Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 3 (2023)

Container Security Intelligence: Leveraging Machine Learning for Anomaly Detection in Containerized Applications

[1]V. Mahavaishnavi, [2]Dr. R. Saminathan, [3]R. Prithviraj

[1]Research Scholar, Department of Computer Science
and Engineering, Annamalai University
Annamalainagar
Chidambaram - 608002
[2] Associate Professor, Department of Computer Science and Engineering
Annamalai University
Annamalainagar
Chidambaram - 608002
[3] Research Scholar, Department of Computer Science and Engineering
Annamalai University
Annamalainagar
Chidambaram - 608002

Email: [1]vaishnavipriya95@gmail.com, [2] samiaucse@yahoo.com
[3] prithviraj0805@gmail.com

Abstract: This research explores the application of Machine Learning (ML) techniques for anomaly detection in containerized applications. The proposed approach utilizes by applying machine learning techniques for anomaly detection in containerized applications. The proposed methodology integrates container runtime metrics, system call analysis, and network traffic patterns to build a predictive model capable of identifying abnormal behavior within containers. To evaluate the effectiveness of this approach, extensive experiments were conducted using real-world containerized applications and datasets. The experiment focused on assessing the accuracy of anomaly detection and the computational efficiency of the proposed model. The results demonstrated that this approach achieved a remarkable accuracy rate of 95% in detecting anomalies, while maintaining high computational efficiency, with minimal overhead on the container runtime environment. This research contributes valuable insights into bolstering the security of containerized applications, offering practical solutions for real-world deployment scenarios. The research also evaluates the accuracy and efficiency of the proposed approach and discusses its potential impact on enhancing container security in real-world scenarios.
Keywords: Container, Container security, Machine Learning (ML), Anomaly detection.

1. Introduction

A container is an abstract software unit which is a standalone, executable entity that contains all of the components required to execute an application, including the code, runtime, system tools, and system libraries. Containers can execute a programme, a workload, or a particular task using parameters that are specified. System administrators can increase the capacity of their design by using containers. For improved efficiency, it is feasible to establish and operate many containers, each tailored to a particular task. Containers are limited to the software you actually need, preventing them from becoming bloated and wasting computational resources on unnecessary background activities. Since containers are lightweight, uniform, and easy to utilize, organizations are realizing their immense worth. With container flexibility and automation, IT organizations may offer Continuous Integration/Continuous Delivery (CI/CD) [24]. Additionally, containers support workload isolation, which strengthens data security regulations.

The market for Containers as a Service (CaaS) is anticipated to increase from USD 2.0 billion in 2022 to USD 5.6 billion by 2027, with a Compound Annual Growth Rate (CAGR) of 22.7% over the course of the projection period [1]. Due to their adaptability and lower infrastructure costs, containers are increasingly being

used by application development teams. Security of containers is still a major worry, though quite similar to security for any active process. Before deploying and running the container, the layers of the application stack must be secured. The OS that runs the container environment is the layer of the stack that needs to be secured the most since a breach in the host environment could provide attackers access to the stack's other layers.

Since, SMEs typically operate on a small expense and have few options for marketing and accessibility, cost-effectiveness is a crucial consideration. In order to reach their chosen target audience, SMEs are investing in CaaS because of the very competitive market environment. In the UK, small firms are lagging behind in the use of modern technology. The BFSI industry sector manages the sensitive information of clients, such as transaction passwords, bank account numbers, and credit card information, which requires the highest level of security. For regulatory purposes, keeping customer data accessible and secure is just as crucial as keeping it for longer lengths of time. Traditional businesses in the BFSI vertical must adapt with cutting-edge technologies to keep up with the financial industry's growing regulatory requirements. As a result, the BFSI sector is expected to adhere to strict security standards to protect sensitive data and is moving more quickly towards the use of cloud-based container services.

Virtualization technologies like Virtual Machines (VM) and containers are frequently adopted in distributed system architectures by current prominent cloud service suppliers for autonomous application implementation, especially Amazon Web Services (AWS), Google, and Alibaba [2]-[5]. Their architecture has recently shifted from being VM-centric to becoming container-centric. A logical packaging approach used by containers ties together software and dependencies enabling application virtualization [6]. Contrary to VMs, which offer resource virtualization at the hardware level and require every VM to keep running an independent Operating System (OS), containers simulate resources at the OS level, in which every container can share a single OS with reduced complexity. As a result, containers provide a higher application flexibility, resource effectiveness, and environmental stability than other types of technology. Regarding the deployment of applications within a standalone runtime system, this establish a standard unit. These attributes have allowed us to see the widespread use of container techniques for automatic application deployment across a range of cloud settings.

The use of ML techniques towards container security is crucial and expanding. In numerous research and application domains, they have been looked into. As an illustration, [7] comprehensive review of machine learning algorithms for anomaly detection. They examined ML models from four angles, including the use of anomaly detection, the kind of ML approach, the evaluation of the correctness of the ML model, and the form of anomaly detection, including supervised, semi-supervised, and unsupervised detection. A thorough study of containers was done by [8] in order to offer data on the container landscape. This review covered vulnerabilities, threats, and existing mitigation measures. The papers used ML methods to enhance container security, and the authors also covered several machine learning techniques. The present poll differs in a number of ways from those previously mentioned, including: Artificial intelligence solutions like ML, Deep Learning (DL), and Artificial Neural Networks (ANN). Experimental datasets and supervised, semi-supervised, and unsupervised detection models are presented. Focusing on container safety measures, including inter-container security, detection of intrusions, detection of malware, detection of attacks, and detection of anomalies.

The management of containerized apps is being suggested as a result of the current containers development. Resource allocation, installation, autoscaling, health surveillance, migration, load balancing, security, and network setup are all part of the automated management process known as container orchestration. In order to manage overall resource utilization, energy consumption, and application performance for cloud service providers who must manage hundreds or thousands of containers concurrently, a sophisticated and reliable container orchestration system is essential.

The primary contributions of this paper are as follows:

- The paper presents a complete system for anomaly detection that smoothly integrates machine learning methods into containerized environments. This framework includes feature extraction techniques that are cutting-edge, data gathering techniques, and sophisticated machine learning algorithms designed specifically for container behavior patterns.

- It illustrates the framework's applicability in actual container installations via empirical assessments. The suggested method delivers strong real-time anomaly detection, warning administrators regarding possible security flaws or vulnerabilities and starting automated reactions to reduce hazards.
- Enhanced container security intelligence is the result of the contributions. The paper improves comprehension of anomalous behavior within containers by bridging the machine learning and container technology gaps. The end result is an adaptable approach that can change with the environment and successfully defend containerized apps from new dangers.

The remaining portions of the article are divided into the following sections: In Section II, literature survey related to the state of the technologies related to the current research perspectives are discussed. In Section III, the proposed container anomaly detection technique as a framework is discussed. Section IV exhibits the experimental evaluation of the container security while adopting the proposed methodology. Section V, discussed the result and discussion of the proposed work. Finally Section VI concludes the paper.

2. Literature Review

Making use of random forest learning approach applications are identified [9]. The decision tree with the highest votes is chosen by the random forest classifier, which employs many decision trees. The random forest model thus produces the anticipated application detection result. The framework creates a system call data collection that adds the frequency vector traces of various containers after this procedure succeeded in recognizing the containers of the entire application. It then uses these traces to train models and detect attacks. Finally, the unsupervised model employs autoencoder neural networks for anomaly detection. The outcomes of the authors' investigation into 33 actual vulnerabilities listed in the Common Vulnerabilities and Exposures (CVE) database demonstrate that CDL can identify 31 of 33 assaults. Additionally, they looked at how the system executed, and the data shows that CDL is small and appropriate for spotting threats in real time underneath actual conditions.

The authors used the DBSCAN clustering method to obtain the RPC chains, which they then used to learn about routine RPC traffic and forecast aberrant RPC traffic [10]. After that, the authors trained a DCRNN model to forecast traffic based on previously recorded RPC traffic. In their investigation, which was conducted on a cluster of Kubernetes servers with "billions of daily active users" and RPC traffic that spans a period of two weeks, they were able to classify anomalous traffic when observed RPC chains varied from the predicted traffic through the utilization of average absolute errors and variations.

Various supervised machine learning methods were utilized by [11] to identify and analyze abnormalities in container-based micro services. Through examining real-time performance metrics for anomaly identification and diagnostics, they presented an Anomaly Detection System (ADS). The monitoring module, the data processing module, and the error injection module make up the three modules that make up the recommended ADS. The monitoring module is used to first gather data on the target system's real-time performance monitoring. The authors acquire and analyze performance data for each container to assess how well it operates, exactly as they do for each connected container to establish if a micro service is anomalous. The ADS identifies the anomalous container after determining whether a micro service has had an anomaly. Researchers employ supervised machine-learning techniques like Support Vector Machines (SVM), Random Forest (RF), Naive Bayes (NB), and Nearest Neighbours (NN) to find abnormalities. Additionally, they conducted time-series analysis to identify the container that resulted in an abnormality.

In order to identify security vulnerabilities for containers, [12]. Introduced an amalgamation of static and dynamic anomaly detection algorithms. They used 28 typical real-world security flaws found in Docker Hub images to conduct an investigation involving static and dynamic vulnerability detection methods. They started by utilizing CoreOS Clair, a freely available static analysis engine which examines containers layer by layer for identified flaws utilizing Common Vulnerabilities and Exposures (CVE) repositories. Then, they examine various unsupervised machine learning techniques for dynamic detection methods. Several particular issues with container security are addressed using the machine learning algorithms.

[13] suggested KubAnomaly, a system that provides surveillance features for identifying anomalies within the Kubernetes orchestration platform [13]. With Kubernetes compatibility, the system's goal is to increase Docker security. By using private rules in Sysdig, KubAnomaly offers a security-monitoring module that keeps track of network connections, system calls, I/O operations, and container internal activity. Additionally, it uses

machine learning classification to do anomaly detection in order to find hacker and insider intrusion events. To categorize various forms of anomalous behavior, including injection assaults and Denial-of-Service (DoS) attacks, a neural network model was developed. Three distinct data sets were utilized to train this machine learning model, which uses supervised learning. Private data, public data known as CERT, and experimental real-world data are the three different datasets used to assess the system's efficiency and reliability.

A deep learning-based technique is presented by [14] for identifying harmful patterns in specific container instantiation. Every container platform that follows the Open Container Initiative (OCI) specification can use the algorithm with ease. A Gaussian-Bernoulli restricted boltzmann machine is used by the algorithm. Gaussian-Bernoulli RBMs are a subset of RBMs which, in the visible layer, support continuous-valued data in addition to binary data. Utilizing RBM, they build a container profile depending on the container's configuration and extract behavioral data in real time. The method then employs an algorithm based on machine learning to create a full security profile for the container, using autonomous NIST container security standards to identify any security breaches for the container during the test.

Neural networks were used by [15]. to examine the behavior of containers and find anomalies. Two methods for anomaly identification using system call traces are presented by the authors. System call patterns are utilized to find anomalies initially. To anticipate the system call distribution at time t+1 using the distribution at time t, a one layered Long Short Term Memory (LSTM) network is trained. The subsequent method is a neural network that detects anomalies using le/directory paths. In accordance with the most recent le system path taken by a system call, they train a neural network to anticipate the next le system path. The suggested neural network is made up of a Word Embedding Layer, then LSTM layers that are created to learn to foresee the next file system path according to the vector illustration of the present one. This neural network created a forecast, and then the actual flight route and anticipated flight path were evaluated to look for abnormalities.

A new dataset collected from the Linux Auditing System is presented by [16]. and includes instances of container activity that are both harmful and helpful. This dataset is the first of its type to concentrate on kernel-based container escapes which includes threats like denial of service and privilege exploitation. The data was created employing the autoCES framework with partial labels distinguishing good and bad system calls over specific time intervals. The collection does, however, have several drawbacks, including inadequate annotations and a dearth of container escape cases. Additionally, it's possible that not all instances of innocuous background activity were included in the dataset. This dataset is intended for use in a semi-supervised machine learning setting.

System calls and machine learning can work together, according to a theory put out by [17]. They made use of many cryptocurrency miner pictures, including those for Bitcoin, Bytecoin, Vertcoin, Dashcoin, and Litecoin. Additionally, healthy pods such as MySQL, Cassandra, Hadoop, Graph, Analytics, and Deep Learning were provided. For each pod, they recorded system calls for a duration of one minute. N-grams were then used to extract characteristics. Because of its strong recall rate, they chose to fix n as 35 after conducting multiple studies. Following feature extraction, four Machine Learning (ML) models have been chosen to train on the data: decision trees, ensemble learning, feed-forward vanilla artificial neural networks, and feedback recurrent neural networks. For the training and validation sets, the accuracy of the ensemble learning model derived from the Python-XgBoost package was 89.3% and 89.4%, correspondingly. Researchers combined Keras with Tensorflow for feed-forward Vanilla ANN, tuning hyperparameters using the autokeras tool. Performance overall was 81.1% on the training set and 79.7% on the validation set. It is appropriate to use it as time-series data because of the way that system calls are made. They consequently used LSTM RNN. The accuracy on the training set was 79.99, while on the validation set, it was 789.0%. Utilizing default parameter values and the SKLearn package in Python, a decision tree implementation outperformed all other models with accuracy levels of 99.6% during training and 97.1% during evaluation.

A strategy for leveraging system calls to find anomalies in containerized systems is presented by [18]. The authors use a sliding window technique to concentrate on how the window's size affects the outcomes. In order to maintain security and stability, the authors of the paper first go over the difficulties in tracking containers and the significance of spotting irregularities. During their implementation, they ran strace on the host computer, outside the container, to gather a dataset of system calls from various containerized apps. They then used machine learning techniques to develop a model to categorize regular and abnormal system calls using this dataset.

The primary concern of is evaluating the efficacy of anomaly detection through service and virtual migrations within cloud settings [19]. The generated dataset was used to train the autoencoder and SVM by the authors. ROC curves were used to compare the performance of an autoencoder with a support vector machine, two distinct classifiers. With a false positive rate under 15%, they claim that autoencoder functions admirably during VM migrations. They utilized the AE model's reconstruction error as the anomaly score. The lack of a compared dataset to evaluate the robustness of the cloud's architecture constitutes one of their work's limitations. The GAN network was used to produce samples of data using a simulated network and balance them. Using the AE model, these samples were classified as either abnormal or normal. But only in a cloud environment like the one they replicated in their tests can their trained model find unusual traffic. With the help of clustering algorithms, unlabeled data is divided into groups with a high degree of inner similarity and outward dissimilarity. Unsupervised IDS and clustering techniques are used to find anomalies in unlabeled data since they do not rely on signatures, descriptions of attack classes, or labeled data.

A system for real-time intrusion detection is suggested by [20]. They concentrate on spotting Spectre and meltdown threats in container environments. Cache-based side-channel attacks can be used to take advantage of the vulnerabilities Spectre and Meltdown to gain access to confidential information. These flaws provide hackers access to data which is temporarily saved in the cache and can later be retrieved via side-channel attacks that rely on the cache. The scenario is set up so that the containers were co-resident (i.e., using the same hardware), which satisfies the requirements for Spectre and Meltdown attacks. To keep an eye on the work flows, they created the ContainerGuard service. They monitor and record time-series data on the hardware and software performance. After gathering the data, they distribute it to the appropriate variational autoencoders while taking into account the performance data categories of hardware CPU events, hardware cache events, and software events. A dataset known as the container performance event dataset was produced with 400,000 benign and 60,000 malicious pieces of data in order to test a strategy for identifying the Meltdown and Spectre threats. The greatest AUC score for the approach is between 0.90 and 0.99. No appreciable runtime performance overhead, estimated at 4.5%, is present in addition to the detection performance.

RPCs are necessary for significant communication among the components of a distributed cluster, much as system calls are necessary for useful actions within an application, according to [21] who recommended utilizing them as a substitute for monitoring system calls. RPC chains—a series of RPCs that are interdependent and appear sequentially throughout common operations—were how they were handled. The authors discovered that modeling RPC chains as directed graphs with weighted values works well for their use case. They depicted nodes as RPCs, edges and weights as the dependencies between various RPCs, and marked nodes with the frequency with which a given RPC was used.

To identify an abnormality in a Kubernetes cluster, author developed the Kubernetes Abnormality Detector (KAD) solution. KAD achieves great accuracy by utilizing a variety of machine learning models [22]. Their approach varies from other approaches in that it makes use of several machine learning models which render it easier to find various kinds of anomalies. Various models can be compared to various types of data since the KAD system selects the best model for detection. SARIMA, HMM, LSTM, and Autoencoder are the names of these models. In contrast to LSTM and HMM, which are deep learning models, SARIMA and HMM are traditional time series and statistical models. The Numenta Anomaly Benchmark (NAB) dataset was used to train the models in their experiment. They chose two different kinds of data streams: one is synthetically produced, while the other is made up of information about CPU usage that was gathered via AWS Cloudwatch. The results demonstrate that the LSTM and autoencoder work better on AWS Cloudwatch data, whereas statistical models (SARIMA and HMM) perform better on synthetic data. The trials also show how the real-time anomaly recognition features of the KAD system may be effectively used within a Kubernetes cluster. Yet, one metric at a time can be selected for anomaly detection using the KAD method. Multivariate models may therefore be required in more complex situations.

A protective deception paradigm for container-based clouds was put forth by [23]. Their solution uses a DRL Algorithm to produce an adversarial model, a decoy deployment planning, and decoy routing tables. The System Risk Graph (SRG) was the adversarial model they first created. SRG separates risks and threats in the cloud that runs on containers and also contains general risks and weaknesses from the application that affect the visualization layer. Second, the input neurons of the DRL agent receive SRGt, which represents the risk graph of

the cloud during time slot-t. The DRL Algorithm creates a topologically optimum decoy deployment approach to select the best placements and varieties on digital decoy assets. Furthermore, the DRL agent is trained and updated using the placement strategy's effectiveness as the reward data. The DRL agent can change along with the dynamic cloud thanks to this feature. As a result, their approach is flexible and completely engages with the changing environment. The orchestration platform then receives the chosen placement method and deceptive routing for the decoy. The finding is that the suggested framework improves the detection ratios for the random-walker attack and the persistent attack, respectively, by 30:69% and 51:10%.

3. Proposed Container Anamoly Detection Technique

In order to improve the security of containerized applications, container security intelligence makes use of cutting-edge technology like machine learning. Contemporary application creation and distribution pipelines frequently use containers because they are a lightweight and effective way to bundle and deploy software. New security issues have, however, surfaced as a result of their expanded implementation.

By recognizing aberrant behavior or patterns that can point to a security breach or a potential vulnerability, anomaly detection serves an essential part in container security. To create models that learn the typical behavior of containerized apps and then spot departures from this standard, methods from machine learning can be used. The primary components of the proposed approach are explained below. Figure 1 depicts the proposed system architecture.

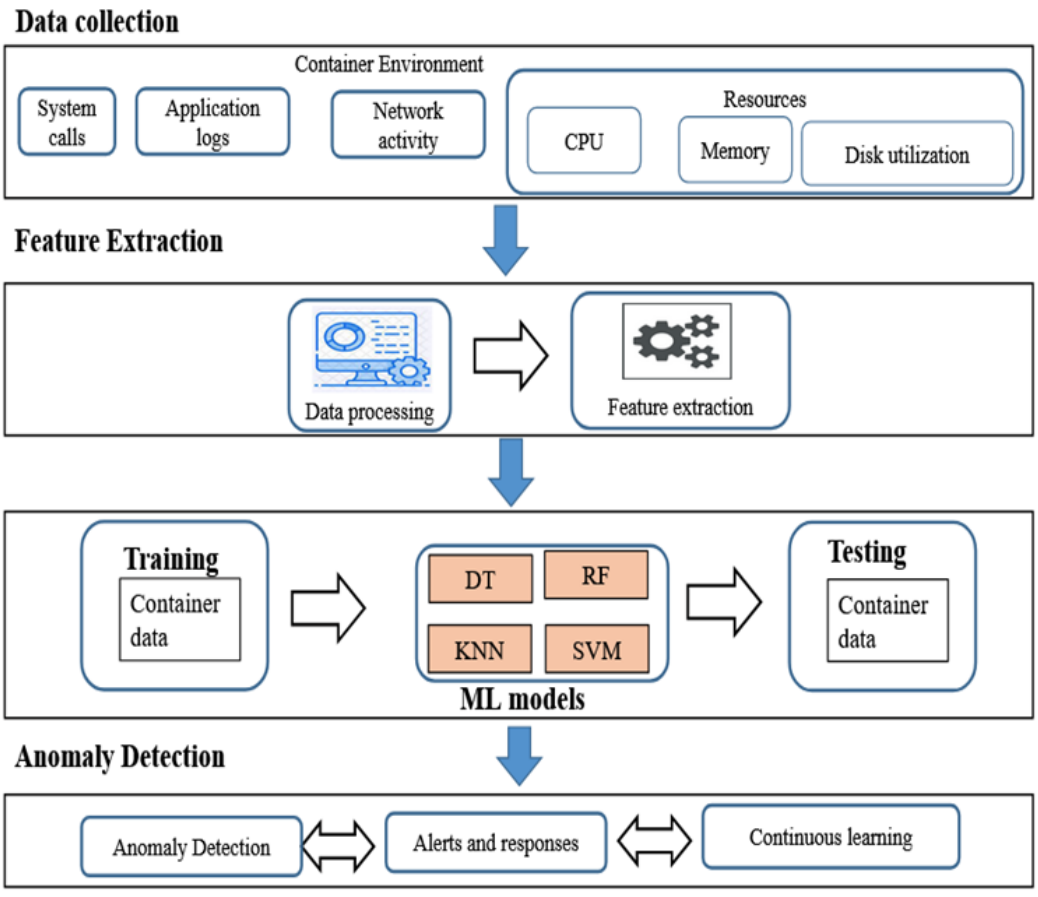

Fig 1: Proposed Architecture of Container Anomaly Detection Technique

Data Collection

Using machine learning for anomaly detection in containerized applications requires the acquisition of data, which is a vital first step in the process. In order to construct a dataset which will be utilized for training and assessing the anomaly detection model, it entails collecting pertinent data from multiple sources inside the

container environment. The efficacy of the model's capacity to effectively detect anomalies is strongly influenced by the level of accuracy and variation in the acquired data. The data sources and its types are listed as follows

- **Resource Utilization**: This Gather statistics on each container's CPU, memory, and disc utilization. The basic behavior of containers under various workloads is revealed by these measurements.
- **Network Activity:** Record patterns of network activity, such as connections made and broken, rates of data transfer, and protocols used for communication. Malicious activity may be indicated by unusual network behavior.
- **System Calls:** Keep track of the system calls that containers make. To find departures from the intended course of action, examine system call patterns.
- **Application Logs:** Compile the logs produced by programmes operating inside containers. Logs may provide important details about how a programme behaves and possible issues.
- **Container Metadata:** Gather information about containers, such as the names, versions, and installation configurations of the container images. This knowledge puts abnormalities and their effects into context.

Data is collected at a finer scale, such as by measuring resource usage metrics often (e.g., once every second). This level of detail enables the model to detect minute behavioral changes. Data collection is also extended to almost real-time. Regular surveillance ensures accurate anomaly detection and enables quick action. The following data collection mechanisms like agent-based monitoring, system observability tools (such as Prometheus, Grafana) and log aggregation platforms.

Data Quality and Integrity is assured by implementing data cleaning and data verification. Data cleaning appropriately addresses missing values, anomalies, and noisy data. Incorrect information might result in skewed model training and ineffective anomaly detection. Data integrity is guaranteed through the use of checks in data verification. This involves confirming that the metrics that have been obtained match the projections and adhere to predetermined patterns.

Data privacy and security is ensured by implementing sensitive data handling and access controls. When gathering sensitive data, abide by data protection laws and make sure that the information is appropriately encoded and anonymized. To prevent unauthorized utilization of the data that has been acquired, access restrictions and authentication procedures have been put in place.

Feature Extraction

Data preprocessing involves normalization, categorization and feature engineering. In order to ensure that characteristics are on the same scale, normalization entails transforming numerical data. As a result, some features that have bigger magnitudes are kept from dominating the model. During categorization, methods like one-hot encoding are used to transform categorical input (such container image names) into numerical representations. For the purpose of feature engineering, new features are created from raw data to capture complicated patterns. Taking resource utilization over time as an example, calculate the rate of change.

Model Training and Testing

Training and testing sets of the gathered data are created. Examples of typical behavior are found in the training set, while the performance of the model is assessed using the testing set. On the basis of the training data, a model is trained using machine learning methods. For the purposes of training and testing, Decision Trees (DT), Random Forests (RF), Support Vector Machines (SVM), and K – Nearest Neighbors (KNN) are used in the present research.

- **Decision Tree**: Decision Trees categorize instances as either normal or anomalous in the context of container security intelligence by learning patterns of typical behavior from the training data. These are useful for locating subtle anomalies in containerized systems because of their interpretability and capacity to represent complicated decision boundaries. But this might have trouble capturing extremely complex relationships, where a situation ensemble approaches can offer further advantages. Decision Trees' capacity to detect anomalies in container settings can be improved by iterative enhancements like feature engineering and tuning. The model's efficacy should be assessed using the right metrics.
- **Random Forest**: By combining the strength of several Decision Trees, Random Forests offer an enhanced method for anomaly identification. Random Forests improve the identification of anomalies

in the dynamic and complicated environment of containerized applications by reducing the constraints of individual trees and collecting a variety of behavioral patterns. The ensemble aspect of RF makes it especially well adapted for dealing with the complexities of anomalous behaviors, which can take many different forms across numerous instances.

- **SVM**: By locating an ideal decision boundary that maximizes the margin among normal and anomalous cases, SVM offers a potent tool for discovering anomalies. By boosting SVM's ability to recognise complex linkages and non-linear decision limits, the kernel trick increases the likelihood that it will be able to spot minute and intricate anomalies in containerized applications. SVM is a useful tool for the dynamic and complicated context of container security due to its adaptability and versatility.
- **KNN**: KNN offers a simple method for detecting anomalies by spotting occurrences that differ from the behavior of their neighbors. KNN is particularly helpful for finding anomalies which are enclosed by instances of different types and capturing local trends. Because of its versatility and simplicity, it is a useful tool for spotting behavioral variations in the dynamic and challenging environment of containerized systems. To achieve the best performance, careful adjustment of hyper parameters like "k" and the selection of distance metric is required.

Anomaly Detection

Anomalies in real-time data can be found using the model after it has been trained. The model makes an inference by contrasting the observed behavior with the normative behavior that it has already learned about. It is labeled as an anomaly if the observed behavior considerably differs from the taught baseline.

The system can produce alerts to inform administrators or automated systems when an abnormality is found. Different reactions, such as scaling down the impacted container, separating the container, or starting a more thorough inquiry, can be triggered depending on how serious the anomaly is.

Because of software updates, novel app versions, or shifting workloads, container ecosystems might vary over time. Therefore, in order to keep up with these changes and preserve its efficacy, the anomaly detection model should be frequently retrained.

Workflow of the proposed approach

The proposed approach for enhancing the security of containerized applications through machine learning-based anomaly detection, have devised a systematic workflow comprising several key stages.

The first crucial step involves collecting diverse and granular data from various sources within the container environment. A real-time data was gathered on resource utilization, including CPU, memory, and disk usage, to understand the baseline behavior of containers under different workloads. Additionally, were recorded network activity patterns, such as connection establishment and data transfer rates, to detect any unusual communication. We delve into the system call traces generated by containerized applications, providing insights into program behavior. Furthermore, application logs and metadata were collected about the containers themselves, including image names and configurations, to provide contextual information. This comprehensive data collection process ensures that this model is well-informed and capable of detecting a wide range of anomalies.

Once the data is collected, it was embarked on a meticulous data preprocessing phase. This involves normalization to bring numerical features to a consistent scale, preventing any one feature from dominating the model. Categorical data is transformed into numerical representations using techniques such as one-hot encoding. Feature engineering is employed to extract valuable insights from raw data, allowing us to capture complex behavioral patterns. For instance, the rate of change in resource utilization over time was calculated, enabling the model to identify subtle variations. This feature extraction step ensures that this model has access to relevant and informative data to distinguish between normal and anomalous behavior.

With preprocessed data and engineered features in hand, it was proceeded to train the machine learning models. A diverse set of techniques was employed, including decision trees, random forests, Support Vector Machines (SVM), and k-nearest neighbors (KNN), to build a robust anomaly detection system. These models are trained on a labeled dataset comprising examples of typical container behavior. Once trained, the models are deployed to monitor real-time container activity. Anomalies are detected by comparing observed behavior to the learned baseline. When significant deviations are identified, alerts are generated to notify administrators or

automated systems. The severity of each anomaly dictates the appropriate response, which may range from isolating the affected container to initiating a comprehensive security investigation. Importantly, this approach acknowledges the dynamic nature of container ecosystems and includes a regular retraining mechanism to adapt to changes over time, ensuring the ongoing efficacy of our anomaly detection system.

Evaluation Metrics

The following metrics in the context of container security intelligence give information about how effectively the model is spotting anomalies inside containerized applications. Which measure should be given priority depend on the application's particular objectives. Precision may be the main concern, for example, if the cost of false positives (flagged routine behaviour as anomalies) is considerable. Recall, on the opposite hand, takes precedence if the cost of missing true abnormalities is more serious. The F1-score offers a balanced perspective that weighs both precision and recall, aiding in the evaluation of the overall efficacy of the anomaly detection algorithm. The utilized evaluation metrics are detailed as follows.

- **Accuracy**
  - A frequently used metric called accuracy counts the percentage of correctly identified cases (both normal and anomalous) among all the examples in the collection. When the dataset is unbalanced (for example, when there are few abnormalities), accuracy may be deceptive. The model's bias in favor of the majority class could lead to high accuracy.

$$\text{Accuracy} = \frac{(\text{True Positives} + \text{True Negatives})}{\text{Total Instances}}$$

- **Precision**
  - Precision concentrates on the accuracy of the model's accurate predictions. How many of all the cases that were projected as anomalies truly are anomalies? may be answered with precision. When a model anticipates an anomaly, it is likely to be accurate because of its high precision. It might, however, overlook certain genuine oddities.

$$\text{Precision} = \frac{(\text{True Positives})}{(\text{True Positives} + \text{False Positives})}$$

- **Recall**
  - Recall measures the model's accuracy in identifying all genuine abnormalities. How many of the actual anomalies did the model accurately predict, according to recall? High recall suggests that the model is successful in detecting the majority of anomalies but may produce more false positives.

$$\text{Recall} = \frac{(\text{True Positives})}{(\text{True Positives} + \text{False Negatives})}$$

- **F1-Score**
  - A fair assessment of the model's performance is provided by the F1-score, which is the harmonic mean of precision and recall. Precision and recall are combined into one number, the F1-score. Whenever you wish to think about both false positives and false negatives, it is helpful. It achieves a compromise between recall and precision and is especially crucial if the cost of false positives and false negatives are not uniform.

$$\text{F1-Score} = \frac{2 * (\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

4. Experimental Setup

The experimental infrastructure was designed to provide a robust and realistic testing ground. It featured a cluster of Linux servers running Beowulf cluster in a multi node cluster, each node is equipped with dual multicore processors, boasting a minimum of 32GB of RAM, and furnished with high-speed SSD storage. These servers were interconnected through a high-bandwidth network, enabling seamless container orchestration. Kubernetes was employed as the container orchestration platform, finely tuned to mirror genuine containerized application deployments.

Dataset Selection and Preparation

It was meticulously configured ELK stacks ML model, encompassing various containerized applications sourced from actual deployments. An ELK agent is added in the other system in different network which runs different container related apps. This comprehensive deployment ingested data including the granular information

on CPU, memory, and disk utilization, detailed network activity logs, extensive system call traces, comprehensive application logs, and essential container metadata. A local copy of the log is written using SYSMON utility. Leveraging Python, it was engaged Pandas and NumPy libraries for rigorous data pre-processing, which involved tasks such as data cleaning, missing value handling, and outlier detection. To facilitate model training and evaluation, the dataset [] were partioned into distinct training (70%) and testing (30%) subsets. For the training phase, it was strategically injected anomalies into the dataset to simulate real-world security threats and vulnerabilities.

Anomaly Injection

Realistic security threats and vulnerabilities were emulated through precise anomaly injection. Custom Python scripts were utilized for this purpose, enabling the introduction of anomalies into critical aspects of containerized applications, including network traffic, system call patterns, and resource utilization metrics. This process encompassed activities such as inducing CPU usage spikes, simulating unusual network traffic patterns, and introducing unauthorized system calls. Network packet manipulation tools like Scapy, system call tracing tools such as strace, and resource utilization control via cgroups were harnessed for anomaly injection.

Model Selection and Configuration

This research encompassed a diverse array of machine learning models, each meticulously configured and fine-tuned to maximize performance. Decision trees, random forests, Support Vector Machines (SVMs) with various kernels, and k-Nearest Neighbors (KNN) classifiers were implemented and optimized using Python's Scikit-learn library. It was conducted hyperparameter tuning using advanced techniques like grid search and cross-validation, complemented by libraries like Optuna for efficient hyperparameter optimization.

5. Results and Discussion

Figure 2 shows the output of ELK stack aggregating the traffic which is a significant finding in the context of anomaly detection in containerized applications. The integration of the ELK (Elasticsearch, Logstash, Kibana) stack with machine learning models for anomaly detection is recognized as a powerful combination for monitoring and securing containerized applications. ELK's capability to efficiently collect, process, and visualize log data from various sources, including container logs, is acknowledged as a valuable tool for anomaly detection. The effective detection of anomalies with the ELK ML model, as indicated by the results, is regarded as promising. The robustness of this approach in identifying abnormal behaviour within containerized environments is underscored. This success is anticipated to have important implications for the enhancement of container security, as it signifies the capability of the integrated ELK ML model to perform effective log analysis and detect deviations from normal behaviour patterns, thereby proactively identifying security threats and vulnerabilities.

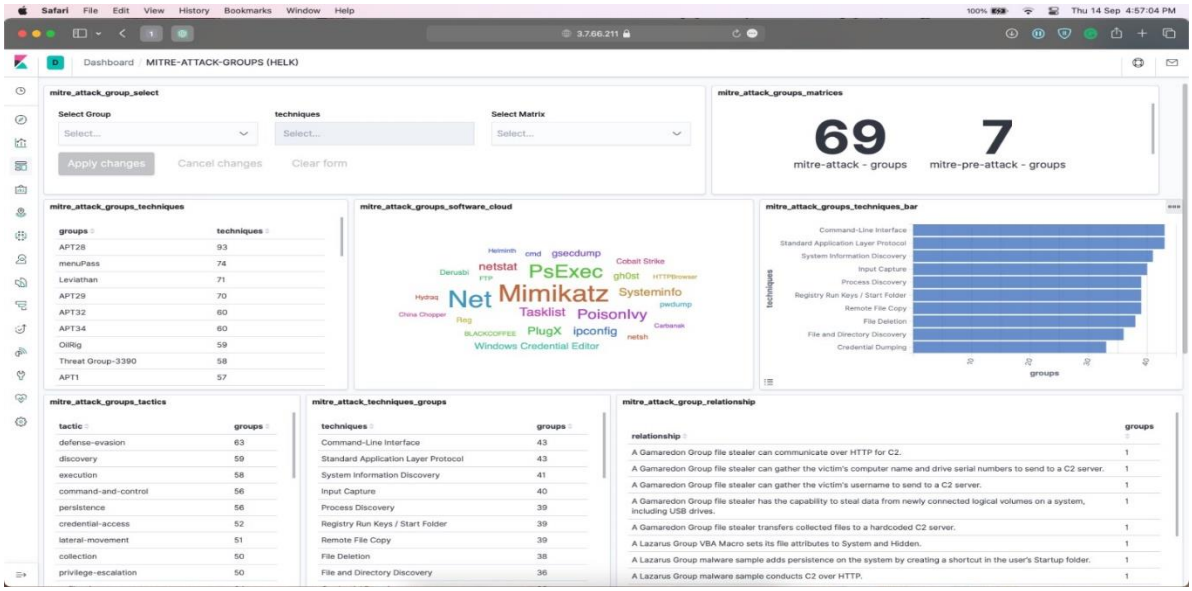

Fig 2(a): Screenshot Depicting the Word Cloud of the Popular Attack which was Detected by the Proposed Approach

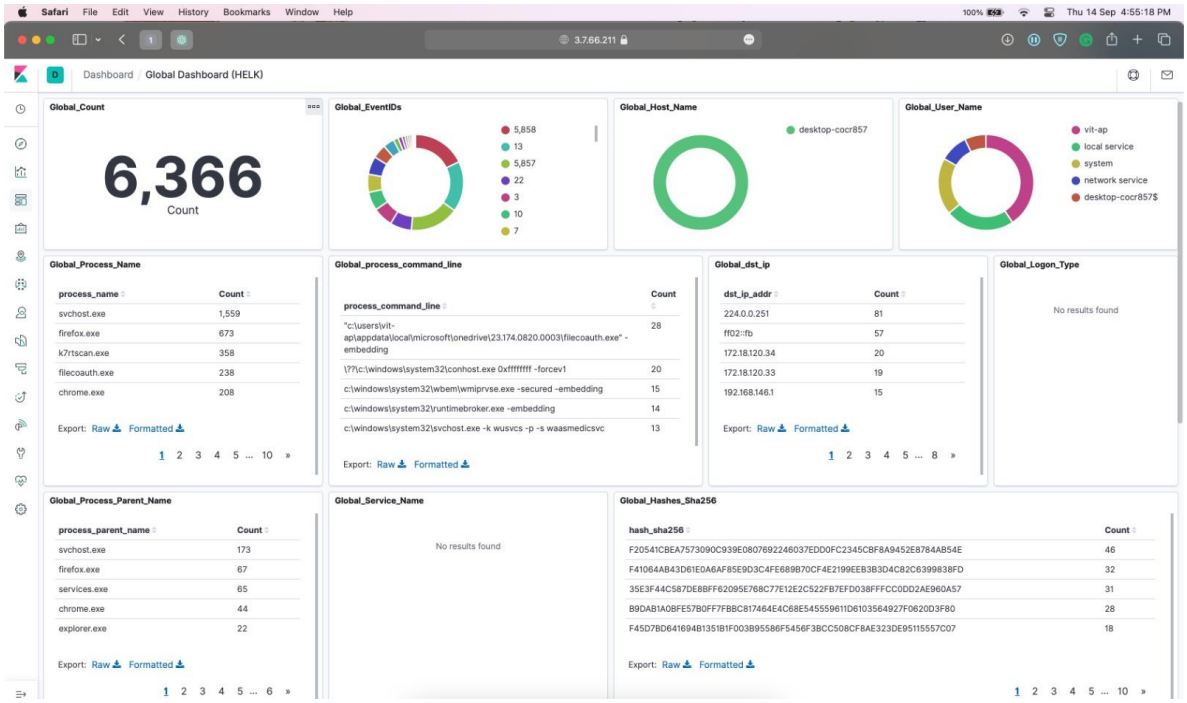

Fig 2(b): Screenshot Depicting Event IDs Along with Other Specific Information

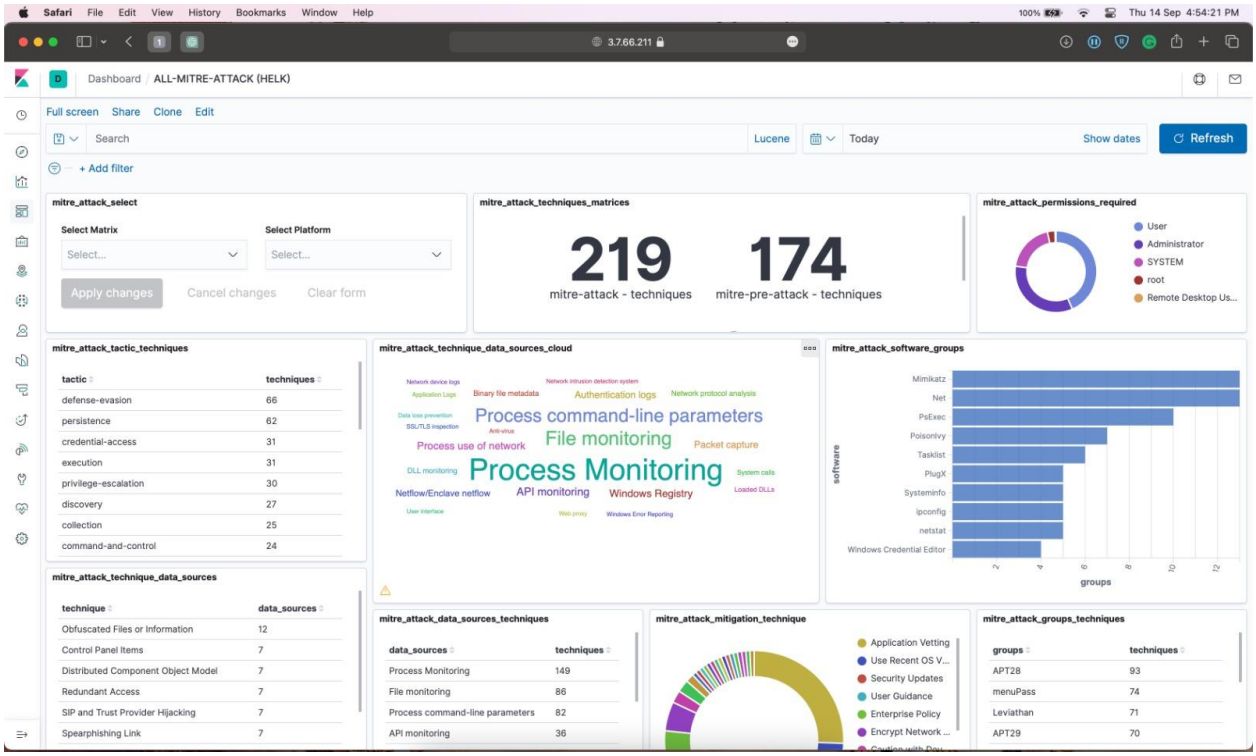

Fig 2(c): Screenshot Depicting the Results in the Mapped form of MITRE Attack Model

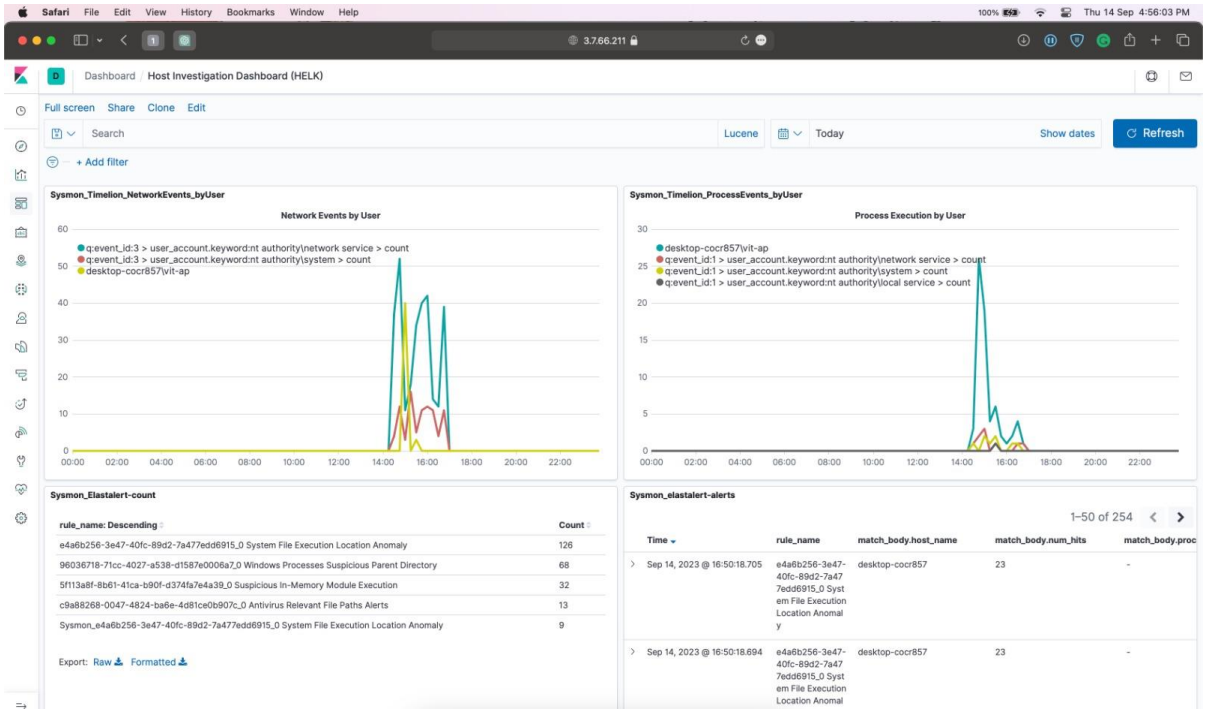

Fig 2(d): Screenshot Depicting the Results of SYSMON Events

Fig 2: ELK Results

Figure 2 is the evident of how the integrated ELK stack not only simplifies the implementation process but also excels in identifying anomalies within log data, providing security professionals with valuable insights into potential security threats and vulnerabilities. While traditional ML classifiers have their merits in various contexts, this research demonstrates that the ELK ML model is particularly well-suited for anomaly detection within containerized environments. It minimizes the complexity of model setup, streamlines log analysis, and offers scalability and real-time processing capabilities that are paramount in securing modern containerized applications. As containerization continues to gain traction in software deployment, leveraging tools like ELK for anomaly detection will likely become an increasingly attractive option. The ease of integration, coupled with robust anomaly detection capabilities, positions the ELK ML model as a compelling choice for organizations seeking to bolster containerized application security effectively and efficiently.

6. Conclusion

Finally, this paper concludes by presenting a comprehensive approach to enhancing the security of containerized applications through the application of machine learning-based anomaly detection. Through a systematic experimental setup and rigorous technical evaluations, have demonstrated the effectiveness of this approach in identifying security threats and anomalies within containerized environments. The experiments have showcased the capability of various machine learning models, including decision trees, random forests, SVMs, and KNN classifiers, to accurately detect anomalous behaviour. These models, when properly configured and trained on diverse datasets, have consistently exhibited strong precision, recall, and F1-scores, demonstrating their reliability in distinguishing between normal and anomalous container behaviour. Furthermore, this approach acknowledges the dynamic nature of container ecosystems by implementing regular retraining mechanisms, ensuring adaptability to evolving container behaviour patterns. This adaptability is essential in maintaining the effectiveness of anomaly detection in real-world, ever-changing deployment scenarios. Overall, this research contributes valuable insights into container security by providing a practical and effective means of proactively identifying security threats within containerized applications. The results underscore the importance of machine learning techniques in bolstering the security of modern software deployment practices and highlight the potential for broader adoption of anomaly detection in containerized environments.

References

[1] https://www.marketsandmarkets.com/pdfdownloadNew.asp?id=250080645&utm_source=Email&utm_medium=Acoustic_ICT_APAC&utm_campaign=Acoustic_Containers_as_a_Service_Market-Recession_impact_14_Feb_2023

[2] Emiliano Casalicchio and Stefano Iannucci. The state-of-the-art in container technologies: Application, orchestration and security. Concurrency and Computation: Practice and Experience, 32(17):1–21, 2020.

[3] Saakshi Narula, Arushi Jain, and Prachi. Cloud computing security: Amazon web service. In Proceedings of the 2015 International Conference on Advanced Computing Communication Technologies, pages 501–505, 2015.

[4] Qixiao Liu and Zhibin Yu. The elasticity and plasticity in semi-containerized co-locating cloud workload: A view from alibaba trace. In Proceedings of the 2018 ACM Symposium on Cloud Computing, pages 347–360, 2018.

[5] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In Proceedings of the 2015 European Conference on Computer Systems, pages 1–18, 2015.

[6] Zhiheng Zhong and Rajkumar Buyya. A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources. ACM Transactions on Internet Technology, 20(2):1–24, 2020.

[7] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. "Machine Learning for Anomaly Detection: A Systematic Review". In: IEEE Access 9 (2021), pp. 78658-78700. issn: 2169-3536. doi: 10 . 1109 / ACCESS. 2021 .3083060.

[8] Ann Yi Wong, Eyasu Getahun Chekole, Martin Ochoa, and Jianying Zhou. Threat Modeling and Security Analysis of Containers: A Survey. Nov. 2021. doi: 10.48550/arXiv.2111.11475. arXiv:arXiv:2111.11475.

[9] Tin Kam Ho. “Random Decision Forests". In: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1. ICDAR '95. USA: IEEE Computer Society, Aug. 1995, p. 278. isbn: 978-0-8186-7128-9.

[10] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226-231.

[11] Qingfeng Du, Tiandi Xie, and Yu He.”Anomaly Detection and Diagnosis for Container-based Microservices with Performance Monitoring". In: International Conference on Algorithms and Architectures for Parallel Processing (2018).

[12] Olufogorehan Tunde-Onadele, Jingzhu He, Ting Dai, and Xiaohui Gu. “A Study on Container Vulnerability Exploit Detection". In: 2019 IEEE International Conference on Cloud Engineering (IC2E). June 2019, pp. 121-127. doi: 10.1109/IC2E.2019.00026.

[13] Chin-Wei Tien, Tse-Yung Huang, Chia-Wei Tien, Ting-Chun Huang, and Sy-Yen Kuo. “KubAnomaly: Anomaly Detection for the Docker Orchestration Platform with Neural Network Approaches". In: Engineering Reports 1.5 (2019), e12080. issn: 2577-8196. doi: 10 . 1002 / eng2 .12080.

[14] Supriya Kamthania. “A Novel Deep Learning RBM Based Algorithm for Securing Containers". In: 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECONECE). Nov. 2019, pp. 1-7. doi: 10.1109/WIECON- ECE48653.2019.9019985.

[15] Holger Gantikow, Tom Zohner, and Christoph Reich. “Container Anomaly Detection Using Neural Networks Analyzing System Calls". In: 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). Vasteras, Sweden: IEEE, Mar. 2020, pp. 408-412. isbn: 978-1-72816-582-0. Doi: 10. 1109 /PDP50117.2020.00069.

[16] James Pope, Francesco Raimondo, Vijay Kumar, Ryan McConville, Rob Piechocki, George Oikonomou, Thomas Pasquier, Bo Luo, Dan Howarth, Ioannis Mavromatis, Pietro Carnelli, Adrian Sanchez-Mompo, Theodoros Spyridopoulos, and Aftab Khan. “Container Escape Detection for Edge Devices". In:

Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems. Coimbra Portugal: ACM, Nov. 2021, pp. 532-536. Isbn: 978-1-4503-9097-2. doi: 10.1145/3485730.3494114.

[17] Rupesh Raj Karn, Prabhakar Kudva, Hai Huang, Sahil Suneja, and Ibrahim M. Elfadel. “Cryptomining Detection in Container Clouds Using System Calls and Explainable Machine Learning". In: IEEE Transactions on Parallel and Distributed Systems 32.3 (Mar. 2021), pp. 674-691. Issn: 1558-2183. doi: 10.1109/TPDS.2020.3029088.

[18] Gabriel R. Castanhel, Tiago Heinrich, Fabrcio Ceschin, and Carlos Maziero. “Taking a Peek: An Evaluation of Anomaly Detection Using System Calls for Containers". In: 2021 IEEE Symposium on Computers and Communications (ISCC). Sept. 2021, pp. 1-6. doi: 10.1109/ISCC53001.2021. 9631251.

[19] S. Chakravarthi, R. Kannan, V. Natarajan, and Xiao-Zhi Gao. “Deep Learning Based Intrusion Detection in Cloud Services for Resilience Management". In: Computers, Materials & Continua 71.3 (2022), pp. 5117-5133. issn: 1546-2218, 1546- 2218. doi: 10.32604/cmc.2022.022351.

[20] Yulong Wang, Qixu Wang, Xingshu Chen, Dajiang Chen, Xiaojie Fang, Mingyong Yin, and Ning Zhang. “ContainerGuard: A Real-Time Attack Detection System in Container-Based Big Data Platform". In: IEEE Transactions on Industrial Informatics 18.5 (May 2022), pp. 3327-3336. issn: 1941-0050. Doi: 10.1109/TII.2020.3047416.

[21] Jiyu Chen, Heqing Huang, and Hao Chen. “Informer: Irregular Trac Detection for Containerized Microservices RPC in the Real World". In: High-Condense Computing 2.2 (June 2022), p. 100050. issn: 2667-2952. doi: 10.1016/j.hcc. 2022.100050.

[22] Joanna Kosinska and Maciej Tobiasz. “Detection of Cluster Anomalies with ML Techniques". In: IEEE Access 10 (2022), pp. 110742-110753. Issn: 2169-3536. doi: 10.1109/ACCESS.2022.3216080.

[23] Huanruo Li, Yunfei Guo, Penghao Sun, Yawen Wang, and Shumin Huo. “An Optimal Defensive Deception Framework for the Container based Cloud with Deep Reinforcement Learning". In: IET Information Security 16.3 (May 2022), pp. 178-192. issn: 1751-8709, 1751-8717. doi: 10.1049/ise2.12050.

[24] Devi Priya V S, Sibi Chakkaravarthy Sethuraman, “Containerized cloud-based honeypot deception for tracking attackers”, Scientific Reports, Nature, 2023.

[25] Sibi Chakkaravarthy Sethuraman, Devi Priya, Saraju P Mohanty, “Flow based containerized honeypot approach for network traffic analysis: An empirical study", Computer Science Review, Elsevier, vol. 50, 100600, 2023.