Apollon: A Robust Defence System against Adversarial Machine Learning Attacks in Intrusion Detection Systems

Antonio Paya[a,*], Sergio Arroni[a], Vicente García-Díaz[a] and 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 growing threat of Adversarial Machine Learning (AML) attacks is posing a significant challenge for the effectiveness of Intrusion Detection Systems (IDS). In this paper, we propose *Apollon*, a novel defence system that can protect IDS against AML evasion and extraction attacks. *Apollon* leverages a set of heterogeneous classifiers to detect intrusions and applies a *Multi-Armed Bandits (MAB)* model to dynamically select the optimal classifier or ensemble of classifiers for each input. Therefore, *Apollon* can prevent attackers from learning the IDS behaviour 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 normal inputs. We also demonstrate that *Apollon* can prevent attackers from learning the IDS behaviour in realistic training times. Our results suggest that *Apollon* is an effective and efficient defence 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 [1].

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 [2, 3]. 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 [4].

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 [5, 6, 7].

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 [8]. 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.

Apollon utilizes a diverse range of classifiers to detect intrusions and employs a *Multi-Armed Bandits (MAB)* model to select the best-suited classifier or a combination of classifiers in real-time for each input, enabling it to achieve this objective without compromising its performance on normal inputs.

In this way, *Apollon* can prevent attackers from learning the behavior of the IDS in realistic training times, adding a layer of semi-randomness 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

*Corresponding author

antoniopaya@outlook.com (A. Paya); sergioadgi38@gmail.com (S. Arroni); garciavicente@uniovi.es (V. García-Díaz); albertogomez@uniovi.es (A. Gómez)

ORCID(s): 0000-0003-3733-3805 (A. Paya); 0000-0002-4907-8576 (S. Arroni); 0000-0003-2037-8548 (V. García-Díaz); 0000-0003-2037-8548 (A. Gómez)

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 [9]. 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) [10].

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

HIDS, on the other hand, monitors the activity on individual hosts, such as servers or workstations. HIDS can detect attacks that may not be visible to NIDS, such as attacks that occur within encrypted traffic or attacks 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 [2, 3, 1]. 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.

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 [4]. 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.

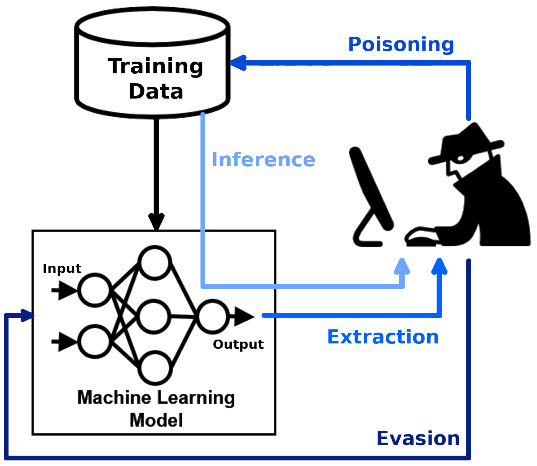

Figure 1: Adversarial Machine Learning (AML) attacks by ART [11]

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[12].

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. 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 grey/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. 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. 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. 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 [13]. 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 [14], upper confidence bound (UCB) [15], and Thompson sampling [16]. 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 [17]. 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.

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 are accessible. This section will examine some of the most prevalent datasets, highlighting their essential qualities and applications.

3.1.1. Discarded datasets

This project did not make use of several other IDS datasets, including Darpa 1998/99 [18], KDD 99 [19], and NSL-KDD [19]. 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 [20, 21].

3.1.2. CIC-IDS-2017

A highly utilized IDS dataset in contemporary literature is the CIC-IDS2017 [22], 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 [23]. 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 [24].

3.1.3. CSE-CIC-IDS-2018

The CSE-CIC-IDS2018 dataset was developed using AWS resources in a simulated enterprise network environment in 2018 [25]. 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 [26]. 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 [26].

3.1.4. 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 [27]. 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 [23]. As a result, experts suggest using this dataset in conjunction with other datasets [24], such as CIC-IDS-2017 or CSE-CIC-IDS-2018, to train a more robust model.

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.

To achieve this goal, researchers conduct various studies and literature reviews to assess and improve the performance of ML-based IDS systems. They also identify potential weaknesses and gaps in existing IDS technologies.

As such, 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)* [34], *Fuzzinessbased Neural Networks (FNN)* [35], *Random Forests (RF)* [36], *Decision Trees (DT)* [37], *Robust transformer based Intrusion Detection System (RTIDS)* [31], and *Support Vector Machines (SVM)* [38].

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 as:

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

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.

F1 Score is a weighted average of precision and the detection rate (DR), 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 as:

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

where precision is defined as:

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

and detection rate is defined as:

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

AUC, or area under the curve, is a metric used for binary classification problems that measures the overall performance of a model across different threshold values. A higher AUC indicates that the model is better at distinguishing between positive and negative classes. The AUC is calculated by plotting the detection rate against the false positive rate (1 - DR) at various threshold settings and calculating the area under the resulting curve.

These metrics can be useful in determining the effectiveness of a model and identifying areas for improvement.

Among the classifiers in Table 1, *Random Forest* [39] and *Decision Trees* [40] 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.

	CIC-IDS-2017			CSE-CIC-IDS-2018			CIC-DDoS-2019			
Classifiers	**Accuracy**	**F1 Score**	**AUC**	**Accuracy**	**F1 Score**	**AUC**	**Accuracy**	**F1 Score**	**AUC**	**Ref**
LR	92.96	90.87	91.50	87.96	88.99	81.54	91.72	87.27	90.23	[28, 29]
FNN	99.61	99.57	99.83	93.00	92.00	100.00	95.55	95.50	95.63	[30, 28, 31]
RF	99.79	99.78	99.98	92.00	94.00	100.00	99.86	99.78	99.82	[26, 30, 32, 3, 33, 28]
DT	99.62	99.57	99.56	88.00	91.00	100.00	99.87	99.78	99.80	[26, 28, 30, 3]
RTIDS	99.35	99.17	98.83	-	-	-	98.58	98.48	98.66	[31]
SVM	96.97	96.99	98.98	61.00	66.00	100.00	94.02	94.98	94.24	[26, 30, 3, 33, 31]

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

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 grey/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 the most powerful kind of AML attacks because they let the attacker know everything about the IDS classifier and the training data. With this knowledge, the attacker can create very specific and complex attacks that can evade the system's defenses and achieve their goals. However, white-box attacks are also the most unrealistic kind of attack because they need the attacker to have a lot of knowledge about the system and its weaknesses, which is often impossible. In most situations, the attacker will only know some or little about the system, making white-box attacks impossible. Therefore, white-box attacks are very uncommon in reality and are usually only done in very focused and well-planned operations.

White-box attacks commonly used on IDS include the *Fast Gradient Sign Method (FGSM)* [41], *Deep-Fool* [42], *Carlini &Wagner attack (C&W)* [43], *Jacobian based Saliency Map Attack (JSMA)* [44], *Basic Iterative Method (BIM)* [45], and *Projected Gradient Descent (PGD)* [46].

3.3.2. Grey/Black-box attacks

In contrast to white-box attacks, grey/black-box attacks are more realistic because they do not require the attacker to have knowledge about the target model. However, these attacks are often less efficient compared to white-box attacks because the attacker's limited knowledge about the system restricts their ability to create highly targeted and sophisticated attacks. Instead, they have to depend on more general techniques that may be less effective in bypassing the system's defenses.

Generative Adversarial Networks (GANs) [47] are frequently used in black-box and grey-box attacks to produce adversarial examples. *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 grey-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 grey/black-box attacks on IDS include *attack-GAN* [5], *DIGFuPAS* [8], *IDSGAN* [6], *VulnerGAN* [7], *ZOO attack* [48], *Boundary attack* [49] and the *HotSkipJump attack (HSJA)* [50]. *ZOO* is a score-based attack that estimates gradients to create adversarial traffic in grey/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 grey/blackbox attacks that employ *Wasserstein-GAN* to generate adversarial traffic. *Wasserstein-GAN (W-GAN)* [51] is a *GAN* variant that trains the generator network with a different objective function called the Wasserstein distance.

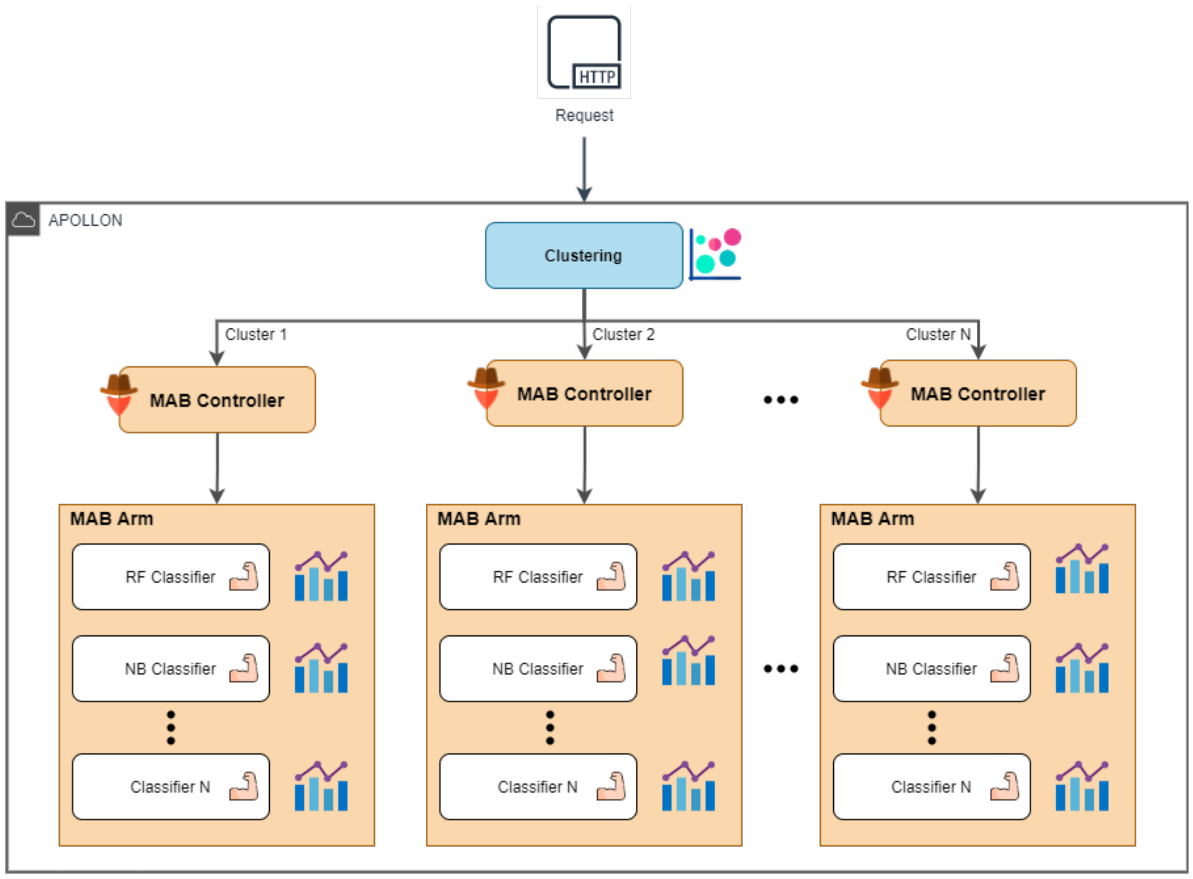

Figure 2: *Apollon* Architecture

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.

4. Apollon

In this paper, we propose a robust defence 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 that rely solely on training with adversarial traffic. The proposed system combines multiple classifiers, a *Multi-Armed Bandits (MAB)* algorithm, and requests clustering to provide robust defence against AML attacks.

The first layer of Apollon involves using multiple classifiers instead of a single classifier that is traditionally used in IDS. These classifiers are trained with adversarial traffic generated by a *W-GAN*, one of the most popular and effective grey/black-box attack methods. The idea behind training with adversarial traffic is to make the classifiers more robust against attacks by exposing them to adversarial examples during the training phase. However, training with adversarial traffic alone is not enough to provide sufficient security against AML attacks, as attackers can still extract the behaviour of the classifiers and use this information to craft more effective attacks.

To address this limitation, the second layer of *Apollon* involves using a*Multi-Armed Bandits (MAB)* algorithm to select the appropriate classifier or set of classifiers to evaluate each request. The *MAB* is responsible for selecting the arm (classifier) to use for each request based on the current state of the system. Each classifier corresponds to an *arm*, and Thomson Sampling is used to select the arm to use for each request. The idea behind using *MAB* is to make the system more robust against AML attacks by dynamically selecting the best classifier or ensemble of classifiers for each request. This layer improves the security of the IDS by making it more difficult for attackers to extract the behaviour of the classifiers.

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 group similar requests together, reducing the complexity of the classification problem and improving the accuracy of the classifiers. The idea behind using cluster-specific classifiers is to make the system more robust against attacks that target specific types of requests, as the classifiers will be better at detecting and blocking these types of requests.

With *Apollon*, we are able to maintain the performance of traditional IDSs when it comes to network traffic without

AML attacks. We achieve this by utilizing the best classifiers for each type of request. Additionally, *Apollon* offers a solution to prevent attackers from easily learning from the behavior of our classifiers through AML techniques.

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

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. These classifiers are trained using adversarial traffic generated by a *Wasserstein Generative Adversarial Network (W-GAN)*.

Training the classifiers with adversarial traffic exposes them to a wide range of attack scenarios, making them more robust against AML attacks. The *W-GAN* generates adversarial traffic that closely resembles real-world traffic but contains subtle variations that can fool traditional IDS. By training the classifiers with this type of data, they can learn to identify and block such traffic more effectively.

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.

Each of the classifiers is trained independently using adversarial traffic generated by the *W-GAN*. The training process involves iterating through the adversarial traffic and adjusting the weights of the classifiers to minimize classification error. Once the training is complete, the classifiers are ready to be used in the next layer of the *Apollon* defence system.

4.2. Multi-Armed Bandit

The second layer of the *Apollon* defence 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 Naive Bayes classifier has a high probability of being correct, but the *Random Forest* and *Multilayer Perceptron* classifiers have lower probabilities, the *MAB* algorithm will select the *Naive Bayes* 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.

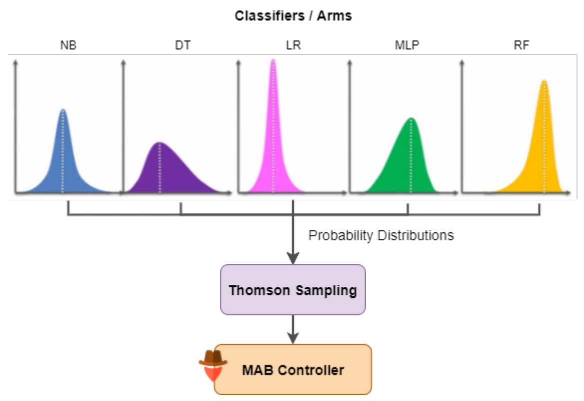

Figure 3: Apollon *Multi-Armed Bandits* algorithm with multiple classifiers

The *MAB* algorithm is constantly updating the probabilities of the different classifiers based on their previous performance. Thus, the system can adapt to changes in the traffic patterns over time, ensuring that the system is always up to date 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 algorithm chooses the arm (ML classifier) that has the highest probability of being the best according to the samples from the posterior distribution.

Algorithm 1 *Apollon* Thompson Sampling

```
Intit $S_i = 0$ y $F_i = 0$ for each arm $i$
for $t = 1, 2, \dots$ do
    For each arm $i$, sample $\theta_i$ of Beta distribution ($S_i + 1$, $F_i + 1$)
    Choose the arm $I_t$ that maximizes $\theta_i$
    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 accuracy of the classification. Figure 3 shows the diagram of the *MAB* algorithm with multiple classifiers.

4.3. Traffic requests clustering

The final layer of the *Apollon* defence system involves clustering the network traffic requests based on their features, and then training a separate version of each classifier for each cluster. By doing this, the system is able to achieve higher accuracy for each cluster by training the classifiers specifically for the traffic patterns in that cluster.

The traffic requests are clustered based on their features, such as their source IP address, destination IP address, protocol, and payload size, among others. Each cluster contains requests with similar features, which enables the system to learn the traffic patterns specific to that cluster.

For each cluster, a separate version of each classifier is trained using only the traffic requests in that cluster. This ensures that each classifier is specifically tuned to the traffic patterns in that cluster.

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 is then used to select the optimal set of classifiers for that cluster, taking into account the performance of each classifier in that specific cluster. Then, the selected classifier or set of classifiers evaluate the request and determine whether it is benign or malicious.

By clustering the network traffic requests and training separate versions of each classifier for each cluster, *Apollon* allows the *Multi-Armed Bandit* algorithm to generate multiple probability distributions for each classifier, depending on the type of request received. This combination of techniques makes it challenging for potential attackers to identify the classifier providing the response, reducing the likelihood of successful imitation.

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. 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 grey/black-box Adversarial Machine Learning attacks to evaluate the ability of our solution to defend against such attacks.

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. Performance Evaluation Metrics

In order to evaluate the performance of our proposed solution, we will use two commonly used metrics: detection rate and accuracy.

The detection rate represents the ability of the classifier to correctly identify attacks, while the accuracy represents the ability of the classifier to correctly identify both attacks and benign traffic. By using both detection rate and accuracy, we will be able to obtain a comprehensive understanding of the performance of our proposed solution. These metrics will help us to determine the effectiveness of our solution in detecting attacks while minimizing false positives.

5.1.2. 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.

The classifiers used in this scenario are the following:

- *Random Forest (RF)*
- *Decision Trees (DT)*
- *Naive Bayes (NB)*
- *Logistic Regression (LR)*
- *Multilayer Perceptron (MLP)*

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

5.1.3. Adversarial Machine Learning Attacks

In this test scenario, we used several grey/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. We compare the accuracy and the detection rate of the classifiers and our proposed solution against the grey/black-box Adversarial Machine Learning attacks. As we mentioned before, we only test with grey/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)*
- *HopSkipJump attack (HSJA)*
- *W-GAN based attacks*

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.

5.2. Results

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 *StandardScaler* and *OneHotEncoder* was utilized. The *StandardScaler* function standardized the datasets by removing the mean and scaling to unit variance, while the *OneHotEncoder* function converted categorical variables into a ML-friendly format to improve performance. These tools ensured that the datasets were in a consistent format and ready for training.

In addition to standardizing the datasets, additional steps were taken to further prepare the data for training our classifiers. One of these steps involved applying the *log1p* function, which performed a $\log(1+x)$ transformation on the data to alleviate the presence of outliers and skewed data in the datasets. Highly correlated variables were also eliminated to remove any redundant information and to prevent multicollinearity. This step was crucial in avoiding instability in our models, thus increasing their accuracy. By applying these pre-processing steps, we ensured that the data was in the best possible format for training our classifiers.

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 a 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 [52] 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.

	CIC-IDS-2017					
Metrics	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**
ROC	0.9998	0.8289	1.0000	0.9972	0.9902	**0.9420**

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

	CSE-CIC-IDS-2018					
Metrics	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**
ROC	0.9945	0.8605	0.9993	0.9984	0.9592	**0.9419**

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

	CIC-DDoS-2019					
Metrics	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**
ROC	0.9817	0.9753	0.9999	0.9999	0.9796	**0.9691**

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

The results obtained from the experiments are presented in the tables 2, 3, and 4. Based on the tables, 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.

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.

	ZOO attack					
Metrics	MLP	NB	RF	DT	LR	**Apollon**
Accuracy	0.7630	0.7160	0.7730	0.7890	0.5210	**0.9304**
Detection rate	0.5260	0.5500	0.5460	0.5780	0.0420	**0.8772**

Table 5
Results of the *ZOO* AML attack

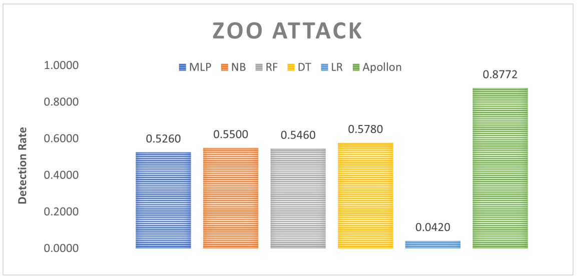

Figure 4: *ZOO* attack detection rate results

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)*, *HopSkipJump 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 [11], 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 and in the Figure 4. 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 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:

	HopSkipJump attack					
Metrics	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**

Table 6
Results of the *HopSkipJump* AML attack

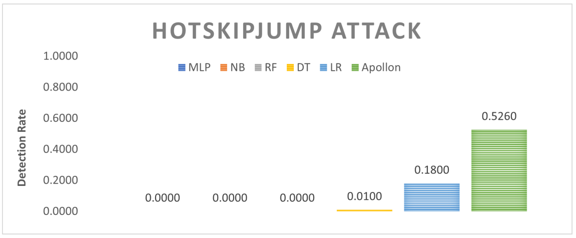

Figure 5: *HopSkipJump attack* Detection Rate results

	W-GAN based attack					
Metrics	MLP	NB	RF	DT	LR	**Apollon**
Accuracy	0.5002	0.4398	0.5001	0.4280	0.3230	**0.4260**
Detection Rate	0.0000	0.0000	0.0000	0.0000	0.0000	**0.4250**

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

- **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 and in the Figure 5. 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, which is not a good score, but still much better than the other classifiers.

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 [6], apdated to the needs of the selected dataset. The results in Table 7 and in the Figure 6 were obtained after launching the attack with 100 epochs.

The *W-GAN* based attack was the most effective attack, reducing the detection rate to zero in all the classifiers. Our solution can mantain a detection rate > 0.4, but with more epochs the attack could be more effective.

The experimental results of the scenario reveals that our solution exhibits greater robustness in comparison to the

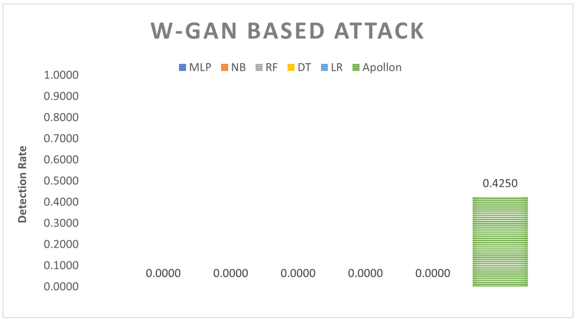

Figure 6: *W-GAN* based attack detection rate results

other classifiers used individually. *Apollon*'s accuracy scores and detection rates consistently surpass the others.

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.

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 or a combination of classifiers in real-time for each input, adding a layer of semi-randomness to the IDS behavior, that makes it more difficult for attackers to detect the IDS behavior and generate adversarial traffic.

Our experimental evaluation on several datasets shows that *Apollon* can successfully detect attacks without compromising its performance on normal inputs, and can prevent attackers from learning the IDS behavior in realistic training times. These results demonstrate that *Apollon* is an effective and efficient 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, only mitigates it increasing the time and effort required by attackers to generate adversarial traffic.

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 explore the use of other datasets, to evaluate the performance of *Apollon* in different network environments.

References

[1] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.

[2] E. E. Abdallah, A. F. Otoom, *et al.*, "Intrusion detection systems using supervised machine learning techniques: A survey," *Procedia Computer Science*, vol. 201, pp. 205–212, 2022.

[3] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE access*, vol. 9, pp. 22351–22370, 2021.

[4] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, 2011.

[5] S. Zhao, J. Li, J. Wang, Z. Zhang, L. Zhu, and Y. Zhang, "attackgan: Adversarial attack against black-box ids using generative adversarial networks," *Procedia Computer Science*, vol. 187, pp. 128–133, 2021.

[6] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 79–91, Springer, 2022.

[7] G. Liu, W. Zhang, X. Li, K. Fan, and S. Yu, "Vulnergan: a backdoor attack through vulnerability amplification against machine learning-based network intrusion detection systems," *Science China Information Sciences*, vol. 65, no. 7, pp. 1–19, 2022.

[8] P. T. Duy, N. H. Khoa, A. G.-T. Nguyen, V.-H. Pham, *et al.*, "Digfupas: Deceive ids with gan and function-preserving on adversarial samples in sdn-enabled networks," *Computers & Security*, vol. 109, p. 102367, 2021.

[9] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE network*, vol. 8, no. 3, pp. 26–41, 1994.

[10] A. Pharate, H. Bhat, V. Shilimkar, and N. Mhetre, "Classification of intrusion detection system," *International Journal of Computer Applications*, vol. 118, no. 7, 2015.

[11] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018.

[12] E. De Cristofaro, "An overview of privacy in machine learning," *arXiv preprint arXiv:2005.08679*, 2020.

[13] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.

[14] M. C. Machado, S. Srinivasan, and M. Bowling, "Domain-independent optimistic initialization for reinforcement learning," *arXiv preprint arXiv:1410.4604*, 2014.

[15] A. Carpentier, A. Lazaric, M. Ghavamzadeh, R. Munos, and P. Auer, "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*, pp. 189–203, Springer, 2011.

[16] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on learning theory*, pp. 39–1, JMLR Workshop and Conference Proceedings, 2012.

[17] H. Park and M. K. S. Faradonbeh, "Analysis of thompson sampling for partially observable contextual multi-armed bandits," *IEEE Control Systems Letters*, vol. 6, pp. 2150–2155, 2021.

[18] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 220–237, Springer, 2003.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "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, 2009.

[20] J. M. Hugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, 2000.

[21] A. M. A. Tobi and I. Duncan, "Kdd 1999 generation faults: a review and analysis," *Journal of Cyber Security Technology*, vol. 2, no. 3-4, pp. 164–200, 2018.

[22] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[23] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[24] J. Shroff, R. Walambe, S. K. Singh, and K. Kotecha, "Enhanced security against volumetric ddos attacks using adversarial machine learning," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[25] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[26] M. Pujari, Y. Pacheco, B. Cherukuri, and W. Sun, "A comparative study on the impact of adversarial machine learning attacks on contemporary intrusion detection datasets," *SN Computer Science*, vol. 3, no. 5, pp. 1–12, 2022.

[27] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019.

[28] B. Thiyam and S. Dey, "Efficient feature evaluation approach for a class-imbalanced dataset using machine learning," *Procedia Computer Science*, vol. 218, pp. 2520–2532, 2023. International Conference on Machine Learning and Data Engineering.

[29] M. Akshay Kumaar, D. Samiayya, P. M. D. R. Vincent, K. Srinivasan, C.-Y. Chang, and H. Ganesh, "A hybrid framework for intrusion detection in healthcare systems using deep learning," *Frontiers in Public Health*, vol. 9, 2022.

[30] S. Huang and K. Lei, "Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Networks*, vol. 105, p. 102177, 2020.

[31] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: a robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.

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

[33] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, (New York, NY, USA), p. 86–93, Association for Computing Machinery, 2019.

[34] R. E. Wright, "Logistic regression.," 1995.

[35] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information sciences*, vol. 378, pp. 484–497, 2017.

[36] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," *Ensemble machine learning: Methods and applications*, pp. 157–175, 2012.

[37] L. Rokach and O. Maimon, "Decision trees," *Data mining and knowledge discovery handbook*, pp. 165–192, 2005.

[38] S. Suthaharan and S. Suthaharan, "Support vector machine," *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, pp. 207–235, 2016.

[39] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[40] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 420–424, 2004.

[41] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[42] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "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, 2016.

[43] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[44] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[45] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[46] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[48] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "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, 2017.

[49] J. Chen and M. I. Jordan, "Boundary attack++: Query-efficient decision-based adversarial attack," *arXiv preprint arXiv:1904.02144*, vol. 2, no. 7, 2019.

[50] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," 2020.

[51] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[52] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.