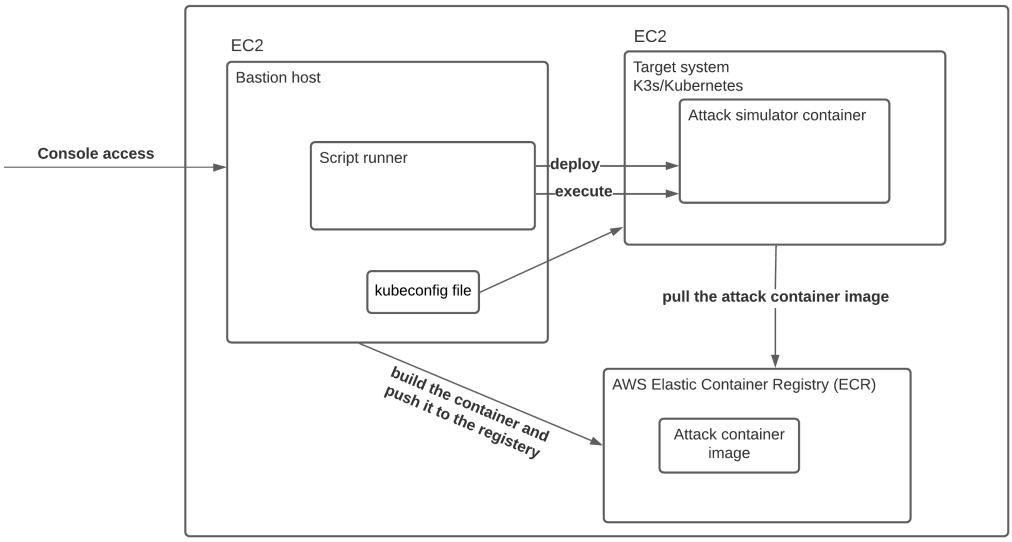


Figure 4.2: Architecture of laboratory environment for the attack simulation testing.

from the external network. In our laboratory environment, it is an Amazon Elastic Compute Cloud (Amazon EC2) instance that hosts the script runner. It is used to trigger the execution of techniques and receive the output of the techniques execution. It provides a remote connection to the private instance with a script runner through the public internet. Bastion host also hosts a kubeconfig file for the target cluster access.

- **Target system** is the cloud native system that is targeted with automated attacks. In our restricted laboratory environment, a target system is an Amazon EC2 instance with a running K3s (a lightweight Kubernetes distribution) Kubernetes cluster.

4.4 Implemented Techniques

To execute attacks, the attack container should be deployed in the target environment. Then one or multiple techniques should be triggered. The log of technique execution is collected at the bastion host and provides whether the technique was successfully run.

We tested our automation tool by running automated techniques in the laboratory environment. We automated 7 techniques from 4 different tactic categories:

- **Execution:** Deploy container, Scheduled Task/Job, Container Lifecycle Hooks.
- **Persistence:** Host Mount.
- **Privilege Escalation:** Privileged Container, Cluster-admin Binding.
- **Discovery:** Cloud Infrastructure Discovery.

Next, we describe more detailed techniques automation.

Technique: Deploy Container

The goal of this technique is to facilitate the execution of malicious commands or to download malware. It may be achieved by deploying a container in the target cluster. The adversary would not usually be able to deploy containers. However, the cluster might be misconfigured, and given too much privileges, it might allow this technique to be executed.

To execute the technique in the target environment, we trigger "dpcn – Simulates deploying container" command from the script runner. The attack automation tool executes the command on the target machine. It pulls the specification file that is stored inside the attack container. Considering that the tool has access to the kubectl, it triggers the *kubectl apply* command to deploy a container.

We present the view of the CLI tool for a script runner after triggering the deploy container technique execution:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
dpcn
deployment.apps/new-deployment created

The container is successfully deployed.
```

To ensure that the technique was executed correctly, we list Pods:

```
$ kubectl get pods
NAME                                READY STATUS    AGE
attack                              1/1   Running   30h
new-deployment-66b6c48dd5-7hvjr     1/1   Running   13m
new-deployment-66b6c48dd5-sp75j     1/1   Running   13m
new-deployment-66b6c48dd5-vrpsr     1/1   Running   13m
```

As shown above, we can see that 3 replicas of the container were created in the cluster after the technique execution.

Technique: Scheduled Task/Job

ID: F-2.2.04.

The goal of this technique is to schedule a job in the environment to achieve persistence or defense evasion. As containers might be recycled/destroyed, a Cron job could recreate the malicious containers for the adversary. Usually, the adversary should not be able to schedule a Kubernetes CronJob. However, in the misconfigured cluster, the service accounts may be given privileges to the Kubernetes apiserver. Such privileges should not be given to service accounts (i.e., machine users of the Kubernetes cluster).

In our case, we simulate the scheduling of a Kubernetes CronJob. To execute this technique, we trigger "scjb – Simulates scheduling a job" from the script runner. Then, the automation tool pulls a job configuration file and deploys a new job using *kubectl apply*.

The configuration file of the job scheduling:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
```

```
    image: busybox:stable
    imagePullPolicy: IfNotPresent
    command:
    - /bin/sh
    - -c
    - date; echo Hello World
  restartPolicy: OnFailure
```

We present the view of the CLI tool for a script runner after triggering the scheduled job technique execution:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
scjb
cronjob.batch/hello created

The job is successfully scheduled.
```

To check that the technique was executed correctly, we list CronJobs:

```
$ kubectl get cronjobs
NAME   SCHEDULE      SUSPEND ACTIVE LAST SCHEDULE AGE
hello  */1 * * * *  False   0      14s           55s
```

As shown above, we can see that a new job was created.

Technique: Host Mount

ID: F-2.2.07.

The goal of this technique is to get persistent access and escalate privileges on the host machine. Access may be gained from the container to the host by mounting a file or folder to a target container using hostPath volume. Usually, the adversary should not be able to add hostPath volume. However, it is possible in a misconfigured system with RBAC that does not forbid doing so.

To execute this technique, we trigger "smhm – Simulates host mount" from the script runner. The tool pulls a specification file stored inside the

attack container. The specification file defines a privileged Pod with a host mount and copy of the public key to the Node's .ssh/authorized_keys file to gain persistent access to the system.

The container specification file:

```
apiVersion: vk
kind: Pod
metadata:
  name: pod
  namespace: default
spec:
  containers:
  - image: busybox
    name: evil-pod
    command:
    ["/bin/sh", "-c", "while true; do sleep 3600; done"]
    args:
    ["-c", " echo ssh-rsa <public_key> <user_name>@<ip_adress>
    >> /host/home/user/.ssh/authorized\_keys ",
    "while :; do echo '.'; sleep 5 ; done"]
    volumeMounts:
    - mountPath: /host
      name: host-root
    securityContext:
      privileged: true
  volumes:
  - hostPath:
      path: /
      type: ""
    name: host-root
```

Technique: Container Lifecycle Hooks

ID: F-2.2.08.

The goal of this technique is to execute a malicious code inside the target system. It may be performed by adding container lifecycle hooks to a Pod, such as postStart and preStop events. Usually, the adversary should not be able to add container lifecycle hooks. This still can be possible in a misconfigured system with RBAC that does not follow the least privilege

principle [41]. Where the least privilege principle implies that a user is given the minimum levels of permissions needed to perform their functions.

To run this technique, we trigger "link – Simulates creating a Pod with life-cycle hooks" from a script runner. It then triggers kubectl to create a new Pod with life-cycle hooks adding postStart and preStop events to the Pod configuration file.

Relevant part of the postStart and preStop events configurations is shown below.

```
lifecycle:
  postStart:
    exec:
      command: ["/bin/sh", "-c",
      "echo Hello from the postStart handler >
      /usr/share/message"]
  preStop:
    exec:
      command: ["/bin/sh","-c","nginx -s quit;
      while killall -0 nginx; do sleep 1; done"]
```

We present the view of the CLI tool for a script runner after the container lifecycle hook technique execution is triggered:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
lfhk
pod/lifecycle-demo created

The pod with a life-cycle hooks is successfully deployed.
```

We list Pods to check that the technique was executed correctly, as it is shown below.

```
$ kubectl get pods
NAME                                READY STATUS  AGE
attack                              1/1   Running 30h
lifecycle-demo                      1/1   Running 50s
```

Technique: Privileged Container Technique

ID: F-4.2.05.

The goal of this technique is to escalate privileges and/or access the host system by getting access or deploying a privileged container. A privileged container has permission to do a wide variety of actions on the host. In the carefully secured cluster, this technique is normally not available to adversaries because the cluster should be configured with least privilege principle.

To execute this technique, we trigger "dprc – Simulates deploying a privileged container". The tool then pulls a configuration file and using *kubectl apply* deploys a privileged container.

We present the view of the CLI tool for a script runner after triggering the privileged container technique execution:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
dprc
deployment.apps/nginx-deployment created

The privileged container is successfully deployed.
```

We list Pods to check that the technique was executed correctly, as it is shown below:

```
$ kubectl get pods
NAME                                READY STATUS   AGE
attack                              1/1   Running  30h
nginx-deployment-645bf95774-fdjmn   1/1   Running  45s
```

Technique: Cluster-Admin Binding

ID: F-4.2.06.

The goal of this technique is to escalate privileges to gain full access to the cluster by creating a binding to the privileged or admin role. Cluster-Admin binding should not be possible in the carefully secured system. In a secured system, users that can have a cluster-admin role should be limited, as well as, users that can create role bindings.

To trigger execution of this technique, we pick "clab – Simulates a cluster admin binding" option on the script runner. After triggering, the tool pulls a configuration file and performs a cluster-admin binding.

The configuration file for the service account creation is shown below:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: myaccount
```

The configuration file for the cluster role binding:

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding
subjects:
  - kind: ServiceAccount
    name: myaccount
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

We present the view of the CLI tool for a script runner after triggering the cluster-admin binding technique execution:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
clab
serviceaccount/myaccount created

clusterrolebinding.rbac.authorization.k8s.io/
cluster-role-binding created
```

```
The cluster admin binding was successfully performed.
```

Technique: Cloud Infrastructure Discovery

ID: F-7.1.04.

The goal of this technique is to discover components of IaaS environments using cloud API and CLI. The adversary may try to enumerate system components, including cloud instances, snapshots, virtual machines, cloud services to define the next steps of attack execution. Gained information may also be used to modify the system's configuration to be publicly accessible. The other use case for the cloud infrastructure discovery is to enumerate the database configurations to define their potential value for the next steps of the attack.

The adversary will not be able to successfully execute this technique if the system has restricted access via Network Policies in Kubernetes.

We execute the shown below command to perform this technique in the target environment.

```
curl http://169.254.169.254/latest/meta-data/
```

We show the output of the technique execution from the script runner:

```
$ Please select a command:
4
Enter the technique short name, as specified
in the all techniques listing:
ascr
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hibernation/
hostname
iam/
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
```

```
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups

Cloud resources are successfully accessed.
```

As we can see from the output of the technique execution we successfully listed meta-data of the cloud environment.

4.5 Combining Techniques into Attacks

We previously showcased the automated execution of separate techniques. It is important to automate different types of techniques to check if a target environment can detect attempts of their execution or if it is resistant to their execution or impact. In case of attack execution, techniques are linked in the attack chains. Typically, the adversary will pick techniques to get access to a system, execute malicious scripts, try to escalate privileges to have more options for the next steps, hide the activity not to be quickly detected and impact the system. The combination of steps varies from attack to attack, but the basic idea of attack construction remains the same. The steps that the adversary performs are usually aligned with MITRE ATT&CK tactics, same steps that we follow in the proposed threat matrix. Out of 11 tactics for cloud native environments, not all tactics have to be utilized as an attack step. Typically, the best impact may be achieved by combining a few techniques from selected tactics appropriate to the target [78].

To utilize automated in this thesis work techniques and showcase building the attack chain, we build an attack based on the techniques listed in the section 4.4. We perform the attack execution in the lab environment. Following the assume breach approach, we assume that we have initial access to the system. The next step is execution, we schedule a privileged container deployment. A privileged container has a mount from the Pod to the file

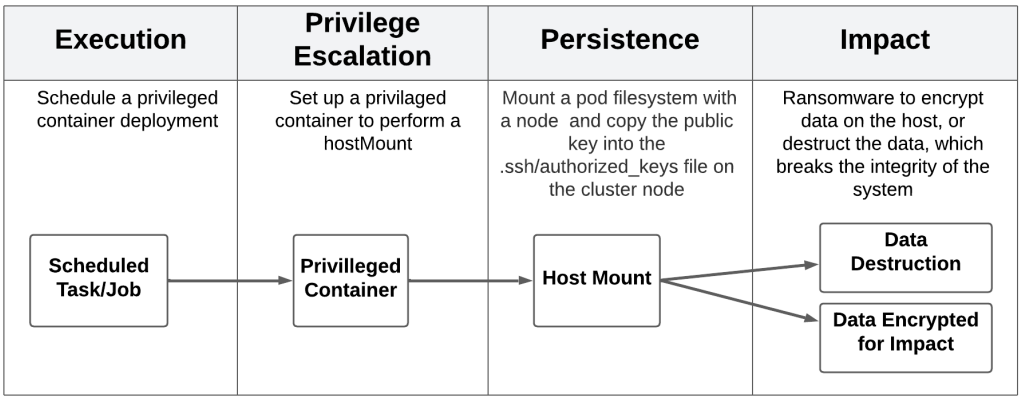


Figure 4.3: Linked techniques into the attack.

system of a Node. As a persistence step, we perform account manipulation by adding a copy of the public key into the .ssh/authorized_keys file on the cluster Node. With persistent access and privileged rights, we can perform different impact-related techniques, such as using ransomware to encrypt data on the host or destructing the data, which breaks the integrity of the system. Figure 4.3 shows linked techniques into the attack.

Chapter 5

Discussion

In this chapter, we discuss the advantages and disadvantages of the research conducted and the artifact created in this thesis work. Firstly, we review the research process. Second, we discuss the improvements proposed to the existing threat matrix. Thirdly, we examine the process of creating the automation tool. Next, we discuss the potentially achievable improvements in purple team exercises using the threat matrix method and automation tool.

5.1 Research Process

The research followed the design science research methodology [79]. This was done due to the fast-evolving nature of the cloud environment. The iterative process of triangulating between cloud native threats, automating testing and improving the purple team's performance required several cycles of creating, experimenting, learning and amending. These development cycles were easier to manage because of the business and operational context as well as the competency provided at Fraktal Oy [5] for the research. For example, the variations in threat naming and selecting the cloud native specific threats from the plethora of possible threats required a reflection from practice. The reflection also provided necessary viewpoints for lean design thinking [80], which improved the usability of the final artifacts. Naturally, continuing to repeat the cycles of iteration would improve the threat matrix, tool and be a perfect aim for additional research because of the fast-evolving nature of the cloud native environment.

5.2 Cloud Native Threat Matrix

The first research question, "What type of new cyber threats do cloud native environments face?" presented a typical challenge in dynamic and open systems. Therefore, answering the first question of what is the cloud native requires defining features for the cloud native system. There are several approaches for cloud native architecture [81], but they do not support the target viewpoint. Hence, the research composed a threat model from components of cloud, containers and CI/CD to keep the focus during the threat survey.

Answering the second sub-question of what are the new cyber threats, especially those targeting the cloud native architecture, required surveying and categorizing processes. First, sourcing the attack vectors opened several options. For example, there are many contemporary studies discussing threats from the perspective of a specific time scope [82], continuous collection of threats [83], generic attack vector models [84] or cumulative threat knowledge base [85]. Since the cumulative approach provides more than a snapshot of the dynamic threat environment, the MITRE ATT&CK knowledge base established the foundation for threat information. Moreover, other cumulative sources like Microsoft, Linux Foundation, Aqua Security, CI/CD pipeline threat research were referred to and correlated when collecting and normalizing feasible presentation of contemporary threats.

Cloud native threat vectors created a large matrix, and the original Excel format appeared inefficient for the users. Therefore, research designed a dashboard for the testing teams to get a more comfortable presentation of the vast amount of data. The purple team in Fraktal Oy appreciated the improved usability.

5.3 Automation Tool

Besides improving the focus and usability, the research's second question, "How to effectively utilize the automated purple team approach for attack automation in cloud native environments?" aimed to examine how automation may improve the testing process. Fortunately, Fraktal's internal automation platform provided an integrated toolbox for automation testing. Therefore, the tool provided in this research did not remain a stand-alone tool. Since the development followed a mature state of policy, language, and integration, the automation tool emerged lightweight and ready to evolve as the demands of in dynamic threat environment and testing process change.

Recognising the ethical hacking principles [86], the research chose to test the tool in a controlled laboratory environment. The testing included seven

techniques from four different categories to keep the sampling broad and deep while keeping the programming work aligned with the focus of the research. During the process, the research observed that not all 11 tactics categorized in the matrix need to apply in a successful attack. However, the best impact may be achieved by combining a few techniques from selected tactics appropriate to the target.

5.4 The Performance Improvement of the Purple Team Evaluation Process

Combining the created matrix and the automation tool provides the following improvements to purple team activities:

- The improved threat matrix helps purple teams to focus on cloud native specific threats and shorten the time for their test planning.
- The automation tool includes a more straightforward work process for security evaluations which standardizes the process and improves its maturity.
- Utilizing automation allows making sure that security weaknesses are not re-introduced during major changes. Automation allows running periodic security checks that do not require manual labor as part of the continuous integration and testing of the service.
- The matrix and tool support iterative development. Thus, purple teams may develop them further as threats and targets evolve. Moreover, the purple teams using the package can start small and elaborate in steps as their competency and objectives evolve.
- A purple team approach allows to develop and verify system detections with a SIEM or other log monitoring systems. This makes it more probable to detect real attacks in a production environment.

Chapter 6

Conclusions

This thesis explores a purple team approach to attack automation in the cloud native environment. It firstly investigates cyber threats that cloud native environments face. As a result of the research, we created a more comprehensive resource of the cloud native threats that we refer to as a *Cloud Native Threat Matrix*. Threats are grouped by the goals of the adversary and by the environment it is related to, such as containers, cloud and CI/CD. The other aspect of this thesis work is the utilization of purple team activities together with attack automation. We built the tool for attack automation. The tool follows the assume breach approach, providing a defense-in-depth security testing of cloud native environments. As a final step, we propose an improvement to the purple team evaluation of the cloud native environments. It combines created *Cloud Native Threat Matrix* with an automated attack techniques execution and active collaboration as a fundamental concept of purple team assessments.

The assume breach approach helps to evaluate the possible system impact, and improve defense capabilities to minimize a possible system impact after the adversary got initial access to the system. The benefit of utilizing automation is the possibility of rerunning such assessments and making sure that security weaknesses are not re-introduced during major changes. In addition, a *Cloud Native Threat Matrix* solves the problem of scattered threat data over to the Internet, providing a coherent and easy-to-use platform. Furthermore, the purple team approach to the automated attack techniques execution helps to improve system defense capabilities based on the active collaboration during the testing process.

6.1 Limitations

The most significant limitation of thesis work is that we could not test a purple team evaluation of the cloud native environments in the actual client setting. The purple team approach requires an active collaboration with an organization's blue team to test or set up detection capabilities. Thus, the evaluation effectiveness cannot be assessed.

6.2 Improvements

The contemporary threat landscape is constantly changing. Thus, a matrix needs to be updated frequently. This is a time-consuming process. Therefore, building an update pipeline that will make this update easier to approach is beneficial. One of the solutions can be building a hybrid matrix that references different matrices to avoid constant updates.

Another possible improvement is expanding the scope of the cloud native threats by adding network threats to the matrix, as discussed in [87].

6.3 Future Work

The next step of the project development is to increase the number of automated techniques, integrate the automation framework into the *Cloud Native Threat Matrix* User Interface (UI) and test the purple team evaluation of the cloud native environments in the client setting.

Bibliography

[1] The Linux Foundation. Cloud native computing foundation charter, 2015. URL: https://github.com/cncf/foundation/blob/main/charter.md.

[2] Red Hat. Kubernetes adoption, security, and market trends report 2022. 2022.

[3] Jay Chen, Nathaniel "Q" Quist, Matthew Chiodi. Cloud threat report 1h 2021. 2021.

[4] The MITRE Corporation. Mitre att&ck, 2015. URL: https://attack.mitre.org/.

[5] Fraktal. Fraktal: Positive takes on cyber security, 2022. URL: https://www.fraktal.fi/about.

[6] IBM Cloud Education. Virtualization a complete guide, June 2019. URL: https://www.ibm.com/cloud/learn/virtualization-a-complete-guide.

[7] VMware. What is a virtual machine?, May 2022. URL: https://www.vmware.com/topics/glossary/content/virtual-machine.html.

[8] Masdari Mohammad, Nabavi Sayyid, Ahmadi Vafa. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66:32, January 2016.

[9] Docker. What is a container?, April 2022. URL: https://www.docker.com/resources/what-container.

[10] Combe Theo, Martin Antony, Pietro Roberto. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3:54–62, 09 2016.

[11] IBM Cloud Team, IBM Cloud. Containers vs. virtual machines (vms): What is the difference?, 2021. URL: `https://www.ibm.com/cloud/blog/containers-vs-vms`.

[12] Docker. Docker overview. URL: `https://docs.docker.com/get-started/overview/`.

[13] Asif Rameez, Ghanem Kinan, Irvine James. Containerization: For over-the-air programming of field deployed internet-of-energy based on cost effective lpwan. In *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 65–70, 2020.

[14] The Linux Foundation. What is kubernetes?, April 2022. URL: `https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/`.

[15] The Linux Foundation. Kubernetes components, April 2022. URL: `https://kubernetes.io/docs/concepts/overview/components/`.

[16] The Linux Foundation. Pods, January 2022. URL: `https://kubernetes.io/docs/concepts/workloads/pods/`.

[17] The Linux Foundation. Nodes, April 2022. URL: `https://kubernetes.io/docs/concepts/architecture/nodes/`.

[18] The Linux Foundation. Kubernetes concepts, 2022. URL: `https://kubernetes.io/docs/concepts/_print/`.

[19] Dragoni Nicola, Giallorenzo Saverio, Lluch-Lafuente Alberto, Mazzara Manuel, Montesi Fabrizio, Mustafin Ruslan, Safina Larisa. Microservices: yesterday, today, and tomorrow. page 16, 06 2016.

[20] Rossel Sander. *Continuous integration, delivery, and deployment: Reliable and faster software releases with automating builds, tests, and deployment.* Packt, 2017.

[21] RedHat. What is ci/cd?, May 2022. URL: `https://www.redhat.com/en/topics/devops/what-is-ci-cd`.

[22] Loukides Mike. *What is devops?* O'Reilly Media., 2012.

[23] Al-Mohannadi Hamad, Mirza Qublai, Namanya Anitta, Awan Irfan, Cullen Andrea, Disso Jules. Cyber-attack modeling analysis techniques: An overview. pages 69–76, 2016.

[24] National Cyber Security Centre. Cyclops blink. malware analysis report, February 2022. URL: https://www.ncsc.gov.uk/files/Cyclops-Blink-Malware-Analysis-Report.pdf.

[25] The MITRE Corporation. Corporate overview, October 2020. URL: https://www.mitre.org/about/corporate-overview.

[26] Rehberger Johann. *Cybersecurity attacks - Red Team Strategies: A practical guide to building a penetration testing program having homefield advantage*. Packt Publishing Ltd, 2020.

[27] Nadean Tanner. *Cybersecurity Blue Team Toolkit*. Wiley, 2019.

[28] Redscan. What is purple teaming? how can it strengthen your security?, May 2022. URL: https://www.redscan.com/news/purple-teaming-can-strengthen-cyber-security/.

[29] XM Cyber. What is a purple team?, March 2022. URL: https://www.xmcyber.com/glossary/what-is-a-purple-team/.

[30] Automated technologies vs manual testing?, August 2018. URL: https://www.packetlabs.net/posts/automated-technologies-vs-manual-testing/.

[31] Stefinko Yaroslav, Piskozub Andrian, Banakh Roman. Manual and automated penetration testing. benefits and drawbacks. modern tendency. *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, 2016.

[32] Redcanaryco. Explore atomic red team, 2017. URL: https://atomicredteam.io/.

[33] Stratus Red Team. Stratus red team, 2021. URL: https://stratus-red-team.cloud/.

[34] FSecureLABS. Fsecurelabs/leonidas: Automated attack simulation in the cloud, complete with detection use cases., 2021. URL: https://github.com/FSecureLABS/leonidas.

[35] Spencer Gietzen. Pacu: The open source aws exploitation framework, August 2018. URL: https://rhinosecuritylabs.com/aws/pacu-open-source-aws-exploitation-framework/.

[36] The MITRE Corporation. Caldera, May 2020. URL: https://caldera.mitre.org/.

[37] Petersen Kai, Feldt Robert, Mujtaba Shahid, Mattsson Michael. Systematic mapping studies in software engineering. *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 17, 06 2008.

[38] Microsoft. Evaluate security postures by using microsoft defender for cloud, 2022. URL: `https://docs.microsoft.com/en-gb/learn/modules/evaluate-security-posture-recommend-technical-strategies-to-manage-risk/3-postures-use-microsoft-defender-for-cloud`.

[39] Argon Security. Software supply chain attacks: 2021 in review. 2022.

[40] Lazarovitz Lavi. Deconstructing the solarwinds breach. *Computer Fraud amp; Security*, 2021(6):17–19, 2021.

[41] Rice Liz. *Container security: Fundamental technology concepts that protect containerized applications*. O'Reilly Media, 2020.

[42] Red Hat. What is container security? URL: `https://www.redhat.com/en/topics/security/container-security`.

[43] Yossi Weizman. Secure containerized environments with updated threat matrix for kubernetes, March 2021. URL: `https://www.microsoft.com/security/blog/2021/03/23/secure-containerized-environments-with-updated-threat-matrix-for-kubernetes/`.

[44] The MITRE Corporation. Valid accounts, May 2017. URL: `https://attack.mitre.org/techniques/T1078/`.

[45] Mercari Security Team. Common threat matrix for ci/cd pipeline. URL: `https://github.com/rung/threat-matrix-cicd`.

[46] Yossi Weizman. Threat matrix for kubernetes, April 2020. URL: `https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/`.

[47] The MITRE Corporation. Deploy container. URL: `https://attack.mitre.org/techniques/T1610/`.

[48] Martin Andrew, Hausenblas Michael. *Hacking Kubernetes*. O'Reilly Media, Inc, 2021.

[49] The MITRE Corporation. Account manipulation: Additional cloud credentials, January 2021. URL: `https://attack.mitre.org/techniques/T1098/001/`.

[50] Dang Wei Lien. Protecting kubernetes against mitre att/ck: Privilege escalation. URL: https://cloud.redhat.com/blog/protecting-kubernetes-against-mitre-attck-privilege-escalation.

[51] Kubernetes. Using rbac authorization, April 2022. URL: https://kubernetes.io/docs/reference/access-authn-authz/rbac/.

[52] The MITRE Corporation. Impair defenses, February 2020. URL: https://attack.mitre.org/techniques/T1562/.

[53] The MITRE Corporation. Impair defenses: Disable or modify cloud firewall, June 2020. URL: https://attack.mitre.org/techniques/T1562/007/.

[54] The MITRE Corporation. Masquerading, May 2017. URL: https://attack.mitre.org/techniques/T1036/.

[55] The MITRE Corporation. Masquerading: Match legitimate name or location, February 2020. URL: https://attack.mitre.org/techniques/T1036/005/.

[56] The MITRE Corporation. Brute force, May 2017. URL: https://attack.mitre.org/techniques/T1110/.

[57] The MITRE Corporation. Network sniffing, May 2017. URL: https://attack.mitre.org/techniques/T1040/.

[58] The MITRE Corporation. Network service discovery, May 2017. URL: https://attack.mitre.org/techniques/T1046/.

[59] The MITRE Corporation. Cloud infrastructure discovery, August 2020. URL: https://attack.mitre.org/techniques/T1580/.

[60] The MITRE Corporation. Use alternate authentication material, January 2020. URL: https://attack.mitre.org/techniques/T1550/.

[61] The MITRE Corporation. Data from cloud storage object, August 2019. URL: https://attack.mitre.org/techniques/T1530/.

[62] The MITRE Corporation. Automated collection, May 2017. URL: https://attack.mitre.org/techniques/T1119/.

[63] The MITRE Corporation. Transfer data to cloud account, August 2019. URL: https://attack.mitre.org/techniques/T1537/.

[64] Jayasinghe Keshani, Poravi Guhanathan. A survey of attack instances of cryptojacking targeting cloud infrastructure. *Proceedings of the 2020 2nd Asia Pacific Information Technology Conference*, 2020.

[65] The MITRE Corporation. Resource hijacking, April 2019. URL: `https://attack.mitre.org/techniques/T1496/`.

[66] The MITRE Corporation. Endpoint denial of service, April 2019. URL: `https://attack.mitre.org/techniques/T1499/`.

[67] The MITRE Corporation. Network denial of service, April 2019. URL: `https://attack.mitre.org/techniques/T1498/`.

[68] The MITRE Corporation. Cloud matrix, April 2022. URL: `https://attack.mitre.org/matrices/enterprise/cloud/`.

[69] The MITRE Corporation. Containers matrix, April 2022. URL: `https://attack.mitre.org/matrices/enterprise/containers/`.

[70] Jen Burns. Help shape att&ck for containers, Jun 2021.

[71] Aqua Security. Cloud native threat report: Attacks in the wild on the container supply chain and infrastructure. 2022.

[72] Dresch Aline, Lacerda Daniel Pacheco, Antunes José Antônio Valle. Design science research. In *Design science research*, pages 67–102. Springer, 2015.

[73] J.E van Aken, Johannes Jentinus Berends. *Problem solving in organizations: A methodological handbook for business and management students*. Cambridge University Press, 2019.

[74] Steve Anson. *Applied incident response*. Wiley, 2020.

[75] National Cyber Security Center. Log4j vulnerability - what everyone needs to know, December 2021. URL: `https://www.ncsc.gov.uk/information/log4j-vulnerability-what-everyone-needs-to-know`.

[76] Ravie Lakshmanan. Dependency confusion supply-chain attack hit over 35 high-profile companies, February 2021. URL: `https://thehackernews.com/2021/02/dependency-confusion-supply-chain.html`.

[77] Matthew Rosenquist. Defense in depth strategy optimizes security. *Intel Corporation*, 2008.

[78] CISA. Cisa releases best practices for mapping to mitre att&ck, June 2021. URL: `https://www.cisa.gov/uscert/ncas/current-activity/2021/06/02/cisa-releases-best-practices-mapping-mitre-attckr`.

[79] Hevner Alan, Chatterjee Samir. *Design research in information systems.* Springer, 2010.

[80] Ahmed Bakhtiyar, Dannhouser Thomas, Philip Nada. A lean design thinking methodology (ldtm) for machine learning and modern data projects. *10th Computer Science and Electronic Engineering (CEEC)*, pages 11–14, 2018.

[81] Kratzke Nane, Quint Peter-Christian. Understanding cloud-native applications after 10 years of cloud computing - a systematic mapping study. *Journal of Systems and Software*, pages 1–16, 2017.

[82] Mukhopadhyay Indraneel. *ICT Analysis and Applications. Lecture Notes in Networks and Systems.* Springer, 2022.

[83] Karagiannis Stylianos, Magkos Emmanouil, Ntantogian Cristoforos, Ribeiro Luis L. *Sandboxing the Cyberspace for Cybersecurity Education and Learning.* Springer, 2020.

[84] Fitch Scott C. Muckin Michael. A threat-driven approach to cyber. 2014.

[85] Lee Tae-Jin Hwang Chan-Woong, Bae Sung-Ho. Mitre att&ck and anomaly detection based abnormal attack detection technology research. *Convergence Security Journal*, pages 13–23, 2021.

[86] Olivier David, Chiffelle Jaquet, Loi Michele. *Ethical and Unethical Hacking.* Springer, 2020.

[87] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. Understanding the security implications of kubernetes networking. *IEEE Security Privacy*, 19(5):46–56, 2021.