Computers & Security 136 (2024) 103546

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Apollon: A robust defense system against Adversarial Machine Learning attacks in Intrusion Detection Systems

Antonio Paya [a,*], Sergio Arroni [a], Vicente García-Díaz [a], Alberto Gómez [b]

[a] Department of Computer Science, University of Oviedo, Science Faculty, Oviedo, Spain
[b] Department of Business Administration, University of Oviedo, Gijón, Spain

ARTICLE INFO

Keywords:
Adversarial Machine Learning
Intrusion Detection Systems
Artificial Intelligence
Cybersecurity
Multi-Armed Bandits

ABSTRACT

The rise of Adversarial Machine Learning (AML) attacks is presenting a significant challenge to Intrusion Detection Systems (IDS) and their ability to detect threats. To address this issue, we introduce *Apollon*, a novel defense system that can protect IDS against AML attacks. *Apollon* utilizes a diverse set of classifiers to identify intrusions and employs *Multi-Armed Bandits (MAB)* with *Thompson sampling* to dynamically select the optimal classifier or ensemble of classifiers for each input. This approach enables *Apollon* to prevent attackers from learning the IDS behavior and generating adversarial examples that can evade the IDS detection. We evaluate *Apollon* on several of the most popular and recent datasets, and show that it can successfully detect attacks without compromising its performance on traditional network traffic. Our results suggest that *Apollon* is a robust defense system against AML attacks in IDS.

1. Introduction

As computer networks continue to grow in size and complexity, the need for effective security measures becomes increasingly critical. Intrusion Detection Systems (IDS) have been developed to address this challenge, providing a means of monitoring network traffic and identifying potential security threats. These systems can analyze network traffic and identify potential security threats such as malware, network intrusions, and denial of service attacks. However, the increasing complexity and diversity of network traffic have made it difficult to accurately classify network traffic using traditional rule-based IDS systems (Thakkar and Lohiya, 2020).

To overcome these limitations, Machine Learning (ML) techniques have been widely adopted in IDS for network traffic classification. These techniques leverage the power of statistical models and algorithms to automatically learn and detect anomalous network traffic patterns, which are indicative of security threats.

ML-based IDS systems offer several advantages over traditional rule-based systems, including higher accuracy, better scalability, and more robustness to evolving network threats (Abdallah et al., 2022; Maseer et al., 2021). However, they also pose new challenges, particularly in terms of security. One of the main challenges is the susceptibility of ML models to Adversarial Machine Learning (AML) attacks (Huang et al., 2011).

AML attacks are a type of cyber-attack that aims to manipulate ML models by feeding them carefully crafted input data, called adversarial examples. These examples are designed to cause misclassification or incorrect predictions, which can be exploited by attackers to bypass the security measures of IDS systems (Zhao et al., 2021; Lin et al., 2022; Liu et al., 2022).

Attackers create adversarial examples by utilizing information obtained from the targeted IDS, including its responses to specific inputs. This information is used to train a model capable of generating adversarial traffic that remains undetectable by the IDS classifier.

As ML-based IDS systems become more prevalent, the threat of Adversarial Machine Learning (AML) attacks becomes more significant (Duy et al., 2021). Therefore, it is crucial to develop effective defense mechanisms to mitigate the impact of these attacks and ensure the reliability and robustness of IDS systems.

In this paper, we propose a new robust defense system against Adversarial Machine Learning attacks on Intrusion Detection Systems called *Apollon*. *Apollon* serves to safeguard IDS from attackers by obstructing their ability to generate adversarial traffic through learning from the behavior of the IDS.

* Corresponding author.
E-mail addresses: antoniopaya@outlook.com (A. Paya), sergioarroni@outlook.com (S. Arroni), garciavicente@uniovi.es (V. García-Díaz), albertogomez@uniovi.es (A. Gómez).

https://doi.org/10.1016/j.cose.2023.103546
Received 30 March 2023; Received in revised form 24 September 2023; Accepted 17 October 2023
Available online 20 October 2023

Apollon utilizes a diverse range of classifiers to detect intrusions and employs a *Multi-Armed Bandits (MAB)* with *Thompson sampling* to select the optimal classifier or a combination of classifiers in real-time for each input, enabling it to achieve this objective without compromising its performance on traditional network traffic.

In this way, *Apollon* can prevent attackers from learning the behavior of the IDS in realistic training times, adding a layer of uncertainty to the IDS behavior that makes it more difficult for attackers to detect the IDS behavior and generate adversarial traffic.

The structure of the paper is as follows. Section 2 provides an overview of the main concepts and techniques used in this paper. Section 3 discusses the related work in the field of AML attacks and IDS classifiers. Section 4 presents the proposed defense system, *Apollon*. Section 5 presents the experimental evaluation of *Apollon*. Finally, Section 6 concludes the paper and discusses future work.

2. Background

In this section, we will provide an overview of the background knowledge required to understand the rest of this work. This includes an introduction to the concepts of Adversarial Machine Learning and adversarial examples, as well as an introduction to the concepts of Intrusion Detection Systems and the challenges they face.

2.1. Intrusion Detection Systems

Intrusion Detection Systems (IDS) are security tools designed to identify and prevent unauthorized access, misuse, and malicious activities in computer networks (Mukherjee et al., 1994). IDS play a critical role in protecting networks from various types of cyber threats, including viruses, malware, and intrusions. IDS operate by monitoring network traffic and analyzing it for suspicious behavior or patterns. There are two main types of IDS: Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS) (Pharate et al., 2015).

NIDS monitors network traffic and analyzes packets to identify potential security threats. It can detect a wide range of network-based attacks, such as port scans, denial-of-service attacks, and data exfiltration. NIDS can be deployed at various points within the network, such as at the perimeter, within the LAN, or at critical junctions within the network.

On the other hand, HIDS monitors the activity on individual hosts, such as servers or workstations. While it can also analyze network traffic specific to the host, its unique strength lies in its ability to inspect system-specific activities, including file system modifications and system call behaviors. This makes HIDS particularly suitable for identifying and detecting malware infections, as these often manifest in changes at the host level. Furthermore, HIDS can detect attacks that may not be visible to NIDS, such as attacks that occur within encrypted traffic or those that originate from within the network.

From this point on, and to facilitate the understanding of the document, we will refer to Network Intrusion Detection Systems as Intrusion Detection Systems.

Intrusion Detection Systems are an essential component of a comprehensive network security strategy. They provide an additional layer of protection beyond firewalls, antivirus software, and other security tools. By detecting and alerting administrators to potential security threats, IDS can help organizations respond quickly and effectively to cyber attacks.

Machine Learning (ML) has emerged as a powerful technique for improving the accuracy and effectiveness of Intrusion Detection Systems (Abdallah et al., 2022; Maseer et al., 2021; Thakkar and Lohiya, 2020). ML algorithms can be used to analyze large volumes of network data and identify patterns that may be indicative of security threats. ML-based IDS can learn from past network activity to identify and flag potential security threats in real-time, even when the attacks are novel or previously unseen. ML-based IDS can also adapt and improve over time as they learn from new data and feedback from security analysts.

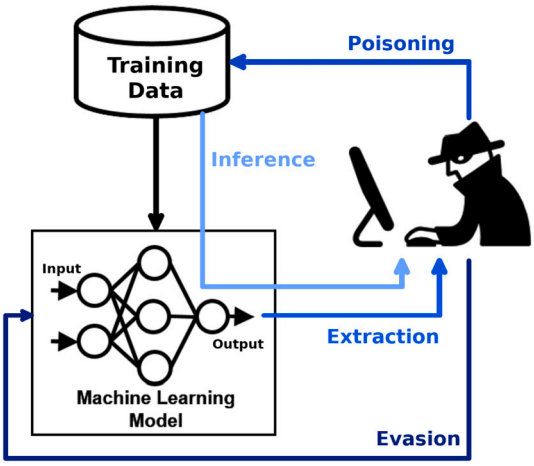


Fig. 1. Adversarial Machine Learning (AML) attacks by ART (Nicolae et al., 2018).

2.2. Adversarial Machine Learning

Adversarial Machine Learning (AML) attacks refer to a set of techniques used to undermine the accuracy, integrity, or security of machine learning (ML) models (Huang et al., 2011). AML attacks can be launched by malicious actors with different objectives, such as stealing sensitive information, manipulating decision-making processes, or compromising the confidentiality and privacy of ML systems (see Fig. 1).

AML attacks can be launched against a wide range of ML models, including deep neural networks, support vector machines, decision trees, and others. The success of an AML attack depends on various factors, such as the type and quality of the target ML model, the sophistication of the attack technique, and the attacker's level of knowledge and resources. According to the taxonomy of the attack, they can be classified into evasion attacks, poisoning attacks, extraction attacks and inference attacks (De Cristofaro, 2020).

2.2.1. Evasion attacks

Evasion attacks in AML refer to a type of attack where the attacker manipulates the input data in a way that the ML model will misclassify it, without changing the underlying characteristics of the data (Biggio et al., 2013). Evasion attacks are typically launched against classification models, such as those used for image recognition or spam detection, and they can be crafted using various techniques, including gradient-based methods, evolutionary algorithms, or gray/black-box attacks. The goal of an evasion attack is to create an adversarial example, i.e., a modified version of the original input data that is similar to the original but is misclassified by the ML model. Evasion attacks pose a significant threat to the security and robustness of ML systems, especially in domains such as malware detection, Intrusion Detection Systems and fraud detection, where accurate classification is critical.

2.2.2. Poisoning attacks

Data poisoning attacks in AML involve manipulating the training data of an ML model to introduce biases or to cause it to learn incorrect patterns (Koh, 2018). Poisoning attacks can be launched at different stages of the ML pipeline, including data collection, preprocessing, and training. The goal of a poisoning attack is to compromise the integrity and accuracy of the ML model by introducing malicious data into the training dataset, which can cause the model to learn incorrect patterns and make incorrect predictions. Poisoning attacks can be launched in a variety of ways, such as by injecting adversarial examples into the

training data, manipulating the distribution of the training data, or introducing outliers into the dataset.

2.2.3. Extraction attacks

Model extraction attacks in AML refer to a type of attack where the attacker aims to extract the details of an ML model without direct access to it (Chen, 2020). This is achieved by leveraging the output of the target ML model to infer the underlying structure, architecture, or parameters of the model. Model extraction attacks can be launched through different channels, such as querying the model with carefully crafted inputs or by observing its behavior in response to various inputs. The goal of a model extraction attack is to steal the target model's intellectual property or use it for malicious purposes such as deploying counterfeit models, stealing sensitive data, or reverse engineering proprietary algorithms. Model extraction attacks can be particularly effective against black-box models where the attacker does not have access to the model's internal structure or parameters.

2.2.4. Inference attacks

AML inference attacks involve an attacker attempting to glean confidential information about the input data utilized by the ML model by scrutinizing the model's output (Yeom, 2017). Inference attacks can be launched against a wide range of ML models, including deep neural networks, decision trees, and support vector machines, among others. The goal of an inference attack is to obtain access to private or confidential information about the input data, such as personal characteristics, financial transactions, or medical records, without having direct access to the data itself. Inference attacks can be launched through different channels, such as analyzing the output distribution of the model, measuring its response time to different inputs, or exploiting the model's decision boundaries.

2.3. Multi-Armed Bandits (MAB)

The *Multi-Armed Bandits (MAB)* is a classic problem in probability theory and Machine Learning, where an agent has to allocate a limited set of resources among competing choices that have uncertain rewards (Kuleshov and Precup, 2014). The agent faces a trade-off between exploiting the choices that have the highest expected rewards based on the current information, and exploring new choices that may yield higher rewards in the future.

The *MAB* problem has many practical applications in various domains, such as clinical trials, adaptive routing, financial portfolio design, and online advertising. Several algorithms have been proposed to solve the *MAB* problem, such as optimistic initialization (Machado et al., 2014), upper confidence bound (UCB) (Carpentier et al., 2011), and Thompson sampling (Agrawal and Goyal, 2012). These algorithms differ in how they balance exploration and exploitation, and how they estimate the expected rewards of each choice.

Thompson sampling is a Bayesian approach that maintains a probability distribution over the unknown reward distributions of each choice, and chooses actions based on sampling from these distributions. Specifically, at each timestep, Thompson sampling samples a reward from each distribution, chooses the action associated with the highest sampled reward, and updates its beliefs about the reward distributions based on the observed reward. This approach has been shown to be effective in many applications, and has a strong theoretical justification in terms of minimizing regret.

Thompson sampling has gained popularity in recent years due to its ability to balance exploration and exploitation in a principled way (Park and Faradonbeh, 2021). By sampling from the probability distributions over the reward distributions, Thompson sampling encourages exploration of all choices while still favoring choices with higher expected rewards. Additionally, the Bayesian framework allows for the incorporation of prior knowledge about the reward distributions, which can be especially useful in scenarios with limited data.

The *MAB* problem is intricately connected to the realm of Reinforcement Learning (RL). In RL, an intelligent agent endeavors to acquire knowledge and develop a strategy, known as a policy, that maximizes its total rewards throughout its interaction with an environment. Over the past few years, RL has exhibited remarkable success in diverse domains. Notably, it has found significant utility in the domain of demand forecasting (Ramos et al., 2022a,b).

In the context of our work, we use the *MAB* algorithm to select the best (IDS) classifier for each network traffic request. This is similar to how *MAB* is used in demand forecasting to select the most optimal forecasting model or determine the best hyperparameters for a given forecasting model. By using *MAB*, we can balance the trade-off between exploiting the classifiers that have the highest expected accuracy based on the current information, and exploring new classifiers that may yield higher accuracy in the future.

3. Related work

This section will present a summary of the distinct datasets for IDS, along with their corresponding classifiers and performance metrics. Our proposed approach will make use of these classifiers. Furthermore, we will explore the typical types of AML attacks used in this domain.

3.1. Intrusion Detection Systems datasets

IDS datasets play a crucial role in assessing and gauging the performance of IDSs. These datasets contain labeled instances of regular and anomalous network traffic that are used to train and assess the precision and efficiency of IDSs. A range of datasets with diverse strengths and features is accessible. This section will examine some of the most prevalent datasets, highlighting their essential qualities and applications.

3.1.1. CIC-IDS-2017

A highly utilized IDS dataset in contemporary literature is the CIC-IDS2017 (Sharafaldin et al., 2018a), which was developed by the Canadian Institute for Cybersecurity (CIC) in a simulated enterprise network environment, gathering network traffic data for five consecutive days. This dataset emulates the actions of 25 users and comprises nearly 80 significant attributes (Ring et al., 2019). Notably, it has an 83% to 17% benign to malicious instance ratio, representing a significant portion of the dataset. The CIC-IDS2017 is considered an accurate depiction of normal traffic distribution in a network and can be utilized individually or combined with other datasets (Shroff et al., 2022).

3.1.2. CSE-CIC-IDS-2018

The CSE-CIC-IDS2018 dataset was developed using AWS resources in a simulated enterprise network environment in 2018 (Sharafaldin et al., 2018b). It consists of data on seven distinct attack categories and comprises nearly 79 important features. With over 450 devices, including servers, computers, and other tools, this dataset is notably large and realistic (Pujari et al., 2022). It is akin to the CIC-IDS2017 dataset, analyzing bidirectional flow packet data, but with more significant features and greater comprehensiveness. Hence, it is widely used in the literature for assessing and benchmarking IDSs (Pujari et al., 2022).

3.1.3. CIC-DDoS-2019

To address the lack of representation of all DDoS (Distributed Denial of Service) attack subtypes in existing datasets, the CIC-DDoS-2019 dataset was created (Sharafaldin et al., 2019). Although the dataset includes simulated network traffic, it strives to present realistic benign data. It features 13 types of DDoS attacks and over 80 significant features. However, it is severely imbalanced, with 50,006,249 DDoS attack records and just 56,863 benign traffic records, making it challenging to train a model on both data types (Ring et al., 2019). As a result, experts suggest using this dataset in conjunction with other datasets (Shroff et al., 2022), such as CIC-IDS-2017 or CSE-CIC-IDS-2018, to train a more robust model.

Table 1
Performance of the *IDSs* classifiers on the selected datasets.

Classifiers	CIC-IDS-2017			CSE-CIC-IDS-2018			CIC-DDoS-2019			References
	Accuracy	F1 Score	AUC	Accuracy	F1 Score	AUC	Accuracy	F1 Score	AUC	
LR	92.96	90.87	91.50	87.96	88.99	81.54	91.72	87.27	90.23	Thiyam and Dey (2023); Akshay Kumaar et al. (2022)
FNN	99.61	99.57	99.83	93.00	92.00	100.00	95.55	95.50	95.63	Huang and Lei (2020); Thiyam and Dey (2023); Wu et al. (2022)
RF	99.79	99.78	99.98	92.00	94.00	100.00	99.86	99.78	99.82	Pujari et al. (2022); Huang and Lei (2020); Abdulhammed et al. (2019); Maseer et al. (2021); Faker and Dogdu (2019); Thiyam and Dey (2023)
DT	99.62	99.57	99.56	88.00	91.00	100.00	99.87	99.78	99.80	Pujari et al. (2022); Thiyam and Dey (2023); Huang and Lei (2020); Maseer et al. (2021)
RTIDS	99.35	99.17	98.83	-	-	-	98.58	98.48	98.66	Wu et al. (2022)
SVM	96.97	96.99	98.98	61.00	66.00	100.00	94.02	94.98	94.24	Pujari et al. (2022); Huang and Lei (2020); Maseer et al. (2021); Faker and Dogdu (2019); Wu et al. (2022); Sahoo et al. (2020)

3.1.4. Discarded datasets

This project did not make use of several other IDS datasets, including Darpa 1998/99 (Mahoney and Chan, 2003), KDD 99 (Tavallaee et al., 2009), and NSL-KDD (Tavallaee et al., 2009). These datasets are no longer commonly employed for evaluating and benchmarking IDS due to their outdated nature. Created in the late 1990s and early 2000s, they do not accurately represent the current landscape of network threats and behaviors. KDD 99 dataset, in particular, has been criticized for its high false positive rate and lack of realism, thereby limiting its usefulness in assessing the performance of modern IDS (Hugh, 2000; Tobi and Duncan, 2018).

3.2. Intrusion Detection Systems ML-classifiers

ML-classifiers have emerged as a promising alternative to traditional IDS for detecting network attacks. This is due to the limitations of traditional IDS in dealing with the complex and dynamic nature of cyber-attacks. In the current digital era, the number and sophistication of malware threats are constantly growing, posing a serious challenge to network security. Therefore, it is essential to have reliable and effective IDS systems in place to protect network systems from potential damage.

Several studies have been conducted to evaluate the performance of various ML-classifiers in detecting network attacks. These studies use datasets such as CIC-IDS-2017, CSE-CIC-IDS-2018, and CIC-DDoS-2019.

Table 1 provides a summary of the most commonly used ML-classifiers and their scores, including accuracy, F1 Score, and AUC, for each of the aforementioned datasets. The ML-classifiers used in these studies are *Logistic Regression (LR)* (Wright, 1995), *Fuzziness based Neural Networks (FNN)* (Ashfaq et al., 2017), *Random Forests (RF)* (Cutler et al., 2012), *Decision Trees (DT)* (Rokach and Maimon, 2005), *Robust transformer based Intrusion Detection System (RTIDS)* (Wu et al., 2022), and *Support Vector Machines (SVM)* (Suthaharan and Suthaharan, 2016).

These classifiers were selected due to their wide adoption and proven effectiveness in diverse Machine Learning applications. Nonetheless, research continues with the development of new and enhanced classifiers, such as LSTM-FCNN (Sahu et al., 2022) or DCNNBiLSTM (Hnamte and Hussain, 2023), further extending the landscape of ML-IDS. By examining the results presented in Table 1, researchers can gain a comprehensive understanding of the performance of these classifiers on the specific datasets, enabling informed decisions when choosing the most appropriate algorithm for their requirements. The detailed explanations of how each of the results were obtained for each classifier can be found in the papers listed in the *References* column. These papers provide comprehensive insights into the methodologies employed and offer further analysis of the performance of each ML-classifier on the respective datasets.

Accuracy, F1 Score, and AUC are three common metrics used to evaluate the performance of machine learning models. Accuracy measures the proportion of correct predictions made by a model out of the total number of predictions and is defined in Equation (1),

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

In the context of IDS, a true positive is defined as a malicious network flow that is correctly identified as malicious by the IDS. Conversely, a true negative refers to normal network traffic that is correctly identified as non-malicious. It's worth noting that in many real-world scenarios, the volume of normal traffic significantly outweighs malicious traffic. As a result, if the positive class in our metrics refers to normal traffic, the values would naturally be higher due to the prevalence of normal traffic. However, in our evaluations, the positive class specifically denotes malicious traffic, ensuring that our metrics provide a balanced and accurate representation of the IDS's performance.

F1 Score is a weighted average of precision and the detection rate (DR) (also known as recall or sensitivity on traditional ML literature), where precision measures the proportion of true positives out of all predicted positives, and the detection rate measures the proportion of true positives out of all actual positives. The F1 Score is defined in Equation (2).

$$F1Score = 2 \cdot \frac{precision \cdot DR}{precision + DR} \tag{2}$$

where precision is defined in Equation (3) and DR is defined in Equation (4).

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$DR = \frac{TP}{TP + FN} \tag{4}$$

The Area Under the ROC Curve (AUC) is a widely used performance metric in ML and binary classification tasks. It quantifies the discriminative power of a classification model by measuring the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. The ROC curve plots the true positive rate (DR) against the false positive rate (1-specificity) for various classification thresholds, where specificity is defined in Equation (5).

$$specificity = \frac{TN}{TN + FP} \tag{5}$$

The AUC represents the integral of this curve and ranges from 0 to 1, where a value of 1 indicates a perfect classifier and a value of 0.5 suggests a random or ineffective classifier. Higher AUC values indicate better model performance in distinguishing between positive and negative instances. The AUC is a popular evaluation metric as it is robust to class imbalance and provides a concise summary of the model' overall performance.

Among the classifiers in Table 1, *Random Forest* (Zhang et al., 2008) and *Decision Trees* (Amor et al., 2004) are found to be some of the most effective classifiers for detecting network attacks. *Random Forest* has gained popularity due to its ability to handle large datasets and its robust performance even when the data contains noise or missing values. *Decision Trees* are also preferred because of their simplicity and interpretability. They enable clear visualization of the decision-making process, making them useful for understanding the factors that contribute to the classification results.

Overall, these classifiers have demonstrated strong performance in the field of IDS and are frequently used by researchers and practitioners. Their effectiveness in detecting intrusions and classifying network traffic makes them valuable tools for maintaining the security and integrity of computer networks.

3.3. Adversarial Machine Learning attacks

ML-based IDSs can learn from data and adapt to new situations, unlike traditional systems that rely on predefined rules. However, ML-based IDSs also face new challenges from attackers who use Artificial Intelligence (AI) to craft sophisticated attacks that can fool or compromise ML models. One such threat comes from the use of Artificial Intelligence (AI) in the form of Adversarial Machine Learning (AML), where attackers use sophisticated techniques to manipulate or subvert ML models. These attacks are attractive to cyber attackers since they can be challenging to detect and prevent. Furthermore, as AI techniques gain popularity in cybersecurity, attackers are incentivized to develop more sophisticated adversarial attacks to evade detection.

Adversarial Machine Learning attacks can be classified as white-box attacks and gray/black-box attacks, depending on the level of knowledge the attacker possesses about the target model.

3.3.1. White-box attacks

White-box attacks are a powerful category of AML attacks that can pose challenges not only to IDS but to any Machine Learning model. Their potency derives from the complete knowledge they assume about the target model and training data. This allows the attacker to craft intricate attacks that can evade system defenses.

Admittedly, such thorough knowledge about the system and its vulnerabilities is often impossible, rendering white-box attacks quite unrealistic in many practical scenarios. Thus, while white-box attacks are potent in theory, they are seldom observed in practice. Common white-box attack methods, applicable to ML models in general, and proven to be effective against IDS, include *Fast Gradient Sign Method (FGSM)* (Goodfellow et al., 2014a), *Deep-Fool* (Moosavi-Dezfooli et al., 2016), *Carlini &Wagner attack (C&W)* (Carlini and Wagner, 2017), *Jacobian based Saliency Map Attack (JSMA)* (Papernot et al., 2016), *Basic Iterative Method (BIM)* (Kurakin et al., 2016), and *Projected Gradient Descent (PGD)* (Madry et al., 2017).

3.3.2. Gray/black-box attacks

Gray/black-box attacks are comparatively more practical as they do not necessitate knowledge about the target model. Yet, the effectiveness of these attacks may be limited by the attacker's restricted knowledge, leading to more generic and potentially less effective techniques.

A prevalent tool used to mount such attacks is the Generative Adversarial Network (GAN) (Goodfellow et al., 2020), a machine learning model that can produce adversarial examples capable of evading detection systems. *GANs* are a type of Machine Learning model consisting of two neural networks: a generator network and a discriminator network. The generator network is trained to produce synthetic data that resembles the real data, while the discriminator network is trained to differentiate between real and synthetic data.

In the context of black-box and gray-box attacks, the generator network can generate adversarial examples that are specifically designed to evade the target system's defenses. The attacker may have access to the system's model scores or just a binary output indicating whether the input was accepted or rejected. This information can be utilized to guide the training of the generator network, enhancing its ability to produce effective adversarial examples.

Recent gray/black-box attacks proven to be effective against IDS include *attackGAN* (Zhao et al., 2021), *DIGFuPAS* (Duy et al., 2021), *IDSGAN* (Lin et al., 2022), *VulnerGAN* (Liu et al., 2022), *ZOO attack* (Chen et al., 2017), *Boundary attack* (Chen and Jordan, 2019) and the *HopSkipJump attack (HSJA)* (Chen et al., 2020). *ZOO* is a score-based attack that estimates gradients to create adversarial traffic in gray/black-box settings. *Boundary attack* and *HSJA* are decision-based attacks that only use binary feedback to craft adversarial inputs. The *IDSGAN*, *attackGAN*, and *DIGFuPAS* are gray/blackbox attacks that employ *Wasserstein-GAN* to generate adversarial traffic. *Wasserstein-GAN (W-GAN)* (Gulrajani et al., 2017) is a *GAN* variant that trains the generator network with a different objective function called the Wasserstein distance. The Wasserstein distance measures the distance between two probability distributions and has properties like smoothness and continuity. Some recent *WGANs* use the Gradient Penalty to enhance the training convergence.

3.4. Adversarial Machine Learning defenses

Due to the increasing number of AML attacks, researchers have developed several defense mechanisms to mitigate the impact of these attacks. In the IDS domain, these defenses can be classified into three categories (Alotaibi and Rassam, 2023): *preprocessing defenses*, *adversarial training defenses* and *adversarial detection defenses*.

It's crucial to briefly mention here the context of our proposed system, *Apollon*, in relation to these established defenses. Rather than replacing or competing with these mechanisms, *Apollon* works in tandem with them, dynamically selecting and optimizing the utilization of these unaltered, pre-existing models.

3.4.1. Preprocessing defenses

In order to mitigate the impact of adversarial perturbations, researchers have devised carefully planned preprocessing techniques. These preprocessing methods aim to reduce the vulnerability of machine learning models to adversarial attacks and enhance their robustness applying carefully planned transformations to the input data before it is fed into the model. These transformations are designed to reduce the vulnerability of the model to adversarial perturbations.

One example of a preprocessing defense technique is *Stochastic Transformation-based Defenses* (Kou et al., 2019). This technique involves applying random transformations to the input data, such as rotations, translations, and scaling, before feeding it into the model. By introducing randomness into the input data, the model becomes less susceptible to adversarial perturbations that are designed to exploit specific features of the input.

Another example of a preprocessing defense technique is *Gradient Masking* (Athalye et al., 2018). This technique involves modifying the gradients of the model during training to make it more difficult for an attacker to compute the gradients needed to generate adversarial examples. This is achieved by adding noise to the gradients or by clipping them to a certain range.

Nevertheless, sophisticated adversarial attacks have proven the inadequacy of these defense mechanisms. The primary shortcomings of these strategies are rooted in their approach: they tend to "confound" or confuse adversaries instead of outright eradicating the presence of adversarial examples (Xu et al., 2020). This means that while they might momentarily disrupt or delay an attacker, they don't provide a long-term solution or a foolproof barrier against these threats.

3.4.2. Adversarial training defenses

Adversarial training is a widely researched topic within the realm of visual computing. Goodfellow et al. (2014b) demonstrated that by retraining a neural network with a dataset comprising both original

and adversarial samples, the network's capability to counter adversarial samples can be markedly improved.

One of the salient features of adversarial training, as detailed in He et al. (2023), is its operational efficiency. Unlike some other defense mechanisms, adversarial training can function seamlessly without imposing any additional processing overhead during its operation.

However, the practical implementation of adversarial training in IDS presents its set of challenges. The foremost among these is the intricate task of obtaining authentic network attack traffic and crafting tailored adversarial attacks against sophisticated IDS setups. Even with a more accommodating threat model that provides defenders with access to malicious datasets, the seamless integration and success of adversarial training remain non-trivial (He et al., 2023).

The data-intensive nature of this approach adds another layer of complexity. As highlighted in Wang et al. (2023), adversarial training techniques have an insatiable appetite for extensive datasets. Given the proprietary nature of many datasets and the challenges associated with data collection, this becomes a significant roadblock. Most IDS systems, in their current configurations, lean heavily on anomaly detection algorithms that focus predominantly on benign data. This data bias makes the infusion of adversarial training into IDS systems a challenging endeavor, primarily due to the limited availability of attack-centric data.

A critical consideration in this discourse is the delicate equilibrium between adversarial resilience and the fidelity of model predictions on clean data (Bai et al., 2021). While the primary aim of adversarial training is to fortify IDS defenses against adversarial threats, there's an inherent risk of diminishing the system's performance on regular, non-adversarial traffic. This trade-off is central to our ongoing research, as the ultimate objective is not just robust defense but also unwavering accuracy and reliability across all data inputs in IDS systems.

3.4.3. Adversarial detection defenses

An alternative approach to combat adversarial attacks involves the implementation of detection mechanisms capable of identifying the presence of adversarial samples (Zizzo et al., 2019). These mechanisms utilize various techniques such as direct classification, neural network uncertainty, or input processing.

In the rapidly evolving landscape of cybersecurity, current detection mechanisms, though promising, have shown limitations in their ability to robustly defend against AML attacks (Zizzo et al., 2019). These mechanisms, developed based on previous threat models, are now being outpaced by the ingenuity of advanced adversarial techniques (Athalye et al., 2018).

One of the core vulnerabilities of these detection systems is their struggle to adapt to the dynamic and ever-changing nature of adversarial attacks. Crafted with precision, adversarial samples are specifically designed to elude and mislead detection systems. This makes it exceedingly difficult to differentiate between genuine and adversarial inputs, especially when the threat landscape is in constant flux (He et al., 2023). As the digital world becomes more interconnected and complex, there's an imperative need to not only enhance our detection capabilities but also to anticipate and preemptively address the next wave of adversarial strategies.

4. Apollon

In this paper, we propose a robust defense system called *Apollon*, which is designed to protect an IDS against AML attacks. *Apollon* is composed of multiple layers to provide better security than traditional IDS and previous works. The proposed system combines multiple classifiers, a *Multi-Armed Bandits (MAB)* algorithm, and requests clustering to provide robust defense against AML attacks.

The first layer of Apollon involves using multiple classifiers instead of a single classifier that is traditionally used in IDS. The concept behind utilizing multiple classifiers is to increase the difficulty for potential attackers attempting to replicate the IDS model. This is because they cannot predict which specific model will be responsible for classifying a given request.

```
Cluster 0: Len of request 274888
Training arm 0 on cluster 0
Training arm 1 on cluster 0
Training arm 2 on cluster 0
Cluster 1: Len of request 358525
Training arm 0 on cluster 1
Training arm 1 on cluster 1
Training arm 2 on cluster 1

Setting reward_sums arm 0 on cluster 0
Setting reward_sums arm 1 on cluster 0
Setting reward_sums arm 2 on cluster 0
Setting reward_sums arm 0 on cluster 1
Setting reward_sums arm 1 on cluster 1
Setting reward_sums arm 2 on cluster 1
```

Listing 1: Apollon training process example.

To dynamically select the optimal classifier or set of classifiers for each input, *Apollon* involves using a *Multi-Armed Bandits (MAB)* with *Thompson sampling*. The *MAB* is responsible for selecting the arm (classifier) to use for each request based on the current state of the system.

Finally, requests are clustered, and there is a version of each classifier for each cluster, trained only with the information of that cluster. Clustering is used to add another layer of uncertainty to the system, as the attacker cannot predict in a simple way which cluster a request belongs to.

Apollon stands as a groundbreaking approach in the realm of intrusion detection. Unlike traditional adversarial training methods that necessitate extensive adversarial data, *Apollon* is ingeniously designed to function effectively without such data. This strategic design not only streamlines the training process but also ensures that *Apollon* remains resilient and adaptable in ever-changing network environments. By eliminating the need for adversarial data, Apollon offers a more pragmatic and scalable solution for intrusion detection.

Furthermore, *Apollon* doesn't alter the underlying models. Instead, it refines their application for peak efficiency. This harmonizes with established defense mechanisms, allowing for a seamless integration and enhancing its adaptability. By selecting the most appropriate classifiers for each request type, *Apollon* maintains the performance standards of conventional IDSs in non-AML network traffic scenarios. Moreover, it devises a strategic methodology to prevent attackers from easily understanding the behaviors of our classifiers using AML techniques. This robust framework ensures that while *Apollon* augments the overall efficacy of existing models, it also fortifies its defenses against AML attacks.

In the face of challenges posed by adversarial training, our proposal, *Apollon*, presents a unique advantage, setting a new standard in the field.

Fig. 2 shows the architecture of *Apollon*, and the flow that a network traffic request follows until it is classified.

Listing 1 shows the output of an Apollon training process example with two clusters and three classifiers. As can be seen, the training data is first divided between the two clusters, the classifiers are trained for each cluster and finally, the rewards are assigned. The Listing 2 shows a selection process example of the classifier for each request, the predicted value and the real value.

In the following, each of the layers that influence the final classification of a network traffic request will be detailed.

4.1. Multiple classifiers

The first layer of *Apollon* involves using multiple classifiers instead of a single classifier, which is typically used in traditional IDS. The

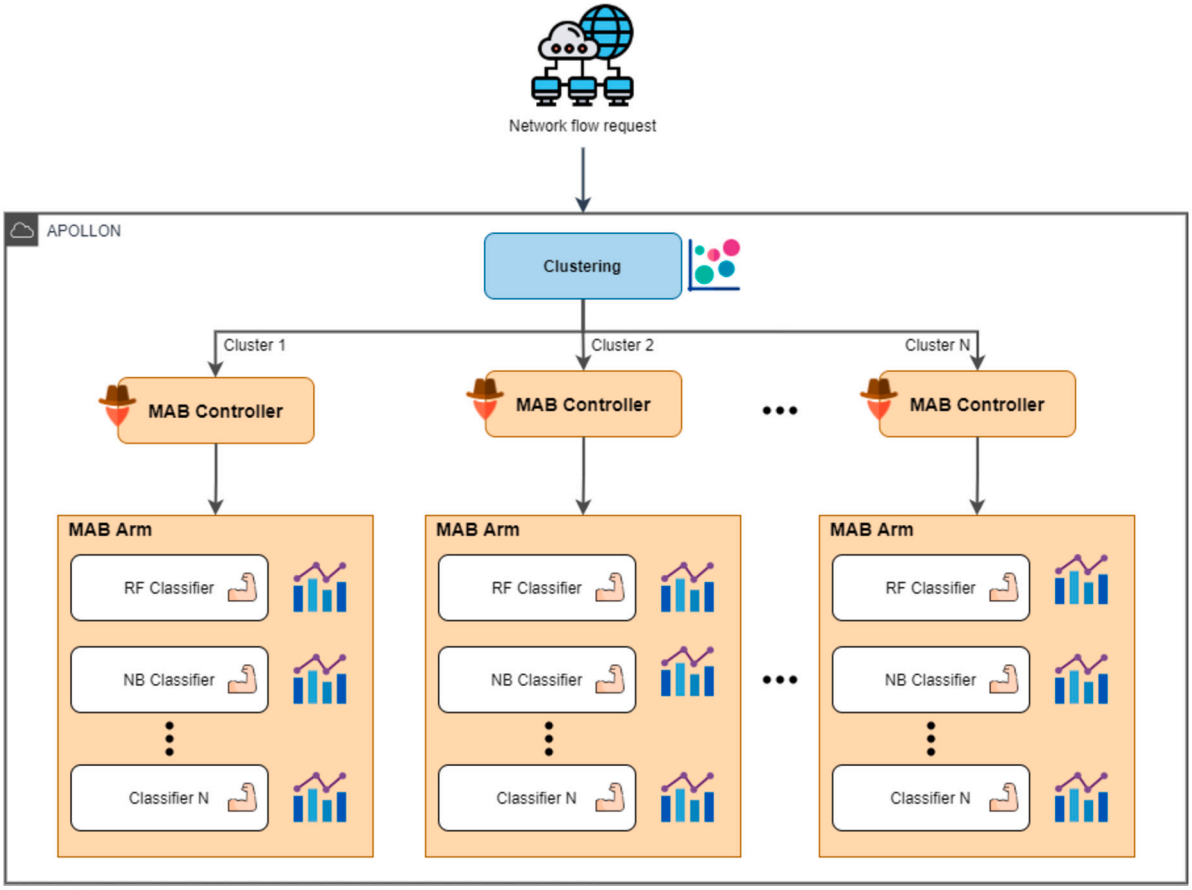


Fig. 2. *Apollon* Architecture.

```
Selected arm: 0.0    Predicted:0    Actual:0
Selected arm: 0.0    Predicted:0    Actual:0
Selected arm: 2.0    Predicted:0    Actual:0
Selected arm: 0.0    Predicted:0    Actual:0
Selected arm: 1.0    Predicted:1    Actual:1
Selected arm: 2.0    Predicted:0    Actual:0
```

Listing 2: Apollon classifier selection process example.

idea of using multiple classifiers is to make it more difficult for the attacker to replicate the IDS model, as he is not able to predict which model is going to classify the request. By utilizing a greater number of diverse classifiers, the system becomes more resilient by introducing greater uncertainty, and better classifiers imply better performance in the *Apollon* system score.

In *Apollon*, we can use any type of classifier. These classifiers can be either the most common ones based on Deep Learning or Machine Learning, or classifiers based on network traffic request forecasting techniques, or the more classical ones based on rule systems.

4.2. Multi-Armed Bandit

The second layer of the *Apollon* defense system involves the use of a *Multi-Armed Bandits (MAB)* algorithm to select the appropriate classifier or set of classifiers for each network traffic request. The *MAB* is responsible for selecting the best classifier or set of classifiers to evaluate whether a request is benign or malicious. This approach avoids the need for manual tuning of thresholds or weights for each classifier.

The *MAB* algorithm works by selecting the arm, or classifier, that has the highest probability of providing the correct classification. In *Apollon*, we use *Thomson Sampling*, which is a popular algorithm for solving the *MAB* problem. Thomson Sampling balances exploration and exploitation of the available classifiers, ensuring that the system selects the optimal classifier or set of classifiers while still being responsive to new and unknown types of traffic.

The *MAB* algorithm in *Apollon* is designed to take into account the different types of classifiers used in the system. For instance, if the *Random Forest* classifier has a high probability of being correct, but the *Naive Bayes* and *Logistic Regression* classifiers have lower probabilities, the *MAB* algorithm will select the *Random Forest* classifier for that particular request. In this way, the *MAB* algorithm ensures that the system selects the optimal set of classifiers for each request, improving the overall accuracy of the classification.

The MAB algorithm is constantly updating the probabilities of the different classifiers based on their previous performance, allowing the system to adapt to changes in the traffic patterns over time and ensuring that the system is always updated with the latest types of attacks.

This update process is described in Algorithm 1. Here, S_i and F_i are the number of observed successes and failures for arm i, respectively, and $theta_i$ is the estimated probability of obtaining a positive reward from arm i. The algorithm uses the Beta distribution as an a priori distribution for the parameters $theta_i$, and updates it with the observed data using Bayes' theorem.

The MAB algorithm samples the posterior distribution and generates a value for each arm, following which the arm with the highest value is selected. However, in cases where there is a tie, indicating that multiple classifiers have an equally high probability of being the best, we resort to an ensemble method, specifically Bootstrap Aggregating, commonly known as Bagging (Lee et al., 2020), to combine the outputs of these selected models. This approach ensures that even when there are

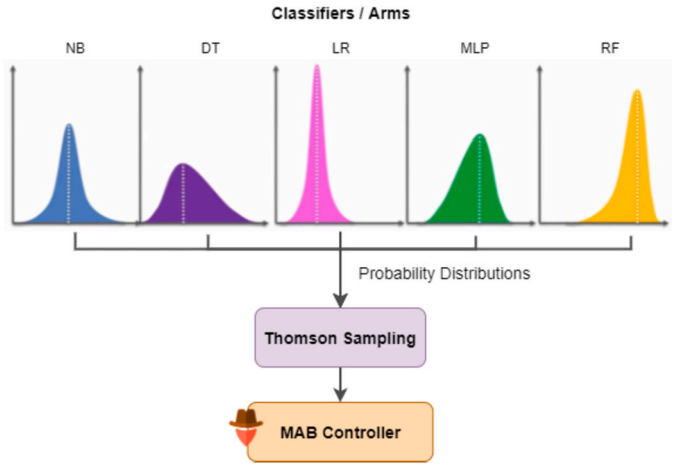


Fig. 3. Apollon *Multi-Armed Bandits* algorithm with multiple classifiers.

multiple 'best' classifiers, the decision-making process remains robust and effective.

Our choice of Bagging as the ensemble method stems from its inherent properties that align well with our objectives. Bagging works by generating multiple versions of a predictor through bootstrapped samples of the training set and then aggregates their predictions. This process inherently reduces variance, making the ensemble less sensitive to the idiosyncrasies of individual classifiers. In the context of our *MAB*-based system, where ties among classifiers indicate closely matched performance, Bagging offers a natural way to harness the collective strength of these classifiers without introducing undue bias.

Algorithm 1 *Apollon* Thompson Sampling.

Intit $S_i = 0$ y $F_i = 0$ for each arm i
for $t = 1, 2, \ldots$ **do**
  For each arm i, sample θ_i of Beta distribution $(S_i + 1,\ F_i + 1)$
  Choose the arm I_t that maximizes θ_i
  Observe the reward X_t of the arm I_t.
  if $X_t = 1$ **then**
    Increment S_{I_t} by one
  else
    Increment F_{I_t} by one
  end if
end for

By using a *Multi-Armed Bandits* algorithm, *Apollon* can dynamically select the optimal classifier or set of classifiers for each network traffic request, making the system more responsive to new types of attacks. The use of *Thomson Sampling* ensures that the system is balanced between exploration and exploitation, improving the overall attacks detection rate of the classification.

The primary reason for our choice of *MAB* over traditional ensemble algorithms is its dynamic adaptability. In the context of defending against AML attacks in IDS, the threat landscape is constantly evolving. *MAB* provides us with the flexibility to adapt over time, allowing us to explore and exploit different classifiers based on their historical performance. This dynamic selection mechanism ensures that our system remains robust even as adversaries adapt their strategies. Traditional ensemble methods, while powerful, operate on a more static combination of models and might not be as agile in responding to changing adversarial tactics.

Fig. 3 shows the diagram of the *MAB* algorithm with multiple classifiers.

4.3. Traffic requests clustering

The final layer of the *Apollon* defense system involves clustering the network traffic requests based on their features, and then training a separate version of each classifier for each cluster. While this may seem to add computational complexity without directly improving traditional performance metrics such as accuracy or detection rate, it plays a crucial role in increasing the robustness of our system against adversarial attacks.

This layer enriches the overall diversity of our system, ensuring the existence of multiple models trained on different clusters of data. This added variation serves to complicate an attacker's task when trying to replicate the behavior of our system through black/gray box attacks. It essentially increases the system's unpredictability, forming a key part of *Apollon*'s robust defense against such attempts.

The traffic requests are clustered using the *K-Means* algorithm (Sinaga and Yang, 2020), which is a popular clustering algorithm that is used in many Machine Learning applications. The *K-Means* algorithm works by randomly selecting k points as the initial centroids, and then iteratively updating the centroids until the clusters converge. In *Apollon*, we use the *K-Means* algorithm to cluster the network traffic requests based on their features, ensuring that requests with similar features are grouped together. These features are the same ones used by the classifiers to evaluate the requests, ensuring that the clustering is based on the same information that the classifiers use to make their decisions.

Each cluster has its dedicated version of every classifier, trained exclusively on the traffic requests in that cluster. This specific tuning to distinct traffic patterns significantly contributes to the system's complexity and enhances its defense against adversarial learning.

When a new network traffic request arrives at *Apollon*, it is classified into the appropriate cluster based on its features. The *Multi-Armed Bandits* algorithm then selects the optimal set of classifiers for that cluster, factoring in the performance of each classifier within that specific cluster. The selected classifier or set of classifiers then evaluate the request to determine if it is benign or malicious.

The use of clustering in *Apollon*, together with the individual training of each classifier on each cluster, allows the *Multi-Armed Bandit* algorithm to generate multiple probability distributions for each classifier, depending on the type of request received. This blend of strategies amplifies the challenge for potential attackers to identify the responding classifier, thereby reducing the probability of successful system imitation.

4.4. Apollon limitations

While *Apollon* offers a novel and robust approach to defending against adversarial attacks in Intrusion Detection Systems, it is crucial to discuss its limitations to provide a well-rounded understanding of its applicability, strengths, and areas for future improvement.

- **Increased Model Training Time:** One of the notable limitations of *Apollon* is the increased computational time required during the model training phase. Although *Apollon* is designed to be computationally efficient during the prediction phase—where it merely samples from pre-defined distributions to select an appropriate model—the training phase is more computationally intensive. This is because *Apollon* needs to generate these distributions for each model in the pool, which can be a time-consuming process. This limitation is particularly relevant in scenarios where rapid model training and deployment are crucial.
- **Model Pool Diversity:** The second limitation pertains to the diversity of the model pool. The efficacy of *Apollon* is intrinsically linked to the diversity and quality of the models it has at its disposal. A pool that lacks diversity in terms of the types of models, their architectures, or their training data may not fully exploit the potential of the *Multi-Armed Bandits* mechanism. This could result in suboptimal performance and may reduce *Apollon*'s overall robustness against a wide array of adversarial attacks.

These limitations offer avenues for future research and development to further enhance the real-world applicability and effectiveness of *Apollon*.

5. Evaluation

In this section, we present the methodology and results of our evaluation, which assesses the performance of *Apollon* in traditional and AML attack environments. The code that was used and created during the evaluation of our proposal is completely open and accessible. It can be found on GitHub.[1]

5.1. Methodology

We designed two scenarios to assess the performance of our proposed solution. In the first scenario, we tested our solution on three datasets: CIC-IDS-2017, CSE-CIC-IDS-2018, and CIC-DDoS-2019. These datasets are comprised of network traffic and web attacks and are commonly used in the evaluation of IDSs. The reason for selecting these datasets is that they are the most popular and widely used in the evaluation of IDSs. Additionally, they are generated to be as similar as possible to real-world network traffic and attacks. Therefore, they are ideal for evaluating the performance of our solution in a traditional IDS environment. However, it is important to note that real-world network traffic and attacks are constantly evolving and becoming more complex, and as such, these datasets may not be representative of the current state of network traffic and attacks.

These datasets are comprised of flow features in a CSV format which were used as inputs to our Machine Learning models. The process of feature selection was undertaken prior to model training. For the CIC-DDoS-2019 dataset, we followed the feature selection process as described in work (Thiyam and Dey, 2023). For the CSE-CIC-IDS-2018 dataset, we referred to the methodology outlined in work (Pujari et al., 2022) for feature selection. Lastly, for the CIC-IDS-2017 dataset, we adhered to the approach recommended in work (Faker and Dogdu, 2019) for selecting features. This systematic feature selection approach enabled us to optimize the performance of our models, enhancing their accuracy, and reducing overfitting and training time.

Due to our lack of powerful machines, we opted to use a representative subset of the datasets instead of the complete data to efficiently train and test our models. For the dataset CIC-DDoS-2019 the whole dataset has been used, while for the dataset CSE-CIC-IDS-2018 the subset from 02-15-2018 has been used. Finally, for the CIC-IDS-2017 dataset, the following subsets of data have been selected:

- *Friday WorkingHours Afternoon DDoS*
- *Friday WorkingHours Afternoon PortScan*
- *Friday WorkingHours Morning*
- *Monday WorkingHours*
- *Thursday WorkingHours Afternoon Infilteration*
- *Thursday WorkingHours Morning WebAttacks*
- *Tuesday WorkingHours*

We compared the results of our solution with the classifiers extracted from related work. In the second scenario, we used several gray/black-box Adversarial Machine Learning attacks to evaluate the ability of our solution to defend against such attacks. In this scenario, the test data exclusively comprise attack instances. Conversely, for the first scenario, the test data are extracted from the respective dataset and incorporate a mix of benign and malicious instances.

The classifiers scores used to compare our solution may be different than those reported in related work due to several factors. Firstly, the machine on which the training is performed may differ from that of related work. This can impact the speed and efficiency of the training process, which in turn can affect the final accuracy of the classifiers. Secondly, the pre-processing of the data may be different between our solution and related work. Pre-processing techniques can greatly impact the quality of the data and hence the performance of the classifiers. Therefore, differences in pre-processing techniques can lead to varying levels of accuracy in the classifiers. It is important to take into account these differences when comparing our solution to related work, and to consider the impact of these factors on the performance of the classifiers.

All the experiments were performed on a *Ubuntu 20.04.5 LTS* machine with an *Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz* processor and 16 GB of RAM memory.

5.1.1. Traditional network traffic and attacks

In this test scenario, we utilized the CIC-IDS-2017, CSE-CIC-IDS-2018, and CIC-DDoS-2019 datasets to train a set of classifiers to compare with our proposed solution. Our goal was to assess the performance of our solution with the traditional network traffic and web attacks.

We have utilized default hyperparameters of the classifiers because the objective of this scenario is not to maximize the performance of these classifiers but rather to compare them with our solution. The classifiers used in this scenario are the following:

- *Multilayer Perceptron (MLP)*: hidden_layer_sizes = (32), max_iter = 200
- *Random Forest (RF)*: n_estimations = 100
- *Decision Trees (DT)*
- *Naive Bayes (NB)*
- *Logistic Regression (LR)*

5.1.2. Adversarial Machine Learning attacks

In this test scenario, we used several gray/black-box Adversarial Machine Learning attacks to evaluate the ability of our solution to defend against such attacks. To simplify the evaluation process, we used the CIC-IDS-2017 dataset to train the classifiers and our proposed solution because it is the most popular dataset. We compare the accuracy and the detection rate of the classifiers and our proposed solution against the gray/black-box Adversarial Machine Learning attacks. As we mentioned before, we only test with gray/black-box attacks because white-box attacks are not realistic in real-world scenarios.

The attacks used in this scenario are the following:

- *Zeroth-order optimization attack (ZOO)* (Chen et al., 2017)
- *HopSkipJump attack (HSJA)* (Chen et al., 2020)
- *W-GAN based attacks* (Lin et al., 2022)

We opted for these particular attacks because they encompass a broad spectrum of potential Adversarial Machine Learning evasion strategies and are among the most widely used and successful.

The adversarial network traffic used in this scenario was generated by modifying real attack traffic while preserving the key characteristics of the attacks. This method ensures that the adversarial traffic maintains the legitimate semantics of the original traffic, an approach adopted from M. Usama et al.'s methodology (Usama et al., 2019).

5.2. Results

Throughout the development of the evaluation we have decided to set a seed, so that the results can be replicated: the *seed = 42*.

Before training the classifiers, a common pre-processing step was performed on the data from all datasets. This step is essential in standardizing the datasets, ensuring that we are working uniformly with each of them. To achieve this, a combination of *sklearn* functions such as *RobustScaler* and *Normalizer* was utilized. To make features less sensitive to outliers, the *RobustScaler* subtracts the median and adjusts the data based on the quantile range. The *Normalizer* scales the input data set to have a norm of 1 and values between 0 and 1.

In addition to standardizing the datasets, additional steps were taken to further prepare the data for training our classifiers. We avoided ex-

[1] https://github.com/antonioalfa22/apollon.

cessive pre-processing, to preserve as much data as possible, so we only eliminated duplicate elements and attributes that had only unique data.

We also encountered some data with missing values (*NaN*) and explored various methods to handle this issue: filling in missing values with a fixed value, with the mean or median of the feature's non-missing values, with a prediction based on the other features and with a prediction based on k-nearest neighbors. Our experiments revealed that using a constant value of 0 to fill in missing values achieved the best performance in the training step.

We apply this pre-processing to all datasets equally. For validating the results, we employed the Walk-Forward Cross-Validation method, which is particularly suitable for time-series data, ensuring that there's no data leakage from the future into the training set. The specifics of our approach are as follows:

- Step Size: After each iteration, we expanded the training set by 8% of the total dataset size. This ensures that the model is trained on an increasing amount of past data while leaving room for validation.
- Validation Set Size: In each iteration, the model was validated on the subsequent 2% of the dataset. This size was chosen to provide a balance between the number of training and validation iterations and the granularity of validation.
- Total Iterations: Given the initial training size, step size, and validation set size, the process was carried out in 5 iterations, ensuring that the model is validated across all possible contiguous training-validation splits.

By using Walk-Forward Cross-Validation, we ensure that the results obtained are representative of the model's ability to predict future data points based on past observations, eliminating the risk of inadvertently training the model on future data. This approach provides a more realistic and robust evaluation of the model's performance on time-series data.

The results of our experiments in the two evaluation scenarios are presented below.

5.2.1. Traditional network traffic and attacks

In this scenario, we trained the selected classifiers on the CIC-IDS-2017, CSE-CIC-IDS-2018, and CIC-DDoS-2019 datasets. Additionally, we trained our proposed solution on the same datasets and with the same classifiers. To train *Apollon*, we use the K-Means (Likas et al., 2003) algorithm to cluster the data into 2 clusters. For each cluster, we train the selected classifiers and we update the *Multi-Armed Bandits* algorithm with the results.

The results obtained from the experiments are presented in the Tables 2, 3, and 4.

In our experiments across different datasets, the *Multi-Armed Bandit* algorithm showed a preference for specific models or combinations thereof. For instance, in the CIC-IDS-2017 dataset, the most frequently chosen combination by *MAB* was Random Forest (RF) and Decision Tree (DT), accounting for approximately 40% of the selections, followed by RF alone at 28%. In the CIC-DDoS-2019 dataset, DT was the choice in 60% of the selections. Similarly, in the CSE-CIC-IDS-2018 dataset, RF was chosen 42% of the time, followed by the combination of RF and DT at approximately 27%. Intriguingly, the models or combinations selected by MAB are generally those that individually exhibit high performance. This observation underscores the efficacy of MAB in dynamically selecting optimal models, thereby enhancing the robustness and adaptability of our IDS.

Based on these results, our solution demonstrates high detection rate and accuracy scores, comparable to the classifiers chosen for comparison. It's important to mention that among all the datasets, *Apollon* doesn't achieve the best or the worst scores. This is due to the fact that *Apollon* internally selects from the same classifiers. Hence, the highest score that *Apollon* can achieve is limited to the best classifier's maximum score, while it can never perform as poorly as the worst classifier since it has other better options.

Table 2
Results of the traditional network traffic and attacks scenario on the CIC-IDS-2017 dataset.

Metrics	CIC-IDS-2017					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.9766	0.6694	0.9996	0.9980	0.9516	**0.9740**
Detection rate	0.9731	0.8049	0.9996	0.9962	0.8360	**0.9420**
F1	0.9760	0.6118	0.9991	0.9959	0.8858	**0.7699**
AUC	0.9998	0.8289	1.0000	0.9972	0.9902	**0.9420**

Table 3
Results of the traditional network traffic and attacks scenario on the CSE-CIC-IDS-2018 dataset.

Metrics	CSE-CIC-IDS-2018					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.9916	0.7280	0.9993	0.9939	0.9593	**0.9064**
Detection rate	0.9367	0.8563	0.9967	0.9961	0.6008	**0.9419**
F1	0.9545	0.5507	0.9963	0.9968	0.6556	**0.7303**
AUC	0.9945	0.8605	0.9993	0.9984	0.9592	**0.9419**

Table 4
Results of the traditional network traffic and attacks scenario on the CIC-DDoS-2019 dataset.

Metrics	CIC-DDoS-2019					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.9998	0.9990	0.9999	0.9999	0.9970	**0.9996**
Detection rate	0.8985	0.8709	0.9932	0.9999	0.7930	**0.9691**
F1	0.9186	0.7148	0.9957	0.9991	0.8541	**0.8521**
AUC	0.9817	0.9753	0.9999	0.9999	0.9796	**0.9691**

Our results demonstrate that even with the integration of new security mechanisms, *Apollon* can still provide high accuracy and detection rate scores in traditional network traffic classification environments. Therefore, *Apollon* is capable of preserving the fundamental functionality of an IDS.

5.2.2. Adversarial Machine Learning attacks

In this scenario, we have launched three types of Adversarial Machine Learning attacks on the selected classifiers and against our solution. These attacks are *Zeroth-order optimization attack (ZOO)*, *Hop-SkipJump attack (HSJA)* and W-GAN based attacks.

The classifiers and the *Apollon* implementation used as targets of the attacks are the ones trained in the previous environment with the CIC-IDS-2017 dataset.

Starting with the *Zeroth-order optimization attack (ZOO)*, we have used the open source implementation provided by ART (Nicolae et al., 2018), and created a Classifier class so that *Apollon* can be used as a model. The attack was launched with the following parameters for each classifier:

- **classifier**: the Classifier class instance with the classifier to be attacked.
- targeted: True.
- learning_rate: 0.01.
- max_iter: 100.

The attack was launched against the classifiers and the results are shown in Table 5. The most frequently chosen combination was RF and DT, making up approximately 39% of the selections, followed by RF alone at 28%. According to the findings, the attack was effective in every scenario, resulting in reduced detection rates across all classifiers. Nonetheless, even though the *Apollon* implementation's accuracy and

Table 5
Results of the *ZOO* AML attack.

Metrics	ZOO attack					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.7200	0.7300	0.7250	0.7400	0.5200	**0.9304**
Detection rate	0.5260	0.5500	0.5460	0.5780	0.0420	**0.8772**
F1	0.5329	0.5550	0.5505	0.5815	0.0457	**0.8835**
AUC	0.7300	0.7400	0.7350	0.7500	0.5300	**0.9400**

Table 6
Results of the *HopSkipJump* AML attack.

Metrics	HopSkipJump attack					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.5002	0.4301	0.5001	0.5051	0.5900	**0.7550**
Detection Rate	0.0000	0.0000	0.0000	0.0100	0.1800	**0.5260**
F1	0.0000	0.0000	0.0000	0.0133	0.2091	**0.5607**
AUC	0.5002	0.4907	0.5000	0.5121	0.6200	**0.7760**

Table 7
Results of the *W-GAN* based AML attack.

Metrics	W-GAN based attack					
	MLP	NB	RF	DT	LR	Apollon
Accuracy	0.5002	0.4398	0.5001	0.4280	0.3230	**0.6260**
Detection Rate	0.0000	0.0000	0.0000	0.0000	0.0000	**0.4250**
F1	0.0000	0.0000	0.0000	0.0000	0.0000	**0.4595**
AUC	0.5000	0.4903	0.5004	0.4887	0.4604	**0.7000**

detection rate scores also declined, they remained considerably superior to those of the other classifiers, with high accuracy and detection rate scores.

To launch the *HopSkipJump attack (HSJA)*, we have used the open source implementation provided by ART, and created a Classifier as in the *ZOO* attack. The attack was launched with the following parameters for each classifier:

- **classifier**: the Classifier class instance with the classifier to be attacked.
- targeted: True.
- max_iter: 100.
- norm: inf.

The results of the attack are shown in Table 6. In this scenario, the RF and DT combination accounted for approximately 36% of the selections, followed by RF at 33%. The attack was very effective in all the classifiers, resulting in reduced detection rates to scores close to zero. The exception was the *Apollon* implementation, which was able to maintain a detection rate > 0.5.

Finally, the *Wasserstein Generative Adversarial Network (W-GAN)* based attack was launched with a custom implementation available in the *Apollon* repository on GitHub. This implementation was based on the *IDSGAN* attack (Lin et al., 2022), updated to the needs of the selected dataset. The results in Table 7 were obtained after launching the attack with 100 epochs. Much like in prior experiments, the RF and DT combination was chosen in approximately 48% of the cases, followed by RF at 20%.

The *W-GAN* based attack was the most effective attack, reducing the detection rate to zero in all the classifiers. Our solution, on the other hand, was able to maintain a detection rate > 0.4.

Interestingly, the MAB choices for these three types of AML attacks closely resemble those observed in the previous scenario conducted on the same dataset. This consistency can be attributed to the fact that Apollon starts each experiment with the same initial distributions for the models. As a result, the selection patterns across different types of attacks are highly similar, with the same models being chosen more frequently than others.

Apollon's improved accuracy and detection rates in AML attacks can be attributed to the inclusion of an uncertainty component in its model selection process. This renders it difficult to train a model solely based on its responses.

The experimental results of the scenario reveal that our solution exhibits greater robustness in comparison to the other classifiers used individually.

However, we also observed that while our solution effectively reduces the effectiveness of the attacks, it does not completely nullify them. This means that there is still room for improvement in terms of enhancing the solution's robustness to further strengthen its resistance against such attacks.

In particular, if we were to generate the attacks with more time, such as by increasing the number of iterations or epochs, it is likely that the effectiveness of these attacks against our solution would increase. It's important to note that we are realistic about the limitations of our proposal. While crafting an impenetrable defense is nearly impossible, our solution aims to significantly increase the time and resources required for a potential attacker to successfully execute an attack. In real-world scenarios, this increase in time and computational cost can render an attack unfeasible or economically unviable, thereby serving as a deterrent and adding an extra layer of security.

6. Conclusions and future work

In conclusion, this paper presents *Apollon*, a new robust defense system against Adversarial Machine Learning attacks on Intrusion Detection Systems. *Apollon* utilizes a *Multi-Armed Bandits* model to select the best-suited classifier in real-time for each input with Thompson sampling, adding a layer of uncertainty to the IDS behavior, that makes it more difficult for attackers to replicate the IDS and generate adversarial traffic.

Our experimental evaluation on several datasets shows that *Apollon* can successfully detect attacks without compromising its performance on normal network traffic data, and can prevent attackers from learning the IDS behavior in realistic training times. These results demonstrate that *Apollon* is an effective defense system against AML attacks in IDS, which can help to enhance the security of critical systems.

Nevertheless, *Apollon* does not completely eliminate the risk of AML attacks; rather, it mitigates the threat by significantly increasing the time and effort required by attackers to generate adversarial traffic. By doing so, *Apollon* adds a layer of complexity that attackers must navigate, thereby serving as a deterrent. This increased time and effort can often translate into higher computational and financial costs for the attacker, making the attack less appealing or even economically unfeasible. While this doesn't make the system entirely impervious to AML attacks, it does raise the bar for what constitutes a successful attack, providing an additional layer of security and resilience in real-world applications.

With the aim of improving the performance and robustness of *Apollon*, we plan to explore the use of other *MAB* models and implementations, such as Bayesian Optimization or Deep Bayesian Bandits. We also plan to explore the use of other classifiers and ML models, such as requests forecasting models, which can be used to predict the expected number of requests in the next time window and compare it with the actual number of requests. Finally, we plan to test *Apollon* with AML attacks on other datasets, such as *CSE-CIC-IDS-2018* and *CIC-DDoS-2019* datasets, and explore the use of other datasets, to evaluate the performance of *Apollon* in different network environments.

CRediT authorship contribution statement

Antonio Paya: Conceptualization, Investigation, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Sergio**

Arroni: Investigation, Software, Visualization, Writing – original draft. **Vicente García-Díaz:** Conceptualization, Validation, Writing – original draft, Writing – review & editing. **Alberto Gómez:** Conceptualization, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the Github with the data in the article.

References

Abdallah, E.E., Otoom, A.F., et al., 2022. Intrusion detection systems using supervised machine learning techniques: a survey. Proc. Comput. Sci. 201, 205–212.

Abdulhammed, R., Faezipour, M., Musafer, H., Abuzneid, A., 2019. Efficient network intrusion detection using pca-based dimensionality reduction of features. In: 2019 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6.

Agrawal, S., Goyal, N., 2012. Analysis of Thompson sampling for the multi-armed bandit problem. In: Conference on Learning Theory. In: JMLR Workshop and Conference Proceedings.

Akshay Kumaar, M., Samiayya, D., Vincent, P.M.D.R., Srinivasan, K., Chang, C.-Y., Ganesh, H., 2022. A hybrid framework for intrusion detection in healthcare systems using deep learning. Front. Public Health 9.

Alotaibi, A., Rassam, M.A., 2023. Adversarial machine learning attacks against intrusion detection systems: a survey on strategies and defense. Future Internet 15 (2), 62.

Amor, N.B., Benferhat, S., Elouedi, Z., 2004. Naive Bayes vs decision trees in intrusion detection systems. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 420–424.

Ashfaq, R.A.R., Wang, X.-Z., Huang, J.Z., Abbas, H., He, Y.-L., 2017. Fuzziness based semi-supervised learning approach for intrusion detection system. Inf. Sci. 378, 484–497.

Athalye, A., Carlini, N., Wagner, D., 2018. Obfuscated gradients give a false sense of security, circumventing defenses to adversarial examples. In: International Conference on Machine Learning. In: PMLR, pp. 274–283.

Bai, T., Luo, J., Zhao, J., Wen, B., Wang, Q., 2021. Recent advances in adversarial training for adversarial robustness. arXiv preprint arXiv:2102.01356.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F., 2013. Evasion attacks against machine learning at test time. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23–27, 2013, Proceedings, Part III 13. Springer, pp. 387–402.

Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, pp. 39–57.

Carpentier, A., Lazaric, A., Ghavamzadeh, M., Munos, R., Auer, P., 2011. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In: Algorithmic Learning Theory: 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings 22. Springer, pp. 189–203.

Chen, J., Jordan, M.I., 2019. Boundary attack++: query-efficient decision-based adversarial attack. arXiv preprint arXiv:1904.02144. vol. 2, no. 7.

Chen, J., Jordan, M.I., Wainwright, M.J., 2020. Hopskipjumpattack: a query-efficient decision-based attack.

Chen, K., 2020. Stealing deep reinforcement learning models for fun and profit. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.-J., 2017. Zoo: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15–26.

Cutler, A., Cutler, D.R., Stevens, J.R., 2012. Random forests. In: Ensemble Machine Learning: Methods and Applications, pp. 157–175.

De Cristofaro, E., 2020. An overview of privacy in machine learning. arXiv preprint arXiv:2005.08679.

Duy, P.T., Khoa, N.H., Nguyen, A.G.-T., Pham, V.-H., et al., 2021. Digfupas: deceive ids with gan and function-preserving on adversarial samples in sdn-enabled networks. Comput. Secur. 109, 102367.

Faker, O., Dogdu, E., 2019. Intrusion detection using big data and deep learning techniques. In: Proceedings of the 2019 ACM Southeast Conference. ACM SE'19, New York, NY, USA. Association for Computing Machinery, pp. 86–93.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2020. Generative adversarial networks. Commun. ACM 63 (11), 139–144.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2014a. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2014b. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C., 2017. Improved training of Wasserstein gans. Adv. Neural Inf. Process. Syst. 30.

He, K., Kim, D.D., Asghar, M.R., 2023. Adversarial machine learning for network intrusion detection systems: a comprehensive survey. IEEE Commun. Surv. Tutor. 25 (1), 538–566.

Hnamte, V., Hussain, J., 2023. Dcnnbilstm: an efficient hybrid deep learning-based intrusion detection system. Telemat. Inform. Rep. 10, 100053.

Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D., 2011. Adversarial machine learning. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp. 43–58.

Huang, S., Lei, K., 2020. Igan-ids: an imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. Ad Hoc Netw. 105, 102177.

Hugh, J.M., 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln laboratory. ACM Trans. Inf. Syst. Secur.

Koh, P.W., 2018. Stronger data poisoning attacks break data sanitization defenses. Mach. Learn. 111, 1–47.

Kou, C., Lee, H.K., Chang, E.-C., Ng, T.K., 2019. Enhancing transformation-based defenses using a distribution classifier. arXiv preprint arXiv:1906.00258.

Kuleshov, V., Precup, D., 2014. Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028.

Kurakin, A., Goodfellow, I., Bengio, S., 2016. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236.

Lee, T.-H., Ullah, A., Wang, R., 2020. Bootstrap aggregating and random forest. In: Macroeconomic Forecasting in the Era of Big Data: Theory and Practice, pp. 389–429.

Likas, A., Vlassis, N., Verbeek, J.J., 2003. The global k-means clustering algorithm. Pattern Recognit. 36 (2), 451–461.

Lin, Z., Shi, Y., Xue, Z., 2022. Idsgan: generative adversarial networks for attack generation against intrusion detection. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp. 79–91.

Liu, G., Zhang, W., Li, X., Fan, K., Yu, S., 2022. Vulnergan: a backdoor attack through vulnerability amplification against machine learning-based network intrusion detection systems. Sci. China Inf. Sci. 65 (7), 1–19.

Machado, M.C., Srinivasan, S., Bowling, M., 2014. Domain-independent optimistic initialization for reinforcement learning. arXiv preprint arXiv:1410.4604.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

Mahoney, M.V., Chan, P.K., 2003. An analysis of the 1999 darpa/Lincoln laboratory evaluation data for network anomaly detection. In: International Workshop on Recent Advances in Intrusion Detection. Springer, pp. 220–237.

Maseer, Z.K., Yusof, R., Bahaman, N., Mostafa, S.A., Foozy, C.F.M., 2021. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. IEEE Access 9, 22351–22370.

Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P., 2016. Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582.

Mukherjee, B., Heberlein, L.T., Levitt, K.N., 1994. Network intrusion detection. IEEE Netw. 8 (3), 26–41.

Nicolae, M.-I., Sinn, M., Tran, M.N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., Edwards, B., 2018. Adversarial robustness toolbox v1.2.0. CoRR. arXiv:1807.01069.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A., 2016. The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, pp. 372–387.

Park, H., Faradonbeh, M.K.S., 2021. Analysis of Thompson sampling for partially observable contextual multi-armed bandits. IEEE Control Syst. Lett. 6, 2150–2155.

Pharate, A., Bhat, H., Shilimkar, V., Mhetre, N., 2015. Classification of intrusion detection system. Int. J. Comput. Appl. 118 (7).

Pujari, M., Pacheco, Y., Cherukuri, B., Sun, W., 2022. A comparative study on the impact of adversarial machine learning attacks on contemporary intrusion detection datasets. SN Comput. Sci. 3 (5), 1–12.

Ramos, D., Faria, P., Gomes, L., Campos, P., Vale, Z., 2022a. Selection of features in reinforcement learning applied to energy consumption forecast in buildings according to different contexts. Energy Rep. 8, 423–429.

Ramos, D., Faria, P., Gomes, L., Campos, P., Vale, Z., 2022b. A learning approach to improve the selection of forecasting algorithms in an office building in different contexts. In: Progress in Artificial Intelligence: 21st EPIA Conference on Artificial Intelligence, EPIA 2022, Lisbon, Portugal, August 31–September 2, 2022, Proceedings. Springer, pp. 271–281.

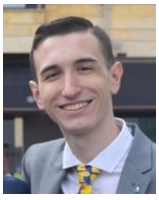
Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A., 2019. A survey of network-based intrusion detection data sets. Comput. Secur. 86, 147–167.

Rokach, L., Maimon, O., 2005. Decision trees. In: Data Mining and Knowledge Discovery Handbook, pp. 165–192.

Sahoo, K.S., Tripathy, B.K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., Burgos, D., 2020. An evolutionary svm model for ddos attack detection in software defined networks. IEEE Access 8, 132502–132513.

Sahu, S.K., Mohapatra, D.P., Rout, J.K., Sahoo, K.S., Pham, Q.-V., Dao, N.-N., 2022. A lstm-fcnn based multi-class intrusion detection using scalable framework. Comput. Electr. Eng. 99, 107720.
Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018a. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSp, vol. 1, pp. 108–116.
Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018b. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy.
Sharafaldin, I., Lashkari, A.H., Hakak, S., Ghorbani, A.A., 2019. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–8.
Shroff, J., Walambe, R., Singh, S.K., Kotecha, K., 2022. Enhanced security against volumetric ddos attacks using adversarial machine learning. Wirel. Commun. Mob. Comput. 2022.
Sinaga, K.P., Yang, M.-S., 2020. Unsupervised k-means clustering algorithm. IEEE Access 8, 80716–80727.
Suthaharan, S., Suthaharan, S., 2016. Support vector machine. In: Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning, pp. 207–235.
Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6.
Thakkar, A., Lohiya, R., 2020. A review of the advancement in intrusion detection datasets. Proc. Comput. Sci. 167, 636–645.
Thiyam, B., Dey, S., 2023. Efficient feature evaluation approach for a class-imbalanced dataset using machine learning. In: International Conference on Machine Learning and Data Engineering. Proc. Comput. Sci. 218, 2520–2532.
Tobi, A.M.A., Duncan, I., 2018. Kdd 1999 generation faults: a review and analysis. J. Cyber Secur. Technol. 2 (3–4), 164–200.
Usama, M., Asim, M., Latif, S., Qadir, J., et al., 2019. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, pp. 78–83.
Wang, Y., Sun, T., Li, S., Yuan, X., Ni, W., Hossain, E., Poor, H.V., 2023. Adversarial attacks and defenses in machine learning-powered networks: a contemporary survey. arXiv preprint arXiv:2303.06302.
Wright, R.E., 1995. Logistic regression.
Wu, Z., Zhang, H., Wang, P., Sun, Z., 2022. Rtids: a robust transformer-based approach for intrusion detection system. IEEE Access 10, 64375–64387.
Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., Jain, A.K., 2020. Adversarial attacks and defenses in images, graphs and text: a review. Int. J. Autom. Comput. 17, 151–178.
Yeom, S., 2017. Privacy risk in machine learning: analyzing the connection to overfitting. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pp. 268–282.
Zhang, J., Zulkernine, M., Haque, A., 2008. Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 38 (5), 649–659.
Zhao, S., Li, J., Wang, J., Zhang, Z., Zhu, L., Zhang, Y., 2021. attackgan: adversarial attack against black-box ids using generative adversarial networks. Proc. Comput. Sci. 187, 128–133.
Zizzo, G., Hankin, C., Maffeis, S., Jones, K., 2019. Adversarial machine learning beyond the image domain. In: Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1–4.

Antonio Payá González is a Ph.D. student developing his thesis on defense systems for Software-Defined Perimeters and Intrusion Detection Systems. He has a Master's degree in Web Engineering of the University of Oviedo. He is currently working for TheNextPangea S.L. on topics related to Machine Learning and Artificial Intelligence applied on cybersecurity fields.



Sergio Arroni del Riego is a student of Software Engineering at the University of Oviedo, passionate about Machine Learning, Artificial Intelligence and Cybersecurity among other Software fields. He is currently working at the Foundation of the University of Oviedo, researching new advances in the field of Artificial Intelligence and Machine Learning.

Vicente García-Díaz is an Associate Professor in the Department of Computer Science at the University of Oviedo, Spain. He is a Software Engineer, PhD in Computer Science. He has a Master's in Occupational Risk Prevention and the qualification of University Expert in Blockchain Application Development. He is part of the editorial and advisory board of several indexed journals and conferences and has been editor of several special issues in books and indexed journals. He has supervised 100+ academic projects and published 100+ research papers in journals, conferences, and books. His teaching interests are primarily in the design and analysis of algorithms and the design of domain-specific languages. His current research interests include decision support systems, health informatics and eLearning.

Alberto Gómez Gómez works for the Department of Business Administration, at the School of Industrial Engineering of The University of Oviedo, Spain. His teaching and research initiatives focus on the areas of Production Management, Applied Artificial Intelligence and Information Systems. He has written several national and international papers. Journal of the Operational Research Society. Artificial Intelligence for Engineering Design, Analysis and Manufacturing. International Journal of Foundations of Computer Science. European Journal of Operational Research. International Journal of production economics. Engineering Applications of Artificial Intelligence. Concurrent Engineering- Research and Applications.