MMLAB Main Seminar

SoK: A Comprehensive Analysis and Evaluation of Docker Container Attack and Defense Mechanisms

Md Sadun haq, Thien Duc Nguyen, Ali Saman Tosun, Franziska Vollmer, Turgay Korkmaz, and Ahmad-Reza Sadeghi

2024 IEEE Symposium on Security and Privacy 2024 (S&P 24')

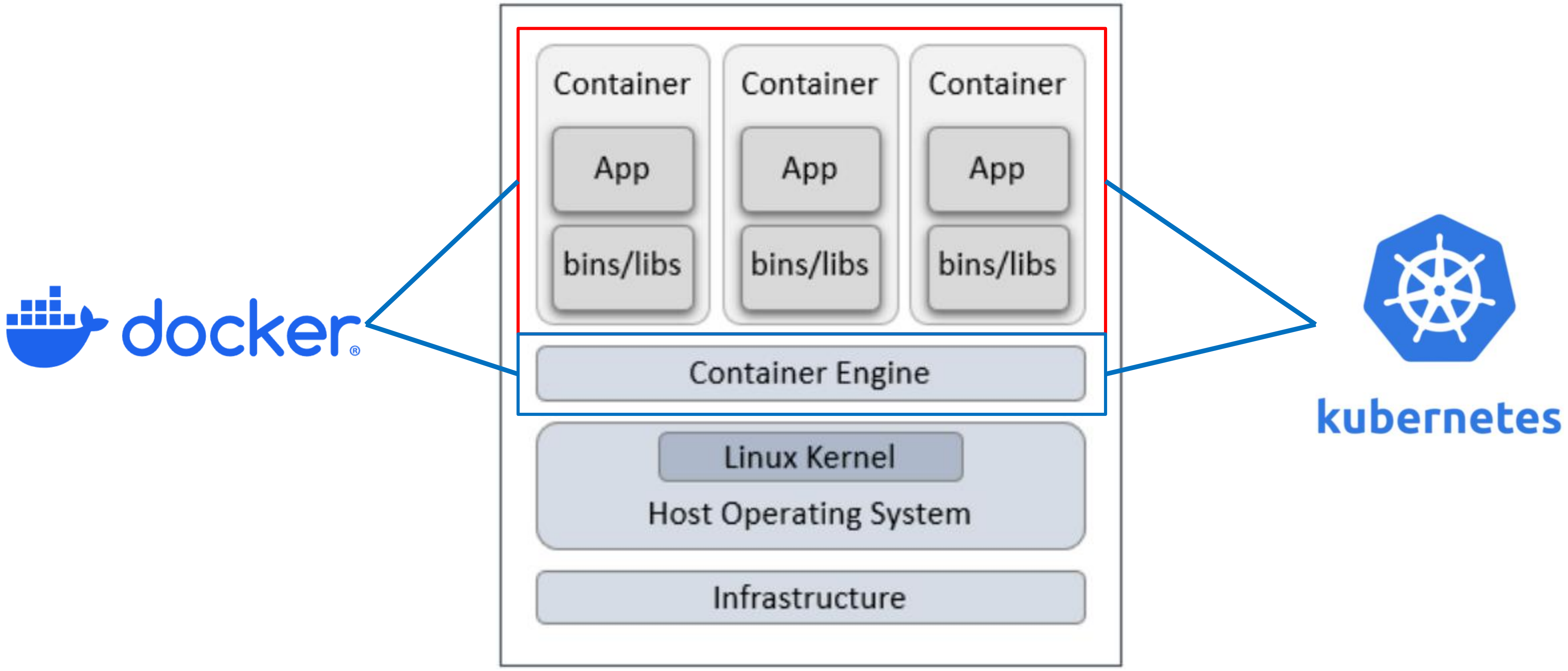
HyeongUk Ko (huko@mmlab.snu.ac.kr)

2025.04.09

Index

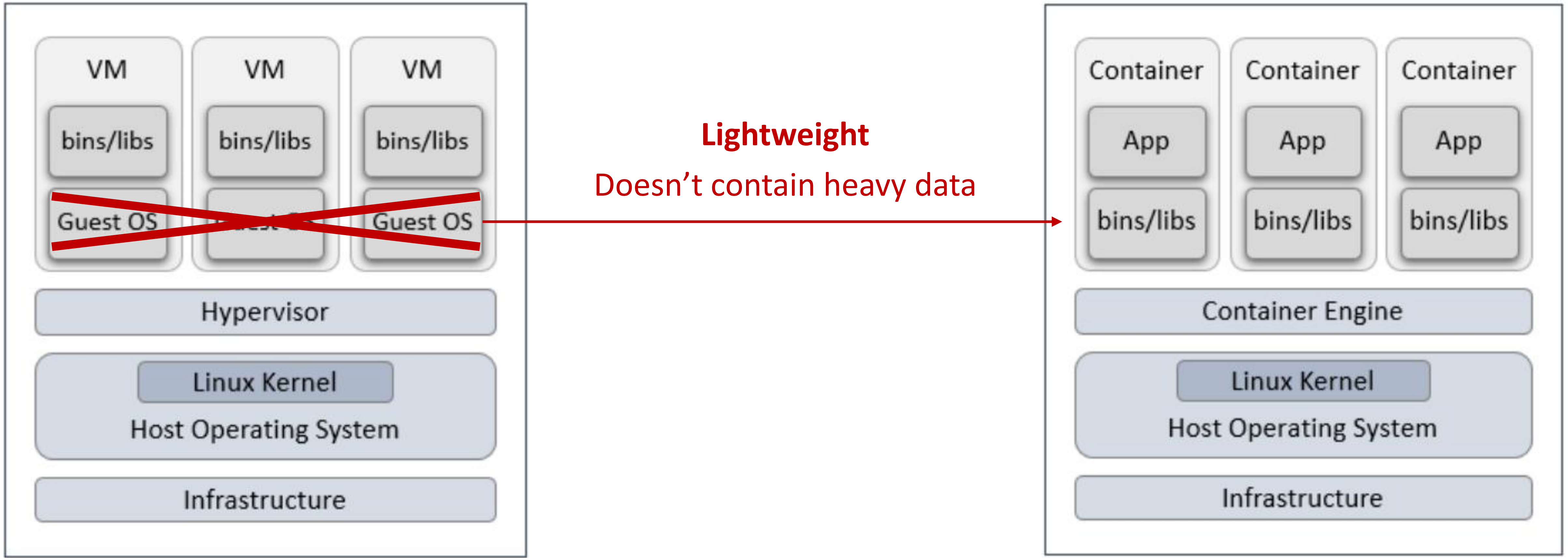
What is Container?

- A **lightweight executable bundle** of software encompassing code, system tools, and other dependencies needed to run an application



What is Container?

- Container can provide **isolation** between various applications, offering **effectivity and portability**



Two Sides of Container

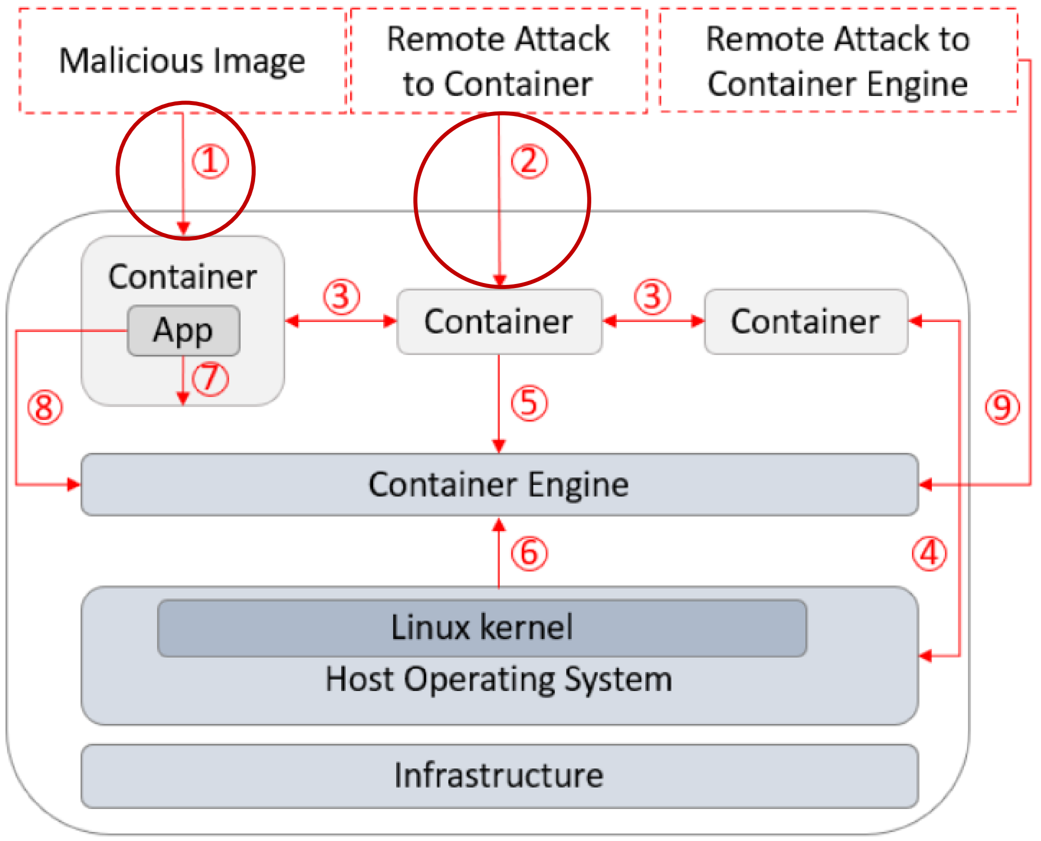
- Container-based technology is increasingly used in developing and deploying applications across various platforms
  - **91% of organizations** are using containers in production, according to Cloud Native Computing Foundation (CNCF) 2024 annual survey [1]

- However, containers possess security risks due to their **weak isolation** from the host OS compared to VMs
  - Any successful attack within a container might compromise the **entire** host system

*Therefore, there is a need for **understanding and systemizing** diverse attack vectors and defenses mechanisms

Index

9 Attack Scenarios



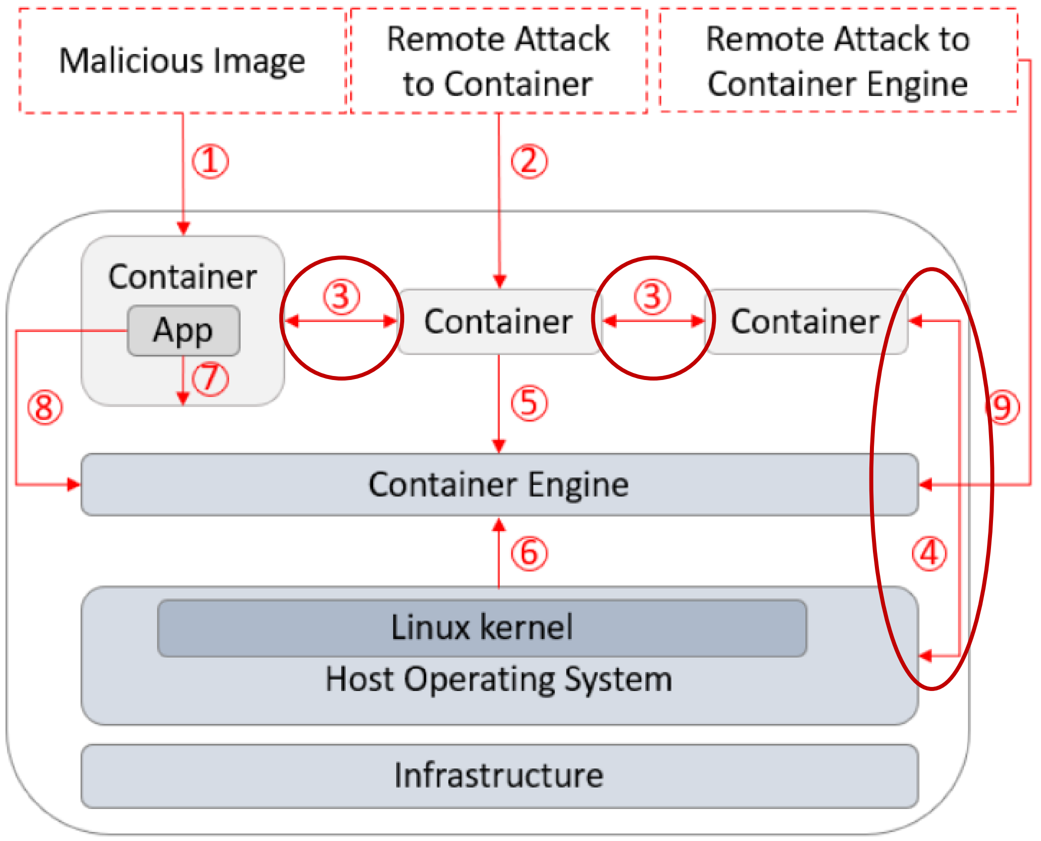
① Malicious image

- User downloads malicious images from a public container repository (e.g., DockerHub)
- It can lead to various exploits – reverse shell, exposing credentials, etc.

② Remote to container

- The vulnerable component or application is located in the container by preexisting issues

9 Attack Scenarios



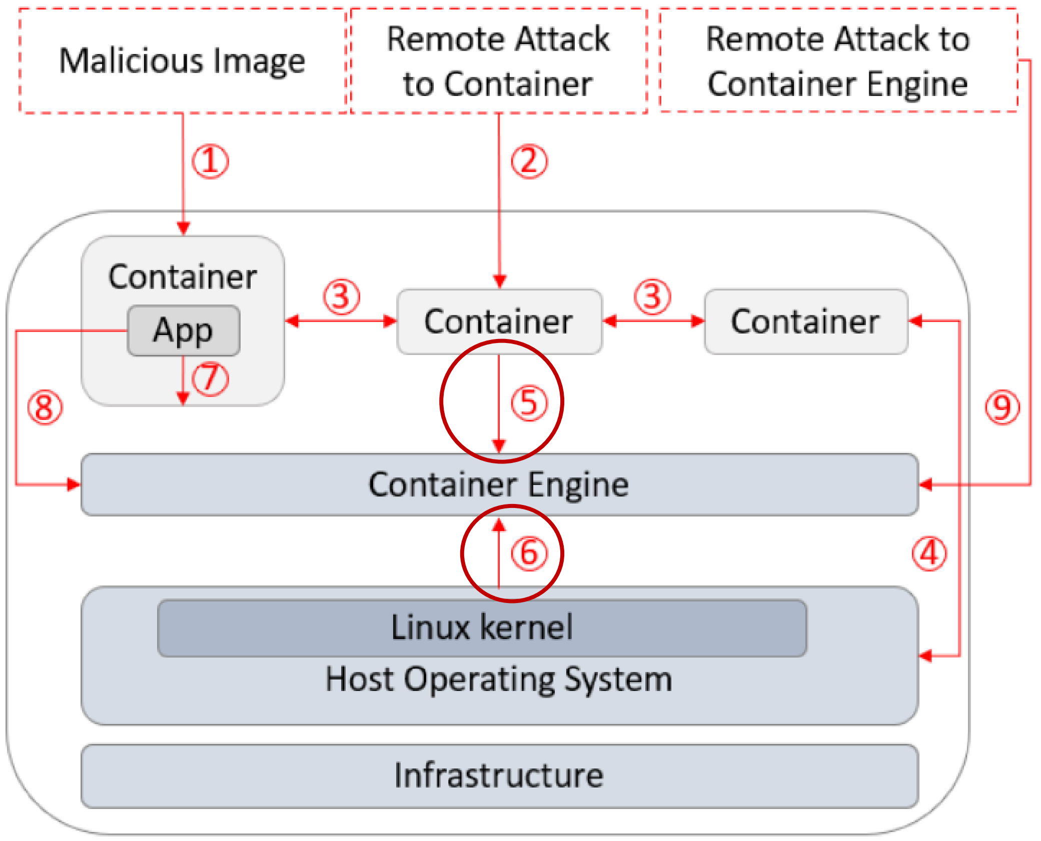
③ Container to container

- A container is attacked by the compromised container
- Attacker can get vital information regarding other containers and cause various exploits – DoS, authentication bypass, etc.

④ Between container and host

- Container to host
- Host to container

9 Attack Scenarios



⑤ Container to container engine

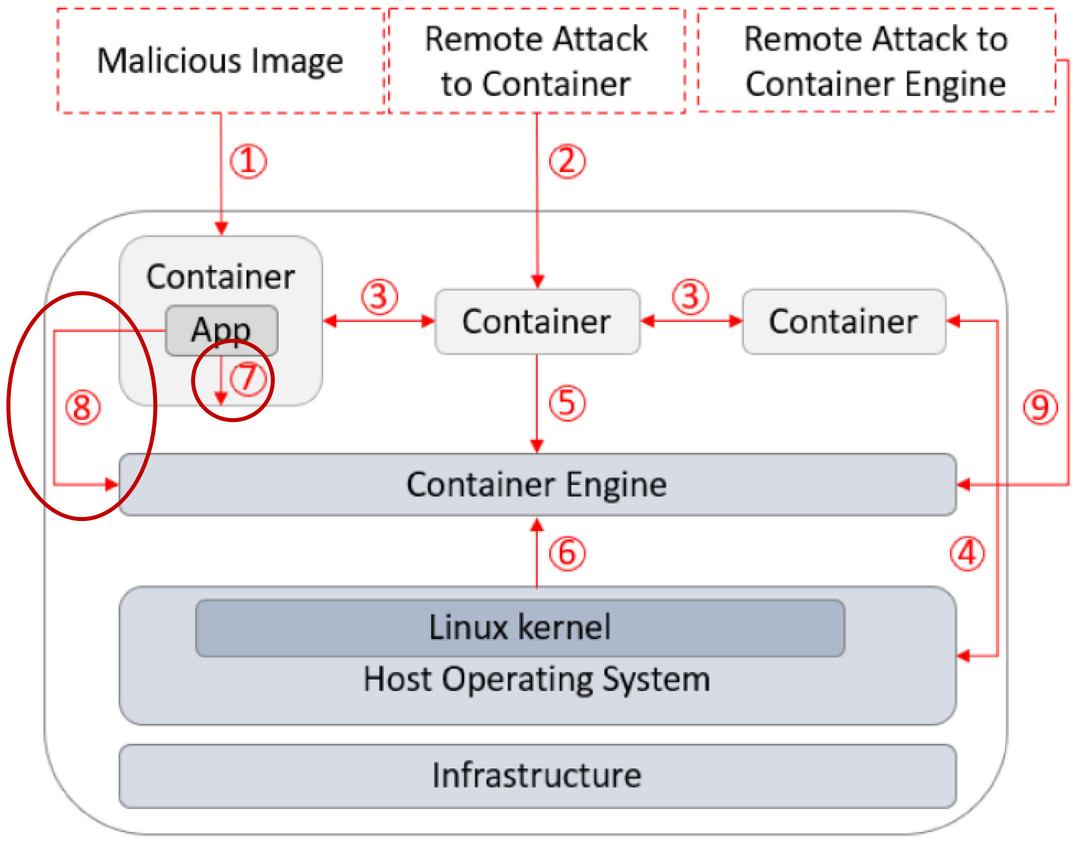
- A container engine is attacked by the compromised container
- Since the container engine usually runs with root privileges, compromising it can grant superuser access

⑥ Host to container engine

- Given malicious host, attacker exploits the container engine

*Straightforward

9 Attack Scenarios



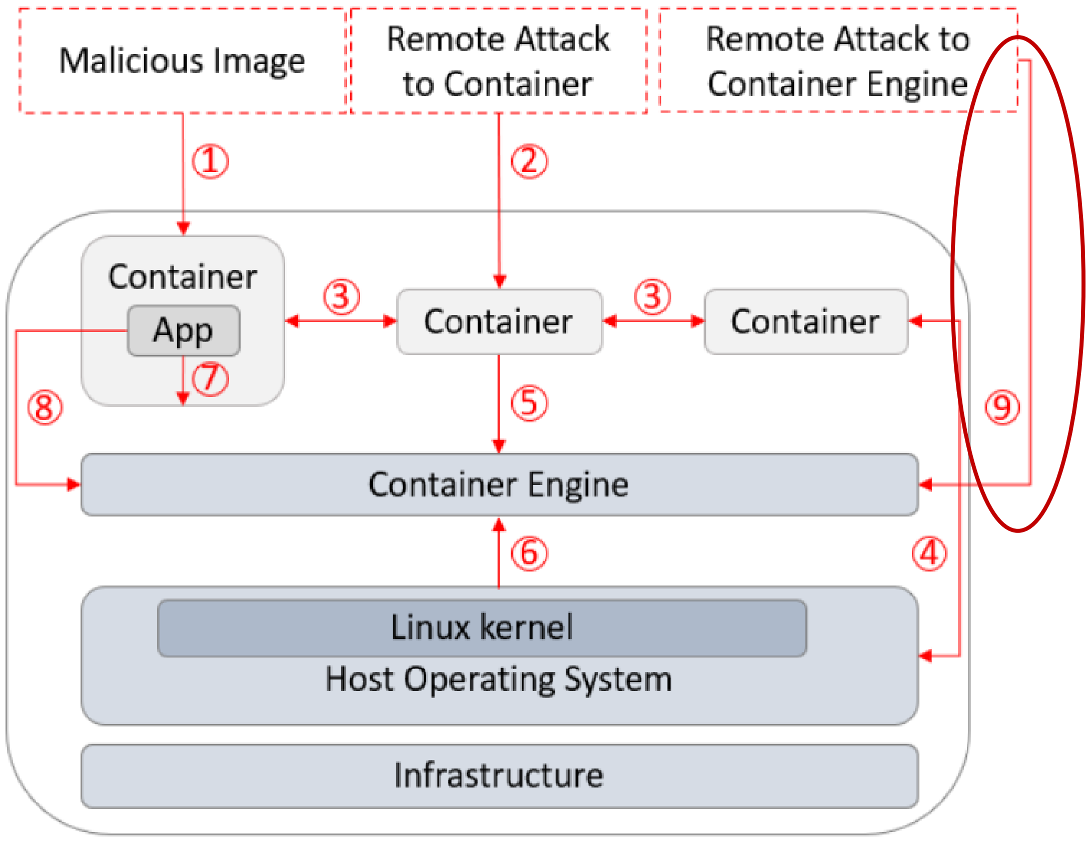
⑦ Application to container

- A container is attacked from an application
- Various attacks can be performed

⑧ Application to container engine

- A container engine is attacked from an application
- It can impact associated containers and orchestrators, leading to broader service disruptions

9 Attack Scenarios



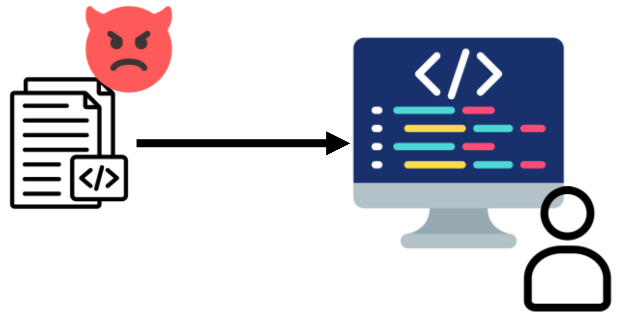
⑨ Remote to container engine

- The vulnerable component is in the container engine
- The adversary scans for exposed ports, and if found, he tries to gain privilege by exploiting the preexisting vulnerability

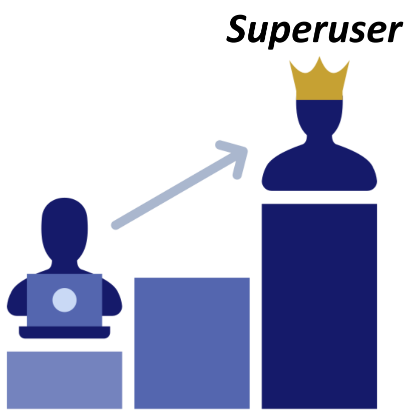
*There are 9 combinable attack paths associated with container architecture

Index

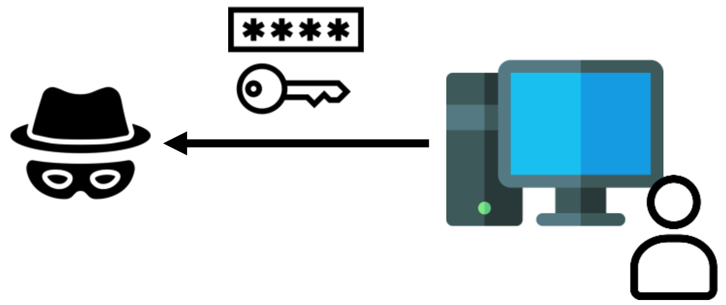
5 Attack Types



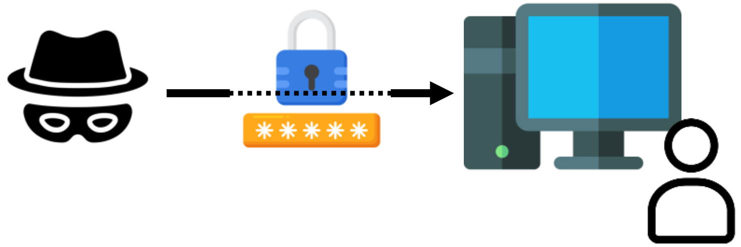
Execute
Arbitrary Code



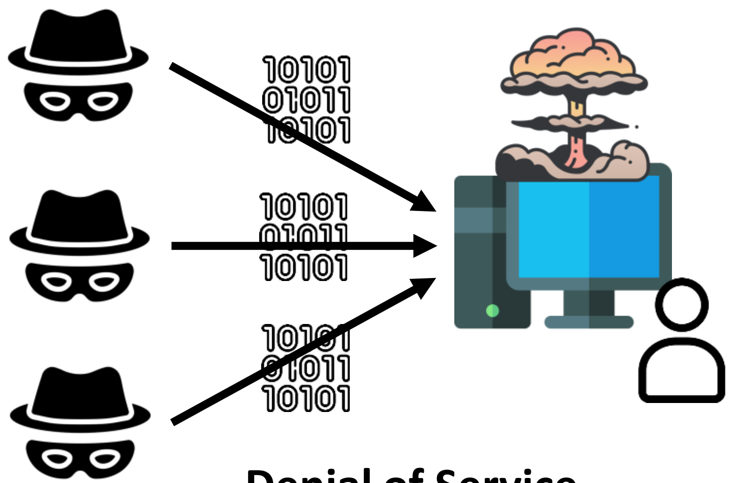
Gain Privilege



Disclose Credential
Information

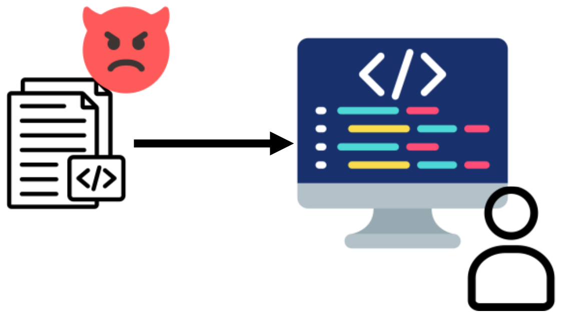


Authentication
Bypass



Denial of Service

Execute Arbitrary Code



Execute Arbitrary Code

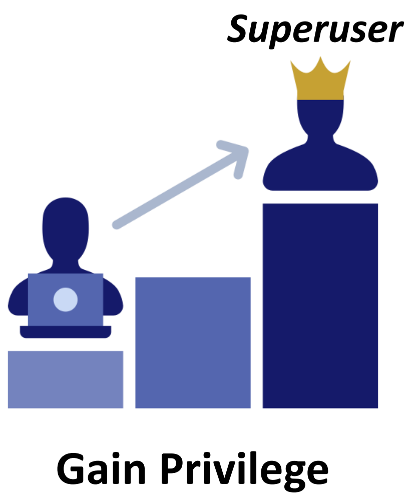
- Definition
  - Attacks which aim to send **arbitrary commands** to the victim to gain unauthorized access or control of the system

- **Return a shell**
  1. An adversary sends crafted commands to databases or web servers
  2. It leads to opening a shell and creating a pathway for the adversary

  *If returning a shell is unable, they may cause other issues – **return-oriented programming**

- Example usage
  - Make a target host download a malicious image or attack a target container remotely

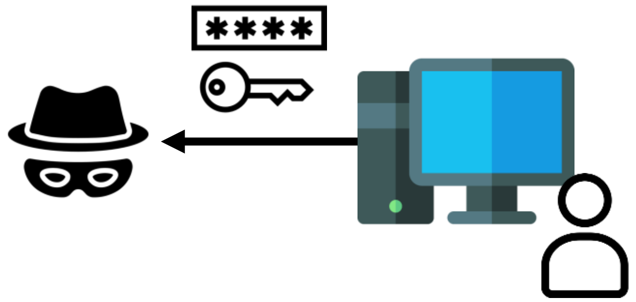
Gain Privilege



- Definition
  - Attacks which aim to gain **superuser privileges** by exploiting
- **Escalation of privilege**
  - Modifications of memory and files can be used to escalate privilege by buffer overflow or modifying the superuser's password, respectively

  *Privilege escalation that involves gaining root access by exploiting vulnerabilities in the kernel is called **kernel escalation**
- Example usage
  - Escalate from container to container engine

Disclose Credential Information



Disclose Credential Information

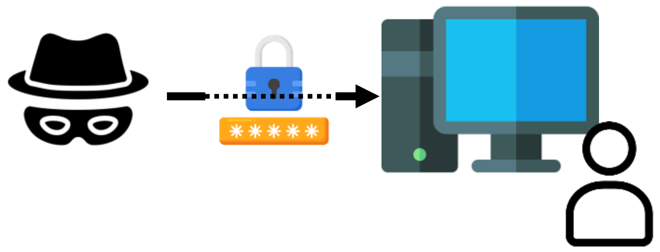
- Definition
  - Attacks which aim to involve unauthorized access and exposure of **credentials** (e.g., usernames and passwords)
  - Additionally, they also aim to gain the **underlying information** of the systems, e.g., file names or directory structures

- **Gain login credentials**
  - Having the contents of '/etc/shadow' or '/etc/passwd' files can give an adversary unlimited access

  *the adversary can also utilize **side-channel attacks** to acquire sensitive information by analyzing CPU or other components

- Example usage
  - Exploit container or host and expose credentials

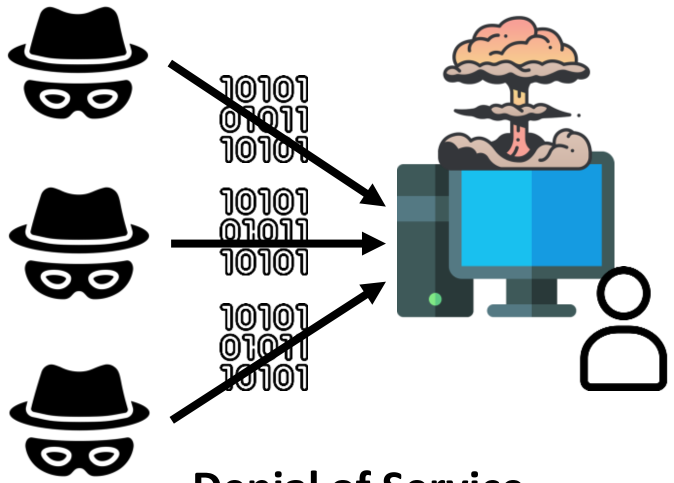
Authentication Bypass



Authentication Bypass

- Definition
  - Attacks which aim to gain **unauthorized access** without providing correct credentials exploiting authentication systems
- **Password bypass**
  - Adversary can bypass the password authentication method and gain access to a person's online account or database
  - SQL injection or integer overflow
- Example usage
  - Exploit the vulnerable application in container or attack flawed design in container/host authentication mechanisms

Denial of Service



Denial of Service

- Definition
  - Attacks which aim to **disrupt** normal system functionalities
- **Consume excessive resources**
  - It focuses on three resources: CPU, Memory, and Network
- Example usage
  - Excessive resource consumption by one container can lead to starvation in others

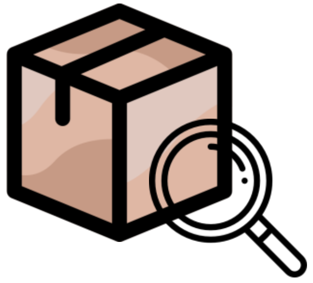
A Taxonomy of Attack Types and Techniques

Attack Types	Attack Techniques	Attack Types	Attack Techniques
Execute Arbitrary Code	Return a shell	**Authentication Bypass**	Password bypass
	Return-oriented programming		Reduced security attributes
	Remote code execution		Integer overflow
Gain Privilege	Escalation of privilege	**Denial of Service**	Crash the application
	Kernel escalation		Consume excessive resources
Disclose Credential Information	Gain login credentials		Brute-force
	File system access		Spoofing
	File name access		DoS overflow
	Channel attacks		Redirect traffic

*More detailed table [2]

Index

4 Defense Mechanism Groups



Static Scanning

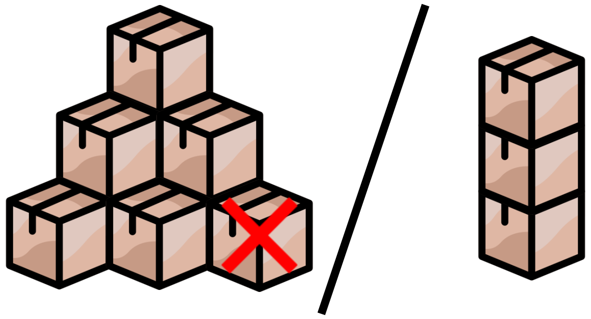
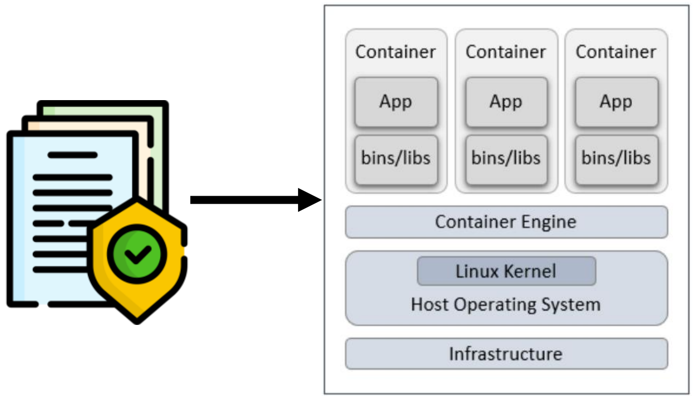
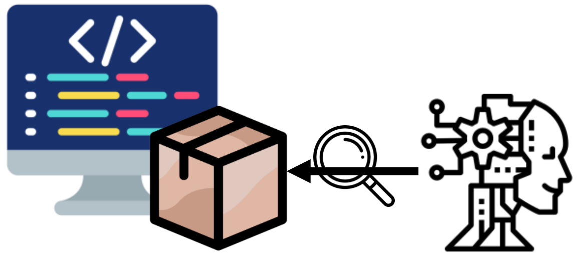


Image Hardening

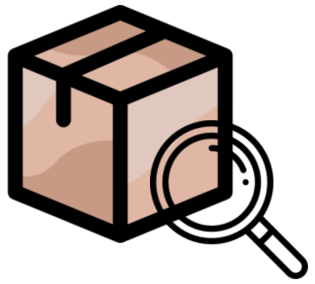


Security Policies &
Practices



Dynamic Anomaly
Detection

Static Scanning



Static Scanning

- Definition
  - The mechanisms which use **scanning tools** to seek known vulnerabilities in containers
- Property
  - Can detect vulnerabilities in non-updated/poisoned/malicious images, and even insecure configurations
- Limitations
  - Cannot detect run-time attacks
  - Can suffer from inconsistencies and false positives
  - Cannot detect zero-day attacks

Image Hardening

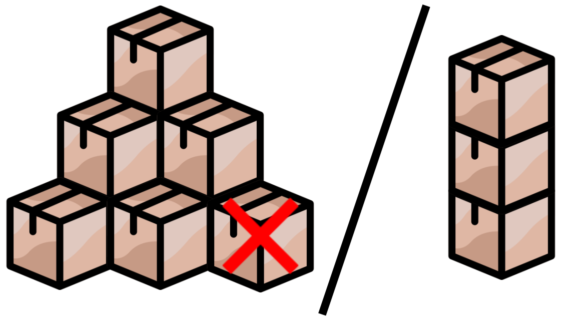
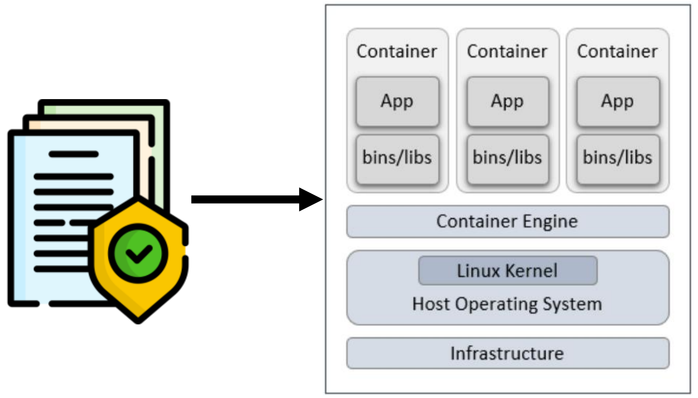


Image Hardening

- Definition
  - The mechanisms which **remove unnecessary packages** to reduce the attack surface or build an application with a **minimal base image**
- Property
  - Can reduce the attack surface in container
- Limitation
  - Removing unnecessary packages is not straightforward and can hamper the execution of applications

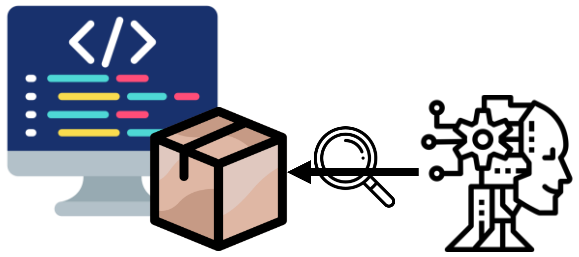
Security Policies & Practices



Security Policies & Practices

- Definition
  - The mechanisms which provide **certain policies and practices** to prevent breaches through containers
- Property
  - Can increase security by providing rigid access to system calls, capabilities, and privileges
- Limitation
  - Hard to implement in practice and may prevent a container from functioning correctly
  - Incur an increase in overhead

Dynamic Anomaly Detection

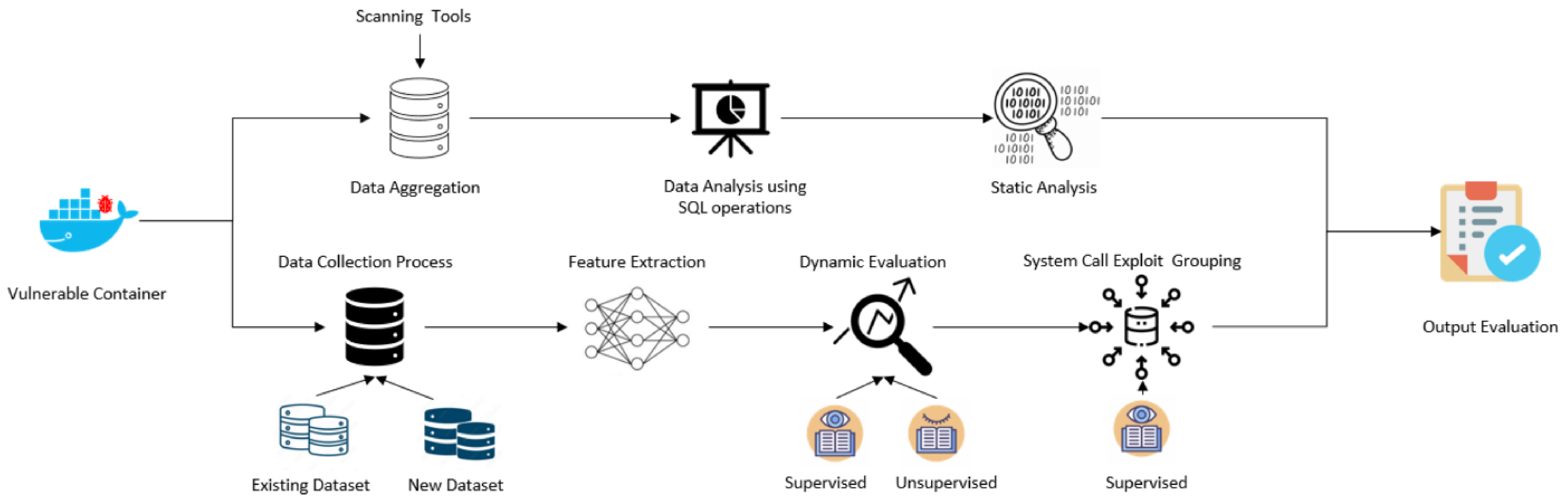


Dynamic Anomaly Detection

- Definition
  - The mechanisms which use **machine learning algorithms** to detect anomalies in the runtime
- Property
  - A model learns very many data and, consequentially, it can determine whether the running container has anomaly or not
  - Some defenses use supervised approaches, some other defenses use unsupervised approaches
- Limitation
  - Building a robust detection model is difficult due to the scarcity of training data and the absence of a benchmark

Index

Evaluation Framework

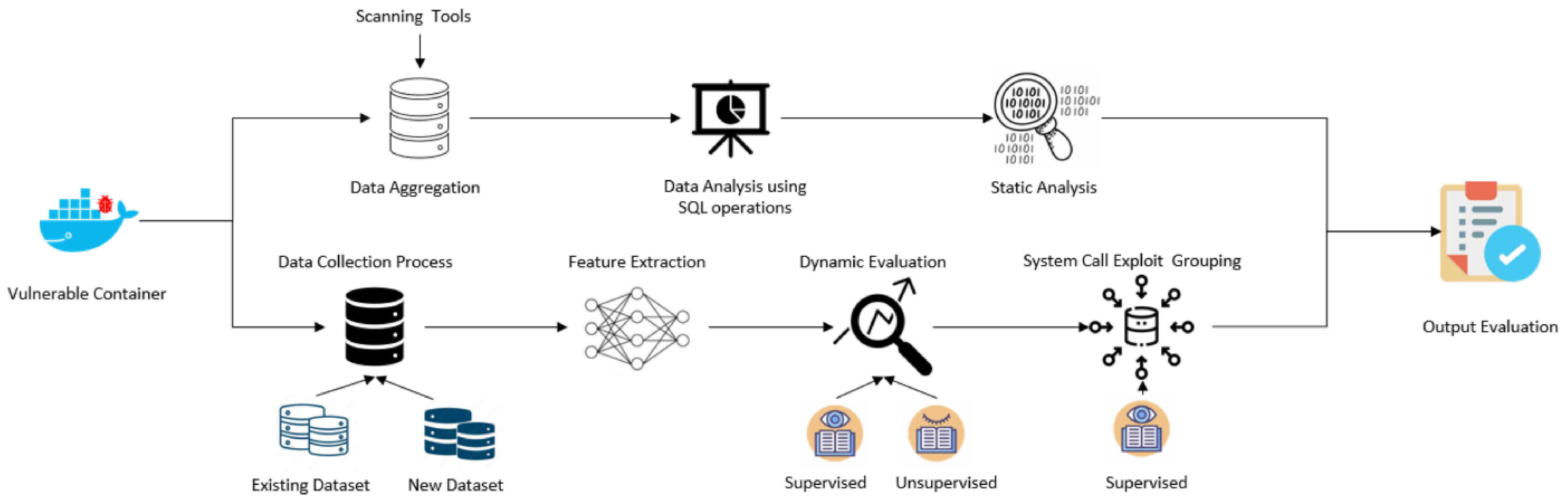


Evaluation Framework

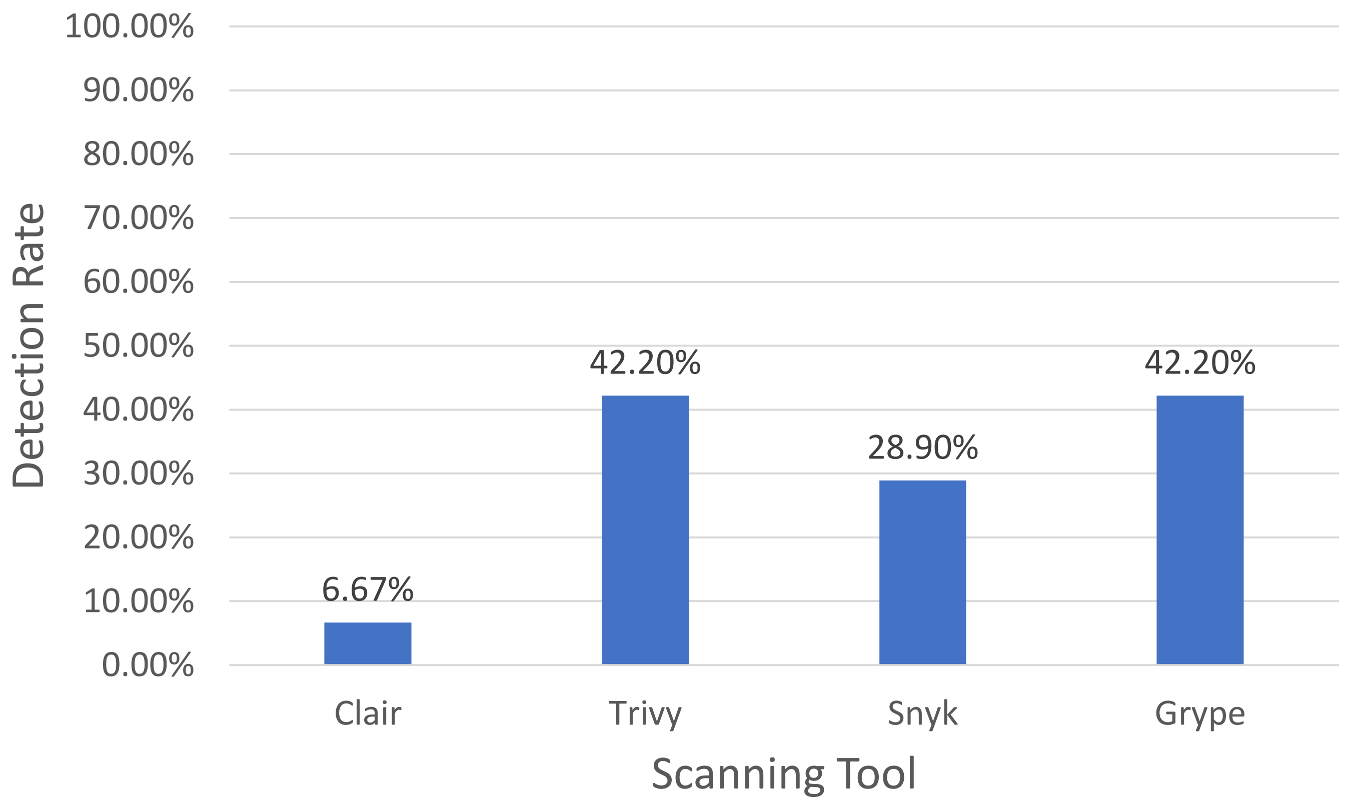
- Dataset
  - Existing dataset: includes the system calls for 41 exploits which are from existing dataset
  - New dataset: includes the system calls for 10 exploits which authors had generated
  - The exploits are carried out at the fourth minute – the data collected in the first three minutes is labeled as benign, while the rest is labeled as malicious

- Training and testing scenarios
  - Reorganizing dataset according to when the subset of the dataset was published
    - ✓ d1 (oldest, 34 exploits), d2 (middle, 9 exploits), d3 (newest, 8 exploits)
  - 3 different scenarios
    - ✓ S1 (train dataset == test dataset), S2 (train dataset != test dataset), S3 (train dataset == test dataset, each dataset is entire dataset)

Evaluation Framework



Evaluation: Static Analysis



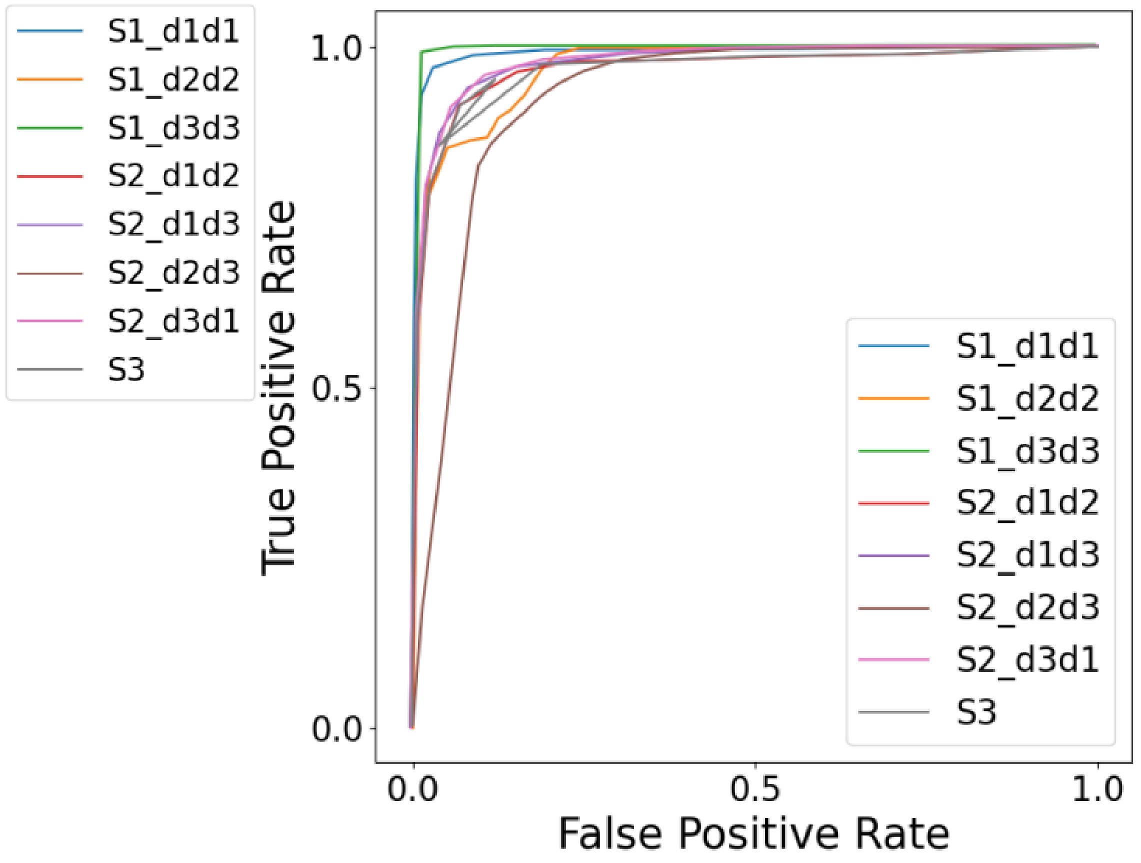
None of the tools provide a detection rate above 50%, even when combining multiple scanning tools (48.9%)

The **inconsistencies** in container scanning results with existing works emphasize the **necessity for enhanced detection tools**

Evaluation: Dynamic Analysis – Anomaly Detection

- Supervised algorithms – (Random Forest, RF)

For S2, the TPR falls under 60% (testing with unseen data)



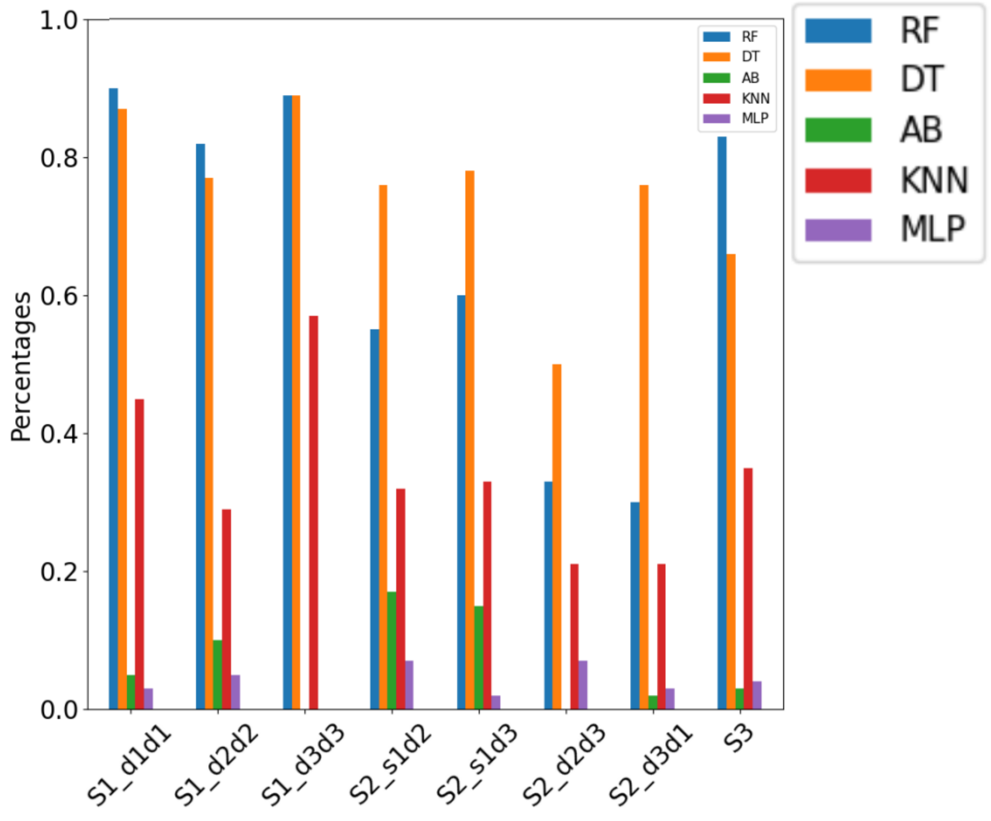


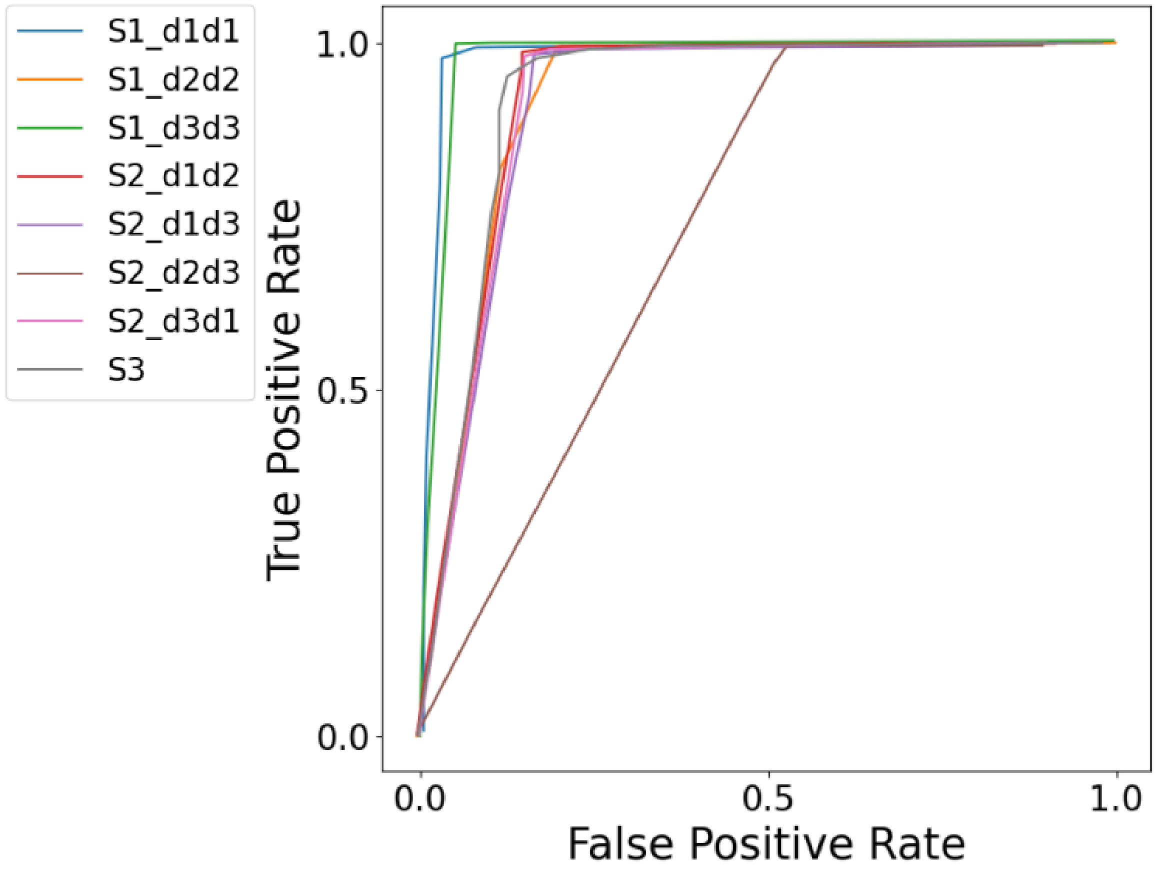
Figure 6: True Positive Rates For Anomaly Class

Evaluation: Dynamic Analysis – Anomaly Detection

- Supervised algorithms – (Decision Tree, DT)

DT performs well except for S2_d2d3

For S2_d2d3, the TPR and FPR is around 50%



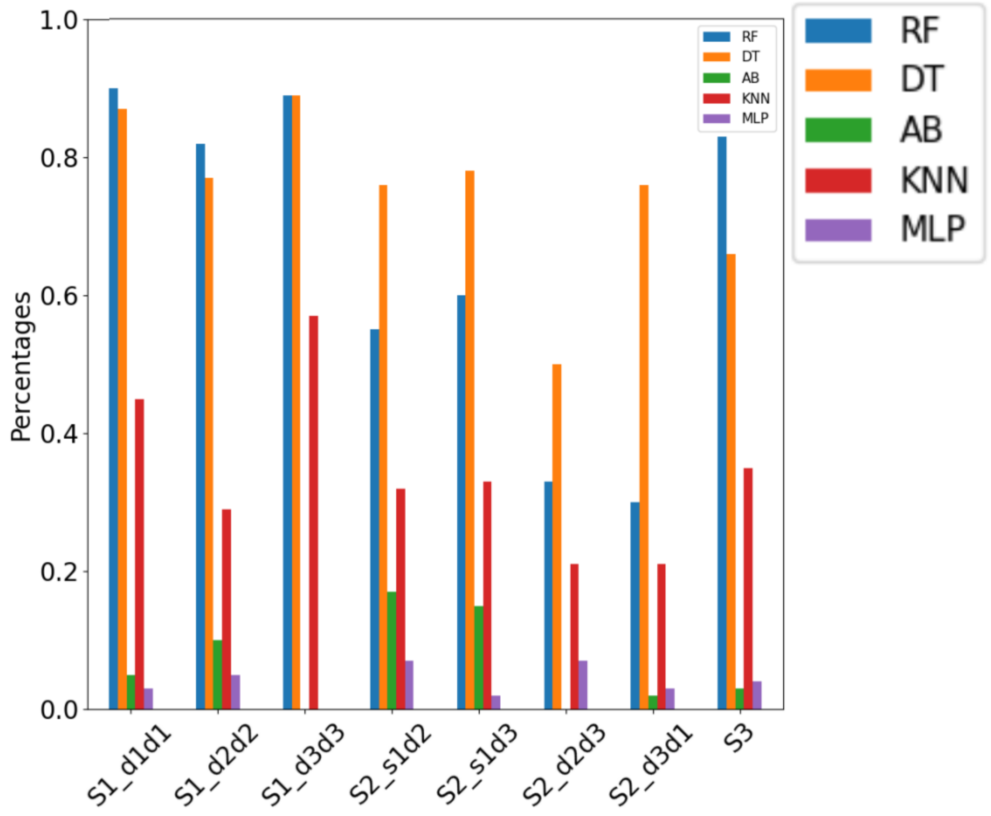
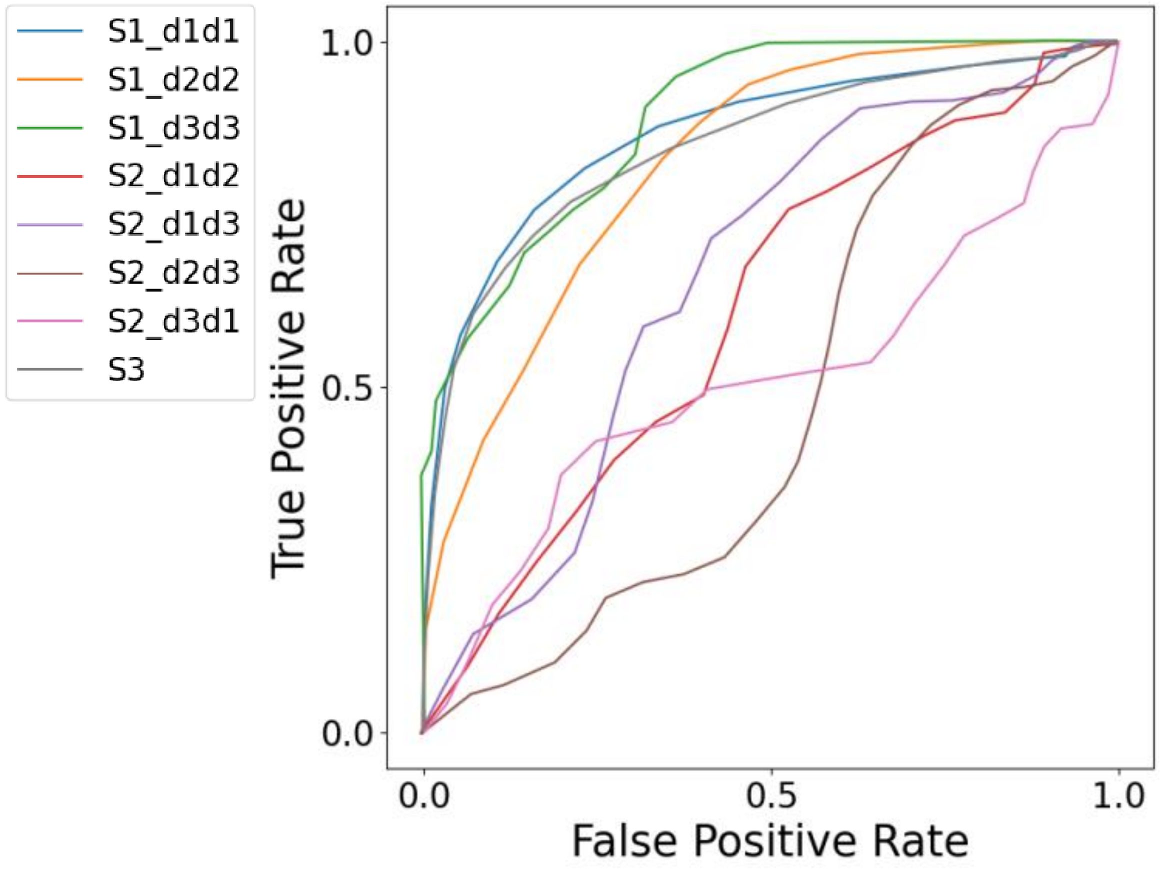


Figure 6: True Positive Rates For Anomaly Class

Evaluation: Dynamic Analysis – Anomaly Detection

- Supervised algorithms – (AdaBoost, AB)

AB performs very bad



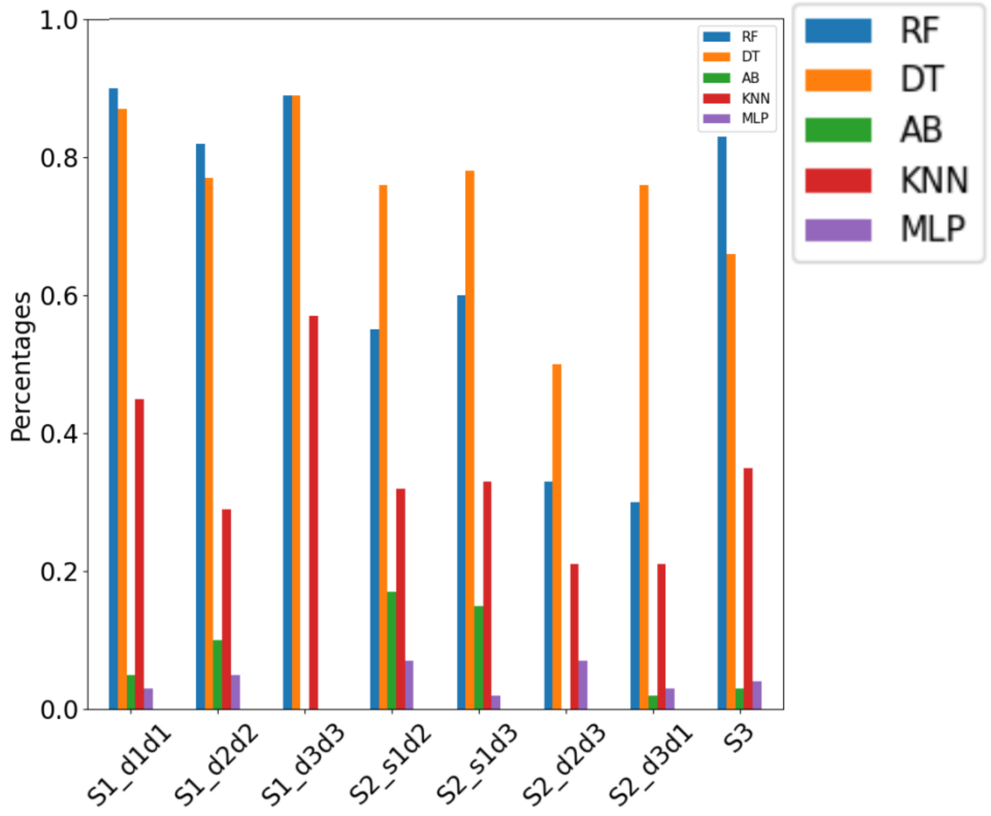
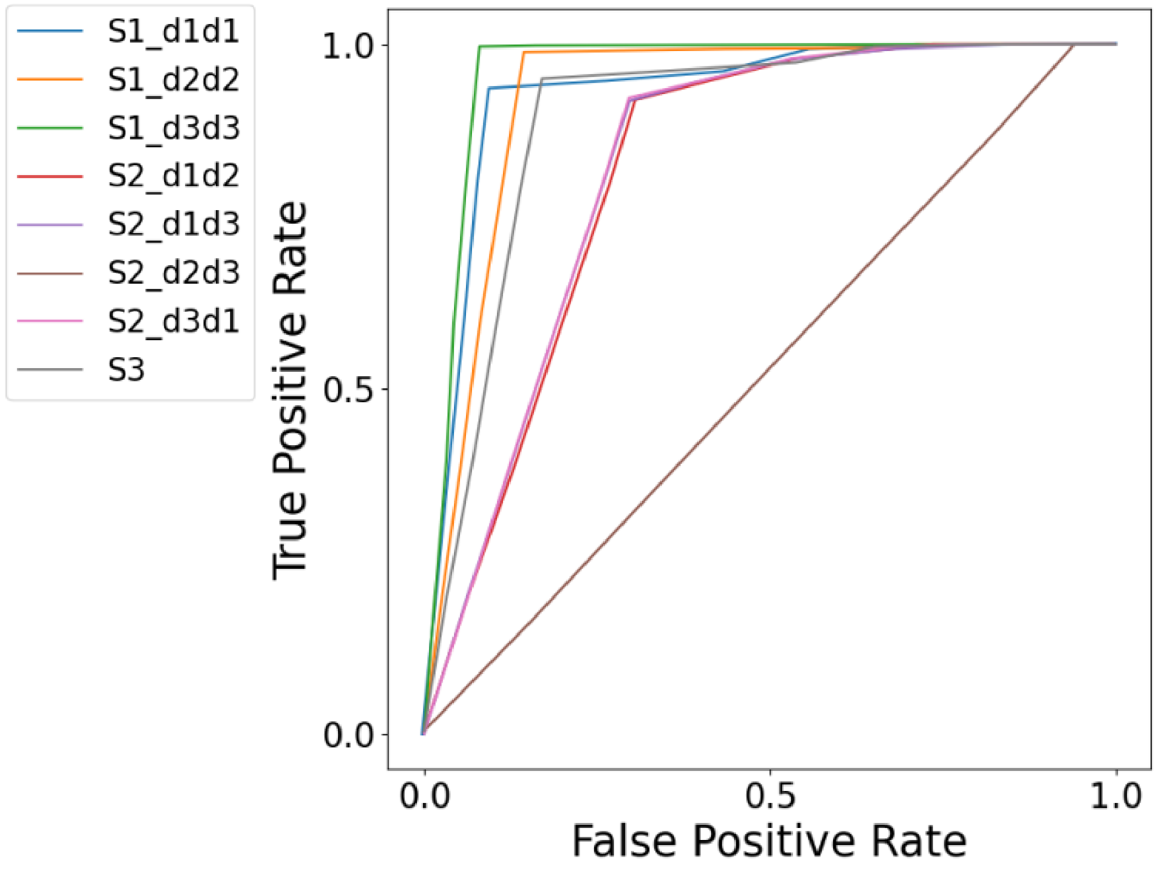


Figure 6: True Positive Rates For Anomaly Class

Evaluation: Dynamic Analysis – Anomaly Detection

- Supervised algorithms – (K-nearest-neighbor, KNN)

KNN does not perform well



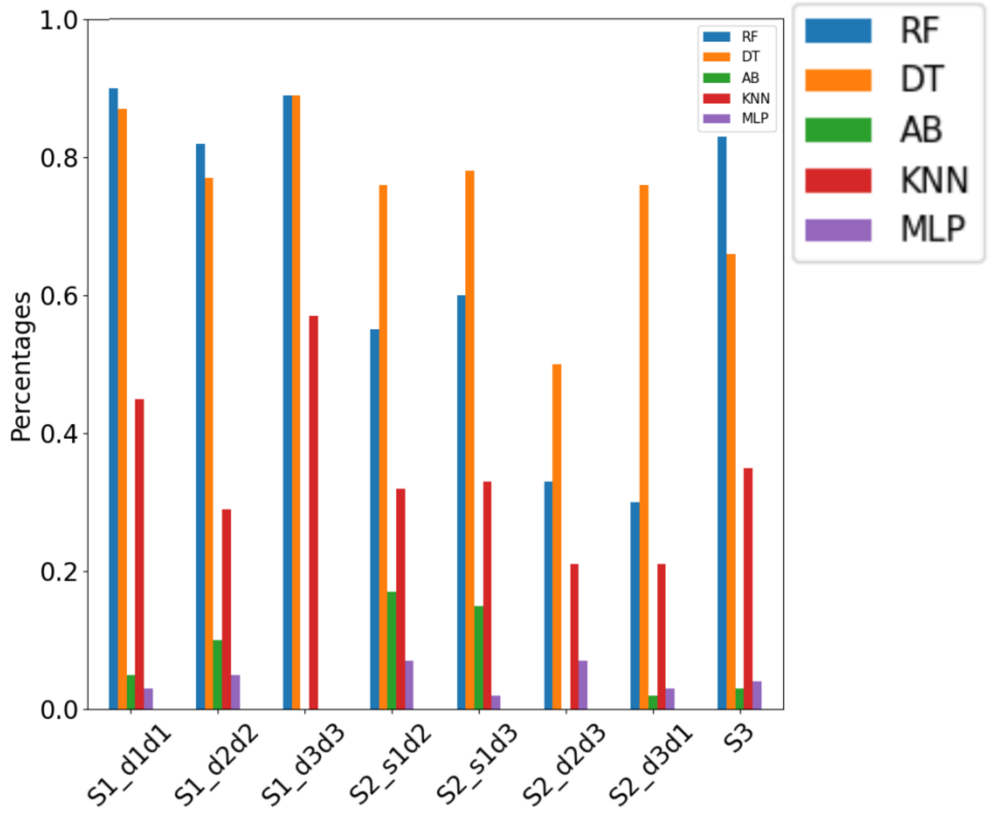
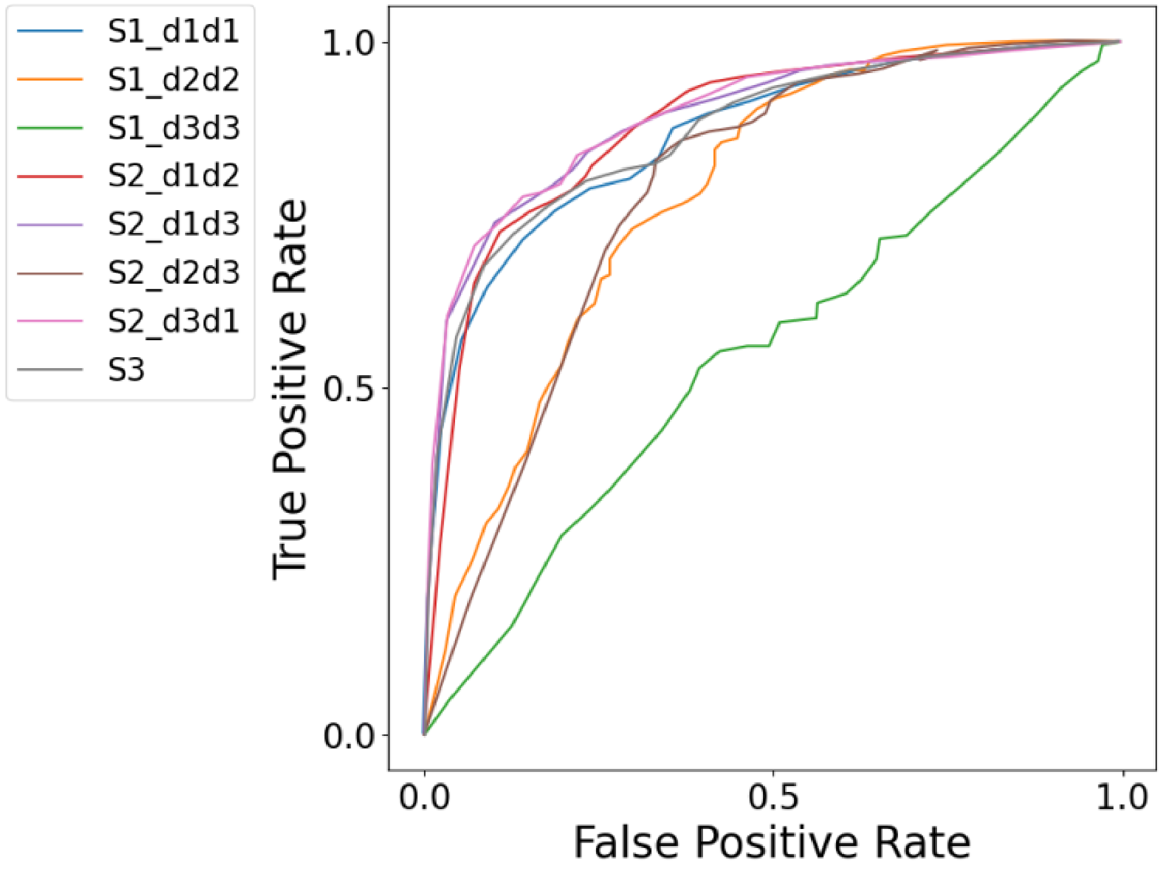


Figure 6: True Positive Rates For Anomaly Class

Evaluation: Dynamic Analysis – Anomaly Detection

- Supervised algorithms – (Multi-layer-perceptron, MLP)

MLP does not perform well



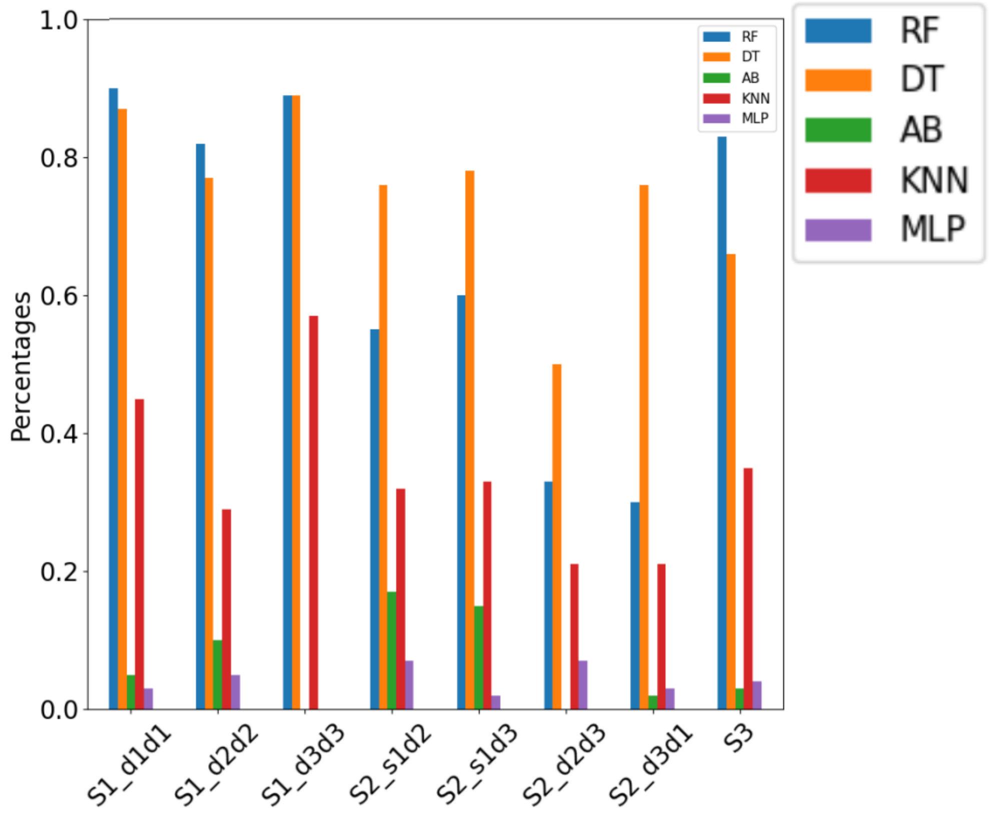


Figure 6: True Positive Rates For Anomaly Class

Evaluation: Dynamic Analysis – Anomaly Detection

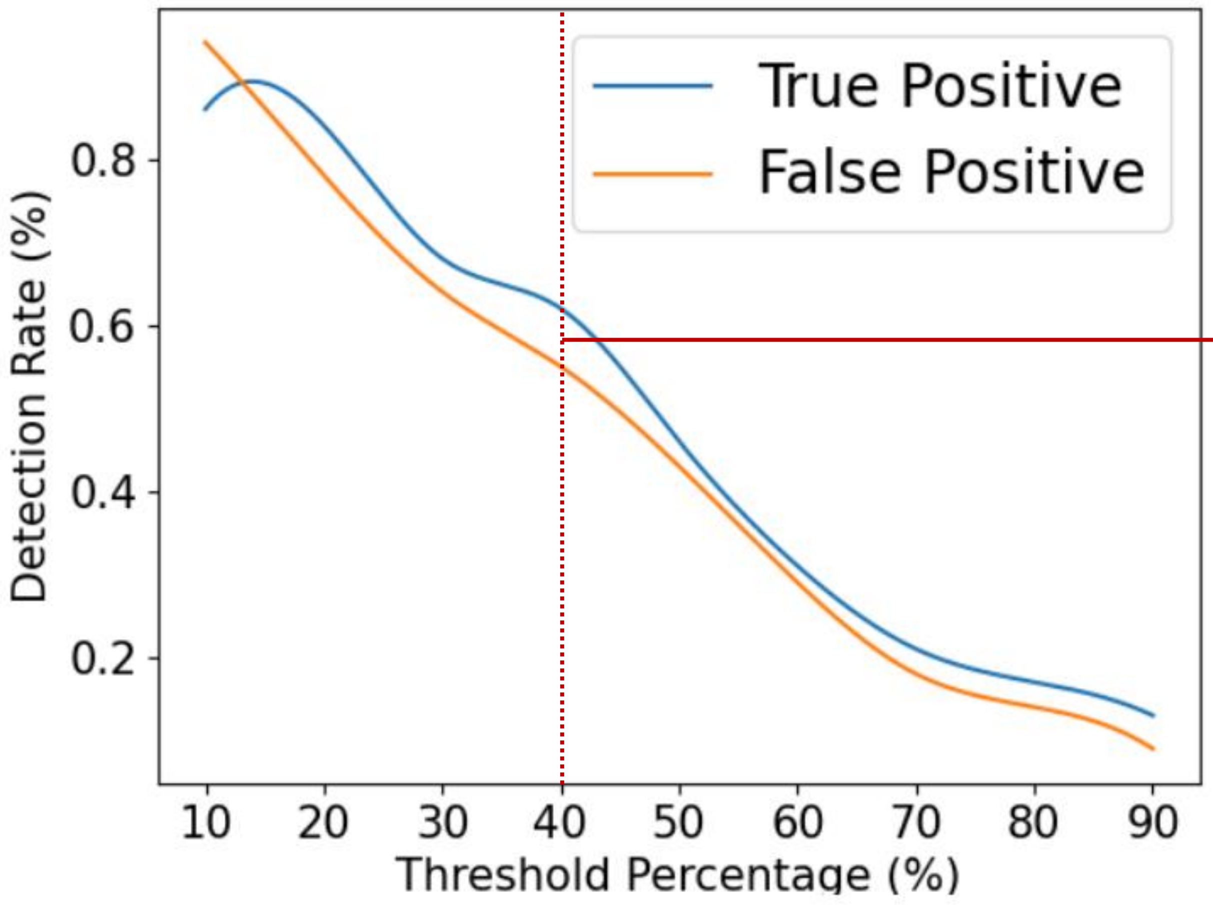
- Supervised algorithms – summary
  - Critical properties
    - ✓ RF and DT seem to classify the anomaly class better than other algorithms
    - ✓ However, neither RF nor DT provide consistent TPRs across the different scenarios, with results falling in S2

    *__Room for improvement__

  - Improvement examples
    - ✓ Performing dimensionality reduction before conducting the classification
    - ✓ Separating the scenarios based on the application and fine-tuning the algorithms

Evaluation: Dynamic Analysis – Anomaly Detection

- Unsupervised algorithms – (AutoEncoder)

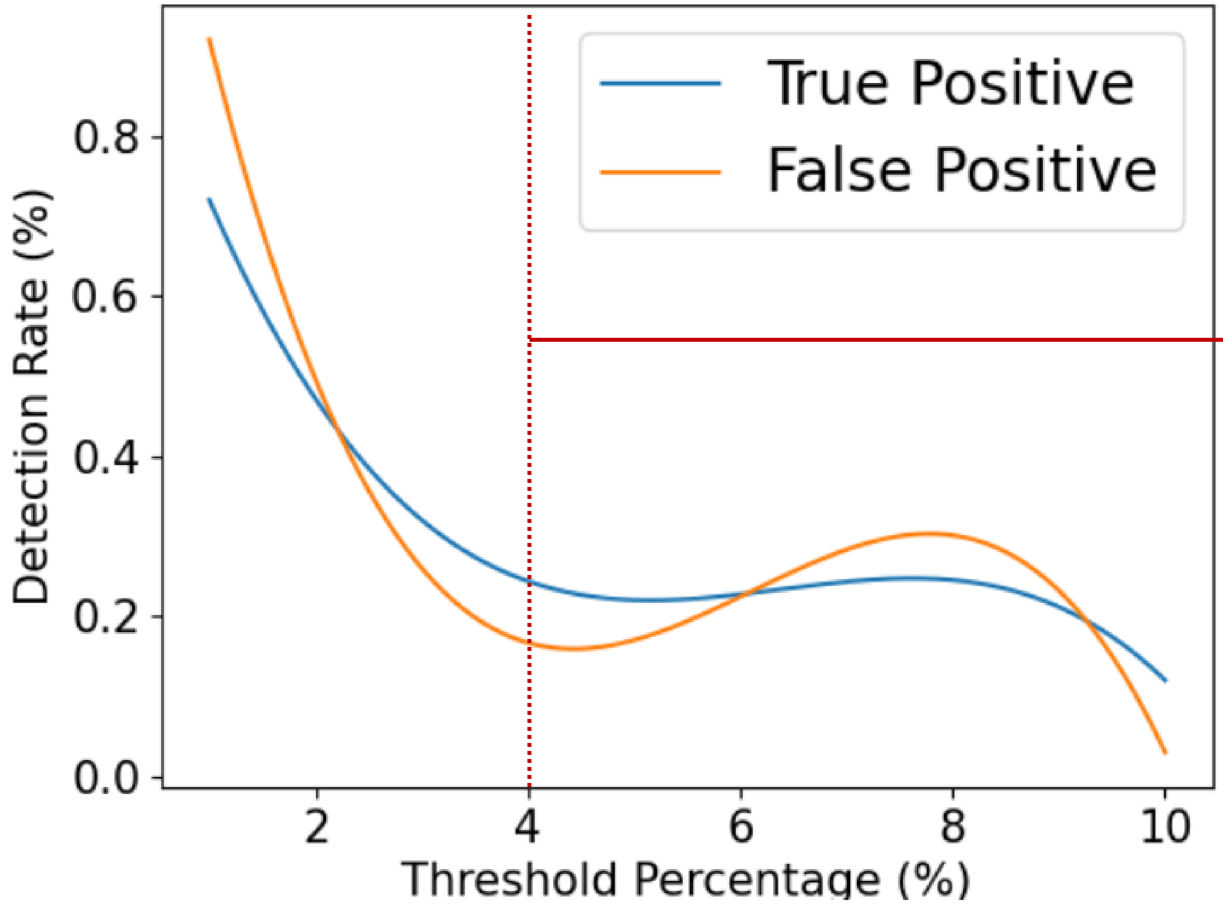


At the 40th percentile, the true positive rate is 62%, and the false positive rate is 54% (8%p difference)

AutoEncoder performs **very poorly** in detecting anomalies

Evaluation: Dynamic Analysis – Anomaly Detection

- Unsupervised algorithms – (K-means)



At the 4th percentile, the true positive rate is 25%, and the false positive rate is 18% (7%p difference)

K-means performs **very poorly** in detecting anomalies

Evaluation: Dynamic Analysis – Anomaly Detection

- Unsupervised algorithms – summary
  - Critical properties
    - ✓ The unsupervised algorithms perform **very poorly** in detecting anomalies, maybe due to their lack of labeled data
    - ✓ Even though they perform better than some supervised algorithms (higher TPR), they also give rise to FPR, which means that the algorithms can’t accurately separate the anomaly class from the benign class

Evaluation: Dynamic Analysis – Attack Type Detection

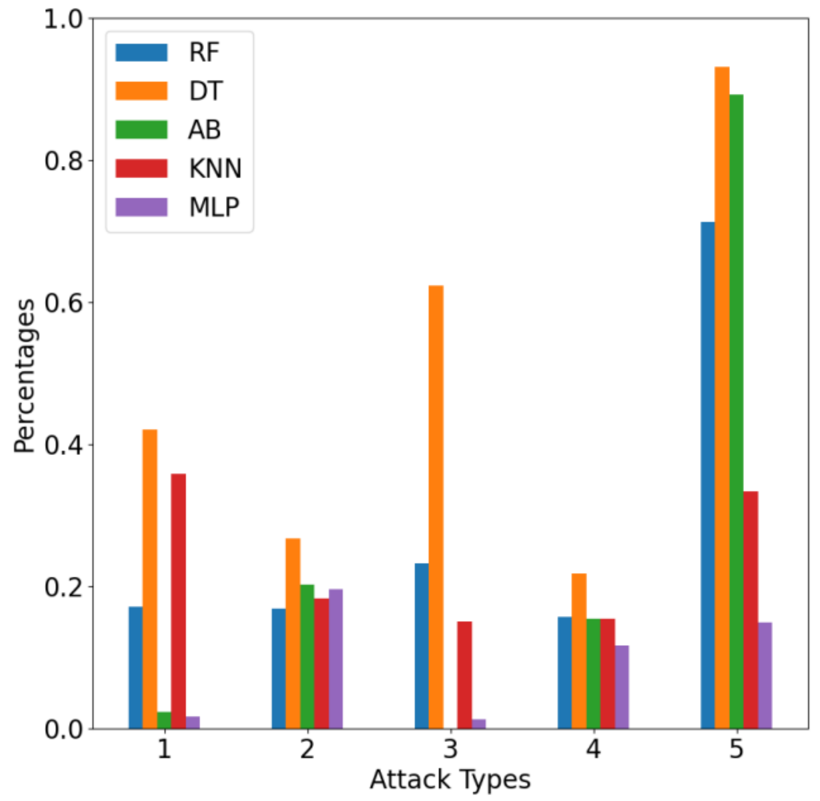


Figure 8: Performance of supervised detection models on different attack types. 1: Execute Arbitrary Code, 2: Gain Privilege, 3: Disclose Credential Information, 4: Authentication Bypass, 5: Denial Of Service

- 1: Execute arbitrary code
  - None of the models can accurately detect this type of attack
- 2: Gain privilege
  - None of the models can accurately detect this type of attack
- 3: Disclose credential information
  - DT detects around 60% for this type of attack

서울대학교 SEOUL NATIONAL UNIVERSITY

Network Convergence & Security Lab

Evaluation: Dynamic Analysis – Attack Type Detection

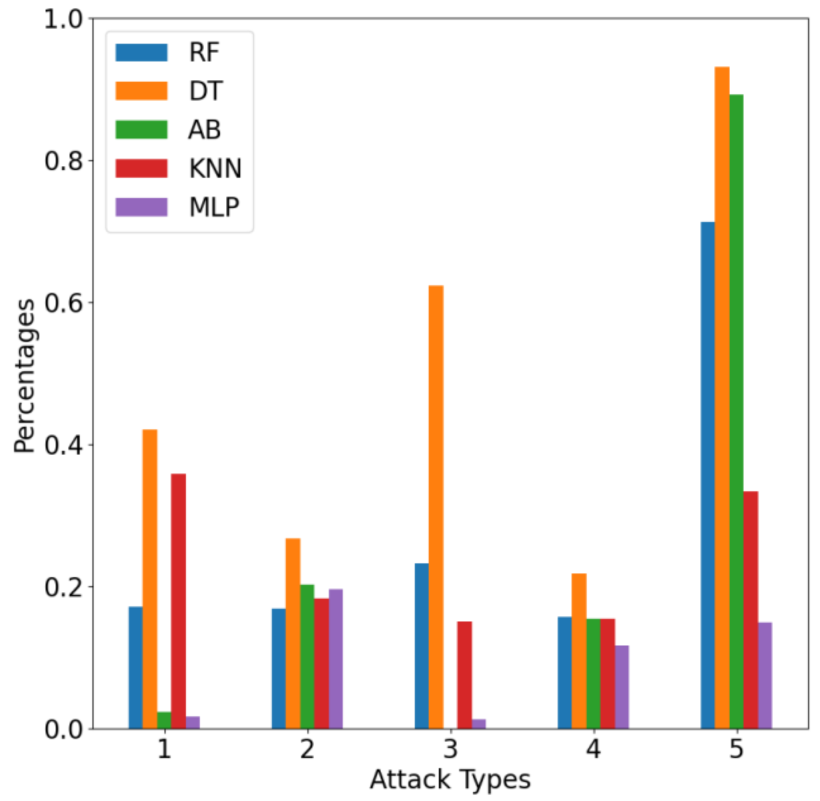


Figure 8: Performance of supervised detection models on different attack types. 1: Execute Arbitrary Code, 2: Gain Privilege, 3: Disclose Credential Information, 4: Authentication Bypass, 5: Denial Of Service

- 4: Authentication bypass
  - None of the models can accurately detect this type of attack
- 5: Denial of service
  - RF, DT, and AB can detect Denial Of Service attacks with high TPRs

*DT performs best for all attack types

Index

Discussion of Potential Improvements

- Static scanning
  - Problem: None of the scanning tools achieve a detection rate greater than 50%
  - Improvement 1: Don’t use scanning tools solely but **combine with other mechanisms**
  - Improvement 2: **Enhance** effectiveness and reduce the delay of **scanning tools** by utilizing collaborative learning for real-time vulnerability updates

- Dynamic Anomaly Detection
  - Problem: All the anomaly detection algorithms have poor performance
  - Improvement 1: **Dimensionality reduction** might provide better results for both supervised and unsupervised algorithms, as this might **reduce the chance of overfitting**
  - Improvement 2: **Improve existing anomaly detection** using dynamic approaches by training and tuning different models with diverse and up-to-date data

Conclusion

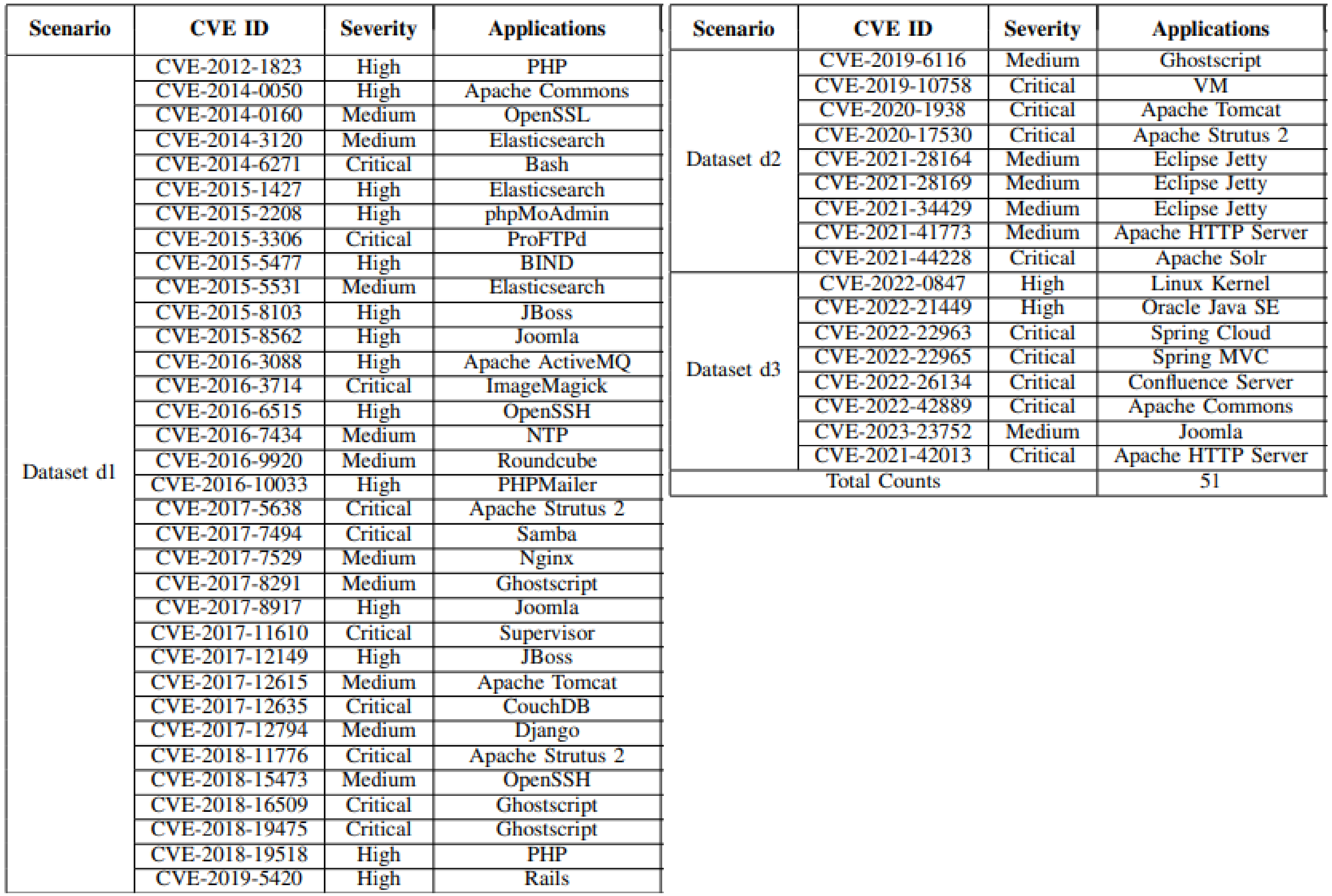
- This paper presents a systematic and comprehensive study of existing attacks and defense mechanisms for containers
  - 9 attack scenarios, 5 attack types, 4 defense mechanism groups
- We evaluate two defense mechanisms and find shortcomings: “Static Scanning” and “Dynamic Anomaly Detection”
- As neither of the defenses can fully protect containers against state-of-the-art attacks according to our findings, further works to secure container-based applications are needed
  - We suggested some improvements

Thank you!

References

- [1] https://www.cncf.io/wp-content/uploads/2025/04/cncf_annual_survey24_031225a.pdf
- [2] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10646668 TABLE 1

Appendix 1: Dataset



Scenario	CVE ID	Severity	Applications
Dataset d1	CVE-2012-1823	High	PHP
	CVE-2014-0050	High	Apache Commons
	CVE-2014-0160	Medium	OpenSSL
	CVE-2014-3120	Medium	Elasticsearch
	CVE-2014-6271	Critical	Bash
	CVE-2015-1427	High	Elasticsearch
	CVE-2015-2208	High	phpMoAdmin
	CVE-2015-3306	Critical	ProFTPd
	CVE-2015-5477	High	BIND
	CVE-2015-5531	Medium	Elasticsearch
	CVE-2015-8103	High	JBoss
	CVE-2015-8562	High	Joomla
	CVE-2016-3088	High	Apache ActiveMQ
	CVE-2016-3714	Critical	ImageMagick
	CVE-2016-6515	High	OpenSSH
	CVE-2016-7434	Medium	NTP
	CVE-2016-9920	Medium	Roundcube
	CVE-2016-10033	High	PHPMailer
	CVE-2017-5638	Critical	Apache Strutus 2
	CVE-2017-7494	Critical	Samba
	CVE-2017-7529	Medium	Nginx
	CVE-2017-8291	Medium	Ghostscript
	CVE-2017-8917	High	Joomla
	CVE-2017-11610	Critical	Supervisor
	CVE-2017-12149	High	JBoss
	CVE-2017-12615	Medium	Apache Tomcat
	CVE-2017-12635	Critical	CouchDB
	CVE-2017-12794	Medium	Django
	CVE-2018-11776	Critical	Apache Strutus 2
	CVE-2018-15473	Medium	OpenSSH
	CVE-2018-16509	Critical	Ghostscript
	CVE-2018-19475	Critical	Ghostscript
	CVE-2018-19518	High	PHP
	CVE-2019-5420	High	Rails
Dataset d2	CVE-2019-6116	Medium	Ghostscript
	CVE-2019-10758	Critical	VM
	CVE-2020-1938	Critical	Apache Tomcat
	CVE-2020-17530	Critical	Apache Strutus 2
	CVE-2021-28164	Medium	Eclipse Jetty
	CVE-2021-28169	Medium	Eclipse Jetty
	CVE-2021-34429	Medium	Eclipse Jetty
	CVE-2021-41773	Medium	Apache HTTP Server
	CVE-2021-44228	Critical	Apache Solr
Dataset d3	CVE-2022-0847	High	Linux Kernel
	CVE-2022-21449	High	Oracle Java SE
	CVE-2022-22963	Critical	Spring Cloud
	CVE-2022-22965	Critical	Spring MVC
	CVE-2022-26134	Critical	Confluence Server
	CVE-2022-42889	Critical	Apache Commons
	CVE-2023-23752	Medium	Joomla
	CVE-2021-42013	Critical	Apache HTTP Server
Total Counts			51