Detecting DDoS Attacks in Microservice Architectures via AI-Based Agents

Giacomo Benedetti[2], Luca Caviglione[2], Alberto Falcone[2], Massimo Ficco[1], Massimo Guarascio[2], and Antonio Guerriero[1(✉)]

[1] Università degli Studi di Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Italy
antguerriero@unisa.it
[2] National Research Council of Italy, Genova and Rende, Italy

Abstract. The widespread adoption of microservices captured the attention of attackers, mainly due to their distributed and dynamic nature. Unfortunately, traditional intrusion detection mechanisms may struggle to accurately and efficiently identify the most effective threats, for instance, DDoS campaigns. To advance in the security of modern microservice architectures, this paper presents an Intrusion Detection Agent (IDA) for the run-time identification and classification of DDoS campaigns. Specifically, the IDA takes advantage of AI to classify attacks in a multi-container environment. To prove the effectiveness of our approach, we collected data from a realistic testbed built on top of the Train Ticket framework. Then, a classification pipeline has been evaluated when used to identify four attack templates, *i.e.*, Standard DDoS, Slow DDoS, GET floods, and SYN floods. Obtained results showcased that the AI-based IDA can correctly handle the considered offensive templates, even when the data is scarce. For instance, when a decision tree is used, our IDA achieves an accuracy of ∼0.991 by considering only 4% of measurements capturing the behavior of containers (*e.g.*, the used RAM or the volume of network traffic).

Keywords: Machine Learning · Intrusion Detection · Denial of Service · Security benchmark · Microservices

1 Introduction

The rapid evolution of MicroService-based Architectures (MSAs) has facilitated the creation of modular, scalable, and resilient systems. As a consequence, they are responsible for the transformation of the development process at the basis of major software frameworks and Internet-wide platforms [4]. Unfortunately, MSAs are highly-distributed and characterized by a large attack surface, which attract the increasing attention of many malicious treat actors [3]. Among the various attack models that proven to be effective, MSAs exhibit an inherent vulnerability against Distributed Denial of Service (DDoS) attacks. In fact, the use of containerized software jointly with their decentralized flavor requires to

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15889, pp. 3–15, 2026.
https://doi.org/10.1007/978-3-031-97603-2_1

extensively use inter-service communication paths, which inherently amplify the attack surface [11].

As a consequence, the research community doubled the efforts aimed at designing and evaluating automatic Intrusion Detection Systems (IDS) tailored for protecting MSAs [7]. Despite deploying heuristics may be a viable option, mitigation techniques leveraging AI proven their effectiveness. Specifically, AI-based solutions can take advantage of the monitoring infrastructure already integrated within MSAs, for instance to assess their performance or enforce service level agreements [9].

This paper presents an IDA endowed with AI capabilities for the run-time identification and classification of DDoS campaigns targeting MSAs. To prove the effectiveness of the proposed approach, we collected data from a realistic testbed implemented via the Train Ticket framework [31], which has been tested with various attack templates. In more detail, we considered a threat actor launching four types of DDoS offensive campaigns, *i.e.*, Standard DDoS, Slow DDoS, GET floods, and SYN floods. We point out that considering DoS-based threats is of major importance, since they can challenge the ability of MSAs of resorting to their intrinsic autoscaling mechanisms and cause major economical losses, for instance due to the over-provisioning of resources [1]. Obtained results showcased that the AI-based IDA can correctly detect and classify the considered DDoS attacks, even when the data is scarce. This aspect is of paramount importance, as effective IDA schemes should be able to balance their effectiveness with the overhead required to the monitoring infrastructure to make the detection possible (see, *e.g.*, [33] and the references therein).

Summing up, the contribution of this work is twofold: *i*) it confirms the effectiveness of the Train Ticket platform when used to experiment with container security, and *ii*) it proposes a prime IDA mechanism for demonstrating that the AI is a suitable tool for spotting DDoS-like attacks, even when in presence of scarce data.

The rest of the paper is structured as follows. Section 2 reviews past research on monitoring and security aspects of microservices, while Sect. 3 showcases the testbed for preparing the dataset used in our experiments. Section 4 describes the proposed AI-based IDA, Sect. 5 introduces the experimental setup, and Sect. 6 discusses numerical results. Lastly, Sect. 7 concludes the paper and outlines future research directions.

2 Related Work

The ability of microservices to handle complex software ecosystems promoted their adoption in several scenarios, such as artificial intelligence frameworks, massive IoT deployments, and cloud-native applications [26]. As a consequence, understanding the security properties of containers, orchestration mechanisms, and the overall deployments is an urgent need. In this vein, a vast amount of research has been done to enhance the security posture of microservices, for instance to make the development pipeline more robust, elaborate best practices, and enforce efficient static analysis mechanisms [13,16,29].

An important aspect concerns the ability of tracking the evolution of microservices both in terms of used hardware resources and software behaviors. Even if monitoring is an essential component to enforce the adherence to quality of service guarantees, it has been also extensively employed to detect anomalous patterns that may reveal an attack [21]. As possible examples, runtime monitoring techniques based on the collection of system calls have been used to spot the presence of Docker containers trying to exfiltrate sensitive data (*e.g.*, cryptographic keys) by means of information hiding techniques [25,32]. Moreover, data describing the evolution of containers (*e.g.*, the depletion of a specific hardware resource) can be mixed with more classical information (*e.g.*, logs) to tune proper heuristics able to spot anomalous behaviors [27]. More fine-grained behaviors (*e.g.*, volumes of syscalls or information on the execution flow) can be used to feed machine learning models or AI-based agents for detecting a wide-range of threats. Methods based on AI proven to be very effective in revealing the presence of stealth cryptomining software exploiting resources of a population of containers [12]. Events related to the lifecycle of a microservice can be exploited as well. A viable approach to train a classifier for spotting security risks or unintended behaviors leverages operations such as traversing part of the filesystem hierarchy (*e.g.*, the `/etc`) or accepting an incoming TCP connection [28].

Indeed, various AI-capable frameworks have been proposed to mitigate attack campaigns against microservices or improve their security levels [10]. As an example, shared network resources among containers can be monitored with the goal of learning indicators that can prevent spoofing attempts [15]. Instead, [22] addressed the problem of identifying low-rate DoS attacks through various AI models. Similarly, [18] showcases the adoption of convolutional networks to detect DDoS campaigns against the application layer of microservices, whereas [3] discusses how XGBoost allows to classify many DDoS offensive techniques. Despite the used technique, being able to collect suitable datasets to train models and tune countermeasures is of paramount importance. Thus, [6] deals with a testbed for collecting information on how a large-scale set of orchestrated containers behaves when in presence of different DDoS attacks.

Despite being based on simple heuristics or on some form of AI, detecting attacks against MSAs requires to have a proper visibility over the hardware and software at runtime. In this perspective, the majority of researchers and industrial developers are agreeing on the effectiveness of the extended Berkeley Packet Filter (eBPF) [19]. In essence, the eBPF can be used to collect a vast set of information, granting visibility on network, hardware and software resources with a unique interface [33]. At the same time, the success of the eBPF is also rooted in its ability of gathering data without the need of instrumenting container images, which are often immutable [2].

3 Testbed and Datasets

`TrainTicket` is an open-source application, developed by FudanSELab, that implements a realistic microservices-based train ticket reservation system. The

aim is to provide to researchers and practitioners a testbed for investigating key software engineering challenges with microservices-based architectures, facing with the limited availability of suitable benchmark frameworks reflective of real-world microservices environments.

It includes 41 interconnected microservices, with each microservice dedicated to specific functional responsibilities. It is implemented by using different programming languages (Java, Node.js, Python, and Go) and exploiting databases like Mongo and MySQL.

`Train Ticket` has been used by Ficco et al. [6] to generate a dataset representing various DDoS attacks to a realistic MSA. Such dataset is publicly available at[1]. The authors simulate the nominal usage with the Locust workload generator, while the DDoS attacks (including Classic DDoS, Slow DDoS, SYN Flood, and GET Flood) were executed using the MHDDoS tool. The performance metrics were collected using CAdvisor and Prometheus tools, with sampling every five seconds across 30-min scenarios under idle and loaded conditions.

The resulting dataset consists of 10 subsets two depicting normal conditions, eight representing different attack scenarios. The dataset contains various features categorized into the following sets:

- *ContainerUsage*: Number of bytes consumed by the container on this filesystem;
- *CPUThrottling*: Total time duration the container has been throttled;
- *CPUUsage*: CPU quota of the container;
- *MemoryUsage*: Current memory usage, including all memory regardless of when it was accessed;
- *NetworkIO*: Cumulative count of bytes received/transmitted;
- *PacketsDropped*: Cumulative count of packets dropped while receiving;
- *PodReady*: Describes whether the pod is ready to serve requests;
- *ResourceLimits*: The number of requested limit resources by a container;
- *PodStatus (Restarts)*: The number of container restarts per container.

Each subset is labeled, distinguishing benign from malicious traffic allowing the identification of the specific performed attack.

4 Design of the AI-Based IDA

In this section, we describe the threat model and the solution approach adopted for revealing incoming attacks. Figure 1 illustrates the reference attack scenario where the AI-based IDA is deployed to safeguard a microservice-based application from DDoS attacks.

In more detail, an MSA is targeted by a DDoS attack, which may take various forms such as Standard DDoS, SYN floods, GET floods, or slow-rate attacks. These attacks aim to exhaust the computational or networking resources of the

[1] https://github.com/iotresearchunisa/MicroservicesDDoS.

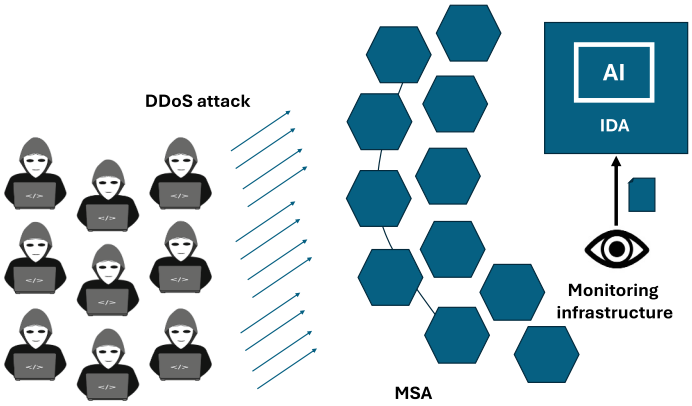


Fig. 1. AI-based IDA application scenario.

microservices, potentially degrading the system's performance or causing service outages.

To counteract these threats, an IDA is deployed alongside the application environment. It acts as the core detection component and is integrated with a monitoring infrastructure (based on tools like `Prometheus`, `CAdvisor`, and so on) that continuously collects runtime data. The monitoring layer gathers metrics such network flows, resource utilization, or other low-level indicators relevant for intrusion detection.

These data are processed and converted into a propositional view, which is used to feed the AI module responsible for real-time analysis. In our solution, we adopted a supervised approach, which means that the AI-based detector is trained on both benign and malicious traffic patterns. Leveraging the input data, it identifies anomalies and classifies ongoing events to detect potential DDoS attacks. When malicious activity is recognized, the system can raise alerts or take mitigation actions, depending on the deployment context.

Our IDA embeds four different supervised machine learning algorithms, each with distinct characteristics and offering trade-offs in terms of accuracy, interpretability, and computational cost. Specifically, we integrated four widely adopted classification models in cybersecurity literature [5,24,30], *i.e.*, K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Gradient Boosting (GB). In the following, we offer a brief overview of the functioning of each algorithm.

K-Nearest Neighbours. It is a non-parametric, instance-based classification method that assigns a label to a test sample by analyzing the labels of its nearest neighbors in the training set [23]. Specifically, given an instance a, the algorithm identifies the k training samples closest to a according to a predefined distance metric (*e.g.*, Euclidean distance, Manhattan distance). Once the top k neighbors have been identified, the class label of a is predicted as the most frequent class among those neighbors. This majority voting mechanism makes KNN suitable

in scenarios where class boundaries are irregular or nonlinear. Although KNN is a relatively simple model compared to more sophisticated ones, it was chosen for its suitability in data stream processing scenarios, where real-time monitoring and attack alerting are essential. Its instance-based nature allows it to adapt to incoming data without the computational overhead of retraining, making it a practical choice for dynamic and evolving environments.

Decision Tree. It is a supervised learning algorithm whose classification process occurs through a hierarchical tree structure [14]. The tree is built starting from a labeled training dataset, recursively partitioning the data based on feature values to maximize the class purity within each resulting subset. In this structure, internal (non-leaf) nodes represent decision points where the data are partitioned based on specific feature thresholds, while leaf nodes correspond to class labels. Given a test instance a, the algorithm traverses the tree from the root to a leaf node following the decision rules at each node, determined by the feature values of a. The final prediction corresponds to the class label of the reached leaf node. Although they exhibit lower performance than other approaches, we also included DT, as they are interpretable and can assist operators in decision-making.

Random Forest. An ensemble-based machine learning algorithm integrates bagging with random feature selection to improve predictive performance and robustness [17]. It works by building a set of decision trees, each trained on a random subset of the training set obtained by sampling with repetition. To further increase the model diversity and reduce the correlation between trees, only a randomly selected subset of features is considered at each split. This strategy mitigates overfitting and improves generalization. The final prediction is obtained by either majority voting in the case of classification problems or by averaging the predictions in the case of regression problems. Aggregating the results of many trees reduces the variance of the overall model, improving the ability to generalize to unseen data. Combining different weak models, RF allows for achieving more effective performances than a single model, hence, in our scenario, it can be used when a limited number of attack examples are available.

Gradient Boosting. An ensemble learning technique builds a predictive model by sequentially combining multiple weak models, typically decision trees [8]. The idea of gradient boosting is to train each new model to correct the residual errors made by the previous one. At each iteration, the algorithm fits a new tree to the negative gradient of the loss function with respect to the predictions of the current model. The final prediction is obtained by summing the weighted outputs of all individual models. It was selected in this study because it tends to be more resistant to overfitting than individual decision trees, especially when regularization techniques are applied (*e.g.*, subsampling or limiting tree depth).

5 Experimental Setup

This section describes the experimental setup employed in this paper. Section 5.1 describes the pipeline used to prepare data for the model learning phase. Section 5.2 presents the metrics used for the evaluation of our approach.

Regarding the classification task of attacks, we built the machine learning algorithms by relying on the implementations provided by the scikit-learn library [20]. Since the dataset has missing values, we applied a mean imputation strategy using the *SimpleImputer* of scikit-learn to handle them. This step was integrated into a pipeline to impute missing values before using the classifiers. Furthermore, KNN was configured with 5 neighbors, while the DT, RF, and GB models were used with the default number of estimators.

5.1 Data Preparation

For the experimentation, we consider the dataset described in Sect. 3 and the four models introduced in the Sect. 4. In detail, to feed the model during the learning phase, we aggregated features contained in the dataset described in Sect. 3. Then, we removed columns with constant values since they are not representative for the learning phase. We removed features from monitoring services – *i.e.*, *grafana*, *prometheus*, and *nacos*. We filtered columns with less than 80% of values different from NaN, and for the remaining columns we filled NaN values with the mean value computed on the column's values. Finally, we normalized all the values using *Min-Max* according to Eq. 1:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

where x represents the original value, x_{min} and x_{max} are the minimum and maximum value of the considered column, respectively.

5.2 Experimental Metrics

To evaluate the performance of the machine learning models, we consider standard classification metrics derived from the confusion matrix: Accuracy (Acc), Precision (Prec), Recall (Rec), and F1-score. These are defined as follows:

– *Acc* measures the proportion of correctly classified instances:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}; \tag{2}$$

– *Prec* measures the proportion of true positive predictions among all positive predictions:

$$Prec = \frac{TP}{TP + FP}; \tag{3}$$

– *Rec* measures the proportion of actual positives that were correctly predicted:

$$Rec = \frac{TP}{TP + FN}; \tag{4}$$

– *F1* is the harmonic mean of Precision and Recall:

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec}. \tag{5}$$

Here, *TP*, *TN*, *FP*, and *FN* represent the number of true positives, true negatives, false positives, and false negatives, respectively. A positive corresponds to a sample classified as a DDoS attack, while a negative is a sample classified as normal behavior.

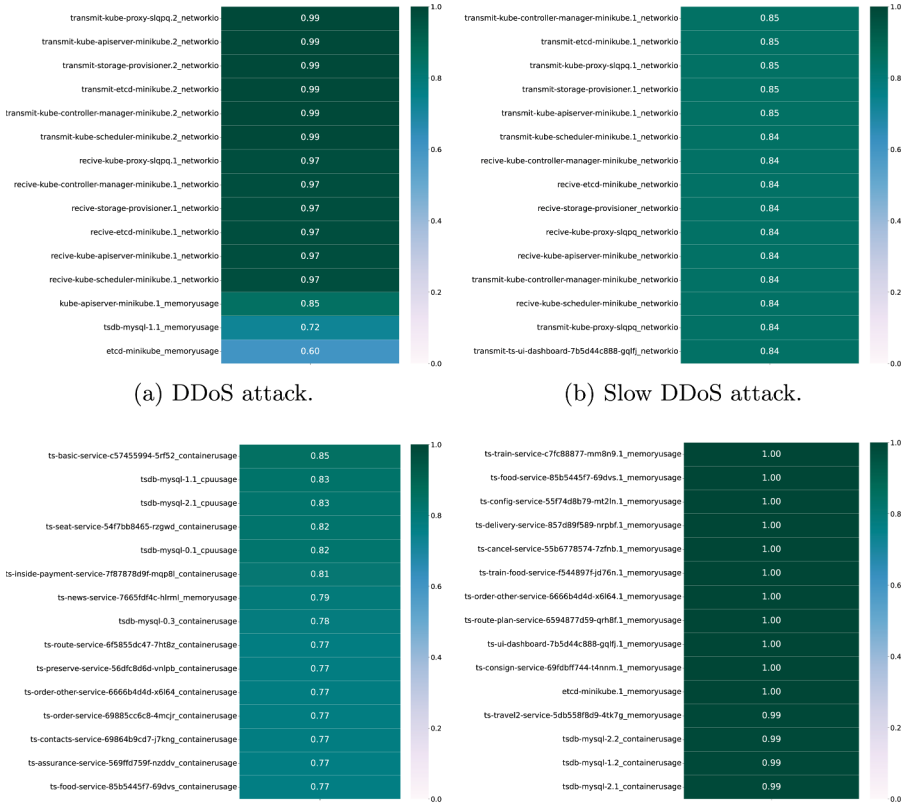
6 Numerical Results

Our results demonstrate that the AI-based IDA efficiently detects and classifies DDoS attacks in microservice environments. Among the evaluated models, Gradient Boosting consistently outperforms other algorithms across varying sample sizes, achieving an accuracy of 1.000 and F1-score of 1.000 with only 2–4% of the original dataset. Random Forest also achieves perfect performance at these sample levels. Even with only 0.25% of the sample, Gradient Boosting reaches an 83% accuracy, significantly outperforming K-Nearest Neighbors (KNN), which yields only 46.2% accuracy and an F1-score of 0.330.

Table 1 outlines the evolution of model performance as a function of the data sample size. The Decision Tree model crosses 90% accuracy with just 1% of data, while Random Forest and Gradient Boosting exceed 99% accuracy at the same threshold. KNN, while simpler and more computationally efficient, shows limited performance, requiring at least 2% of the data to exceed 95% accuracy.

To understand the internal decision-making process of the models, especially for explainability, we conducted a feature correlation analysis using Random Forest as the reference. Figure 2 illustrates the top 15 features correlated with each of the four DDoS attack types:

– *Standard DDoS Attacks:* Figure 2a shows the most influential features are memory usage metrics, particularly for core services such as `kube-apiserver`, `tsdb-mysql`, and `ts-train-service`, with correlation values ranging from 0.47 to 0.85. This suggests that memory exhaustion is a distinguishing indicator of standard DDoS behavior.
– *Slow DDoS Attacks:* These are primarily characterized by consistent and high correlation in both transmitted and received network I/O across key system containers (*e.g.*, `kube-*` and `etcd`), all showing correlation values between 0.84 and 0.85 (see, Fig. 2b). This uniformity reflects the subtle but sustained pressure slow attacks place on network interfaces.
– *GET Flood Attacks:* These attacks show high correlation with container usage and CPU usage in database-intensive and user-facing services, such as `ts-basic-service`, `tsdb-mysql`, and `ts-seat-service`. Correlation values are typically around 0.77 to 0.85 (see, Fig. 2c).



(a) DDoS attack. (b) Slow DDoS attack.

(c) GET flood attack. (d) SYN flood attack.

Fig. 2. Top-15 correlated features for the four attacks considered. Each feature name is fully reported.

– *SYN Flood Attacks:* The most correlated features – up to 1.00 – are memory usage metrics for frontend and service-routing components, including `ts-ui-dashboard`, `ts-delivery-service`, and `ts-route-plan-service`. Container and pod readiness metrics (*e.g.*, `rabbitmq_podready`) also contribute significantly, indicating the impact of SYN floods on service availability and startup (see, Fig. 2d).

This correlation analysis supports the effectiveness of feature selection and model design by revealing how different types of DDoS attacks affect system behavior differently. Notably, the IDA maintains robustness even under severe data scarcity, making it suitable for real-world deployments with a constrained monitoring infrastructure.

Figure 3 further confirms these findings by comparing F1-scores across algorithms and sample sizes. The steep performance improvement, especially between

Table 1. Performance metrics of different machine learning algorithms across varying sample percentages.

Sample pcg	Algorithm	Performance metrics			
		Acc	Prec	Rec	F1
0.25%	K-Nearest Neighbors	0.462	0.283	0.462	0.330
	Decision Tree	0.699	0.746	0.699	0.704
	Random Forest	0.769	0.828	0.769	0.742
	Gradient Boosting	**0.829**	**0.858**	**0.829**	**0.826**
0.5%	K-Nearest Neighbors	0.610	0.714	0.610	0.569
	Decision Tree	0.790	0.870	0.790	0.803
	Random Forest	0.903	0.915	0.903	0.902
	Gradient Boosting	**0.988**	**0.989**	**0.988**	**0.988**
1%	K-Nearest Neighbors	0.850	0.862	0.850	0.852
	Decision Tree	0.923	0.945	0.923	0.925
	Random Forest	0.996	0.996	0.996	0.996
	Gradient Boosting	**0.999**	**0.999**	**0.999**	**0.999**
2%	K-Nearest Neighbors	0.951	0.955	0.951	0.950
	Decision Tree	0.929	0.942	0.929	0.929
	Random Forest	**1.000**	**1.000**	**1.000**	**1.000**
	Gradient Boosting	0.999	0.999	0.999	0.999
4%	K-Nearest Neighbors	0.976	0.976	0.976	0.976
	Decision Tree	0.991	0.991	0.991	0.991
	Random Forest	**1.000**	**1.000**	**1.000**	**1.000**
	Gradient Boosting	**1.000**	**1.000**	**1.000**	**1.000**

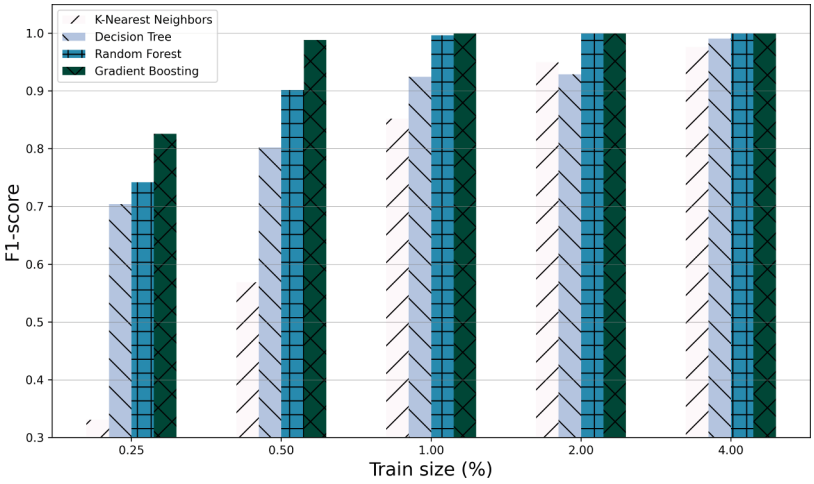


Fig. 3. A comparison of the F1-scores for the machine learning algorithms across different training dimensions. Each bar in the chart represents the performance of a specific algorithm within each training dimension.

0.5% and 1% of the training data, highlights a practical threshold for efficient model training in constrained environments.

7 Conclusions

In this paper, we presented an IDA mechanism based on AI for improving the security posture of modern MSAs. Specifically, we used four different supervised approaches to classify various DDoS attacks, *i.e.*, Standard DDoS, Slow DDoS, GET floods, and SYN floods. As shown, results obtained in a realistic testbed built on top the Train Ticker platform demonstrated the effectiveness of our approach. As an example, with an IDA leveraging a decision tree, we achieve an accuracy of $\sim$0.991 by considering only 4% of measurements capturing the evolution of containers.

Future work aims at refining the approach. For instance, a relevant part of our ongoing research is devoted to investigate mechanisms based on transfer learning, especially to "port" our approach to other similar scenarios. Furthermore, we are also working towards the development of a small prototype for conducting measurements. Specifically, we are interested in quantifying the resources needed to run the IA-capable IDA and the overheads caused to the monitoring infrastructure for allowing the detection.

Acknowledgments. This work was partially supported by Project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

1. Bremler-Barr, A., Czeizler, M., Levy, H., Tavori, J.: Exploiting miscoordination of microservices in tandem for effective DDoS attacks. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications, pp. 231–240. IEEE (2024)
2. Cassagnes, C., Trestioreanu, L., Joly, C., State, R.: The rise of eBPF for non-intrusive performance monitoring. In: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–7. IEEE (2020)
3. Castro, J., Laranjeiro, N., Vieira, M.: Detecting dos attacks in microservice applications: approach and case study. In: Proceedings of the 11th Latin-American Symposium on Dependable Computing, pp. 73–78 (2022)
4. De Lauretis, L.: From monolithic architecture to microservices architecture. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 93–96. IEEE (2019)
5. Douiba, M., Benkirane, S., Guezzaz, A., Azrour, M.: An improved anomaly detection model for IoT security using decision tree and gradient boosting. J. Supercomput. **79**(3), 3392–3411 (2023)
6. Ficco, M., et al.: A benchmark for DDoS attacks detection in microservice architectures. In: ICNC 2025 (2025)
7. Flora, J., Antunes, N.: Evaluating intrusion detection for microservice applications: benchmark, dataset, and case studies. J. Syst. Softw. **216**, 112142 (2024)
8. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
9. Giamattei, L., et al.: Monitoring tools for DevOps and microservices: a systematic grey literature review. J. Syst. Softw. **208**, 111906 (2024)

10. Haq, M.S., Nguyen, T.D., Tosun, A.Ş., Vollmer, F., Korkmaz, T., Sadeghi, A.R.: SoK: a comprehensive analysis and evaluation of docker container attack and defense mechanisms. In: 2024 IEEE Symposium on Security and Privacy (SP), pp. 4573–4590. IEEE (2024)
11. Jayalath, R.K., Ahmad, H., Goel, D., Syed, M.S., Ullah, F.: Microservice vulnerability analysis: a literature review with empirical insights. IEEE Access (2024)
12. Karn, R.R., Kudva, P., Huang, H., Suneja, S., Elfadel, I.M.: Cryptomining detection in container clouds using system calls and explainable machine learning. IEEE Trans. Parallel Distrib. Syst. **32**(3), 674–691 (2020)
13. Kim, Y., Park, C., Hong, D.: A robust framework for comprehensive container image vulnerability assessment. IEEE Access (2025)
14. Kingsford, C., Salzberg, S.L.: What are decision trees? Nat. Biotechnol. **26**(9), 1011–1013 (2008)
15. Kommula, A., Hu, Y.H.F., Hoppa, M.A., Olatunbosun, S.: Machine learning techniques to enhance container network security. In: 2020 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 622–627. IEEE (2020)
16. Leppänen, T., Honkaranta, A., Costin, A.: Trends for the DevOps security. A systematic literature review. In: International Symposium on Business Modeling and Software Design, pp. 200–217. Springer (2022)
17. Liu, Y., Wang, Y., Zhang, J.: New machine learning algorithm: random forest. In: Liu, B., Ma, M., Chang, J. (eds.) ICICA 2012. LNCS, vol. 7473, pp. 246–252. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34062-8_32
18. Marimuthu, S.: Detecting application layer DDoS attacks among microservices using spatio-temporal graph convolution networks. Ph.D. thesis, California State University, Northridge (2023)
19. Miano, S., Risso, F., Bernal, M.V., Bertrone, M., Lu, Y.: A framework for eBPF-based network functions in an era of microservices. IEEE Trans. Netw. Serv. Manage. **18**(1), 133–151 (2021)
20. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
21. Pereira-Vale, A., Fernandez, E.B., Monge, R., Astudillo, H., Márquez, G.: Security in microservice-based systems: a multivocal literature review. Comput. Secur. **103**, 102200 (2021)
22. Perez-Diaz, J.A., Valdovinos, I.A., Choo, K., Zhu, D.: A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access **8**, 155859–155872 (2020)
23. Peterson, L.E.: K-nearest neighbor. Scholarpedia **4**(2), 1883 (2009)
24. Primartha, R., Tama, B.A.: Anomaly detection using random forest: a performance revisited. In: 2017 International Conference on Data and Software Engineering (ICoDSE), pp. 1–6 (2017). https://doi.org/10.1109/ICODSE.2017.8285847
25. Saenger, J., Mazurczyk, W., Keller, J., Caviglione, L.: VoIP network covert channels to enhance privacy and information sharing. Futur. Gener. Comput. Syst. **111**, 96–106 (2020)
26. Singleton, A.: The economics of microservices. IEEE Cloud Comput. **3**(5), 16–20 (2016)
27. Steidl, M., Leitner, M., Urbanke, P., Gattringer, M., Felderer, M., Ristov, S.: Understanding microservice runtime monitoring data for anomaly detection with structural equation modeling. In: International Conference on Product-Focused Software Process Improvement, pp. 404–413. Springer (2024)

28. Tien, C.W., Huang, T.Y., Tien, C.W., Huang, T.C., Kuo, S.Y.: KubAnomaly: anomaly detection for the docker orchestration platform with neural network approaches. Eng. Rep. **1**(5), e12080 (2019)
29. VS, D.P., Sethuraman, S.C., Khan, M.K.: Container security: precaution levels, mitigation strategies, and research perspectives. Comput. Secur. **135**, 103490 (2023)
30. Ying, S., et al.: An improved KNN-based efficient log anomaly detection method with automatically labeled samples. ACM Trans. Knowl. Discov. Data **15**(3) (2021). https://doi.org/10.1145/3441448
31. Zhou, X., et al.: Benchmarking microservice systems for software engineering research. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, pp. 323–324 (2018)
32. Zuppelli, M., Guarascio, M., Caviglione, L., Liguori, A.: No country for leaking containers: detecting exfiltration of secrets through AI and Syscalls. In: Proceedings of the 19th International Conference on Availability, Reliability and Security, pp. 1–8 (2024)
33. Zuppelli, M., Repetto, M., Schaffhauser, A., Mazurczyk, W., Caviglione, L.: Code layering for the detection of network covert channels in agentless systems. IEEE Trans. Netw. Serv. Manage. **19**(3), 2282–2294 (2022)