Harpe: Using Adversarial Machine Learning to dodge Intrusion Detection Systems in Software-Defined Perimeters

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
Software-Defined Perimeters
Intrusion Detection Systems
Artificial Intelligence
Cybersecurity
W-GAN

ABSTRACT

With the increasing reliance on *Software-Defined Perimeters* (*SDPs*) for network security, it is important to understand the limitations and vulnerabilities of this architecture. This paper proposes a framework, *Harpe*, that generates attacks on *SDP*-based networks to demonstrate the potential weaknesses of this architecture. *Harpe* generates malicious network traffic that mimics normal traffic to evade the *Intrusion Detection System* (*IDS*) component of *SDPs*. Utilizing *Adversarial Machine Learning* techniques, it learns how the *IDS* operates and finds ways to evade detection. The effectiveness of the proposed framework is tested in two different environments, a traditional network architecture and an *SDP*-based network. The results show that *Harpe* is capable of significantly reducing the detection capabilities of the *IDS* in both environments, highlighting the need for further research and improvement of *SDP* security mechanisms.

1. Introduction

The rise of modern cyberattacks has highlighted the inadequacies of traditional network security methods [1, 2]. With the increasing adoption of *cloud computing*, *mobile devices*, and the *Internet of Things (IoT)*, traditional perimeter-based security models have become increasingly porous and difficult to maintain [3, 4]. This has led to a growing need for more dynamic and flexible security solutions that can adapt to the constantly changing threat landscape. *Software-Defined Perimeters (SDPs)* have emerged as a promising solution to this problem, as they aim to provide a more secure way to protect against modern cyberattacks by creating a *Zero Trust* [5] network environment. This means that access is granted only to authorized users and devices, and all network traffic is continuously monitored and analyzed for suspicious activity. This approach helps to prevent unauthorized access to sensitive data and resources, and minimizes the risk of a successful cyber attack. Additionally, *SDPs* allow for a more dynamic and flexible approach to network security, as they can easily adapt to changes in the threat landscape and new technologies [6].

To ensure the security of the infrastructure, *SDPs* must be able to detect intrusions and attacks in real time. This task is performed by *Intrusion Detection Systems (IDSs)*, which have become an integral part of several software infrastructures, as they are responsible for detecting and alerting on suspicious or malicious activity within a network or system. *IDSs* can be based on various technologies, such as signature-based approaches, anomaly-based approaches, and rule-based approaches. New classification techniques based on *Machine Learning* have led to significant improvements in the efficiency and accuracy of *IDSs* [7].

However, *IDSs* are not immune to attacks, and one increasingly common and sophisticated type of attacks are those using *Adversarial Machine Learning (AML)* [8] techniques. These attacks are a particularly dangerous type of attack for *IDSs* because they involve the manipulation of *Machine Learning* models through the injection of malicious or misleading data. This can be achieved through a variety of methods, such as the use of carefully crafted data samples known as adversarial examples, or the modification of model parameters through techniques such as *gradient descent* or *gradient-based* optimization [9].

The goal of *AML* attacks is to cause the *Machine Learning* to make incorrect decisions, which could potentially be used to evade detection or cause false alarms in the context of an *IDS*. *AML* attacks are particularly dangerous because they can be difficult to detect and defend against, as they often involve subtle changes to data that are not easily noticeable to humans.

In the context of *SDPs*, *IDSs* are more effective due to the small number of connections and requests allowed. The *Zero Trust* security model and the *SDP* architecture greatly reduces the number of connections and requests that the *IDS* has to monitor, as it only needs to focus on the small number of connections that tries to authenticate with the *SDP*. This reduction in the type of traffic allowed makes it more difficult to create attacks using *AML* techniques, making attacks that generated random noise unfeasible. However, this increased difficulty does not make *SDP*-based network architectures completely invulnerable.

In this paper, we introduce a new attack framework called *Harpe*, which targets *IDSs* in *SDP*-based networks. *Harpe* aims to show that *SDP* network architecture is not immune to current cyber threats, highlighting the need for additional research and enhancement of SDP security measures. *Harpe* is designed to bypass the security measures

*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)

of *SDPs*, by generating attacks that mimic normal network traffic. The goal of *Harpe* is to evade detection by the *IDS*, allowing an attacker to make unauthorized requests to the *SDP* and, potentially gain access to sensitive data or make denial-of-service attacks.

One of the key features of *Harpe* is its use of a *Wasserstein Generative Adversarial Network (WGAN)* with *Gradient Penalty* [10] (a variant of the *Generative Adversarial Network (GAN)* [11]) to generate malicious inputs that can fool the *IDS* into thinking they are normal network traffic. This allows *Harpe* to generate highly targeted attacks that are specific to the characteristics of the *SDP's IDS*.

To test *Harpe*, we used the open source implementation of *SDP* from Waverley Labs [1], and integrated an *IDS* with some of the most commonly used classifiers.

The structure of the paper is as follows. In Section 2 we present the background information necessary to understand the rest of the paper, including a description of *SDPs*, *IDSs*, and *AML* attacks. In Section 3 we present the related *IDS* classification techniques and *AML* attacks. Section 4 presents *Harpe* and its implementation, Section 5 presents the evaluation of our attack method, and, finally, Section 6 presents our conclusions and future work.

2. Background

In this section, we will provide an overview of the concept of *Software-Defined Perimeters (SDP)* and their role in network security. Next, we will discuss the concept of *Intrusion Detection Systems (IDSs)*, the available datasets, and their role in the context of *SDPs*. Finally, we will also discuss the basics of *Adversarial Machine Learning (AML)* and its potential for exploitation in cyberattacks. This information will provide the necessary context for the subsequent sections of this paper.

2.1. Software-Defined Perimeters

The *Software-Defined Perimeter (SDP)* is a security framework developed by the Cloud Security Alliance (CSA) to dynamically protect networks [12, 13]. The *Software-Defined Perimeter* framework was adapted from the *Global Information Grid (GIG) Black Core* initiative proposed by the USA Department of Defense (DoD) [14]. The CSA modified the generalized DoD workflow for commercial use and aligned it with National Institute of Standards and Technology (NIST) security controls. The *SDP* follows a strategy based on the *Zero Trust* security model, where the identity of the device or application that will initiate the communication is verified and authenticated before granting access to the infrastructure [15].

In this model, there is a clear separation between its two main components (*PDP* and *PEP*), which operate in the control plane and data plane respectively. The *Policy Decision Point (PDP)* includes trust and policy engines where authorization and access decisions are made based on the information provided by the *Policy Enforcement Point (PEP)*.

The architecture of SDP frameworks consists of three main components (see Figure 1).

2.1.1. SDP Controller

The *SDP Controller* is the central component of the *SDP* framework and plays a crucial role in the exchange of control messages. It acts as a trusted agent between the *SDP Initiating Host* and the backend security controls. The *SDP Controller* includes tasks for device identification and authentication, as well as determining which programs or services are authorized for each device [12, 13]. These tasks are crucial for the proper functioning of *Intrusion Detection Systems (IDS)*, which are used to identify and alert on unauthorized access or malicious activity within the network.

2.1.2. SDP Initiating Host

The *SDP Initiating Host* is the device that initiates the connection to the backend security controls. This device is responsible for sending the control messages to the *SDP Controller* and for receiving the authorization response from the *SDP Controller* [12, 13]. Once the authentication is completed, a mutual *TLS tunnel* is created that connects the client (IH) to the service or application for which it has authentication.

2.1.3. SDP Accepting Host or Gateway

Accepting Hosts are the devices that are instructed to accept certain authorized services or applications. The *SDP Accepting Host* is protected by a firewall that allows only the authorized services or applications to communicate with the *SDP Initiating Host* [12, 13].

2.2. Intrusion Detection Systems

Intrusion Detection Systems (IDS) are security technologies designed to detect and alert on unauthorized access or malicious activity within a network. They are an important tool in protecting against cyber threats, such as malware, viruses, and hacking attempts. *IDSs* can operate in real-time, analyzing network traffic as it occurs, or they can be configured to analyze logs or other stored data. They use a variety of techniques to detect intrusions, such as analyzing network traffic patterns, examining system logs, and looking for known malicious activity indicators. *IDSs* play a critical role in maintaining the security and integrity of a network, and are commonly used in conjunction with other security measures such as firewalls and antivirus software.

Machine Learning (ML) techniques have increasingly been used in *IDSs* in recent years [17, 18, 19] due to their ability to analyze large amounts of data and identify complex patterns that might not be easily detected by traditional rule-based approaches. *ML-based IDSs* are capable of learning from past data and adapting to new patterns of behavior, which makes them more effective at detecting new and evolving threats. Additionally, *ML* can be used to improve

[1]https://www.waverleylabs.com

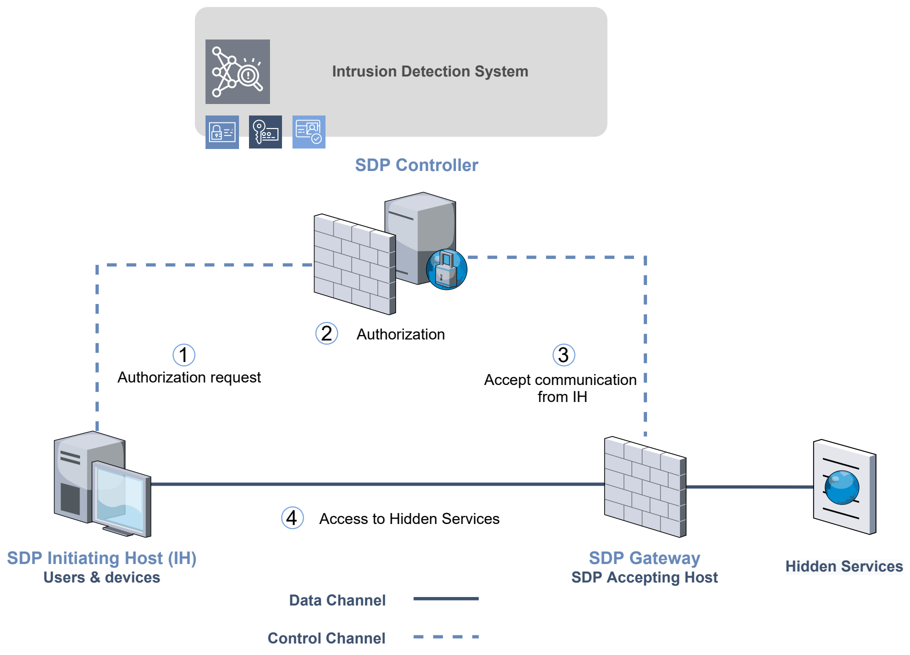

Figure 1: *Software-Defined Perimeter* architecture [16]

the accuracy and efficiency of *IDSs* by reducing the number of false positives and false negatives.

2.3. Datasets overview

Intrusion Detection Systems datasets are an important resource for evaluating and benchmarking the performance of *IDSs*. This datasets are typically collections of labeled examples of normal and abnormal network traffic, which are used to train and test the accuracy and effectiveness of *IDSs*. There are several datasets available, each with its own characteristics and strengths. In this part, we will review some of the most popular datasets and discuss their key features and uses.

2.3.1. CIC-IDS2017

The *CIC-IDS2017* dataset [20] was created in a simulated enterprise network environment and captures network traffic data over a five-day period. The data was collected to mimic the behavior of 25 users and contains approximately 80 significant attributes [21]. It is currently one of the most widely used *IDS* datasets in modern literature and contains a high ratio of benign to malicious examples (83% vs 17%). It can be used individually or in conjunction with other datasets, as it accurately reflects the normal distribution of benign and malicious traffic in a network [22].

2.3.2. CSE-CIC-IDS2018

The *CSE-CIC-IDS2018* dataset [23], created by the Canadian Institute for Cybersecurity (CIC), was collected in a simulated enterprise network environment using AWS resources in 2018. It contains data on seven different attack types and has approximately 79 significant features . The dataset includes data from over 450 devices, including servers, computers, and other devices, and is notable for its large size and high level of realism [24]. It is similar to the *CIC-IDS2017* dataset in that it includes packet analysis of network traffic with bidirectional flow, but is larger and more comprehensive. As a result, it is widely used in the literature for evaluating and benchmarking IDSs [24].

2.3.3. CIC-DDoS2019

The *CIC-DDoS2019* dataset [25] was created to address the lack of representation of all subtypes of DDoS attacks in existing datasets. While the dataset includes simulated network traffic, efforts were made to create realistic benign data. It includes 13 types of DDoS attacks and has more than 80 significant features. However, the dataset is heavily imbalanced, with 50,006,249 records of DDoS attacks and only 56,863 records of benign traffic, making it difficult to use for training a model on both types of data [21]. It is therefore recommended to use this dataset in conjunction with other datasets, such as *CIC-IDS2017* or *CSE-CIC-IDS2018*, to train a more robust model [22].

2.3.4. ADFA IDS - UNSW-NB15

The *UNSW-NB15* dataset [26] was developed at the University of New South Wales (UNSW) in Canberra by the Cyber Range laboratory. It is notable for its use of raw network packets that are a hybrid of real normal activities and synthetic contemporary attack behaviors. The dataset includes nine attack types and a total of 49 features, and is correctly labeled [21]. Unlike many other *IDS* datasets, the *UNSW-NB15* dataset is well-balanced, making it an excellent choice for training models.

2.3.5. *Unused datasets*

There are many other *IDS* datasets, such as *Darpa 1998/99* [27], *KDD 99* [28] or *NSL-KDD* [28], which have not been used in this project. The *Darpa 1998/99* and *KDD 99* datasets are no longer commonly used for evaluating and benchmarking *IDS* due to their outdated nature. These datasets were created in the late 1990s and early 2000s, and do not accurately reflect the current landscape of network threats and behaviors. In particular, the *KDD 99* dataset has been criticized for its high rate of false positives and its lack of realism, which makes it less useful for testing the performance of modern *IDS* [29, 30].

On the other hand, datasets such as *UNSW-NB15* and *CIC-IDS2017* are more widely used due to their larger size, greater realism, and more balanced distribution of benign and malicious examples .

2.4. Adversarial Machine Learning

Adversarial Machine Learning (AML) refers to the use of *Machine Learning* techniques to deceive or mislead models. According to the taxonomy of the attack and with Emiliano de Cristofaro [31], they can be classified into poisoning attacks, evasion attacks, inference attacks, and extraction attacks.

- **Poisoning attacks**: This attack involves modifying the training data in a way that causes the model to perform poorly on specific types of inputs. These attacks can be targeted at specific data points or can be more general, affecting the model's overall performance. Poisoning attacks can be difficult to detect, as the modified data can be indistinguishable from normal data.

- **Evasion attacks**: This attack involves generating inputs that are specifically designed to evade detection by the model. These inputs, known as adversarial examples, are modified versions of normal inputs that have been slightly perturbed in a way that is not easily noticeable to humans, but can cause the model to misclassify them. Evasion attacks can be conducted using a variety of techniques, such as gradient-based optimization or *ML*.

- **Inference attacks**: This attack consists of inverting the sense of the information in a Machine Learning model. The purpose is to get information from the model, which was not intended to be shared. There are three main sub-types of inference attack, *Membership Inference Attack (MIA)* which is to discover whether or not a piece of data has been used in training, *Property Inference Attack (PIA)* which attempts to infer model parameters and statistics, and Data Reconstruction which attempts to replicate the training samples.

- **Extraction attacks**: This attack is based on obtaining data from Machine Learning models. It uses a series of requests to the model, which allows it to intuit or predict the behavior of a model. These attacks can be easily detected as they need to send many suspicious requests. This type of attack it is intended to reveal the inner workings of the model, once enough knowledge is obtained, the model can be replicated or profit can be made from this information.

In the context of *IDSs*, *AML* can be used to attack the *IDS* by generating adversarial examples that are designed to evade detection. These attacks can be difficult to defend against [32], as they rely on the ability to generate inputs that are specifically designed to deceive the *IDS*. Some of the most common evasion attacks types used in *IDS* are the *Model extraction attacks* and the *Input reconstruction attacks*. In this context, and depending on the level of knowledge of the attacker, adversarial attacks can be classified into three types:

- **White-box attacks**: In a white-box attack, the attacker has complete access to all information about the ML-based *IDS*, including the training data and learning model architecture, decision and parameters (gradient, loss function, etc.). This gives the attacker a significant advantage, but fortunately, this type of attack is generally not practical in most real-world situations.

- **Black-box attacks**: In a black-box attack, the attacker has no knowledge of the *IDS* model architecture, parameters, or training data.

- **Gray-box attacks**: Gray-box attacks involve an attacker who has some level of knowledge about the ML-based *IDS* and may have limited access to the training data. In this scenario, the attacker does not have complete information about the system, but has enough information to attack it and cause it to fail. This type of attack is considered more realistic because it takes into account the fact that an attacker may have partial knowledge of the system they are targeting.

It is worth noting that in academic literature, the term "black-box attacks" is sometimes used to refer to "gray-box attacks". In this article, we use the terms "gray/black-box" to refer specifically to the type of gray-box attacks described above, to distinguish them from the more general use of the term "black-box" in the literature.

3. Related Work

In this section, we will review the different *IDS* classifiers and their scores for the selected datasets. These classifiers will be utilized later to simulate the *SDP IDS*. Furthermore, we will detail the various types of *AML* attacks that are employed in this context.

3.1. Intrusion Detection Systems classifiers

As the number and sophistication of malware threats continue to grow, it is increasingly important to have effective *IDS* in place to protect network systems. To ensure the effectiveness of *IDS* systems, researchers conduct a range

Classifier	Accuracy	F1 score	AUC	Ref
IGAN-IDS	99.79	99.79	99.98	[36]
ANN	99.28	99.17	99.28	[19]
Fuzziness-based NN	99.61	99.57	99.83	[36, 37]
CNN	99.48	99.44	99.94	[36, 19]
RTIDS	99.35	99.17	98.83	[37]
MLP	99.46	99.46	99.41	[24, 38]
ROS + MLP	99.55	99.55	99.88	[36]
Random Forest	99.79	99.78	99.98	[24, 36, 39, 19, 40]
Decision Trees	99.62	99.57	99.56	[24, 36, 19]
SMOTE + SVM	97.00	97.04	98.97	[36]
SVM	96.97	96.99	98.98	[24, 36, 19, 40, 37]
ROS + SVM	96.98	97.04	98.96	[36]
RUS + SVM	96.45	96.55	98.96	[36]
Naive Bayes	93.90	93.53	97.49	[36, 19, 40]

Table 1
Performance of the *IDSs* classifiers on the *CIC-IDS2017* dataset.

Classifier	Accuracy	F1 score	AUC	Ref
MLP	62.00	89.00	100.00	[24, 38]
Random Forest	92.00	94.00	100.00	[24, 36, 39, 19, 40]
Decision Trees	88.00	91.00	100.00	[24, 36, 19]
SVM	61.00	66.00	100.00	[24, 36, 19, 40, 37]

Table 2
Performance of the *IDSs* classifiers on the *CSE-CIC-IDS2018* dataset.

Classifier	Accuracy	F1 score	AUC	Ref
Fuzziness-based NN	95.55	95.50	95.63	[36, 37]
RTIDS	98.58	98.48	98.66	[37]
SVM	94.02	94.88	94.24	[24, 36, 19, 40, 37]

Table 3
Performance of the *IDSs* classifiers on the *CIC-DDoS2019* dataset.

of studies and literature reviews to identify and address potential vulnerabilities. These studies are ongoing and are crucial in the ongoing development of *IDS* systems to keep pace with the evolving threat landscape.

Tables 1, 2, 3, and 4 show the *IDS* classifiers and their scores (accuracy, F1, and AUC) for the *CIC-IDS2017*, *CSE-CIC-IDS2018*, *CIC-DDoS2019*, and *UNSW-NB15* datasets, respectively. In these datasets and the most commonly used datasets (*KDDD 99* and *NSL-KDD*), the most commonly used and best-performing classifiers are *Random Forest*, *Decision Trees*, and *SVM*. *Random Forest* [33], in particular, has gained popularity due to its ability to handle large datasets and its robust performance when dealing with noise and missing values in the data. *Decision Trees* [34] are also popular due to their interpretability and simplicity, as they allow for clear visualization of the decision-making process. *SVMs* [35] are favored for their ability to handle high-dimensional data and their versatility in dealing with different types of classification tasks. Overall, these classifiers have demonstrated strong performance and are frequently used in the field of *IDS*.

3.2. Adversarial Machine Learning Attacks

IDSs that use *AI* are more effective at detecting threats than those that rely on traditional rule-based approaches, but they are also more vulnerable to adversarial attacks. These

Classifier	Accuracy	F1 score	AUC	Ref
IGAN-IDS	82.53	82.86	97.09	[36]
Fuzziness-based NN	81.21	78.58	97.02	[36, 37]
CNN	80.52	76.61	96.72	[36, 19]
MLP	72.00	73.00	90.00	[24, 38]
ROS + MLP	76.13	76.97	96.29	[36]
Random Forest	88.86	77.28	95.18	[24, 36, 39, 19, 40]
Decision Trees	73.52	76.36	86.56	[24, 36, 19]
SMOTE + SVM	71.50	73.77	94.44	[36]
SVM	68.49	70.13	95.15	[24, 36, 19, 40, 37]
ROS + SVM	68.32	70.00	95.08	[36]
RUS + SVM	67.16	70.45	94.98	[36]
Naive Bayes	61.80	65.27	90.13	[36, 19, 40]

Table 4
Performance of the *IDSs* classifiers on the *UNSW-NB15* dataset.

attacks involve manipulating the input data used to train the *AI* model in such a way that it fails to correctly identify a threat. *Adversarial Machine Learning* attacks can be difficult to detect and prevent, which makes them attractive to attackers. Additionally, as *AI* techniques become more widely used in cyber security, there is a growing incentive for attackers to develop more sophisticated adversarial attacks to evade detection.

3.2.1. White-box Attacks

The *white-box* attacks are the most effective type of *AML* attacks because they allow the attacker to have full knowledge of the *IDS* classifier and the training data. This level of knowledge allows the attacker to craft highly targeted and sophisticated attacks that can bypass the system's defenses and achieve their desired outcome. However, *white-box* attacks are also the least realistic type of attack because they require the attacker to have extensive knowledge of the system and its vulnerabilities, which is often not feasible. In most cases, the attacker will only have partial or limited knowledge of the system, making *white-box* attacks infeasible. As a result, *white-box* attacks are relatively rare in practice and are typically only used in highly targeted and carefully planned operations.

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

The *FGSM* attack generates adversarial examples by adding a small perturbation to the input data using the *gradient-based* method. *BIM* and *PGD* are variants of the *FGSM* attack that iteratively add perturbations to the input data to generate more effective adversarial examples. *DeepFool* is a type of untargeted attack that involves calculating the minimum distance between the original input and the decision boundary of a machine learning model. The *JSMA* is a method for generating adversarial examples that uses the model's *Jacobian*, or forward derivatives, to iteratively perturb the features or components of the input. Finally, the *C&W* attack is a method for generating adversarial examples by formulating the search for an adversarial sample as an optimization problem. The objective of this optimization problem is to minimize the distance between the original input and the adversarial example, while also ensuring that the model incorrectly classifies the adversarial example.

3.2.2. Grey/Black-box Attacks

Grey/Black-box are more realistic than *white-box* attacks because they do not require the attacker to have full knowledge of the *IDS* classifier and the training data. However, *grey/black-box* attacks are generally less efficient than *white-box* attacks because the attacker's lack of knowledge about the system limits their ability to craft highly targeted and sophisticated attacks. Instead, the attacker must rely on more general techniques that may be less effective at bypassing the system's defenses.

Many *grey/black-box* attacks use *Generative Adversarial Networks (GANs)* [11] to generate adversarial examples. *GANs* are a type of *Machine Learning* model that consists of two neural networks: a generator network and a discriminator network. The generator network is trained to produce synthetic data that is similar to the real data, while the discriminator network is trained to distinguish between real and synthetic data.

In the context of *grey/black-box* attacks, the generator network can be used to produce adversarial examples that are designed to bypass 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 used to guide the training of the generator network and improve its ability to produce effective adversarial examples.

Some of the most recent *grey/black-box* attacks on *IDS* are *attackGAN* [47], *DIGFuPAS* [48], *IDSGAN* [49], *VulnerGAN* [50], *Zeroth-order optimization (ZOO) attack* [51], *Boundary attack* [52], and *Hop Skip Jump Attack (HSJA)* [53].

ZOO is a *grey/black-box* score-based attack that directly estimates the gradients for generating adversarial traffic. *Boundary attack* and *HSJA* are *grey/black-box* decision-based attacks that only knows a binary output indicating whether the input was accepted or rejected.

The *IDSGAN*, *attackGAN*, and *DIGFuPAS* are *grey/black-box* attacks that use *Wasserstein GAN (WGAN)* to generate adversarial traffic.

Wasserstein GAN (WGAN) [10] is a variant of *GANs* that uses a different objective function for training the generator network. *WGAN* introduces a new objective function called the *Wasserstein* distance, which is used to train the generator network. The *Wasserstein* distance is a measure of the distance between two probability distributions and has several desirable properties, such as being smooth and continuous. Some of the most recent *WGANs* uses the *Gradient Penalty* [10] to improve the convergence of the training process.

However, attacks with *WGANs*, like *IDSGAN*, generate adversarial examples by training a generator with random noise as input. This approach is not effective because the generator network alters key characteristics of an attack in an effort to evade detection [54]. This is not a viable tactic, as these characteristics are crucial for the attack to be successful.

4. Harpe

This research project proposes a new framework for generating malicious network traffic in *Software-Defined Perimeter (SDP)* based networks called *Harpe*. *Harpe's* goal is to demonstrate that, although *SDP*-based networks are much more secure than traditional networks, they are not invulnerable and, further research is needed on how to improve them. To demonstrate this, *Harpe* generates malicious traffic that looks enough like normal traffic to fool the *Intrusion Detection System (IDS)* located in the *SDP Controller*. This *IDS* is responsible for filtering traffic directed to the *SDP Controller* (the only accessible element of the architecture) and allowing only authorization and authentication requests, blocking possible attacks such as denials of service, among others.

Generating malicious traffic that can fool an *SDP's IDS* is not a simple task. Creating network traffic that closely resembles normal traffic to evade detection can be difficult, as the limited types of requests that are allowed make it challenging to create variations without being detected. Furthermore, generating malicious traffic requires specific characteristics that are associated with successful attacks. These characteristics are not always easy to identify, as they can vary depending on the type of attack and the *IDS* being targeted.

4.1. Harpe architecture

The architecture of *Harpe* is depicted in Figure 2. In this architecture, The *W-GAN* with Gradient Penalty is composed of two key elements: the *Traffic Generator* and the *Discriminator*.

In order to generate malicious traffic data, the *Harpe* framework requires access to two types of datasets: one containing benign or normal *SDP* traffic samples and another containing samples of known attack traffic. A feature selection process is performed on the attack traffic samples to reserving the functional characteristics of the traffic by taking into consideration the method in Usama's study [54]. To generate adversarial traffic, the *Traffic Generator* utilizes normal traffic as a base, and incorporates noise derived from chosen features of known attack traffic. The generated traffic is sent to the *Grey/Black-box IDS*, which only provides information on whether or not the traffic is detected as an attack. The generated traffic already labeled as benign or attack is passed to the *Discriminator*. The *Discriminator* is a multi-layer neural network that is trained to distinguish between the normal traffic and the generated adversarial traffic. Also, the *Discriminator* is trained to learn and imitate the *IDS*'s decision process based on the *IDS*'s output. Finally, the *Discriminator* sends feedback to the *Traffic Generator*, whose gradient is back-propagated from the *Discriminator* to improve the quality of the generated traffic.

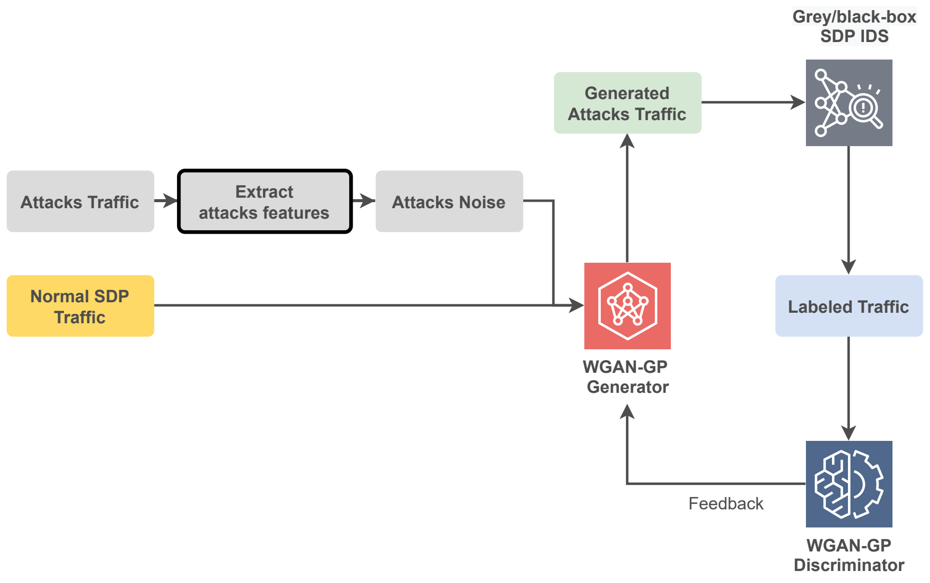

Figure 2: *Harpe* architecture.

4.1.1. *Feature selection*

In order to generate attacks on *IDS* in network environments we have used the *CIC-IDS2017*, *CSE-CIC-IDS2018*, *CIC-DDoS2019*, and *UNSW-NB15 IDS* datasets. These datasets contain benign and the most up-to-date common attacks, which resemble the true real-world data (as *PCAPs* files). To generate the CSV files with which the framework works, we have used the *CICFlowmeter-V4.0* [55] tool from the Canadian Institute For Cybersecurity. The *CICFlowmeter-V4.0* is a tool that extracts features from *PCAP* files and generates CSV files as output.

After acquiring the necessary data, the next step is to identify the characteristics of the attacks that we want to maintain during the generation of adversarial traffic. The process of extracting features from traffic records for *IDSs* is thoroughly discussed by Lee et al [56]. The authors of this work have developed a four-layer feature extraction scheme that is based on the nature of the attack found in network traffic flows. These four steps of feature extraction are *Intrinsic* features, *Time-based* features, *Host-based* features, and *Content-based* features.

The first step in the feature extraction process involves identifying *intrinsic* features of the network traffic flow. These features are essential for determining the validity of the traffic and any alteration in these features will render the traffic invalid. The second step in the feature extraction process involves extracting *time-based* features. Any changes to the *time-based* features of network traffic flow will invalidate the network traffic characteristics. Then, the *host-based* traffic features were extracted and, in combination with *intrinsic* and *time-based* features, are necessary for detecting *slow-probe* attacks. Any changes to these features will not only alter the *host-based* information, but also invalidate the traffic flow. Finally, the content-based features are not required for denial of service attacks. Any alteration in these features will not invalidate the attack. In this framework, have used the content-based features to generate adversarial traffic.

4.1.2. *Data Generation*

$$Gen_{Loss} = \mathbb{E}_{M \in normal\,Data, N} Dis(M, N_{attacks}))) \quad (1)$$

The *Traffic Generator* is a neural network with five linear layers, each with a *ReLU* activation function. *ReLU* is a rectified linear unit activation function that is used to add non-linearity to the network ($ReLU(x) = \max(0, x)$). The generator loss is calculated based on the *Discriminator*'s output, which reflects the similarity between the generated adversarial traffic and the normal traffic (see Equation 1). The generator training performs updates to the generator's weights based on the gradient information from the similarity score. The final goal is to minimize the generator loss.

$$Dis_{Loss} = \mathbb{E}_{s \in P_{normal}} Dis(s) + \mathbb{E}_{s \in P_{attack}} Dis(s) \quad (2)$$

The *Discriminator* is a neural network with five linear layers, each with a *LeakyReLU* activation function ($LeakyReLU(x) = \max(0.01x, x)$). The discriminator loss is calculated based on the results of the *Grey/Black-box IDS*'s output, which labels the generated adversarial traffic as either benign or attack (see Equation 2). The P_{normal} and P_{attack} are the normal and attack traffic samples predicted by the *IDS*, respectively.

The *Generator* and *Discriminator* architectures are described in Table 5, and the *W-GAN* hyperparameters are described in Table 6.

4.1.3. *Grey/Black-Box IDS*

The *grey/black-box IDS* in our framework represents the *ML-based IDS* used by the *SDP Controller* to filter incoming

Operation	Units	Non-linearity
Traffic Generator (Gen)		
Linear	*input_dim*	ReLU
Linear	*input_dim//2*	ReLU
Linear	*input_dim//2*	ReLU
Linear	*input_dim//2*	ReLU
Linear	*input_dim//2*	ReLU
Discriminator (Dis)		
Linear	*input_dim*	ReLU
Linear	*input_dim* ∗ 2	ReLU
Linear	*input_dim* ∗ 2	ReLU
Linear	*input_dim* ∗ 2	ReLU
Linear	*input_dim//2*	ReLU

Table 5
Architecture of the *Traffic Generator* and *Discriminator*.

Hyperparameters	
Optimizer	RMSProp
Learning Rate	0.0001
Batch Size	128
Iterations	150
Weight Clipping	0.01
Discriminator Iterations	5

Table 6
W-GAN hyperparameters.

requests. *Harpe* only needs a binary response on the data it generates to label them as detected or undetected attacks. To simulate this *IDS*, we have used some of the most widely used and best performing classifiers from related work.

4.2. Integration with the SDP

The *Harpe* framework is integrated with the *SDP* as shown in Figure 3. The attacker will correspond to the *SDP IH* in the *SDP* network architecture. The *SDP IH* will generate the adversarial traffic using the *Traffic Generator* and send it to the *SDP Controller*. Next, the *IDS* will filter the incoming traffic and *Harpe* will label the samples if they are detected or undetected attacks. The *Discriminator* will be trained based on the *IDS*'s output and send the updated weights to the *Traffic Generator*. Once we have a trained *Traffic Generator*, we can generate adversarial traffic that is similar to the normal traffic and is undetected by the *IDS*.

In a typical *SDP* architecture, the *SDP IH* needs to authenticate with the *SDP Controller* to gain access to the requested resources. The only type of request in the *SDP* network that can be attacked is the authentication step. Authenticated requests between an *SDP IH* and an *SDP AH* typically have a short period of validity, making it difficult for prolonged attacks such as denial of service *(DoS)* to occur. It should be noted that, in an *SDP* network, there are two types of situations that make it difficult to carry out attacks on the network.

- The network is fully enclosed, where the components are fixed and cannot be altered. As a result, no external devices can initiate requests and to launch an attack, one must be part of the network. In this case, the locations and values of the *SDP* IHs must be fixed and maintained throughout their lifetime in order to be connected to the network.
- The *SDP* network allows for some flexibility in the *SDP IH*, with the *SDP Controller* being responsible for validation and verification. In this case, the *SDP IH* can be moved to a different location, but the *SDP Controller* must validate the new location. This case is the selected scenario for the *Harpe* framework as it allows more requests types to train the *Traffic Generator*.

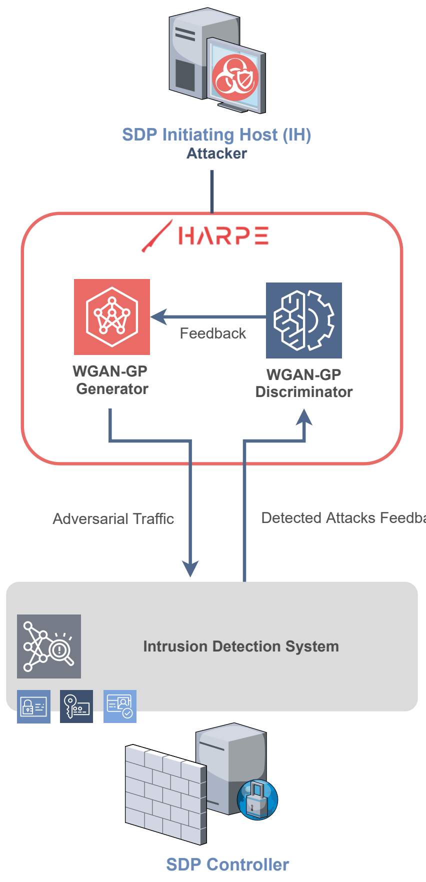

Figure 3: *Harpe* integration with the *SDP*.

5. Evaluation

This section introduces the preparation of our experiments to evaluate our proposed framework. For this purpose, we have deployed an *SDP* network, generated traffic simulating normal behavior to train an *IDS* model and trained our framework to generate adversarial traffic.

The code that was used and created during the evaluation of our proposal is completely open and accessible. It can be found on the popular code sharing and collaboration platform, *GitHub* [2].

[2]https://github.com/antonioalfa22/Harpe

5.1. Methodology

To evaluate the performance of our proposed framework, we have designed two environments to test its ability to generate adversarial traffic that can fool an ML-based *IDS*. Additionally, we have compared our solution with other existing *grey/black-box* attacks.

To evaluate the performance of our solution, we first employed *IDS* datasets from previous studies that were used in traditional network architectures in our test environment. This approach allows us to assess our solution's ability to generate adversarial traffic that can evade detection by existing classifiers.

Secondly, we have designed a test scenario that simulates the behavior of a *SDP* network. The objective of this evaluation environment is to assess the ability of our solution to generate adversarial traffic that can evade detection by the *IDS* in the *SDP Controller*. This will provide a comprehensive understanding of the effectiveness of our solution in modern network environments and its potential to circumvent security mechanisms.

Finally, we have compared our solution with some of the most popular *grey/black-box* attacks in the field. This comparison will allow us to assess the performance of our solution in comparison to other existing solutions.

In this study, we have specifically focused on evaluating the performance of our solution against denial of service *(DoS)* attacks. This is because *DoS* attacks are a prevalent type of attack and are present in all of the datasets used in our analysis. Despite the presence of other types of attacks in these datasets, focusing on *DoS* attacks allows us to have a specific and consistent benchmark to evaluate our solution's performance.

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. Evaluation Metrics

In order to thoroughly assess the effectiveness of our proposal, we have selected two metrics that were utilized in a previous study, called *IDSGAN* [49]. These metrics are known as the *Detection Rate (DR)*(described in equation 3) and the *Evasion Increase Rate (EIR)* (described in equation 4). By utilizing these metrics, we are able to evaluate the performance of *Harpe*, both directly and comparatively. This allows us to gain a comprehensive understanding of how well our proposal performs in comparison to other similar methods, and how well it performs overall. By using these metrics, we can ensure that our proposal is evaluated in a rigorous and objective manner, which will help us identify areas for improvement and make necessary adjustments for optimal performance.

$$DR = \frac{AttacksTP}{AttacksTP + AttacksFN} \tag{3}$$

$$EIR = 1 - \frac{AdversarialDetectionRate}{NormalDetectionRate} \tag{4}$$

5.1.2. Traditional IDS Environment

In the traditional environment test scenario, we utilized the *CIC-IDS2017*, *CSE-CIC-IDS2018*, *CIC-DDoS2019*, and *UNSW-NB15 IDS* datasets to train a set of classifiers. Our goal is to further train *Harpe* using data from these datasets in order to showcase its capability to generate adversarial traffic and evaluate its performance against other existing solutions in the field.

The classification algorithms used for the *grey/black-box IDS* are *Support Vector Machines (SVM)*, *Naive Bayes (NB)*, *Multilayer Perceptrons (MLP)*, *Logistic Regression (LR)*, and *Random Forests (RF)*.

5.1.3. SDP IDS Environment

After demonstrating the efficacy of our solution in generating adversarial traffic that can evade detection by *grey/black-box IDSs* in traditional network environments, it is necessary to extend the evaluation of our solution to its ability to exploit vulnerabilities in *Software-Defined Perimeter (SDP)*-based networks.

In order to evaluate the performance of our solution in a *SDP*-based network architecture, we have designed a test scenario that simulates the behavior of a *SDP* network using the Waverley Labs implementation. This test scenario consists of a *SDP* network that is composed of a *SDP Controller*, two *SDP Initiating Hosts (IHs)*, and one *SDP Accepting Host (AH)*.

We have deployed a traffic generator to generate normal traffic in the *SDP* network. This traffic generator is responsible for generating traffic that simulates the behavior of a *SDP* network. Additionally, we have included the *CICFlowmeter-V4.0* tool to generate the *grey/black-box IDS* dataset from the traffic generated.

Finally, we have trained our *grey/black-box IDS* with the *Support Vector Machines (SVM)*, *Naive Bayes (NB)*, *Multilayer Perceptrons (MLP)*, *Decision Trees (DT)*, *Logistic Regression (LR)*, and *Random Forests (RF)* classifiers.

5.1.4. Comparison with existing AML attacks

In order to evaluate our solution's performance against other existing solutions, we compared *Harpe* with the *IDSGAN*, *Zeroth-order optimization (ZOO) attack*, and *Hop Skip Jump Attack (HSJA)* attacks.

We chose these attacks because they are some of the most popular *grey/black-box* attacks in the *IDS* context, and have been shown to generate adversarial traffic that can evade detection by *IDSs* in traditional network environments.

We did not compare our solution with other *grey/black-box* attacks because *IDSGAN* allows us to evaluate our solution's performance against other similar *WGAN*-based attacks, such as *attackGAN* or *DIGFuPAS*.

In this evaluation scenario, we have only used the *CIC-IDS2017* dataset to train the *grey/black-box IDS* and generate the adversarial traffic. This is because the *CIC-IDS2017* dataset is the only dataset that is used by all of the attacks that we are comparing our solution with.

	Accuracy (%)		Detection Rate (%)		
	Origin	Adversarial	Origin	Adversarial	EIR
MLP	78.16800	45.15392	87.23630	0.00000	1.00000
SVM	85.52806	40.46552	82.57736	0.00000	1.00000
NB	72.22800	39.49882	75.73822	0.00000	1.00000
LR	85.47430	40.57214	82.59692	0.81922	0.99014
RF	74.81544	42.18392	80.70917	0.00000	1.00000

Table 7
Performance of the *IDSs* classifiers with adversarial traffic on the *UNSW-NB15* dataset.

	Accuracy (%)		Detection Rate (%)		
	Origin	Adversarial	Origin	Adversarial	EIR
MLP	89.70249	43.17501	97.80681	0.00000	1.00000
SVM	93.95127	48.35813	95.48084	0.00000	1.00000
NB	97.83296	47.85404	95.74709	0.00000	1.00000
LR	92.30762	47.07897	93.84646	0.00000	1.00000
RF	97.92636	54.34136	91.54804	0.00000	1.00000

Table 8
Performance of the *IDSs* classifiers with adversarial traffic on the *CIC-IDS2017* dataset.

5.2. Results

Below, we present the results of our experiments in the two evaluation environments. The functional characteristics used by *DoS* attacks are intrinsic and time-based, namely *Flow Duration, Active Mean, Average Packet Size, Packet Length Std, Flow IAT Mean, PSH Flag Count, Idle Max* [23, 54].

5.2.1. Traditional IDS Environment Results

Prior to the training of the *IDS* models, we performed a common pre-processing step on the data from all datasets. This step is crucial as it allows us to standardize the datasets, ensuring that we are working in the same way with each of them. To accomplish this, we use a combination of *sklearn* functions such as *StandardScaler* and *OneHotEncoder*. The *StandardScaler* function is used to standardize our datasets by removing the mean and scaling to unit variance. Meanwhile, the *OneHotEncoder* function is used to convert categorical variables into a format that can be provided to machine learning algorithms to improve performance. These tools help us to ensure that our datasets are in a consistent format and ready for training. In addition to standardizing the datasets, we also perform additional steps to further prepare the data for training our *IDS* models. One of these steps is to apply the *log1p* function, which is used to perform $\log(1+x)$ transformation on the data. This transformation is applied to help alleviate the presence of outliers and skewed data in our datasets. Additionally, we also eliminate highly correlated variables. This is done to remove any redundant information and to prevent multi-collinearity. This is an important step, as highly correlated variables can lead to instability in our models, thus reducing their accuracy. By applying these additional pre-processing steps, we are able to ensure that the data is in the best possible format for training our *IDS* models.

In this study, we begin our analysis with the *UNSW-NB15* dataset. The results of this analysis can be observed

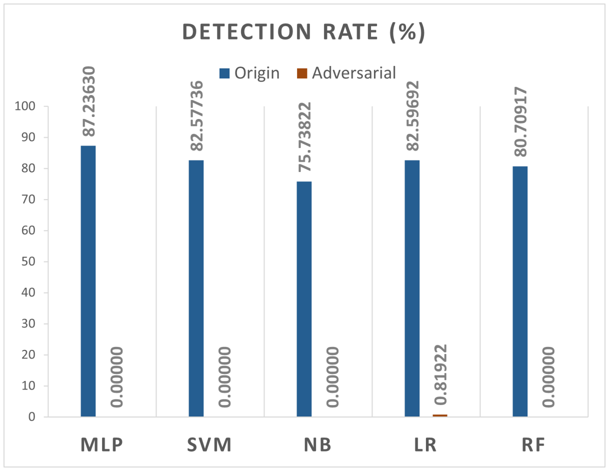

Figure 4: *UNSW-NB15* detection rate.

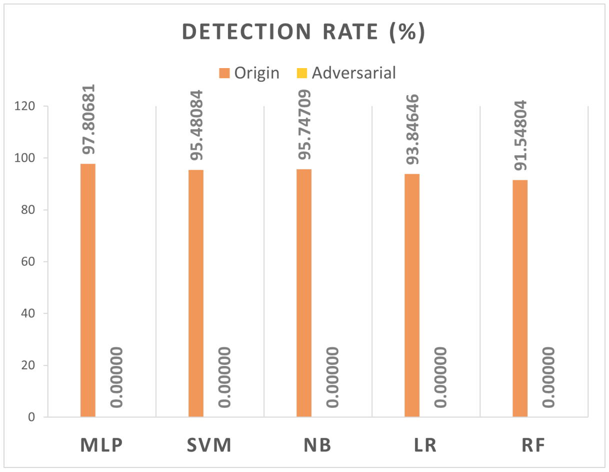

Figure 5: *CIC-IDS2017* detection rate.

	Accuracy (%)		Detection Rate (%)		
	Origin	Adversarial	Origin	Adversarial	EIR
MLP	89.03803	47.04617	94.76926	0.03464	0.99963
SVM	95.45637	45.35497	91.99673	0.00000	1.00000
NB	96.72028	48.55712	92.82482	0.00000	1.00000
LR	91.52872	44.39624	97.91123	0.47354	0.99516
RF	98.20424	56.16009	98.97865	0.00000	1.00000

Table 9
Performance of the *IDSs* classifiers with adversarial traffic on the *CSE-CIC-IDS2018* dataset.

in Table 7 and in the Figure 4. From the table, it is clear to see that the *Detection Rate (DR)* and *Evasion Increase Rate (EIR)* metrics are quite high. This indicates that almost all the adversarial traffic generated with our *Harpe* framework is able to fool all the *IDS* models, and is able to evade detection as an attack.

In the *CIC-IDS2017* (Table 8 and Figure 5), *CSE-CIC-IDS2018* (Table 9 and Figure 6) and *CIC-DDoS2019* (Table 10 and Figure 7) datasets, as in the previous dataset, we can see how by generating adversarial network traffic with the *Harpe* framework we greatly reduce the *Detection Rate*

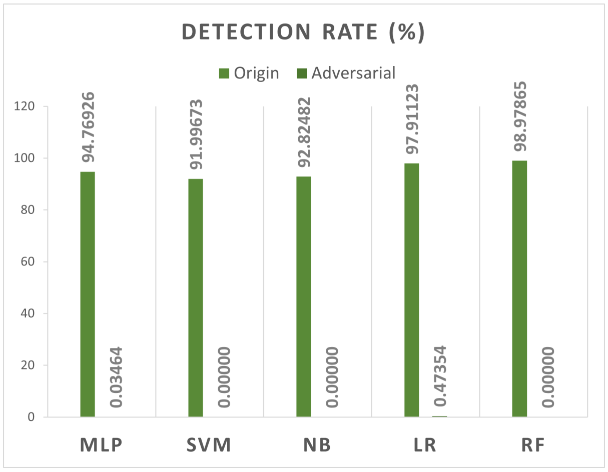

Figure 6: *CSE-CIC-IDS2018* detection rate.

	Accuracy (%)		Detection Rate (%)		
	Origin	Adversarial	Origin	Adversarial	EIR
MLP	87.73115	48.70946	87.06869	0.00000	1.00000
SVM	85.45995	44.68655	84.43590	0.00000	1.00000
NB	86.16776	47.13445	85.14369	0.00000	1.00000
LR	83.96291	45.23399	85.20754	0.00000	1.00000
RF	91.42235	46.93493	84.20779	0.00000	1.00000

Table 10
Performance of the *IDSs* classifiers with adversarial traffic on the *CIC-DDoS2019* dataset.

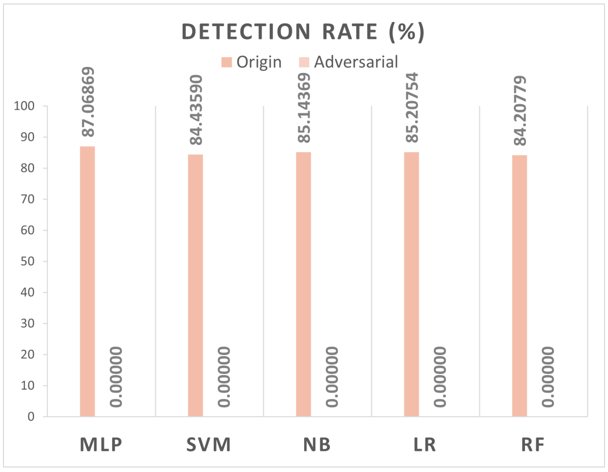

Figure 7: *CIC-DDoS2019* detection rate.

and obtain values very close to the maximum in the *Evasion Increase Rate*. As a consequence, the *accuracy* of the IDS classification algorithms is greatly reduced.

These results indicate that the *Harpe* framework is able to successfully create adversarial attacks on network traffic that are practically undetectable by the majority of traditional *IDS* systems, as well as other solutions such as *IDSGAN*, *attackGAN*, and *DIGFuPAS*. This is a significant achievement as it highlights the framework's ability to bypass the security measures put in place by these systems.

5.2.2. SDP IDS Environment Results

In order to test our solution in an *SDP-based* network environment, we needed to generate our dataset. To do this, we developed a tool that simulates traffic in an *SDP network* from two *SDP Initiating Hosts (IH)* to one *SDP Accepting Host (AH)* using an *Apache* web server. In an *SDP* network, a connection between an *SDP IH* and an *SDP AH* must be authorized by the *SDP Controller* prior to any communication taking place. Because of this, we only stored authentication requests in our dataset, as they are the only type of requests that can be targeted by attacks.

In general, once a connection is authorized, the *SDP IH* is authorized to access the *SDP AH* for as long as the *SDP controller* specifies or until the *SDP IH* changes some of its characteristics, such as IP address, geolocation, etc. This results in very few authorization requests for the *SDP IH*. This results in very few authorization requests for a test environment such as the one in this project, where there are only two *SDP IHs*. For this reason, we have configured the *SDP Controller* so that the *SDP IH* authorization only lasts for one request. This way the *SDP IH* has to make an authorization request for each request it wants to make to the *SDP AH* web server.

Once we can generate normal or benign traffic, we need to generate attack traffic to train the *grey/black-box IDS*. To generate this attack traffic, we used the *Slowloris* [57] attack. *Slowloris* is a type of *DoS* attack that is used to target web servers. The attack works by consuming all of the available connections on a web server, making it unable to serve legitimate requests. This is accomplished by the attacker opening a large number of connections to the web server and keeping them open for as long as possible.

The generated *SDP*-based network traffic dataset is completely open and can be found in the *GitHub* repository. The dataset is composed of two files in *CSV* format, one with the benign traffic and the other with the traffic generated when executing a *Slowloris DoS* attack. The dataset has 84 network flow features extracted from the *pcap* files with full packet payloads using the *CICFlowmeter-V4.0* tool. The subset with the benign data has 2193 authentication requests made from two *SPD IHs*. The subset with malicious data has 1203 requests made on one of the *SPD IHs* without prior authentication.

Table 11 describes the results obtained after training the ML-based *IDS* for the *SDP-based* network and their *Detection Rates (DR)* for data generated with *Harpe*. In this scenario, we were only able to achieve detection rates close to 50% during the training of the *IDS* classification models. This may be due to the small amount of data available, as well as to the small variation between them (only two devices make authentication requests, and one of these devices is the one that subsequently performs the attacks). However, in real environments and with a larger number of devices, we assume that *IDSs* can achieve much higher *DR* scores.

In this environment, *Harpe* is able to significantly reduce the ability of an *IDS* to distinguish whether a request is a *DoS*

	Accuracy (%)		Detection Rate (%)		
	Origin	Adversarial	Origin	Adversarial	EIR
MLP	76.35870	49.68038	52.71739	20.03842	0.61989
SVM	72.75557	64.00435	37.43255	17.10665	0.54300
NB	56.04148	55.26820	41.92895	1.25832	0.96999
LR	68.30679	45.23399	40.49852	8.04726	0.80130
RF	75.28038	57.34791	55.13747	29.51201	0.46476

Table 11
Performance of the *IDSs* classifiers with adversarial traffic on the generated *SDP*-based network dataset.

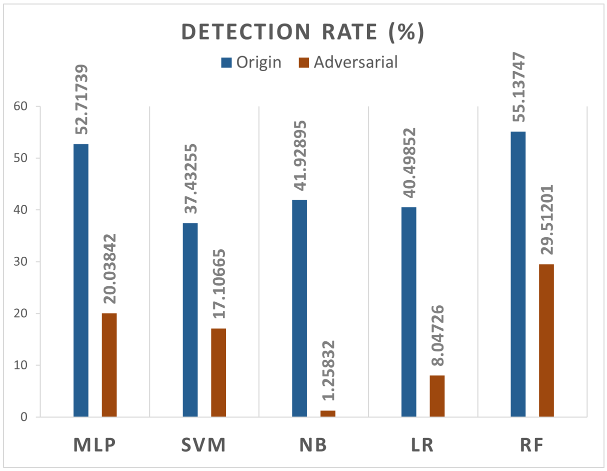

Figure 8: *SDP*-based network generated dataset detection rate.

	Accuracy (%)				
	Origin	ZOO	HSJA	IDSGAN	Harpe
MLP	78.16800	76.30000	50.02000	50.02000	43.17501
NB	72.22800	71.60000	43.01000	43.98000	47.85404
LR	85.47430	52.10000	59.00000	32.30000	47.07897
RF	74.81544	77.30000	50.01000	50.01000	54.34136

Table 12
Accuracy of the *IDSs* classifiers with adversarial traffic generated with *ZOO*, *HSJA*, *IDSGAN* and *Harpe*.

attack or a real authentication request, and performs better against classifiers using *NB* and *LR*.

5.2.3. Comparison with existing AML attacks Results

In this section, we compare the results obtained with *Harpe* with the results obtained with the *IDSGAN*, *Zeroth-order optimization (ZOO) attack*, and *Hop Skip Jump Attack (HSJA) AML* attacks.

Table 12 shows the accuracy of the *IDSs* classifiers when using the selected *AML* attacks on the *CIC-IDS2017* dataset. The results show that *Harpe* is not the best attack in terms of accuracy reduction in all classifiers. However, it still performs well, especially against the *MLP* classifier, where it achieves an accuracy of 43.17501% (the best result among the attacks).

Table 13 shows the detection rate of the *IDSs* classifiers when using the selected *AML* attacks on the *CIC-IDS2017* dataset. In this case, *Harpe* is the best attack in terms

	Detection Rate (%)				
	Origin	ZOO	HSJA	IDSGAN	Harpe
MLP	87.23630	52.60000	0.00000	0.00000	0.00000
NB	75.73822	55.00000	0.00000	0.00000	0.00000
LR	82.59692	4.20000	18.00000	0.00000	0.00000
RF	80.70917	54.60000	0.00000	0.00000	0.00000

Table 13
Detection Rate of the *IDSs* classifiers with adversarial traffic generated with *ZOO*, *HSJA*, *IDSGAN* and *Harpe*.

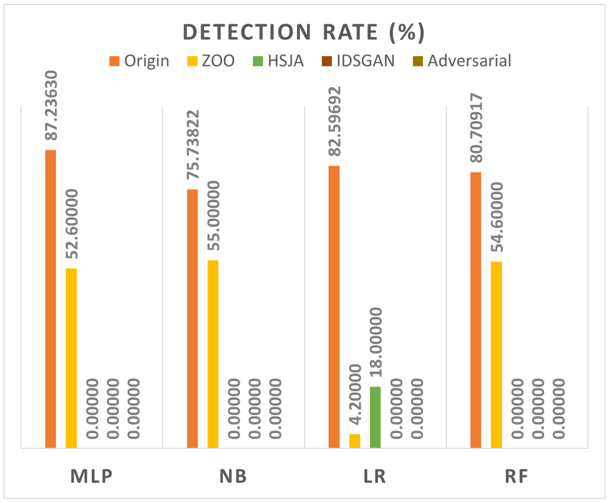

Figure 9: AML attacks detection rate comparison.

of detection rate reduction in all classifiers, together with *IDSGAN*.

Although *Harpe* and *IDSGAN* have similar results in terms of accuracy and detection rate reduction, *Harpe* is a better solution for real-world scenarios. *Harpe* maintains the features necessary for the proper functioning of an attack, such as intrinsic and time-based on DDoS attacks, while *IDSGAN* does not. This means that *Harpe* is more effective in real-world scenarios where these features must be maintained.

6. Conclusions and Future Work

The proposed *Harpe* attack framework have the main objective of demonstrating that *Software-Defined Perimeters (SDP)* are not completely invulnerable as one of their components, the *Intrusion Detection System (IDS)*, can be fooled by *Adversarial Machine Learning* techniques. *Harpe* is able to generate adversarial traffic that closely mimics benign traffic, effectively evading the *IDS* in an *SDP* network while still maintaining the functionality of the original attacks. This is achieved by carefully modifying only those characteristics that do not affect the behavior of the attacks, making them difficult to detect. Furthermore, the framework's ability to generate adversarial traffic is achieved by training a neural network (*W-GAN with Gradient Penalty*) using only binary information received from the *IDS*. This makes the framework practical for use in real-world environments, where limited information about the target system is readily available.

Our experiments have shown that *Harpe* framework can successfully generate adversarial traffic in an *SDP* network that can spoof the *IDS* and trigger denial of service *(DoS)* attacks. The results of this study have important implications for the security of *SDP* networks. The proposed framework highlights the need for further research to improve the security of *SDPs*. Our framework can be used as a tool for evaluating and improving the security of *SDPs*.

In order to improve *Harpe*, we are currently working on the development of a new dataset for a larger *SDP* network with more devices. To achieve this, we plan to implement a real-world network architecture and collect all the generated information to create a dataset with more comprehensive data. We also aim to include other types of attacks in addition to denial-of-service attacks in this dataset. Once we have a more robust dataset, we plan to retrain the *IDS* classifiers and compare the results with traffic generated by our framework.

Additionally, we plan to test the framework with the existing and new proposed *AML* defenses techniques, such as *Adversarial Training* and *Adversarial Detection*.

References

[1] R. Al Nafea and M. A. Almaiah, "Cyber security threats in cloud: Literature review," in *2021 International Conference on Information Technology (ICIT)*, pp. 779–786, IEEE, 2021.

[2] M. A. S. Bubukayr and M. A. Almaiah, "Cybersecurity concerns in smart-phones and applications: A survey," in *2021 International Conference on Information Technology (ICIT)*, pp. 725–731, IEEE, 2021.

[3] M. Karimi and P. Krishnamurthy, "Software defined ambit of data integrity for the internet of things," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 737–745, IEEE, 2021.

[4] Y. Palmo, S. Tanimoto, H. Sato, and A. Kanai, "Complementary methods of iot reliability for embedding iot devices into sdp," in *2021 IEEE 11th International Conference on Consumer Electronics (ICCE-Berlin)*, pp. 1–6, IEEE, 2021.

[5] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Zero trust architecture (zta): A comprehensive survey," *IEEE Access*, 2022.

[6] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (sdp): State of the art secure solution for modern networks," *IEEE network*, vol. 33, no. 5, pp. 226–233, 2019.

[7] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[8] 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.

[9] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[10] 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.

[11] 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.

[12] J. Koilpillai and N. A. Murray, "Software defined perimeter (sdp) and zero trust," *Cloud Security Alliance (CSA)*, 2020.

[13] P. Kumar, A. Moubayed, A. Refaey, A. Shami, and J. Koilpillai, "Performance analysis of sdp for secure internal enterprises," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2019.

[14] D. Enterprise, "Department of defense global information grid architectural vision," *DoD*, 2007.

[15] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," tech. rep., National Institute of Standards and Technology, 2020.

[16] J. Koilpillai and N. A. Murray, "Software defined perimeter (sdp) and zero trust," *Cloud Security Alliance (CSA)*, 2020.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] S. Kariyappa and M. K. Qureshi, "Defending against model stealing attacks with adaptive misinformation," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2020.

[33] 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.

[34] 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.

[35] M. Mohammadi, T. A. Rashid, S. H. T. Karim, A. H. M. Aldalwie, Q. T. Tho, M. Bidaki, A. M. Rahmani, and M. Hosseinzadeh, "A comprehensive survey and taxonomy of the svm-based intrusion detection systems," *Journal of Network and Computer Applications*, vol. 178, p. 102983, 2021.

[36] 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.

[37] 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.

[38] A. Rosay, K. Riou, F. Carlier, and P. Leroux, "Multi-layer perceptron for network intrusion detection," *Annals of Telecommunications*, vol. 77, no. 5, pp. 371–394, 2022.

[39] 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.

[40] 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.

[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] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

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

[54] M. Usama, M. Asim, S. Latif, J. Qadir, *et al.*, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pp. 78–83, IEEE, 2019.

[55] A. H. Lashkari, Y. Zang, G. Owhuo, M. Mamun, and G. Gil, "Cicflowmeter," 2017.

[56] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM transactions on Information and system security (TiSSEC)*, vol. 3, no. 4, pp. 227–261, 2000.

[57] E. Damon, J. Dale, E. Laron, J. Mache, N. Land, and R. Weiss, "Hands-on denial of service lab exercises using slowloris and rudy," in *proceedings of the 2012 information security curriculum development conference*, pp. 21–29, 2012.