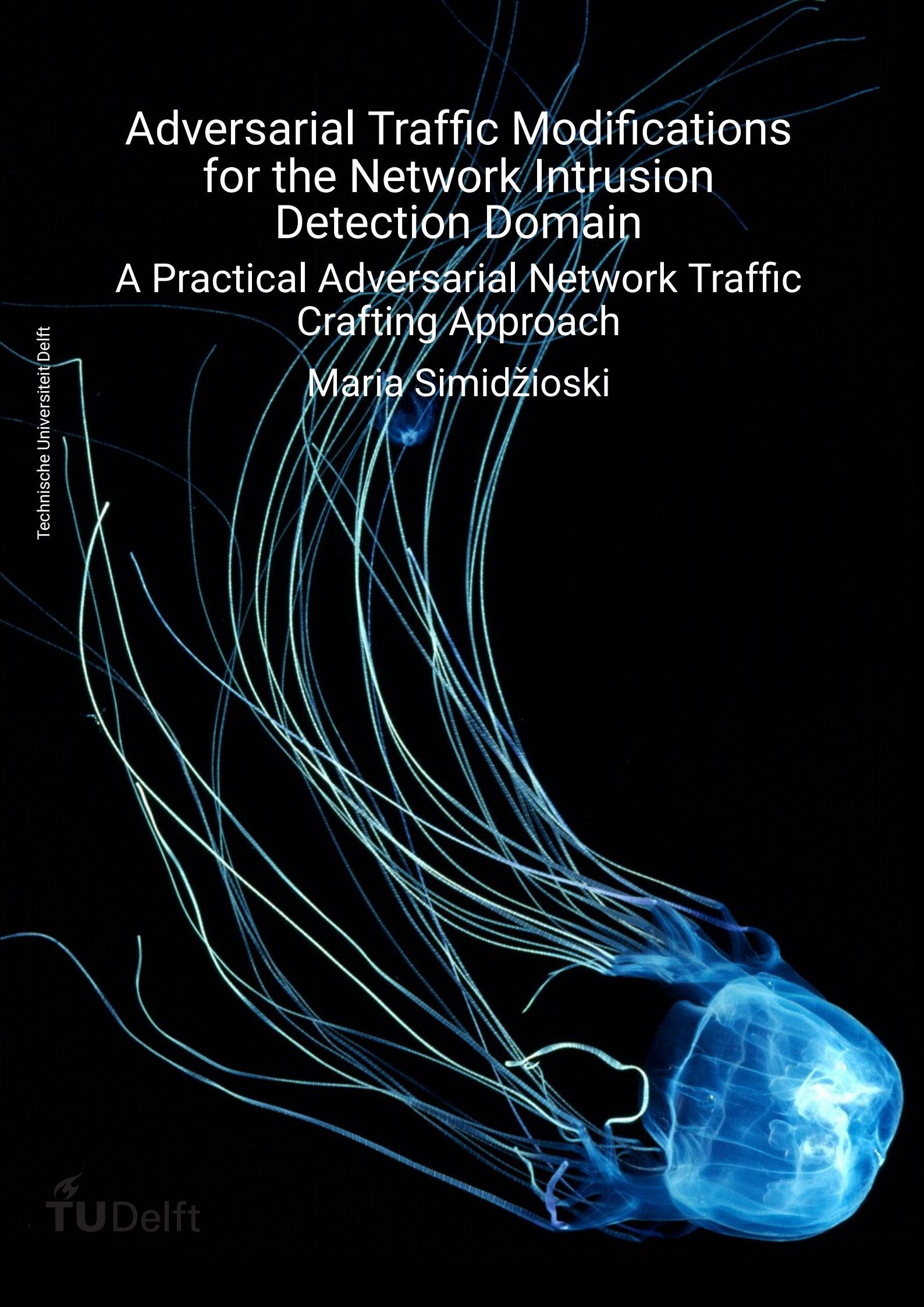
Adversarial Traffic Modifications for the Network Intrusion Detection Domain

A Practical Adversarial Network Traffic Crafting Approach

Maria Simidžioski

Technische Universiteit Delft



TU Delft

Adversarial Traffic Modifications for the Network Intrusion Detection Domain

A Practical Adversarial Network Traffic Crafting Approach

by

Maria Simidžioski

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Specialization: Cyber Security

at the Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science.
to be defended publicly on 9 July 2021

Thesis committee:	Dr. ir. S.E (Sicco) Verwer,	TU Delft , supervisor
	Prof. dr. Catolijn M. Jonker,	TU Delft
	Ir. Daniël Vos,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis was written for a partial fulfilment of the requirements of the Computer Science master at the Delft University of Technology with a specialization in Cyber Security. I would like to express sincere gratitude to my supervisor Dr. ir. S.E (Sicco) Verwer and Ir. Daniël Vos for their excellent support and guidance throughout the duration of this thesis. Their insightful feedback and suggestions have helped me shape and perform this research. Special thanks to Prof. dr. Catolijn M. Jonker for being part of the thesis committee and taking the time to read and evaluate my work. I would also like to thank the other students working on their theses in Sicco's group. The weekly meetings were always a possibility for discussing different topics from which I have learned a lot. Lastly, my appreciation goes to my family and friends for their constant support and encouragement through this interesting and challenging period.

The cover picture is taken from Flickr [14]. It represents an Australian Box Jellyfish, which is a transparent and venomous type of jellyfish. Its transparent colour makes it very difficult to be detected as it blends in with the environment. The cover picture represents the objective of this thesis, creating adversarial network packets that will remain unnoticed on the target network.

Maria Simidžioski
Delft, July 2021

Abstract

Adversarial attacks pose a risk to machine learning (ML)-based network intrusion detection systems (NIDS). In this manner, it is of great significance to explore to what degree these methods can be viably utilized by potential adversaries. The majority of adversarial techniques are designed for unconstrained domains such as the image recognition domain, where these methods apply alterations to the pixels in a picture. Therefore, the applicability of these techniques to the NIDS domain is very limited. Related work on adversarial techniques for NIDS generally considers feature-space techniques, which cannot be applied in a practical situation since only the extracted network traffic features are modified and not the actual network traffic. To solve these limitations, a traffic-space approach for creating adversarial examples for evading ML-based NIDS is proposed and assessed with several classification models. The proposed constrained adversarial crafting method is based on the Iterative Fast Gradient Sign Method (IFGSM) and is called the Constrained Iterative Fast Gradient Sign Method (CIFGSM). A constraint set is added as a penalty term to the loss function of the optimization to ensure that the adversarial values remain within the valid space. Additionally, an L_2 regularization term is used to minimize the distance between the original and adversarial network traffic samples. The proposed method is evaluated and shown to be an effective way for generating realistic and practical adversarial evasion packets. To achieve this, network packet components and their characteristics are defined as a constraint set which can be used for the optimization task and a custom adversarial loss function is created that encapsulates the different elements of this optimization problem. Furthermore, multiple models are evaluated to test the transferability of this method. Conclusively, the proposed method is evaluated in a realistic scenario, where adversarial packet captures are crafted and examined. Where other state-of-the art works only modify the network traffic features in feature-space or on a connection level only and do not apply their method in a real world scenario, this work modifies the packet captures on a per-packet level which is subsequently used to evaluate flow based classification models.

Contents

1 Introduction

The motivation to study this topic comes from the broad adoption of machine learning techniques in the intrusion detection field, which often serve as a crucial part for identifying and preventing attacks. The machine learning-based solutions that are used across different NIDSs, are mostly focused on detecting specific types of attacks and are very sensitive to adversarial attacks. In this work anomaly-based NIDS are used for evaluating the possibilities of evasion through adversarial modifications. These systems classify suspicious traffic based on a previously selected set of statistical features that are representative of the traffic on the target network. Machine learning solutions are applied to intrusion detection classifiers to increase the detection rate and performance of these classifiers. In the last decade, there is an increasing number of adversarial attacks developed against a wide range of these machine learning techniques. The goal of these adversarial modifications is to deceive the target classifiers into thinking that the malicious packets are benign. As new adversarial attacks are being developed and applied to intrusion detection systems, their performance is reduced. Therefore it is of great importance to explore the extent to which adversarial techniques can be used in the intrusion detection field.

While there exist many attacks for network traffic, the defences against these attacks are also getting better. Especially since most of the transmitted traffic is encrypted, defensive solutions are now focusing on the meta-data of the traffic to detect adversarial packets instead of relying only on the packet payload. For attackers to remain undetected, it is important to modify packets including the meta-data to hide the malicious payload and bypass the intrusion detection system. Adversarial machine learning provides great possibilities for this, and therefore it is important to study, understand, map, the possibilities, and the extent to which existing adversarial techniques can be used for the generation of evasive network traffic.

1.1. Motivational Example

The goal of adversarial machine learning is to transform a point x according to the decision boundaries of a classifier, such that it is classified in a different class. Figure 1.1 shows the transformations of points to the opposite class. The background colours represent the different probability areas from the classifier where the middle line is the 0.5 probability. The blue and yellow points that cross this line are considered successful adversarial examples as they are now classified by the model. The dashed line between two points is the direction in which the original point moves and the red point represents the final adversarial point which is wrongly classified by the model. The coloured areas in the plot represent the decision surface of the model and the lines represent the probabilities that the point x will be classified in class $y = 1$.

The same idea is used for the modification of network traffic, where one point represents one packet of the network traffic data.

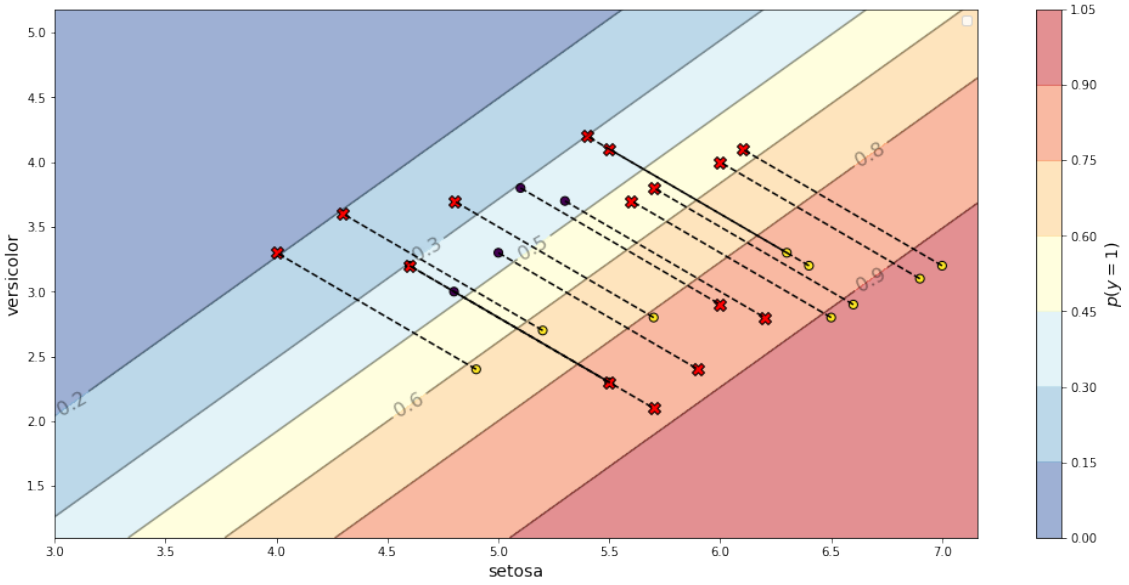


Figure 1.1: Example of adversarial point transformations.

Existing adversarial techniques have been very successful for many domains. However, when applied to network intrusion detection the outcome of the transformations is not valid. In the examples presented under Figure 1.2 two states are shown of the same packet. The first state is a sample of several packets in a packet capture file, and the second state is the modified version of the packet according to the values obtained from the Fast Gradient Sign Method (FGSM). It is clear from these examples that the adversarial modifications produce an invalid packet that cannot be used for a real attack as it will be dropped by the receiving host.

1.2. Problem Statement

Most of the existing techniques for generating adversarial examples and defences proposed against classification models mainly focus on the image recognition field. These techniques are only limitedly applicable to network intrusion detection. There are very few machine learning based adversarial attack techniques proposed and evaluated on network traffic, considering both evasion and poisoning attacks of which the majority is proposed for flow based data. Related work on adversarial attacks for NIDS mostly considers **feature-space attacks** which cannot be applied in a realistic scenario. For these attacks, only the extracted features are modified and not the actual network traffic. The issue with this approach is that the network traffic cannot be inferred or reconstructed from these modified features again. In a realistic scenario, the features cannot be altered by the adversary before the packets are sent to the target IDS. Instead, the adversary has to find a way to directly modify the original network traffic before it is received by the target NIDS.

It is not clear to what extent adversarial-based **traffic-space attacks** would be successful when an adversary has no or limited knowledge of the target NIDS or the benign traffic on the target network. Furthermore, insufficient information has been provided about which features can be altered when crafting adversarial examples for network traffic and how these affect the validity of the network traffic. This is of importance for both the traffic-space and feature-space

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	205.174.165.73	192.168.10.9	TCP	66	8080 → 1841 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
2	0.001660	205.174.165.73	192.168.10.9	TCP	60	8080 → 1841 [ACK] Seq=1 Ack=195 Win=30336 Len=0
3	0.132783	205.174.165.73	192.168.10.9	HTTP	182	HTTP/1.1 200 OK
4	0.134532	205.174.165.73	192.168.10.9	TCP	60	8080 → 1841 [FIN, ACK] Seq=129 Ack=196 Win=30336 Len=0
5	10.140363	205.174.165.73	192.168.10.9	TCP	66	8080 → 1845 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
6	10.142064	205.174.165.73	192.168.10.9	TCP	60	8080 → 1845 [ACK] Seq=1 Ack=195 Win=30336 Len=0
7	10.266350	205.174.165.73	192.168.10.9	HTTP	182	HTTP/1.1 200 OK
8	10.268278	205.174.165.73	192.168.10.9	TCP	60	8080 → 1845 [FIN, ACK] Seq=129 Ack=196 Win=30336 Len=0
9	20.271444	205.174.165.73	192.168.10.9	TCP	66	8080 → 1846 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
10	20.272013	205.174.165.73	192.168.10.9	TCP	60	8080 → 1846 [ACK] Seq=1 Ack=195 Win=30336 Len=0
11	20.435710	205.174.165.73	192.168.10.9	TCP	182	8080 → 1846 [PSH, ACK] Seq=1 Ack=195 Win=30336 Len=128 [TCP segment of a reassembled PDU]
12	20.435914	205.174.165.73	192.168.10.9	HTTP	60	HTTP/1.1 200 OK (text/html)
13	20.437313	205.174.165.73	192.168.10.9	TCP	60	8080 → 1846 [FIN, ACK] Seq=132 Ack=196 Win=30336 Len=0
14	20.535928	205.174.165.73	192.168.10.9	TCP	66	8080 → 1847 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
15	20.536945	205.174.165.73	192.168.10.9	TCP	60	8080 → 1847 [ACK] Seq=1 Ack=1859 Win=33024 Len=0

(a) Original packet capture

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	205.174.165.73	192.168.10.9	TCP	1000	13534 → 1841 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=30 MSS=1460 SACK_PERM=1 WS=128
2	965.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1841 [ACK] Seq=1 Ack=195 Win=30336 Len=30
3	1930.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1841 [PSH, ACK] Seq=1 Ack=195 Win=30336 Len=158
4	2895.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1841 [FIN, ACK] Seq=129 Ack=196 Win=30336 Len=30
5	3860.000000	205.174.165.73	192.168.10.9	TCP	1000	13534 → 1845 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=30 MSS=1460 SACK_PERM=1 WS=128
6	4825.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1845 [ACK] Seq=1 Ack=195 Win=30336 Len=30
7	5790.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1845 [PSH, ACK] Seq=1 Ack=195 Win=30336 Len=158
8	6755.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1845 [FIN, ACK] Seq=129 Ack=196 Win=30336 Len=30
9	7720.000000	205.174.165.73	192.168.10.9	TCP	1000	13534 → 1846 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=30 MSS=1460 SACK_PERM=1 WS=128
10	8685.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1846 [ACK] Seq=1 Ack=195 Win=30336 Len=30
11	9650.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1846 [PSH, ACK] Seq=1 Ack=195 Win=30336 Len=158
12	10615.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1846 [PSH, ACK] Seq=129 Ack=195 Win=30336 Len=33
13	11580.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1846 [FIN, ACK] Seq=132 Ack=196 Win=30336 Len=30
14	12545.000000	205.174.165.73	192.168.10.9	TCP	1000	13534 → 1847 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=30 MSS=1460 SACK_PERM=1 WS=128
15	13510.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1847 [ACK] Seq=1 Ack=1859 Win=33024 Len=30
16	14475.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1847 [PSH, ACK] Seq=1 Ack=1859 Win=33024 Len=158
17	15440.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1847 [FIN, ACK] Seq=129 Ack=1860 Win=33024 Len=30
18	16405.000000	205.174.165.73	192.168.10.9	TCP	1000	13534 → 1848 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=30 MSS=1460 SACK_PERM=1 WS=128
19	17370.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1848 [ACK] Seq=1 Ack=195 Win=30336 Len=30
20	18335.000000	205.174.165.73	192.168.10.9	TCP	1000	[TCP Retransmission] 13534 → 1848 [PSH, ACK] Seq=1 Ack=195 Win=30336 Len=158

(b) Adversarial packet capture

Figure 1.2: Example of network packets after the application of unconstrained adversarial modifications.

attacks to consider for maintaining the validity of the network traffic and keep the functionality of the original attacks unchanged.

In many cases, the alterations need to have constraints defined for each separate changeable feature where the relations between the features are also considered. To overcome this limitation, several works have tried to modify the packets randomly assuming that the attacker has full knowledge of the workings of the NIDS and the extracted features. This is done with a feedback loop where a range of different fragmentation and delay values are applied. Based on the response of the NIDS, these values are modified to be under a predefined threshold. Finally, the values that reduce the performance of the NIDS the most are selected for the attack. However, this is also not a realistic approach since the attacker may not always have full knowledge of and access to the target model (white-box). This approach has also a greater computational complexity and overhead since it is constantly guessing and evaluating the proposed values. Additionally, there is no evaluation of how distinct attacks perform on different types of classifier, for example on packet based vs. flow-based classifiers, and which type of classification method is more robust. Lastly, previous works have put little focus on possible defences that can be applied to mitigate adversarial attacks in network intrusion detection scenarios. One reason for the deficiency of existing defences is that very few attacks against NIDS have been proposed. Some of the proposed defence techniques include normalisation, randomised modelling and testing, adversarial retraining, and feature selection. However, each of these defence techniques has its limitations. One such limitation is the ambiguity between the NIDS and the hosts on the target network. Moreover, most of the defences are designed for signature-based NIDS which also inspect the payload of the packets.

1.2.1. Research Question

Based on the previously defined problem statement and the identified challenges, several research questions are defined which aim to answer the identified limitations. The central research question is:

To what extent can adversarial optimization techniques be used for the generation of adversarial network traffic in a constrained domain while preserving the validity of the original traffic?

To answer this question several sub-questions are defined where each subquestion focuses on a different research step.

RQ1: Which adversarial traffic evasion techniques exist for network traffic and what is the effect of applying these on a target system with limited knowledge?

- How can network traffic be manipulated to create adversarial network packets in a practical setting ?
- To what extent do the crafted adversarial packets succeed at evading the target classification model ?

RQ2: How are adversarial modifications represented through the network traffic features ?

- What is the effect of applying adversarial modifications to the extracted network traffic features ?
- Which constraints and feature relations need to be considered for crafting valid adversarial packets ?

RQ3: To what extent can practical adversarial network packets be crafted using adversarial optimization techniques, while also considering the existing domain constraints?

- How can network traffic domain constraints be included in the optimization function that is used for the generation of adversarial examples?
- To what extent does the proposed method succeed at evading the target classification model and are the adversarial network packets transferable to other models?

1.2.2. Contribution

The most important contribution of this work is the proposed adversarial crafting techniques that can be used for the generation of valid adversarial network traffic in a practical setting. Where other works solely focus on generating adversarial traffic in the feature-space, this work takes a step forward and uses these adversarial crafting techniques in traffic space where realistic adversarial network packets are generated. To accomplish this, (1) network packet elements and their characteristics are formulated as a constraint set which can be used for the optimization tasks,(2) a custom adversarial loss function is created that encapsulates the different components of this constrained optimization problem, and (3) the proposed method is evaluated in a realistic, practical scenario, where adversarial packet captures are crafted and examined. Where other state-of-the art works only modify the network traffic features in feature-space or on a connection level and do not apply their method in a real-world scenario, this work modifies the network packets in a practical manner, on a per-packet level, which subsequently

are used to evaluate the flow-based classification models. This work demonstrates that applying adversarial modifications within the network traffic constrained domain is an effective way for generating realistic adversarial evasion packets. The results from these practical experiments confirm that the majority of generated adversarial network packets are valid and have preserved their functionality in the sense that they are correctly received on the target network. However, more experiments are neccessary to evaluate whether the initial functionality of the attacks remains also unchanged.

- Propose an adversarial crafting method for the chosen set of features that can find adversarial values in the constrained domain and generate adversarial network packets. In the proposed method, network packet elements and their characteristics are formulated as a constraint set which can be used for the optimization tasks and a custom loss function is created that encapsulates the different components of this constrained optimization problem.
- Evaluate the proposed method in a practical manner and evaluate the evasion rate and transferability of the proposed method on multiple ML-based classification models.

1.2.3. Proposed Solution

To try to solve these issues a **traffic-space** approach for crafting adversarial network traffic against anomaly-based NIDS is proposed and evaluated. This approach can be divided into five main phases as shown in Figure 1.3.

The first phase is to evaluate which network traffic modifications are feasible to apply to the network packets, whether these modifications can be accomplished while maintaining the validity of the network packets, and whether these modifications contribute to and are successful at evading the target NIDS. A formal definition is given for all proposed traffic modifications. First the modifications are applied directly to the network packets, next the extraction of statistical features occurs, and finally the data is preprocessed and made ready for classification. To verify that the original attacks remain valid, both the original network traffic and the modified network traffic are evaluated on a Virtual Machine (VM), where the traffic is replayed and inspected using a network protocol analyzer tool Wireshark[1]. To determine if and how successful the proposed attacks are at evading the target model, the adversarial network traffic is tested on several machine learning-based classification models and the success rate of the original network traffic and the adversarial network traffic are compared. In addition the accuracy of the models on the original and adversarial examples is evaluated.

The second phase consists of examining the features that are changed as a result of the proposed network traffic modifications. Additionally, constraints are identified that will be used to ensure the validity of the modified network packets. These are determined according to the official specifications of network traffic protocols and the examination of a large internet backbone traffic data set. The constraints consist of minimal and maximal values that the features can have as well as the interfeature relations and other specific requirements associated with network traffic. There is also a subset of features defined which cannot be modified due to practical limitations. The detailed description and explanation of these constraints can be read in Chapter 6.
After the set of effective attacks and feature constraints are defined, the third phase is the adversarial example crafting phase. Here, a method is proposed that aims to find adversarial

[1] https://www.wireshark.org/

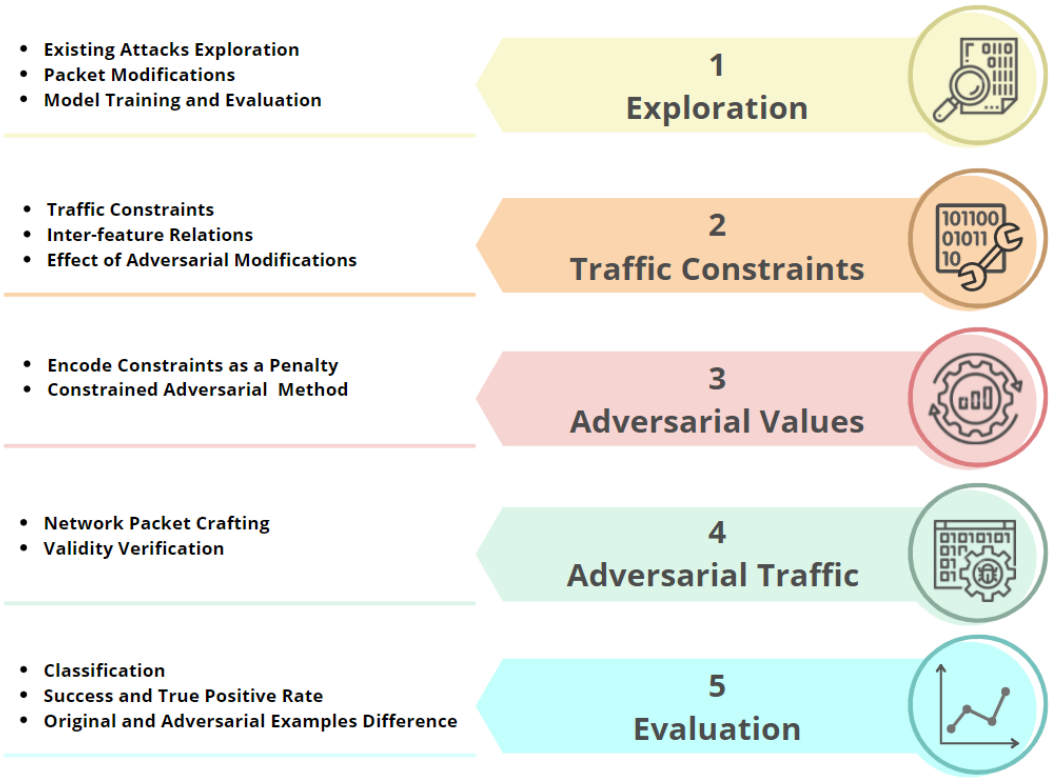


Figure 1.3: Outline of the proposed solution.

feature values that will have the highest evasion rate while not violating the imposed domain constraints. For this purpose, a surrogate classification model is selected based on which the values are optimized. The assumption here is that the adversary has knowledge of both the classifier and the features used in the classification process. Moreover, the set of adversarial values is crafted such that it is directly applicable for generating the actual network packets in a practical setting. While previous works have used an approximation algorithm to map the adversarial values in a practical setting, this method directly finds feasible and legitimate adversarial network traffic values.

The fourth phase consists of taking a set of adversarial values and modifying the actual network traffic packets according to these values. For this purpose the packet manipulation tool Scapy[2] is used. The packet captures are processed and each packet is modified according to the adversarial values obtained for that packet in the previous step. After these alteration again the original and modified network traffic are replayed and inspected to ensure the original functionality of the packets is preserved.

In the last phase, the optimized traffic-space attacks will be evaluated with different classification models to see whether these attack will also work in a gray-box scenario where the target NIDS are not known while crafting adversarial examples. The performance of the target model and the success of the adversarial examples will be examined by measuring the success rate and accuracy. Since there exists no other similar method for the generation of legitimate adversarial traffic that can be used for comparison with the method proposed in this work, several adversarial methods used for crafting feature space attacks will be considered.

[2]https://scapy.net/

2

Preliminaries

In this chapter, background information is provided about several topics and methods that are mentioned throughout this report. The different techniques used in this work are mentioned and explained.

2.1. Adversarial Machine Learning

Adversarial machine learning is a field which focuses on assessing the robustness of machine learning models to adversarial exampled. The main goal of adversarial machine learning is to generate examples that will be wrongly classified by the targeted model. In the sections below the generation of adversarial examples with different methods is explained.

2.1.1. Adversarial Perturbations

An adversarial perturbation is applying a modification to an input point $x_0 \in \mathbb{R}$ which belongs to class c_0, to create a point $x' \in \mathbb{R}$ that belongs to class c_i, where $c \neq c_0$. This perturbation can be targeted, where the point is modified such that it belongs to a target class c_t, or it can be untargeted, where the perturbation is used to just move the new point x' away from the current class c_0. The most common way of modifying the input point x_0 is to add the perturbation $\epsilon \in \mathbb{R}$ to x_0. Then the adversarial point can be defined, as described in equation 2.1. The three general perturbation methods are the minimum norm, maximum norm, and regularization-based pertrubation, these are specified below.

$$x' = x_0 + \epsilon \tag{2.1}$$

- **Minimum Norm Perturbation**

  This method tries to find an adversarial point x that will be classified to class $c \neq c_0$ while also trying to minimize the perturbation magnitude.

$$\begin{aligned} \min_x \quad & ||x - x_0|| \\ \text{s.t.} \quad & max(j \neq t) g_j(x) - g_t(x) \leq 0 \end{aligned}$$

- **Maximum Norm Perturbation**

  The maximum norm method tries to find an adversarial point x by perturbing the original input by a perturbation magnitude that is bounded by η and is not allowed to exceed this space.

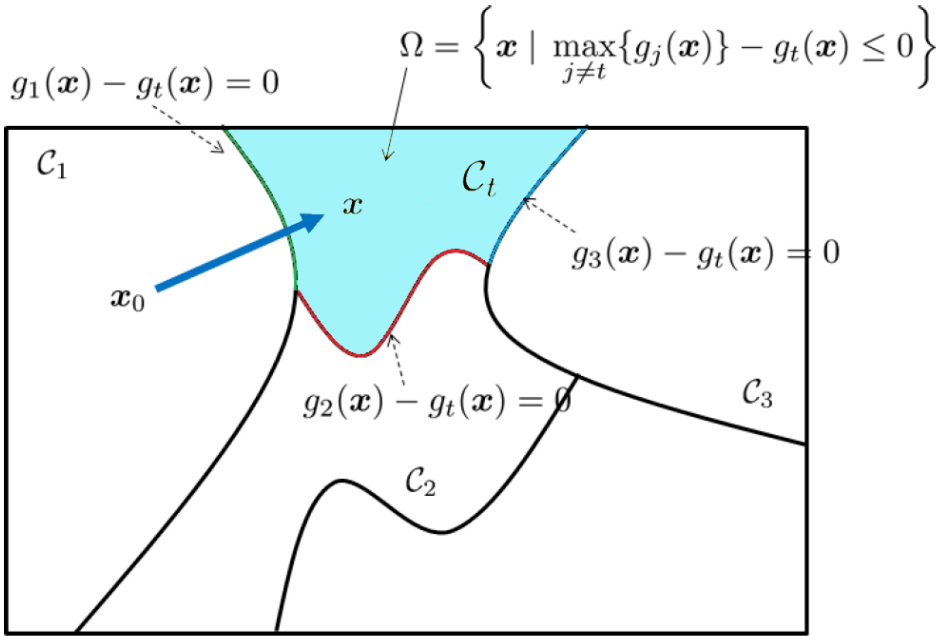


Figure 2.1: Example of an adversarial point x that belongs to class C_t that satisfies all the constraints defined in the constraint set Ω as shown by Chan in [10].

$$\min_x \quad max(j \neq t) g_j(x) - g_t(x)$$
$$\text{s.t.} \quad ||x - x_0|| \leq \eta$$

- **Regularization-based Perturbation**

  Perturbations can be used for constrained domains by defining adding regularization terms to the loss function. As shown in 2.1 the set of modifications is limited by multiple different constraints which are represented through the curved lines. Only correct values for x can be obtained only if x remains within the blue region.

$$\min_x \quad ||x - x_0|| + \lambda \; max(j \neq t) g_j(x) - g_t(x), \; \lambda > 0$$

Here the norm can be chosen according to the goals of the adversary and the norm will define the geometry of the adversarial perturbations. This can be, for example, the L_1 norm, L_2 or the L_∞ norm. The parameter η represents the maximum allowed boundary of the perturbation and λ is called the Lagrange Multiplier and represents the regularization magnitude or how much effect the constraint will have on the perturbation of input point x_0. By using the regularization term, two opposite function objectives are being optimized and the magnitude of λ controls which objective will be weighted more. By choosing λ to be a smaller number, the emphasis is on the first objective $||x - x_0||$ while if λ is larger, the emphasis will be on the second objective $max(j \neq t) g_j(x) - g_t(x)$.

2.1.2. Vector Norm

To minimize an objective function, different norms can be used. A norm is used to calculate a distance between two vectors. Given a vector x, a norm that calculates the distances for each

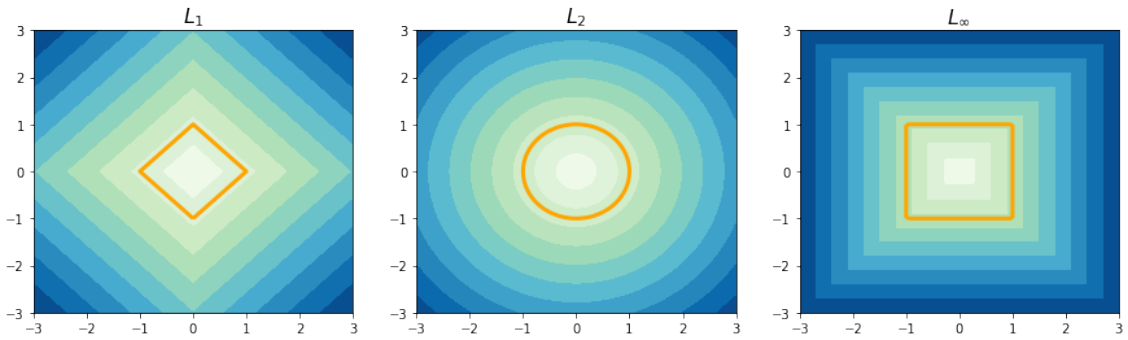


Figure 2.2: Visual representation of different vector norms

point x_i can be defined as follows:

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

When $p = 1$ the norm is called the L_1 norm or the Manhattan distance. This norm represents the absolute distance between two points in the vector. This norm is calculated as the sum of the absolute distances of the points in a vector.

$$\||x\|_1 = \sum_i |x_i| = |x_1| + |x_2| + ... + |x_i|$$

When $p = 2$, the norm is the L_2 norm or Euclidean distance. This norm represents the shortest distance between two origin of the vector and a point in the vector and is calculated as the square root of the sum of squared points in the vector.

$$\|x\|_2 = \sqrt{\left(\sum_i |x_i|^2\right)} = \sqrt{x_1^2 + x_2^2 + ... + x + i^2}$$

When $p = \infty$, the norm is the L_∞ norm or Chebyshev distance. This norm represents the maximum absolute distance between two vector points. The maximum value of the vector is the result of this norm.

$$\|x\|_\infty = \max_i |x_i|$$

When comparing these norms, the L_1 norm can provide more solutions and is more robust to outliers, however, because it calculates the absolute distance, it has no closed-form solution and the found solutions will also be approximations. This makes it a function that is not differentiable and therefore also more computationally expensive than the L_2 norm. The L_2 norm on the other hand takes the square and provides only one answer, the minimal distance between two points. This norm is less computationally expensive and it is also differentiable, however it is more prone to outliers and less robust than the L_1 norm. A visual representation of the previously discussed vector norms can be seen in Figure 2.2.

Loss Function

This loss function is measuring how good (or bad) the performance of the model is. This is done by calculating the negative log likelihood of the probabilities of the predicted output and

the original output. When the predicted output is far from the original output, the loss will be large, otherwise if the predicted output is close to the original output, the loss will be small. The loss function is used as a penalty in the learning phase of the classification model. In the context of adversarial attack models, the loss is maximized instead of minimized with the goal of tricking the classification model to misclassify the input points.

Binary Cross Entropy Loss

The Binary Cross Entropy loss is used to measure the (dis)similarity between the original input label and the predicted output.

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n} Y_i log(Y_i') + (1-Y) log(1-(Y_i')$$

Where n is the number of samples, Y is the original output, Y' represents the predicted output. The probability of the predicted output Y' is calculated using the sigmoid function:

$$S(z) = \frac{1}{1+\exp^{-z}}$$

Mean Squared Error

The Mean Squared Error (MSE) is calculated as the average of the squared difference between the original output and the predicted output. The MSE is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2$$

Here n is the number of samples, Y is the original output, and Y' is the predicted output.

Hinge Loss

The Hinge Loss is created by training margin-based classification models such as support vector machines. This loss is usually used for input that has labels in the range of (-1,1) instead (0,1).

$$L = max(0, 1 - y * f(x))$$

It works by increasing the loss if the prediction is too close to a predefined margin, in other words the hinge loss penalizes not only the wrong prediction but also predictions that are not confident.

Gradient Descent

Gradient Descent is an optimization method that tries to find the local minimum of a function that is differentiable. The gradient descent optimization for an objective function f works by taking an initial point x and computing the derivative f' of the function f at point x. Then the output of this first step is used for the initial guess of the model. f'(x) represents the slope of the tangent and the sign of this slope determines the search direction, whether the point x should be increased or decreased to minimize f(x). The function f(x, y) is defined as:

$$f(x,y) = \left(\frac{7xy}{e^{x^2+y^2}}\right): \quad \begin{bmatrix} 7e^{-x^2-y^2}y\left(1-2x^2\right) \\ 7e^{-x^2-y^2}x\left(1-2y^2\right) \end{bmatrix} \tag{2.2}$$

The gradient of the vector $\Delta f(x, y)$ is calculated as follows:

$$\begin{bmatrix} \frac{\partial}{\partial x}(f) \\ \frac{\partial}{\partial y}(f) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x}\left(\frac{7xy}{e^{x^2+y^2}}\right) \\ \frac{\partial}{\partial y}\left(\frac{7xy}{e^{x^2+y^2}}\right) \end{bmatrix} = \begin{bmatrix} 7e^{-x^2-y^2}y\left(1-2x^2\right) \\ 7e^{-x^2-y^2}x\left(1-2y^2\right) \end{bmatrix} \tag{2.3}$$

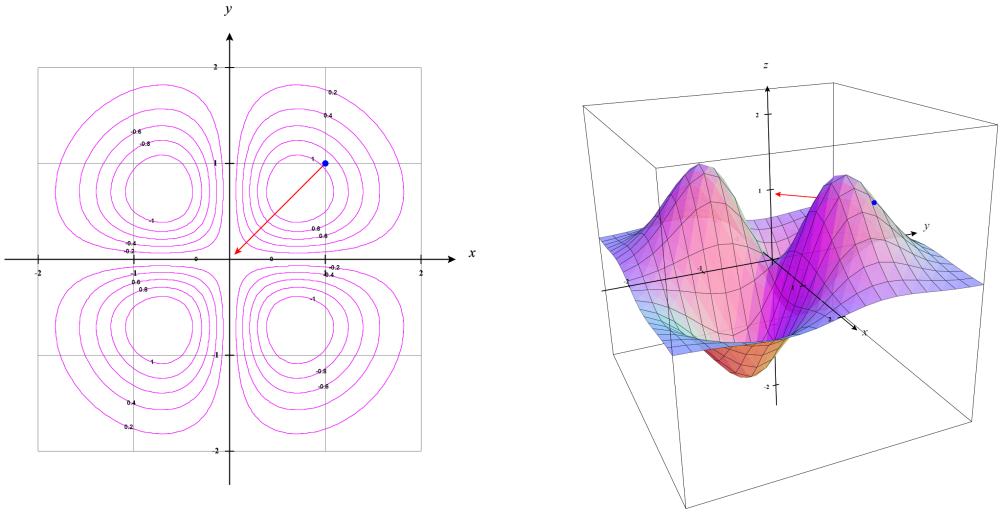


Figure 2.3: Surface plot of the cost function $f(x, y)$ and point x that moves in the direction of the gradient. The gradient descent method is used to find the optimum of this function. Gradient descent will converge when the optimal points is located where the gradient of the cost function is zero.

If the initial start is the point (1, 1) and the learning rate is 0.01, the gradient descent for the variable x after three iterations is the following:

$$7e^{-1^2-y^2}y\left(1-2^2\right) \tag{2.4}$$

1. $x_1 = x_0$ - 0.2 * $(7e^{-x^2-y^2}y\left(1-2x^2\right))$ = 1 - 0.2 * $(7e^{-1^2-1^2}\left(1-2*1^2\right))$ = 1.568
2. $x_2 = x_1$ - 0.2 * $(7e^{-x^2-y^2}y\left(1-2x^2\right))$ = 1.568 - 0.2 * $(7e^{-1.568^2-1^2}\left(1-2*1.568^2\right))$ = 1.307
3. $x_3 = x_2$ - 0.2 * $(7e^{-x^2-y^2}y\left(1-2x^2\right))$ = 1.307 - 0.2* $7e^{-1.307^2-1^2}y\left(1-2*1.307^2\right)$ = 0.894

After three iterations, we see that the value of x is decreasing slowly towards the local minimum of the function.

Penalty

Penalty functions are used to transform a constrained optimization problem to a set of unconstrained problems. By solving these unconstrained subproblems, the methods should converge to an optimal value for the constrained problem. The penalty function is added to the original objective function and this penalty is multiplied with a constant λ that determines the magnitude to which this penalty can influence the optimization. The objective function and penalty move in different directions, this means that if the penalty magnitude is too large, it will be more difficult to find the optimal solution or if the penalty magnitude is too small, an optimal solution will be found faster but at the cost of a larger number of violated constraints.

Different functions exist for measuring if and how much the defined constrains are violated. There exist interior-point methods and exterior point methods for solving constrained optimization problem. For exterior-point methods a penalty is added to the objective function if it finds an optimal point that is located outside of the feasible region defined by the constraints set, this penalty will force the method to search closer to the feasible region. For interior point methods, also called barrier methods, the search space is only the feasible region, and when an optimal point is found that lies too close to the boundaries of the feasible region, a penalty will be added to the cost. Three of the most commonly used penalty functions are presented below.

- **Inverse penalty**

$$P(x) = \sum_{i=1^n} (\frac{1}{c(x_i)})$$

- **Logarithmic penalty**

$$P(x) = \sum_{i=1^n} log(c(x_i))$$

- **Quadratic penalty**

$$P(x) = \sum_{i=1}^{n} (max[0, c(x_i)])^2)$$

2.1.3. Adversarial Example Crafting Methods

In this subsection, several existing adversarial sample crafting methods that are relevant for this problem will be mentioned and explained. Multiple adversarial methods are based on the calculation of the gradient descent to maximise the loss of the optimization function.

- **LBFGS** The Limited-memory Broyden Fletcher Goldforb Shanno method [47] is one of the first proposed adversarial crafting methods. This method is used to create adversarial points that will be misclassified by the target model. The goal of this method is to increase the probability of the opposite class y' by using a line search. For an input point $x\_0 \in^n$ that belongs to class y, LBFGS tries to find an adversarial point $x' \in R^n$ that will belong to class $y' \in 1, ...k$ where $y' \neq y$ while minimizing the L_p distance.

$$\min_x \; c * ||x - x'||_p + L(\theta, x', y) \text{s.t } x' \in [0,1]^n$$

  Here x' represents the adversarial point, θ are the model weights, $L(\theta, x', y)$ is the loss function and c is a constant that is randomly initialized. This method is an untargeted method, it does not direct the input point to a specific class, instead it direct the points further away from the original class. This method is expensive because the optimization problem is NP hard and there exists no closed form solution, by using a line search, the optimal solution can be approximated.

- **FGSM**

  One of the first and most simple adversarial attacks is called the Fast Gradient Sign Method [19]. This attack works by calculating the gradient of the loss with respect to the input samples. This method is l_∞ bounded and is a one-step method. The intuition behind this

is that the output of the model is considered as a function of the input data. In this case, a trained model is taken and instead of changing the model parameters to decrease the loss, the features of the data are modified (in the case of images, the pixels are changed) with the goal of increasing the loss by keeping the model parameters constant. The result of the gradient calculation is a vector with features that corresponds to the size of the input data. This vector contains the gradients that give and an indication of how the loss changes after the proposed modifications. From the gradient vector, a sign vector is created by taking the sign of the gradient, this will determine the direction in which the sample will be modified. This vector is filled with a value of -1 for a negative gradient, a value of 1 for positive gradients, and 0 if the gradient is 0. Then this sign vector is multiplied with a predefined value epsilon and a vector is created that is filled with +/- epsilon values. The adversarial sample is created by adding this vector with epsilon values to the original sample.

$$x' = x + \epsilon \cdot sign(\Delta_x L(\theta, x, y))$$

Here, x represents the original input, x' represents the adversarial input, ϵ is the magnitude of adversarial perturbation, and $sign(\Delta_x L(\theta, x, y))$ is the gradient of the loss function $L(\theta, x, y)$ which is taken with respect to x' and evaluated at x with model weights θ. This method is not computationally expensive because it works with the sign of the gradient instead of the value.

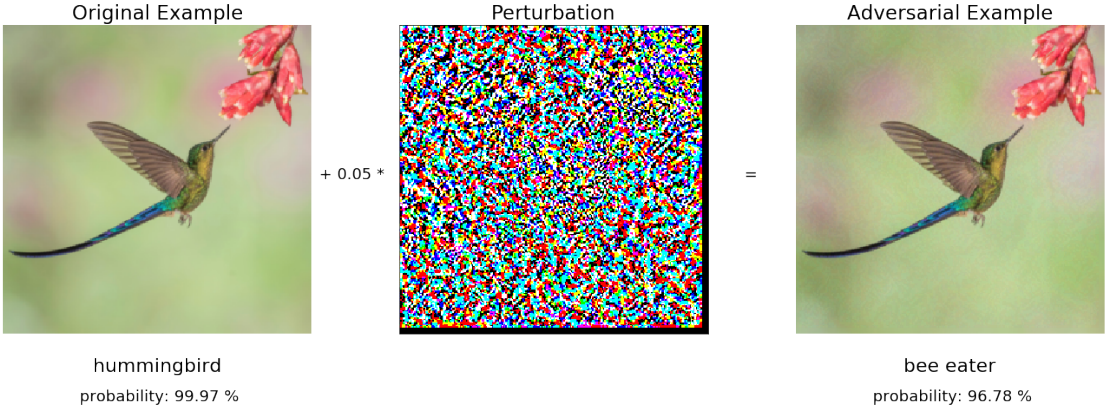


Figure 2.4: Example of an adversarial perturbation generated by the FGSM method with $\epsilon = 0.05$. Here the original image is an image of hummingbird and after this image is perturbed it is classified as a bee eater.

- **IFGSM**

  The Iterative Gradient Sign Method [19] is an extension to the original Fast Gradient Sign method which applies the perturbation to the input sample in multiple steps using a smaller perturbation size as opposed to the FGSM method. This method also imposes bounds on the adversarial example x. By applying smaller perturbations and clipping the new point to the allowed range, it ensures that no local optimum is missed, which increases the possibility that adversarial points with a smaller distance to the original point will be found.

$$X'_{t+1} = Clip\{x'_t + \alpha \cdot sign(\Delta_x L(\theta, x'_t, y))\}$$

Here α represents the magnitude of the perturbation for each iteration, x'_t represents the adversarial point at iteration t and x'_{t+1} represents the point in the next iteration $t+1$. $\Delta_x L(\theta, x'_t, y)$ is the adversarial loss function and y the class of x'.

- **PGD**

  The Projected Gradient Descent Method (PGD) [29] is a generalized version of the IFGSM which projects the adversarial points to a valid range in the $\epsilon - L_\infty$ space of the original example, for each iteration t. The method starts by applying a random perturbation that is within the L_∞ ball of the input point, then calculates the gradient and moves to the direciton where the loss is highest. If the perturbation is not within the L_∞ ball, it gets projected onto the L_∞ ball. These steps are repeated for several iterations until convergence.

$$X'_{t+1} = Proj\{x'_t + \alpha \cdot sign(\Delta_x L(\theta, x'_t, y))\}$$

2.2. Network Traffic

Network Traffic can be described as the amount of data that is flowing through a network at a given point in time. Network traffic information is kept in network packets. A network packet consists of the payload or content and control information. The control information is encapsulated in the header and trailer of the packet. According to the OSI-model [1] a network packet is called a Network Layer protocol data unit. Here a distinction is made between an IP packet, an Ethernet frame from the Data Link Layer, and segments and datagrams of the Transport Layer. Figure 2.5 shows all the components of a network packet. All information is encapsulated in the Ethernet frame. The payload of the Ethernet frame contains the IP frame, the payload of the IP frame contains the TCP segment and the payload of this segment contains the data that is sent over the network through the Application Layer. The network traffic information is captured by using the packet capture (pcap) API. The captured traffic can either be directly processed or it can be saved in a pcap file if further investigation is necessary.

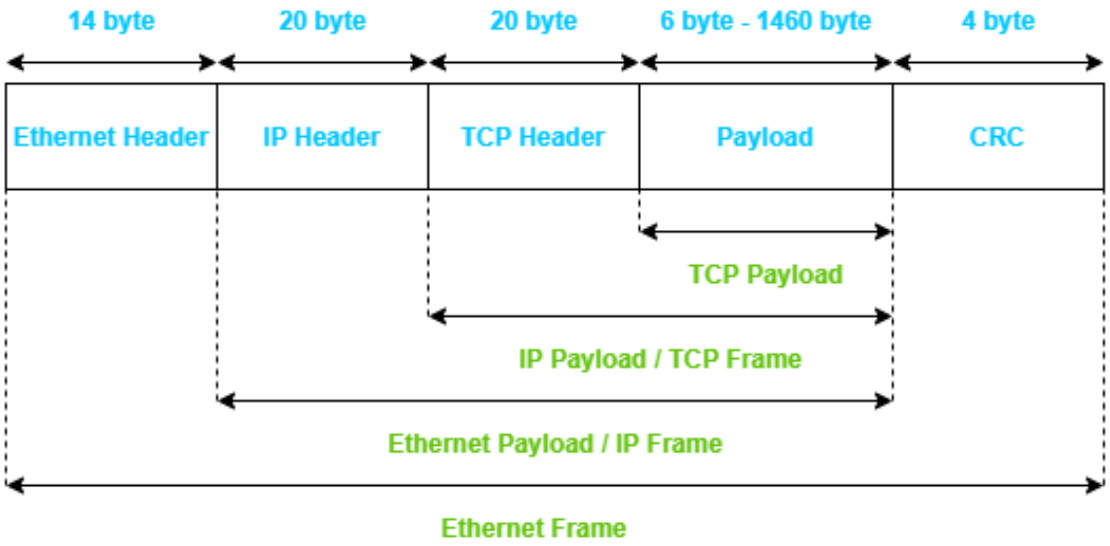


Figure 2.5: Example of a packet and its components

[1] https://www.iso.org/ics/35.100/x/

Signature-based Techniques

These systems are based upon comparing the payload found in the network packets to a database of pre-existing known signatures of malicious network traffic. The main limitation of these systems is that traffic can be slightly modified such that signatures do not match and the NIDS can only recognise already known malicious forms of traffic, but cannot detect zero-day attacks for instance. Another limitation is the amount of false positives that these systems generate, which in many cases need to be manually filtered by security analysts. Because traffic is more often encrypted, these techniques are also losing their effectiveness in many scenarios where the only other option remains detection through statistical anomalies. Lastly, changing the contents of a packet is a complex task as the functionaltiy of the initial traffic may be broken.

Anomaly-based Techniques

The anomaly based intrusion detection systems are solving the limitations of the signature-based systems. These systems generally work by learning what the normal or usual behaviour of a network looks like and alerting to any deviations that are captured on the network. However, these systems also have their limitations as many are trained to detect only specific types of attacks and they need to be fine-tuned according to the usual behaviour in the network which is changing over time. The other major limitation of these systems is that they are based on machine learning classifiers which are known to be sensitive to adversarial modifications. Therefore, it is essential to take this aspect into consideration when deploying intrusion detection systems and know how robust these anomaly-based NIDS can be against adversarial evasion attacks.

Machine learning has been used for the detection of malicious traffic in networks. For this purpose, various solutions are researched, developed, and evaluated. A range of different supervised classification algorithms exist that are used for the classification of both packet and flow-based data. Amongst the evaluated models for classifying malicious network traffic are the Decision Tree and Random Forest algorithms, Support Vector Machines [28], Logistic Regression [26], Neural Networks [31, 50] and Naive Bayes. All of these models have provided high accuracy results on the NLS-KDD, DARPA and CIC datasets, as well as custom data with extracted packet and flow level features [5, 13, 48]. However, there is no single model that is the most suitable to be used for this problem. The types of attacks that are present in the data play a role in the accuracy and detection rate of the used classifier. Some classifiers are shown to work well for detection brute force and denial of service attacks, while others are better in detecting botnet traffic [2]. A general conclusion about the comparison of classification models is the difficulty to make sound conclusions about superiority of certain classifiers because of varying experimental settings. These can include the granularity of the data, the feature set, and the selection and types of attacks that are included in the training and testing phase. [5, 7, 17]. Therefore, to select a representative set of classifiers for the evaluation of adversarial attacks it is important to include multiple models that are good at detecting different types of attacks.

2.3. Evaluation Metrics

For the evaluation of the existing adversarial techniques and the proposed method, several metrics are selected which measure different aspects of the researched problem.

Performance

The success rate measures how many predicted outputs are missed by the model. It is called the success rate because it measure how successful the evasion attack is, the greater the number of missclassified samples, the more successful the attack is. The accuracy metric measures how accurate the model is at predicting the output.

$$Success\ Rate = 1 - \frac{TP}{TP+FN}$$

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

Network Traffic Validity

To measure the validity of the practically modified packets, several metrics are defined which take into account the packet constraints that were defined, the actual transfer of the modified packets over the network and the effectiveness of the attack with these modified packets. The Invalid Packet Rate (IPR) represents the fraction of invalid packets of the total traffic as shown by the network traffic analysis tool Wireshark. The dropped packet rate (DPR) is represented as the fraction of dropped packets from the total number of packets when the traffic is replayed on a network. The violated constraints rate (VCR) is the number of packets that violate one or more constraints compared to the total number of valid packets. Another metric that is used to evaluate the vlaidity of the adversarial network packets is the similarity between the original and adversarial examples. This is measured using the Euclidean distance between the original and the adversarial point. When the distance between the original and adversarial examples is small, the adversarial example is a more realistic sample. The similarity of the data is additionaly evaluated through visualizing all the data points generated with the different methods with heatmaps.

$$IPR = \frac{invalid\_packets}{total\_packets}$$

$$DPR = \frac{dropped\_packets}{total\_packets}$$

$$VCR = \frac{violated\_packets}{total\_packets}$$

$$S(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2}$$

3

Literature Review

In this chapter a literature review will be conducted of the existing works that have research adversarial techniques for network intrusion detection. First, a review is conducted of the existing network traffic evasion techniques and possible defences, where evasion techniques applied to malware samples are also considered. Subsequently, the research gap will be identified based on the existing literature and state-of-the-art works. Multiple techniques are reviewed that cover both attack and defence strategies for network traffic and malware domains.

3.1. NIDS Evasion Techniques

Since the adoption of intrusion detection systems, the detection rate has been extensively researched and improved. Consequently the use of various evasion techniques and countermeasures has also been studied and developed through the years resulting in creative techniques for traffic manipulation. The scope of this research are anomaly-based systems, but the techniques for both will be reviewed as many of the proposed modifications can be used against both types of NIDS. The existing adversarial attacks can be divided into feature-space attacks where the extracted features are modified to evade the target classifiers and traffic-space attacks where the network packets are directly altered before the feature extraction.

3.1.1. Traffic-space Adversarial Attacks

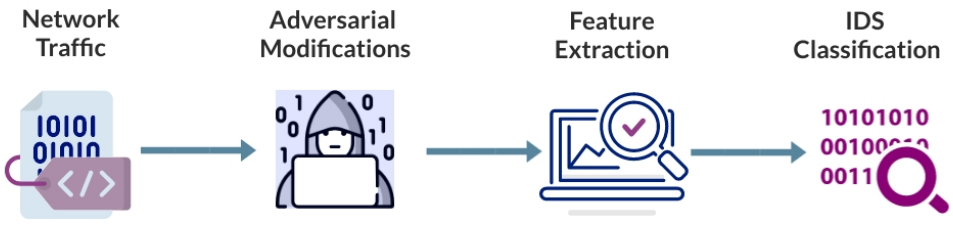


Figure 3.1: Traffic-space Attack

Traffic space adversarial attacks are modifications directly applied to the network traffic. In this category, techniques such as packet splitting or payload modification are applied with the

goal of masking the true malicious contents of the packet. The traffic space attacks are bounded by various constraints which need to be taken into consideration by the attacker when performing modifications. Additionally, there is only a limited set of attributes that an attacker can directly influence and a set of attributes that are indirectly affected by the modifications that the attacker performs. Therefore these attacks can be considered more challenging and complex for achieving satisfying results. In the following works adversarial attacks are performed in traffic space, meaning that the attacks are performed directly on the network packets. The success of these adversarial modifications is also based upon the direct evaluation of the modified traffic against existing ML-based detectors.

One of the first techniques that have been used for this purpose are specifically developed to evade signature-based NIDSs. [41] propose several different attacks. One proposed attack is the insertion of malicious packets in an existing network traffic stream. Another attack that the authors propose is a combination of multiple modifications such as fragmenting packets, sending packets out of order, modifying the TTL value of the packets, and creating overlapping fragments. Similar techniques have also been used in multiple works where the focus is put on transforming a network packet and specifically manipulating the length by padding and fragmentation [11, 35, 43].

A Polymorphic Blending Attack (PBA) is proposed in [15, 16] which performs modifications on the packet payload to make it look like a benign packet such that the content is not matched by the NIDS. The normal traffic is examined and accordingly a profile is created that matches it. Then a malicious packet is crafted using an algorithm to find an encryption key that matches the benign packet payload. An approximation method is used to find the adversarial examples. Similar white-box techniques have been applied for modification of packet payloads by other works such as the polymorphic shell code attack [9, 11, 49]. Encryption techniques have also been applied for hiding botnet traffic [46]. The modifications are performed by encoding and padding the payload or malware binaries and can be used to disrupt the intrusion detection system as well as evade it. Most of the works have only considered applying the attacks separately. More recent works have also implemented the proposed traffic attacks and explore their effectiveness on novel ML-based NIDS solutions. [25] where the authors propose an obfuscation technique to evade ML-based NIDS. This obfuscation technique is based on modifying different network connection properties such as duplicating, reordering, and dropping packets, adding a delay between packets, and modifying the maximum transmission unit (MTU) value. The authors also propose a defence technique that works by expanding the knowledge base of the classifier, in other words, by adding malicious samples to the training set to increase the robustness of the NIDS.

As a result of the broad adoption of ML-based solutions for detecting malicious traffic or malware, researchers have also explored the use of machine learning for the generation of evasion traffic. In [24] the authors show how to craft adversarial inputs in the network domain and three NIDS are evaluated in an adversarial setting. The authors propose constructing fake packets with similar features as the original packets, changing the time between subsequent packets and fragmenting the packets. The approach used in this work is a white-box approach since the detection model and the features are known and for each packet alteration feedback from the classifier is used. Each attack is tested against the target NIDS and improved. If no successful adversarial examples are found with this attack, the attack is combined with the other proposed techniques.

[23] A novel method Reconstruction from Partial Observation (RePO) is proposed as a new mechanism to build an NIDS. This method uses de-noising auto-encoders that are able to de-

tect different types of network attacks. By using this method the robustness of the NIDS against adversarial attacks is increased. The packet-based NIDS is created by grouping the packets by their sender and receiver IPs. Then a feature vector is created for each packet which contains 29 features, and a decision is made based on a combination of the previous 19 packet features and the current packet features. This technique is proposed as a defense for adversarial samples against ML-based NIDS.

[22] propose a practical traffic space evasion attack and defence for ML-based NIDS. The authors evaluate different ML-based classifiers and also different feature sets. The attacks are defined as a set of network traffic mutations with the goal of mutating as many features as possible. The assumptions are that the features and the target classifiers are not known and to solve this issue, the authors propose a set of features that are commonly used by NIDS. The proposed mutation that is directly applied on the original traffic is changing the time between two packets, the other mutations are only applied on new injected packets, and these mutations consist of padding packets, changing the sequence number of a packet, requesting an already established connection and altering the time-to-live value. The adversarial features are generated using the Generative Adversarial Network (GAN) algorithm where a neural network is used for the generation of these examples. Subsequently, the Particle Swarm Optimization (PSO) method is used to approximate the solutions of GAN by iterative searching the network traffic until it finds the most similar values to the adversarial features. The main limitation of this work is that an approximation algorithm has to be used for mutating the network packets and no constraints are considered for optimizing the adversarial examples. Additionally, the set of modifications is very limited.

Traffic space modifications based on feature space attacks have also been applied to evade malware detectors. In [12] the authors describe a set of possible and valid malware transformations which can be used for this purpose. The analysis is based on the modification of Windows portable executable files and tested against several state-of-the-art ML-based detectors. The adversarial optimization is formulated as a minimisation problem constrained by the size of the adversarial inputs. Then the obtained values are translated to traffic space attacks by inserting data extracted from benign samples. The authors have shown that the modifications are successful in evading many of the available commercial solutions.

3.1.2. Feature-space Adversarial Attacks

In contrast to the traffic space attacks, feature space attacks are performed on an extracted feature vector. An optimization algorithm is selected to find the optimal adversarial examples with a predefined set of alterations that can be performed. One of the first domains were feature space attacks were developed and applied is the image recognition field.

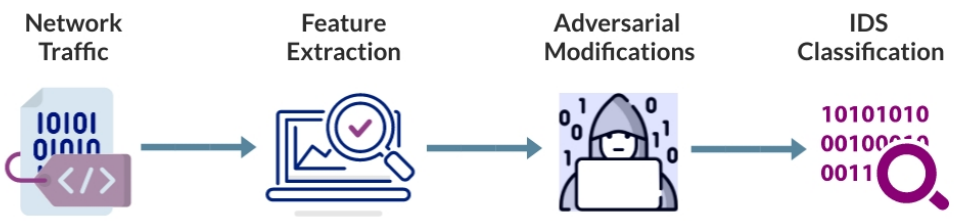


Figure 3.2: Feature-space Attack

However, perturbations on features can be more easily performed on images than on net-

work traffic. This is possible because images have a bigger feature space which can be perturbed and a lot more dimensions. As opposed to images, network traffic has a limited set of features and these features cannot be arbitrarily altered since the functionality and original content of the network traffic must be preserved. This means that the perturbation space of images is unbounded as opposed to the perturbation space of network traffic.

The existing feature-space attacks can be divided into white-box, gray-box, and black-box attacks where the white-box attacks require full knowledge of the feature set and the target classifier, the gray-box attacks require knowledge of a classifier or sample data similar to the data on the target network, while the black-box attacks do not require any knowledge neither of the classifier nor the feature set. Feature space techniques have been used on malware and network traffic are reviewed in this section.

A gradient based method for crafting adversarial is proposed in [6]. This technique is tested several classifiers based on optimization functions. The authors introduce a penalty based on the density function that is supposed to favour adversarial examples that are closer to the benign samples located in a higher density region. The authors also introduce the idea of a surrogate model which can be used for crafting adversarial examples that will attack another target model. The evaluation is done by simulating different attack scenarios, from a scenario where the adversary has very limited knowledge of the classifier to a scenario where the adversary has full knowledge of the target classifier. The approach has been tested in feature space on toy data and in feature space on PDF malware samples by assuming that the features can only be incremented, which translates to the insertion of data in real world samples.

[32] examine the effect of adversarial examples in an encrypted C2 malware detection scenario. A neural network based intrusion detection system is used for evaluation and the adversarial examples are crafted using the Fast Gradient Sign Method (FGSM) combined with additional domain specific constraints to ensure that the malware samples will remain valid. The conclusion of this work was that the adversarial examples succeeded in evading the classifier but also that the robustness of the model could be improved if the classification model was re-trained with the adversarial samples. The validity and functionality of the traffic were not evaluated.

In [20, 21] the generation of adversarial examples for malware with neural networks is explored. A method is proposed that uses gradients to compute perturbations for the malware features in an iterative manner. The features are represented as a binary vector. To maintain the original characteristics of the application, the L_1 norm is used to bound the overall number of modified features. After showing that the proposed attack is successful, the authors propose a defence technique based on feature reduction. The idea is to train detectors only on a feature set that is selected based on the mutual information metric.

Feature space attacks have also been explored on network traffic data. [42] propose a novel network intrusion detection method based on swarm optimization. The Artificial Bee Colony (ABC) algorithm is used for the optimization and a Random Neural Network (RNN) based adversarial intrusion detection system is used as the target IDS. The adversarial attacks are generated using the Jacobian Saliency Map Attack (JSMA) algorithm and based on the obtained accuracy and F1 score results, the authors conclude that this novel method is more robust than the Deep Neural Networks (DNN) used for the same purpose.

[4] present a novel machine learning intrusion detection approach which makes use of a Multilayer Perceptron (MLP) network. The authors first show that the performance of this model is very high for the two datasets on which it is evaluated. To fool the MLP IDS the Model Evasion Attack is proposed. With this technique the generation of adversarial samples was done using the JSMA method. This is done in a white-box setting and the results show that the adversarial examples greatly reduce the accuracy of the model. The authors do not consider

the mapping possibilities from feature to traffic space nor the applicability of this attack in real scenarios.

Three methods are proposed for crafting adversarial examples on network traffic by modifying the flow based features in [1]. The methods that are used are adaptations of the GA, PSO, and GAN algorithms. The adversarial examples are generated and evaluated on two intrusion detection data sets commonly used for research purposes. The authors show that the methods are capable of generating highly evasive network traffic samples. However, the evaluations of the methods are only performed in feature space.

[3] evaluate the robustness of the Random Forest classifier in a network intrusion detection scenario. The adversarial samples are crafted by altering different sets of network flow feature with a minimum of 1 feature and a maximum of 4 features. The feature values are only increased with different step size, no other adversarial methods are used for this purpose. The authors assume that the flow features can be easily increased by adding bytes or delaying the traffic in real scenarios. The detection rate of the Random Forest classifier is heavily reduced for the majority of generated adversarial examples. With this experiment, the authors show how fragile machine learning-based IDS can be even with simple adversarial techniques. In [45] two different methods for generation evasive network flows are proposed. The first method is an adjusted JSMA algorithm that considers constraints from the TCP/IP flow features by grouping them into sets. The other technique combines the Adjusted JSMA and a histogram sketching technique. The authors perform experiments on flow features from the NLS-KDD data set and show that adversarial modifications are successful in a constrained scenario. The experiments are performed only in feature space. The authors also randomly restrict the algorithm to modify a subset of features and show that even with some immutable features this method remains effective. Constraints that are considered are mainly protocol related and for each sample only a resulting modifiable set of features is used.

An extensive research is performed on various feature space attacks such as CW and FGSM on network traffic in [36]. The goal of the authors was to evaluate the robustness of ML-based NIDS in white, grey and black-box setting. Furthermore, a defense method is introduced to improve the robustness by selecting a diverse feature set. The NLS-KDD data set is used for the experiments and adversarial modifications are performed on the network flow features. This is not representative since alterations are only performed in feature space on flow data which cannot be mapped to practical modifications.

[33] propose a black-box optimization technique, reservoir sampling, that can generate adversarial samples which have the transferability property and two attack crafting methods based on Support Vector Machines (SVM) and Decision Trees (DT), they can be transferred to unknown target classifiers and succeed at reducing their detection rate. Here, the assumption is that the target classifier and relevant parameters are not known and the optimization is performed between different pairs of surrogate and target machine learning techniques such as Support Vector Machines SVM and DT. The authors show that adversarial examples created with one classifier do transfer to another classifier also if it is trained on a different data set. Several factors are recognised which make it easier for adversaries to generate adversarial examples.

3.2. Discussion

To summarize, the adversarial attacks that are applied to network traffic can be divided into two categories:

1. **Traffic space evasion attacks**: These attacks are directly applied to the network traffic. Existing packet crafting techniques include fragmentation, padding, delay between two

packets, injection, modifying the TTL value, reordering and duplicating fragments and modifying different flags.

The main limitation of the previous works is that most of the proposed techniques work by randomly guessing values until the target NIDS is fooled. This is done by using feedback loops where different values are tried for the attacks and based on the replies of the target system the best performing parameters are selected. Additionally, the system of works focus on modifying the traffic payload such that it bypasses signature-based IDS and are not necessarily effective for machine-learning based IDS.

Another limitation is that not all of the proposed attacks are used in the same context, that is for the purpose of hiding the already existing attacks or evading the target classifier. Many attacks are performed with denial of service as the main goal or with the goal of probing the network. Because of this, not all attacks are successful in evading the classifier and some only succeed at slightly reducing the detection rate of the NIDS. Aditionally, the proposed attacks have a different effect on packet based NIDS as opposed to flow based NIDS, however this is not further explored and no comparison is made between the different types of NIDS.

2. **Feature space evasion attacks**: These attacks are applied to the features extracted from the network traffic. The feature space attacks proposed by previous works are mainly attacks that have been proposed for and applied to the image recognition field. When these attacks are applied to network traffic, in the reviewed works, the domain constraints are not considered. The adversarial crafting methods that are used in the relevant works: Fast Gradient Sign Metod (FGSM) [19], Carlini and Wagner Attack (C&W) [8], Jacobian-based Saliency Map Attack (JSMA) [34],Generative Adversarial Networks (GAN) [18], Projected Gradient Descent Attack (PDG) [29], Basic Iterative Method Attack (BIM) also knows as Iterative-FGSM [27].

   The main limitation of this approach is that these attacks are not realistic since they cannot be directly applied to the network traffic. Even though the target IDS can be evaded successfully and the accuracy and detection rate can be reduced, these attacks are performed on the already extracted features of the network traffic however an adversary can only modify the actual traffic, not the features that the target NIDS extracts. There is also no known technique which can revert the extracted features into packets such that these can be used in a real scenario. Feature-space attacks might be applicable if only a subset of modifiable features are perturbed using the above mentioned techniques and if these perturbations can then be translated to the traffic space, through applying on of the traffic space attacks considering that the features are only perturbed within their boundaries.

3.3. Research Gap

The main research gap that can be identified from the literature review is that most of the existing methods do not consider domain constraints and have mainly proposed feature space adversarial crafting modifications. Several works limit the perturbations to a set of features that can be modified but do not impose other constraints on these features. In a realistic scenario this will not work since the constraints and restrictions that are imposed on the set of features are more complex and strict and hence cannot be ignored. Because of these reasons, these methods cannot be used for generating practical attacks since this will result in illegitimate network packets. The functionality of the attacks has also not been practically evaluated.

Furthermore, most of the works only consider a small set of modifications such as padding and splitting the packet while there exist many more possibilities for altering a packet. However, the identified challenge are the constraints imposed by specific network protocols and the packet structure which has interfeature relations that cannot be must be considered when applying adversarial perturbations.

1. Related work on adversarial attacks for the network traffic domain mostly considers feature space attacks that are not evaluated in a practical setting.
2. Related work rarely considers the network traffic domain constraints during the generation of adversarial examples.
3. Related work mostly applies a set of modifications that are known to be possible such as padding or splitting a frame, however, there is a larger set of practically applicable modifications which can be used if implemented accordingly to the constraints.
4. The practical applicability of attacks based on optimized adversarial values is not evaluated.
5. It is not clear to what extent adversarial optimization techniques can create a representative set of adversarial feature values.
6. It is not clear which features are altered when crafting adversarial examples for network traffic and how these affect the validity of the traffic.
7. It is unclear to what extent constrained domains are susceptible to adversarial evasion attacks.

3.4. Overview of works and their main characteristics

In this section, an overview is provided of the relevant works and their main similarities and differences.

Paper	Trial and Error/ Random	Adversarial optimization	Extracted feature modifications	Practical feature modifications	Flow-level	Packet-evel	Evaluated on signature-based NIDS	Evaluation on ML-based NIDS	Validity analysis
[41]	✓			✓		✓	✓		✓
[15, 16]	✓	✓	✓	✓		✓			
[9]	✓			✓		✓	✓		
[11]	✓			✓		✓	✓		
[49]	✓			✓		✓			
[25]	✓			✓		✓		✓	
[23, 24]		✓	✓		✓			✓	
[22]		✓		✓		✓			
[12]		✓	✓			✓		✓	
[6]		✓	✓	✓	✓		✓		
[32]		✓		✓		✓			
[20, 21]		✓	✓	✓		✓	✓	✓	
[42]		✓	✓			✓		✓	
[4]		✓	✓	✓			✓		
[1]		✓	✓		✓			✓	
[3]		✓	✓		✓			✓	
[45]		✓	✓		✓			✓	
[36]		✓	✓		✓			✓	

4

Experimental Setup

In this chapter, a detailed overview is given about the data and techniques used during this research.Additionally, the conducted experiments are explained together with the steps that are taken for each experiment.

4.1. Dataset Exploration and Preprocessing

The dataset that is selected for the experiments is the Intrusion Detection Evaluation Dataset (CIC-IDS2017) from the Canadian Institute of Cyber Security [1]. The dataset is generated artificially and tries to mimic real network traffic. It exists of 5 days of traffic of which only one day is completely benign (Monday). In the other 4 days of traffic, various attacks have been generated and captured. This dataset includes the most common attack types such as Brute force, Dos and Web-based attacks and Network scans. Among the included attacks are Infiltration, DDoS, SSH-Patator, FTP-Patator, Heartbleed, SQL Injection, XSS, Port Scan, DoS LOIT, DoS Hulk, DoS GoldenEye, DoS Slowhttp, DoS Slowloris and Botnet Ares.

The authors have generated this dataset for the purpose of analyzing network traffic and developing and evaluating intrusion detection systems. This is done by generating a set of different user profiles which mimic different behaviours and network traffic. In total, there are 12 different machines which generate normal traffic on the target network. These machines are based on various operating systems including Ubuntu, Windows, and Mac operating systems with different versions. The attacks are generated from another network using two machines of which one is Ubuntu-based and the other one is a Windows based machine [44]. A distribution of the attacks can be seen in 4.1. From this figure, it is evident that the DDos attacks and port scan make up for the majority of attacks present in the data.

4.1.1. Packet data

The packet data is directly extracted from the packet captures using Wireshark. The features that are depicted in A.1 are information that can be found in the packet header. This feature set is used in many works for the classification of network packets as it characterises the data through the available header information excluding the payload and is effective for intrusion detection [7, 30]. The packet payload can also be extracted from the packet, however, this is not included in the feature set because different encoding and detection techniques are required to detect anomalous payload compared to network traffic features meta-data.

[1] https://www.unb.ca/cic/datasets/ids-2017.html

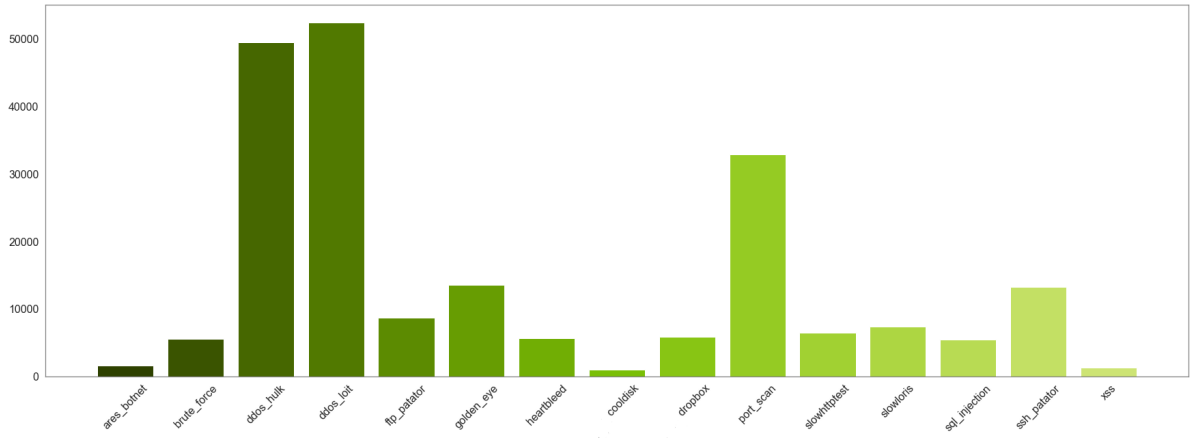


Figure 4.1: Distribution of attacks in the data

4.1.2. Flow data

The flow data is directly extracted from the pcaps using a flow extraction tool provided by the Canadian Institute for Cyber Security (CIC) [2]. This tool combines all packets that are sent forward and backward during a session between two hosts and calculates the features of the data flow. The features that are considered are flow features such as the number of packets and bytes in a flow. Other derived flow features are also used for intrusion detection, however to select additional features, more in depth analysis is needed, however, since this is not the focus of this work, and most of the related works use the same feature set, in the scope of this work this feature set is considered sufficiently representative.

	Number of packets	**Number of flows**
Benign	1055494	405169
Malicious	1054561	121981

4.1.3. Feature Extraction and Preprocessing

The benign data is taken from one day (Monday) where only benign traffic is present. A separate pcap for the benign data and each attack is extracted from the dataset by selecting the connections between the attacker and target IP addresses within the time frame as specified by the creators of the traffic.

The packet captures are processed and the relevant statistical features are extracted from the pcaps. Then a file is generated for each pcap with the relevant features. Next, the NaN values are replaced with 0 and the categorical features are removed. The benign samples are labelled with 0 and the attacks are labelled with 1. To ensure that all attacks are represented, the train set contains at least one sample of each attack present in the data. Because the original dataset is an imbalanced dataset, the benign samples are downsampled to be the same size as the malicious samples. The data is randomly split into a train and test set and is normalised using a MinMax scaler that is fitted to the train set. The same preprocessing steps are taken for the flow features.

[2] https://www.unb.ca/cic/

4.1.4. Classification Models

For the performance evaluation of the proposed technique, five types of classification models are chosen: Logistic Regression (LR), Neural Networks (NN), Support Vector Machines (SVM), Random Forests (RF) and Decision Trees (DT). This set of models is chosen because each model employs a different criterion and learns to classify the samples differently. As all five techniques are explored and evaluated for the goal of intrusion detection, the results can be considered representative for this scope. The hyper-parameter selection is done by trial and error for the Logistic Regression and Neural Network models since these models are implemented in PyTorch where no default settings can be chosen. For the other models, the default set of hyper-parameters is used as selected by the Scikit library. The Logistic Regression model is chosen as a surrogate model for the generation of adversarial examples. This model is chosen for several reasons. First it is one of the simplest classification models and it is interpretable. The method that is used as a basis for the proposed method is the Fast Gradient Sign Method, this method has exact results for linear classification models such as Logistic Regression [19]. Therefore, the Logistic Regression model is chosen as the surrogate classification model in this work.

4.2. Attack Generation

For exploring the possibilities for the optimization of evasion attacks several experiments are conducted. The methods that are used and the steps that are taken for each experiment are listed here.

4.2.1. Initial Practical Adversarial Modifications

This is the first step in the exploration phase that has been practically implemented and tested. Listing all the possible modifications these were implemented using Scapy. With the implemented attacks, the traffic pcaps are modified and rebuild. From these new pcaps the features are extracted as described in the previous section and the data is evaluated with the pre-trained classification models.

1. Select set of modifications
2. Implement modifications with Scapy
3. Read the original pcaps and change accordingly
4. Extract features from new pcaps
5. Perform prepossessing steps from 1.
6. Evaluate on pretrained models.

4.2.2. Constrained Feature-space Adversarial Modifications

Existing implementations are used of the known adversarial method IFGSM. A new method is proposed, which is a modified version of the IFGSM that includes the set of inter-feature relations and boundaries formulated as constraints during the optimization. This method is called the Constrained IFGSM method (CIFGSM). This method is then applied to the pre-processed data as described in the previous section. The resulting data set is evaluated using pretrained models to measure the success rate of the adversarial perturbations. Additionally, the effect of the introduced penalty is evaluated by counting the number of malformed points.

1. Select a set of adversarial crafting algorithms
2. Define penalty function that includes domain constraints

2. Define the custom loss function
2. Implement the constrained attack with custom loss
3. Read the original preprocessed attack data
4. Apply the CIFGSM algorithm to the features
5. Result is a preprocessed adversarial data set
6. Evaluate on pretrained models.

4.2.3. Optimized Practical Adversarial Modifications

For performing the experiment with the adversarial optimized values obtained by using the previous method, the feature values are extracted for each packet in the data. The adversarial values are separated for each original attack in the data. These are then rounded before they are given to the modification functions as floats are not accepted. After the values are modified, new pcaps are generated with adversarial packets and these are then controlled using Wireshark. Additionally the packet captures are replayed with tcpreplay to assess whether the adversarial packets are received. After this check, the new features are extracted, preprocessed, and evaluated.

1. Extract the feature values from the optimized data
2. Apply the pcap modifications with these values
3. Check validity of pcaps
4. Extract features from new pcaps
5. Perform the prepossessing step
6. Evaluate on pretrained models

5

Adversarial Attacks on Network Traffic

In this chapter, an overview will be given of the attacks that are selected for the purpose of evading intrusion detection systems. Several adversarial crafting techniques are selected and applied on a network traffic data set to explore how successful these are at evading the target classifier.

5.1. Threat Model

In this section, the threat model that is used for crafting adversarial examples and evaluating the anomaly-based NIDS will be defined. For this experiment, both a white-box and a gray-box scenario are considered.

Adversary Goal

The goal of the adversary is to evade a target NIDS and successfully attack the victim system. To accomplish this goal, the adversary has to perturb the malicious packets in such a way that the existing attacks in the network traffic will not be recognised by the target NIDS while preserving their original functionality and successfully reaching the network to perform the attack.

Adversary Knowledge

In the exploration phase, the adversary has knowledge of the target NIDS and features that are used but has no knowledge nor a copy of the benign traffic. The assumption here is that publically available 'normal' data is used to train the classifier. In the optimization phase the adversary chooses a surrogate classifier she will use to optimize the adversarial examples, in this phase also the features which are used by the NIDS are known. For the evaluation phase, only the initial model is known. Details about the other target systems which are used for this experiment are considered unknown. This means that both a white-box and gray-box scenario are used.

Adversary Capabilities

The adversary can only modify the original network traffic attacks she has created. In the exploration phase, the attacks are modified based on the general knowledge that is available to the adversary and for the creation of adversarial network packets no additional resources are

used. The adversary is assumed to have access to traffic analysis and modification tools but cannot query the target NIDS.

5.2. Adversarial Network Traffic Attacks

To establish which adversarial evasion attacks are effective against NIDS, several experiments are conducted. In these experiments, a selection of packet modifications is selected and implemented and the effects of the adversarial modifications on both a packet and flow level are explored. This is a necessary step to do since more recent literature on practical adversarial modifications for network traffic is limited and there are special constraints for the modification of network traffic that need to be taken into consideration.

5.2.1. Selection of Adversarial Attacks

There exists different types of attacks for network traffic which have various goals such as making the target system crash or performing reconnaissance on a network. A number of different attacks are considered during the experiments, including DoS, web attacks, port scans, and botnet traffic. The main goal of this research is to explore to what extent attacks can be hidden from the target system. This can be done by modifying the packets in such a way that they are more similar to the benign packets or just different than the malicious model. This means that, for example a SQL Injection attack must retain its functionality and still be an SQL Injection attack after the packets are modified. For this purpose, the subset of attacks that are considered in this study are evasion attacks.

The traffic space evasion attacks can generally be divided into three categories based on where the modifications are performed. The first category are modifications to the packet header, here the packet header information is altered, and this can be done by modifying metadata such as flags or other IP options. The second category is modifications on the payload, here the payload modified by increasing or decreasing the size of the payload data. An example of such attacks are fragmentation and padding. The third category of modifications concerns time-based packet attributes such as the time to live value or the interarrival time between packets. Another technique that is used to evade signature-based intrusion detection systems is modifying the content of the payload. This technique is not considered in this research because the intrusion detection systems that are used for evaluation use statistical attributes of the network traffic and do not examine the actual payload of the packets. The relevant attacks that are considered realistic and are used during this research are listed below.

Fragmentation

When a packet is fragmented into two smaller packets, the header field of the original packet is copied into the newly created packet and the data of the original packet is divided into multiple parts where the first part must be a multiple of 8 and the remaining parts contain the remaining packet. Subsequently, the first part of the data is set to the first fragment and the length is set to length of the first fragment.The More Fragments (MF) flag is also set to one and the fragment offset is updated. The same happens for the second fragment. The fragment information (MF flag and fragment offset) of the original packet is 0, unlike the MF flag and fragment offset of the fragments which contain the previously described information.

Fragmentation is performed as described in the following equation. Here F represents the frag-

mentation function, packet is the size of the original packet, f_i is fragment i and f_r represents a fragment with the remaining bytes. The size of the fragment is arbitrarily chosen as a multiple of 8.

$$F = \begin{cases} packet(f_i), & \text{if packet (mod 8)} \\ packet(f_i) + packet(f_r), & \text{otherwise} \end{cases}$$

On the fragmented packets, reordering and duplication are applied. The reordering and duplication of fragments is possible because the TCP protocol has the capability to receive out-of-order and duplicate packets.

- **Reordered fragments**: This attack is performed by randomly shuffling the fragments of each packet.
- **Duplicate fragments**: This attack is performed by randomly duplicating fragments for each packet.

Padding

Padding is a method which appends a number of zero bytes at the end of the packet payload. This technique is used to fill malicious packets with payloads such that these packets are more similar to the benign packets and the IDS mistakes them for benign packets. In the following equation, the padding function P is described. A padding size is chosen arbitrarily since there exists no method to determine the optimal padding size beforehand.

$$P = \begin{cases} packet + padding * (x - packet), & \text{if packet < x} \\ packet, & \text{otherwise} \end{cases}$$

Equal length packets

The Equal Length attack combines fragmentation and padding, the idea behind this attack is to make all the incoming packets of the same size. When the packet is larger then the chosen size x, the packet is fragmented into multiple x-sized packets. If the packet is smaller than the chosen size x, padding is added to its payload and otherwise the packet remains unmodified.

$$EqL = \begin{cases} P(packet), & \text{if packet < x} \\ F(packet), & \text{if packet > x} \\ packet, & \text{otherwise} \end{cases}$$

Delay

The packet delay attack is a technique where packets are delayed by x seconds. This can be done sequentially or randomly. During this exploration, both ways are tried since there exists no guideline or method to find the optimal delay between packets.

$$Delay = \begin{cases} packet.time + x, & \text{for packet} \in \text{Rand(0, packets)} \\ packet.time + x, & \text{for each packet} \in \text{packets} \end{cases}$$

TTL value modification

The time to live (ttl) is the time span for which a packet is allowed to circulate on a network. After this time span passes, the packet is dropped. For this attack, the time to live of a packet is changed according to a randomly chosen value t.

$$TTL = \begin{cases} packet.ttl + (t - packet.ttl), & \text{if packet.ttl} < \text{t} \\ packet.ttl + (t - packet.ttl), & \text{if packet.ttl} < \text{t} \\ packet.ttl, & \text{otherwise} \end{cases}$$

Packet Injection

With this attack, new dummy packets are created and injected into the existing network traffic. These dummy packets have a random payload. For generating these packets Scapy was used. The packets were generated by selecting a real packet, copying the source and destination address, and randomly filling the payload.

$$Injection = \{dummy\_packet, \quad \text{for packet} \in \text{Rand(0, packets)}$$

Random Src Port

The source port is used by the host network to see the different incoming flows and it is randomly chosen. This attack changes the current source port to a randomly chosen new source port within the existing range of port values.

$$Random\ Src\ Port = \{src\_port, \quad \text{for src\_port} \in \text{Rand(0, port numbers)}$$

Each of the above mentioned attacks requires some values to be selected beforehand, such as the fragment size, or the TTL value. However, to the best of our knowledge exists no realistic method to select values for these attacks that would be effective in evading the target IDS. Nor does a method exist on how to select values such that these do not corrupt the packets and keep the initial malicious functionality of the packets intact.

5.2.2. Results

For this experiment it is assumed that neither the benign traffic nor the target classifiers are known. The adversary has only access to and can modify the malicious packets she created. The attacks that are crafted are arbitrarily chosen. The reason for this is the assumption that an adversary has limited knowledge of the target NIDS and the benign network traffic on the network that she is trying to attack even with sufficient investigation and preparation.

From the results in table 5.1 it can be seen that some of the attacks succeed in bypassing the packet and flow based classifiers while some are less successful. The most successful attacks were the attacks that combined multiple packet modifications such as a combination of padding, splitting, delay and different TTL value. Regarding the sole modifications, fragmenting the packet was the most successful, followed by assigning a random source port to each malicious packet. The attacks that were the least successful for the packet-based classifier were the

delay between packets, the modified TTL value and the injection of dummy packet. The reason that the delay attack was almost not successful is because the classifier looks at the features of each packet separately. Because only the inter-arrival time of the packets was affected by this modification while the other attributes of the malicious packets were not affected, they still remained more different compared to benign packet header features. Therefore, most of the malicious packets were classified correctly. The injection of dummy packets is also not very effective for the packet-based classifiers as opposed to the flow based classifier where it manages to hide the actual attacks. This again is because the packet-based classifiers look at each packet separately and adding dummy packets has no effect on this level. In the flow based classifiers, it succeeds in reducing the detection rate because it affects more of the flow-based features such as an increase in bytes, total packets and duration of the flow.

The most successful attacks ror the flow based classifier were padding, dummy packet injection and adjusting all packets to have equal length. Here, the least successful attacks were reordering and changing the TTL value. Additionally for the flow based classifiers, it can be seen that the SVC and Logistic Regression classifiers were overall more resilient than the Decision Tree and Random Forest classifiers.

	Success Rate							
	Packet				Flow			
Adversarial Modifications	LR	SVC	RF	DT	LR	SVC	RF	DT
Org	.03	.02	.0	.0	.09	.09	.0	.0
Split 8	.9	.89	.0	.0	.04	.04	.12	.9
Split 32	.72	.03	.53	0.6	.15	.16	.33	.83
Split 64	.5	.02	.38	.39	.09	.09	.12	.2
Pad 70	.16	.12	.0	.03	.25	.25	.3	.84
Pad 1600	.03	.0	.14	.07	.2	.42	.34	.66
Delay 05	.16	.12	.0	.07	.09	.09	.12	.2
Delay 15	.06	.05	.0	.0	.09	.09	.12	.2
Reorder	.5	.02	.39	.39	.02	.02	.11	.2
Duplicate	.52	.0	.47	.47	.53	.4	.58	.2
Inject	.07	.04	.0	.0	.43	.21	.38	.83
TTL	.04	.03	.0	.04	.22	.22	.26	.86
Equal 64	.81	.04	.29	.44	.81	.74	.75	.72
Equal 300	.2	.15	.18	.14	.4	.4	.14	.63
Srcport	.39	.28	.19	.16	.08	.08	.79	.57
Combined	.9	.46	.9	.75	.15	.16	.82	.62

Table 5.1: Success rate of the packet and flow based classifiers for each attack

Generally, from the obtained results, it can be concluded that the performance of the flow based classifier is worse than the performance of the packet-based classifiers. In the case of fragmentation, the total packets in a flow drastically changes and in the case of padding this happens with the total bytes. For the time-based attacks also the modifications are reflected only by the duration of the flow. Therefore, it is expected that this feature will not add much to the evasiveness on packet-level.

5.3. Conclusion

From the experiments that were conducted with a chosen set of adversarial samples crafted for network traffic, it is evident that such modifications can have a great effect in reducing the detection rate of different types of classification models. It is important to note that the values for the attacks were chosen randomly, independent of the benign network traffic or classification model settings. The tested data contained different types of attacks (DDoS, Botnet, Port Scans) and some of the adversarial modifications were effective for the majority of malicious samples. However, in some cases it can be concluded that the attacks did not have much effect. There are several reasons for this. The first reason are the arbitrary values that are chosen for this experiment. The same attacks with values optimized on some classification models are likely to have greater success. The second reason why some of these attacks do not work is related to the feature set that the classifiers are trained on. It can be seen that the packet-based classifier was more robust than the flow based classifier. For an adversary who has no knowledge of the target classifier, the used feature set, or benign traffic on the target network, it can be concluded that it is challenging to perform successful evasion attacks. Therefore, it is interesting to explore the possibilities for finding optimal values using adversarial optimization techniques.

6

Network Traffic Constraints

In order to modify the network traffic correctly such that it bypasses the NIDS but keeps its original functionality it is important to know which modifications can be performed. In the case of feature-space attacks, perturbation is performed on all features that are extracted from the network traffic, but as discussed before, this is not realistic since many of these modifications would cause the connection to break or are not applicable in a practical scenario, when directly performed on packet captures.

A network packet is used to send data between two hosts. One such packet contains a payload which represents the encoded data and other header information that is relevant for the connection. The source, destination, protocol and other statistical information that is needed for the host to properly receive the data are included in the packet header. These features are representative for a single packet on the network but do not capture information about the total connection between two hosts. The chosen feature set can be seen in A.1. The packet-based and flow-based classification models use different feature sets for their predictions. The packet-based classifier looks at the packet level and extracts a set of features for each packet, while the flow based classifier looks at the features of a group of packets. Both data sets used in this research are obtained from the same packet capture files. The selected set of features is a representative enough set since these are values that are directly extracted from the packets and more advanced feature sets also use derived features which are mostly obtained through them. A flow represents the number of packets sent between two hosts. Flow based features are extracted directly from the packet captures using the tool Argus. A set of common flow features are used which can directly be extracted from the traffic. This feature set contains both numeric and categorical variables. This feature set captures information about the traffic flow between two hosts and more detailed information about the packets in this flow. The chosen feature set can be seen in A.2.

Since this is an experiment and the attacks are conducted with a limited set of values this is a good representation of legitimately modified network traffic. However, there are infinite possibilities in how the traffic can be modified in terms of values that can be chosen for fragmentation, padding, and delay among other modifications which if chosen wrongly can again cause the resulting packets to be broken and thus the evasion attack to be unsuccessful. For this reason, the other important factor that needs to be considered after having established the set of modifiable features is how these can be changed legitimately and which constraint should be taken into account before the modifications can be applied in a practical scenario where evasion techniques are used in traffic space.

6.0.1. Network Traffic Characteristics

To determine the constraints for network traffic features firstly a network connection needs to be explained. A network connection consists of multiple different network layers which all serve a certain purpose in transferring the data from point A to point B. This functionality is described with the OSI model [1] which contains 7 different layers. However, only limited information can be extracted from the network traffic which mostly contains techniques used in the host layers. The data transmission is performed through protocols which represent different sets of rules specifying how to transfer the data correctly.

The network traffic features can be modified in traffic space only on a packet level by modifying the data contained in the packet capture files. These will then indirectly have impact on the flow-level features. From the packet-level features, the majority of features are considered modifiable except the source address, the destination address, the destination port, and some information about the connection that is derived from other features such as the fragment offset. The destination address and destination port are not modified because the goal of the adversary is to send malicious traffic to the target system. If the destination is altered, this means that the malicious traffic will not be delivered to the right system and that the attack cannot be successfully performed. From the features that can be directly modified in traffic space, there are some inter-feature relationships that need to be considered for crafting a valid packet. Furthermore, modifiable features may have predefined boundaries based on the underlying protocol implementation which also needs to be preserved. In the following subsections, rules will be defined for each of the different protocols and the features that are obtained from these protocols as well as other packet components.

There are also constraints related to the possible adversarial modifications. For example if the delay is very large, the packet will eventually be dropped and this is not the purpose of evasion attacks since the packets need to be delivered correctly in order for the original attack to be performed. Another constraint is the fragmentation size, a packet will only remain valid if its size is a multiple of 8. Several other constraints exist on header features such as minimum and maximum frame size, ttl, delay, and flag combinations. A summary of the relevant network traffic constraints can be observed in Table 6.1.

[1] https://www.iso.org/ics/35.100/x/

Protocol	Feature	Constraint	Relation
Ethernet	frame.len	[46, 1500]	frame.len ≤ 1500 $\implies$ ip.len ≤ 1500 − 46
	frame.time_delta	[0, 30]	frame.time_delta < ip.ttl
IP	ip.len	[20, 1500]	ip.len ≤ 65507 $\implies$ udp.len ≤ 6553520 ip.len ≤ 65507 $\implies$ tcp.len ≤ 6553520
	ip.ttl	[2, 255]	ip.ttl > 255 $\implies$ icmp.type=11
	ip.flags.df	[0, 1]	ip.flags.df = 1 $\implies$ ip.frag.offset = 0
	ip.flags.mf	[0, 1]	ip.flags.mf = 1 $\implies$ frag.offset > 0
	ip.flags.rb	[0, 1]	ip.flags.rb = 0
TCP	tcp.srcport	[0, $2^{16}-1$]	tcp.srcport >0 $\implies$ udp.srcport == 0 tcp.srcport > 0 $\implies$ tcp.dstport > 0
	tcp.dstport	[0, $2^{16}-1$]	tcp.dstport > 0 $\implies$ udp.dstport == 0 tcp.dstport > 0 $\implies$ tcp.srcport > 0
	tcp.ack	[0, $2^{32}-1$]	tcp.flags.ack = 1 $\implies$ tcp.ack > 0 $\text{tcp.ack}_{current} \geq tcp.ack_{previous}$
	tcp.seq	[0, $2^{32}-1$]	$\text{tcp.seq}_{current} \geq tcp.seq_{previous}$
	tcp.window_size	[0, $2^{16}-1$]	tcp.window_size > tcp.ack tcp.window_size > tcp.seq
	tcp.flags.ack	[0, 1]	tcp.ack > 0 $\implies$ tcp.flags.ack = 1
	tcp.flags.cwr	[0, 1]	tcp.flags.fin = 1 $\implies$ tcp.flags.cwr = 0
	tcp.flags.ecn	[0, 1]	tcp.flags.ecn = 1 $\implies$ tcp.flags.syn = 1
	tcp.flags.res	[0, 1]	tcp.flags.res = 0
	tcp.flags.ns	[0, 1]	tcp.flags.ns = 0
	tcp.flags.syn	[0, 1]	tcp.flags.syn = 1 $\implies$ tcp.flags.urg = 0 tcp.flags.syn = 1 $\implies$ tcp.flags.fin = 0
	tcp.flags.fin	[0, 1]	tcp.flags.fin = 1 $\implies$ tcp.flags.ack = 1 tcp.flags.fin = 1 $\implies$ tcp.flags.syn = 0
	tcp.flags.urg	[0, 1]	tcp.flags.urg = 1 $\implies$ tcp.flags.ack = 1 tcp.flags.urg = 1 $\implies$ tcp.flags.syn = 0
	tcp.flags.push	[0, 1]	tcp.flags.push = 1 $\implies$ tcp.flags.ack = 1
UDP	udp.srcport	[0, $2^{16}-1$]	udp.srcport > 0 $\implies$ udp.dstport > 0 udp.srcport > 0 $\implies$ tcp.srcport == 0
	udp.dstport	[0, $2^{16}-1$]	udp.dstport > 0 $\implies$ udp.srcport > 0 udp.dstport > 0 $\implies$ tdp.dstport == 0
	udp.length	[0, 65507]	udp.len ≤ 65507 $\implies$ ip.len ≤ (65535 − 20)
ICMP	icmp.type	[0, 255]	ip.ttl > 255 $\implies$ icmp.type=11 frame.len >=1500 $\implies$ icmp.type = 13

Table 6.1: Constraints and relations between the different features resulting from the protocol specifications and packet analysis.

6.0.2. Packet Attribute Relations

In the packet header there are several features that can be extracted and used for classification. In this section, a specification is provided for each component and the different properties and settings that are enforced by the current implementation of the data transmission protocols.

Ethernet and IP
During the adversarial modifications, the changes that are conducted will be reflected only in

the Ethernet payload. Other Ethernet header information is not affected. In the features that are extracted from the packet header, this is only visible through the frame length which represents the length of the total information included in the Ethernet frame, such as the Ethernet header, IP header, and the packet payload. Because the Ethernet frame has a smaller maximum size than the IP frame, the IP frame is fragmented to fit the Ethernet frame and reassembled upon delivery. An invalid example where the IP length is greater than the frame length can be seen in Figure 6.1. A valid Ethernet frame must also have a minimum of 46 bytes. The IP header length has a constraint to be a minimum of 20 bytes. This has then effect on the total data that can be captured in the Ethernet frame.
In 6.2 a packet is shown where the adversarial attack reduced the value of the IP header length to below 20. As discussed before this is not possible and Wireshark recognises this as a malformed packet. This packet would not be accepted by any host and therefore cannot be used for evasion.

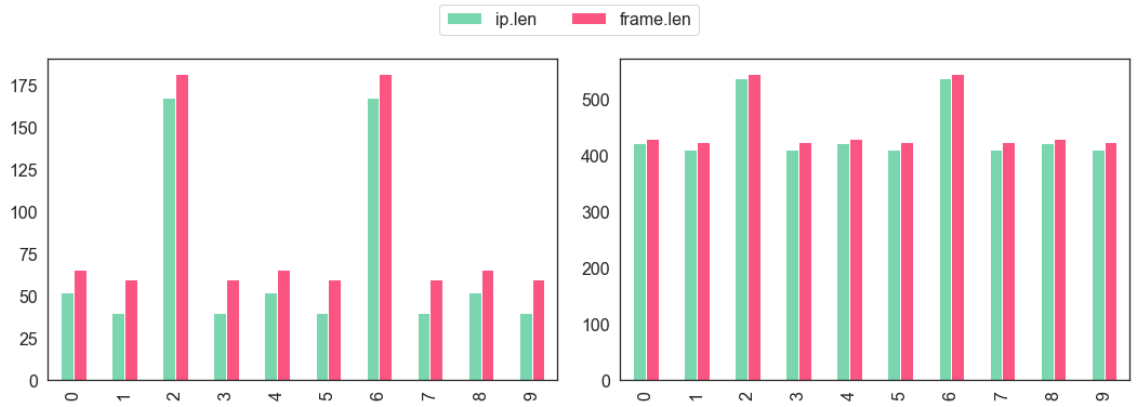


Figure 6.1: IP and Frame values with FGSM 0.25

No.	Time	Source	Destination	Protoc	Length	Info
1	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	354	Bogus IP length (8, less than header length 20)
2	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	348	Bogus IP length (14, less than header length 20)
3	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	470	Bogus IP length (8, less than header length 20)
4	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	348	Bogus IP length (14, less than header length 20)
5	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	354	Bogus IP length (8, less than header length 20)
6	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	348	Bogus IP length (14, less than header length 20)
7	2021-05-25 17:04:06.845844000	Cisco_14:eb:31	Dell_1d:1f:6c	IPv4	470	Bogus IP length (8, less than header length 20)

```
> Frame 4: 348 bytes on wire (2784 bits), 348 bytes captured (2784 bits)
> Ethernet II, Src: Cisco_14:eb:31 (00:c1:b1:14:eb:31), Dst: Dell_1d:1f:6c (b8:ac:6f:1d:1f:6c)
v Internet Protocol Version 4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  v Total Length: 14 bytes (bogus, less than header length 20)
      > [Expert Info (Error/Protocol): Bogus IP length]
```

Figure 6.2: Example of a malformed packet with IP length less than the required minimum for the header.

IP Attributes

The time to live value of a connection is expected not to change much, and this value has no specific relation to other header values of the packet that are considered for the adversarial modifications. The only direct relation is with the delta time of the received frame as this indicates what time has passed before the next packet has been received. If the delta time is larger than the time-to-live, this does not necessarily mean that the packet will be malformed, but in practice this means that the packets are dropped by the host. One such example can be seen in Figure 6.3. In the packet header, there are several features that can be extracted and used

for classification. For all features that are part of the IP protocol a rule is defined. Additionally, some features depend on the values of other features and cannot be directly altered [39].

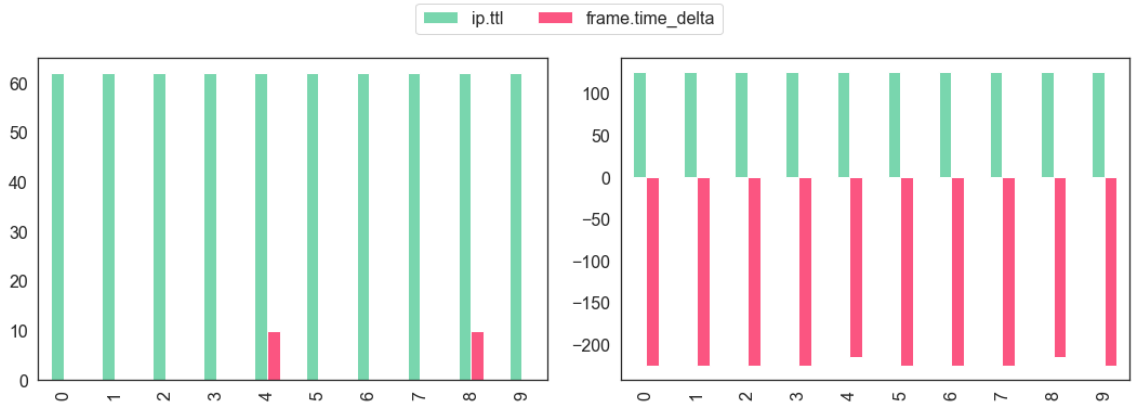


Figure 6.3: Time-based values for FGSM with $\epsilon = 0.25$

The IP flags are used to identify whether a packet is allowed to be fragmented and to represent the location of the fragmented packet relative to the other fragments. The fragmentation indicators are the DF and MF flag as well as the offset and are directly correlated with each other. In 6.4 it is shown how a FGSM attack generates values for the two flags which cannot be set in a real dataframe. Additionally in 6.5 the fragmentation flag is set to 1 while the length of the packet has only been increasing making the fragment offset therefore zero. There are two examples of violated constraints related to the IP header.

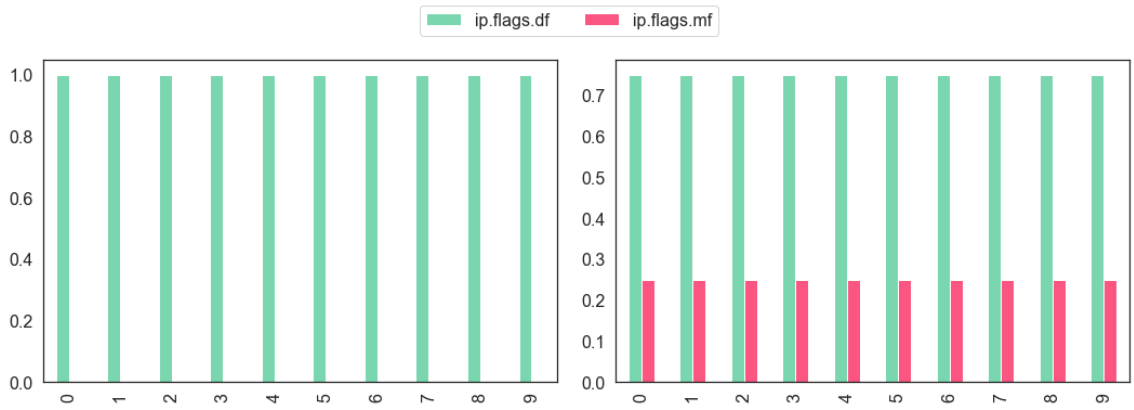


Figure 6.4: MF and DF flag values with FGSM 0.25

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=0000)
2	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=822a)
3	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=822b)
4	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=822c)
5	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=0000)
6	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=d9bc)
7	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=d9bd)
8	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=d9be)
9	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=0000)
10	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=e4bb)
11	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=e4bc)
12	0.000000	205.174.165.73	192.168.10.9	IPv4	1500	Fragmented IP protocol (proto=TCP 6, off=0, ID=e4bd)

```
> Frame 21: 1500 bytes on wire (12000 bits), 1500 bytes captured (12000 bits)
> Ethernet II, Src: Cisco_14:eb:31 (00:c1:b1:14:eb:31), Dst: Dell_1d:1f:6c (b8:ac:6f:1d:1f:6c)
v Internet Protocol Version 4, Src: 205.174.165.73, Dst: 192.168.10.9
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x1d04 (7428)
  v Flags: 0x20, More fragments
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
    Fragment Offset: 0
    Time to Live: 255
    Protocol: TCP (6)
    Header Checksum: 0x4122 [validation disabled]
```

Figure 6.5: MF Flag set with offset 0 Wireshark

IP, TCP and UDP

Other protocols are also encapsulated in the IP frame. The most common ones are the TCP and UDP protocols. There is a relationship between the IP length and TCP/UDP length. The TCP/UDP length can be only as much as the IP length. So the maximum here will never exceed 65,507 bytes. There is also a boundary for the minimal size of an IP frame, which to be valid, needs to have a header of 20 bytes [37, 38, 40]. These are all bounded by the Ethernet frame size, which will fragment large IP data. Therefore, one single packet will never exceed the size of 1500 bytes unless it is a Jumbo frame which can reach up to 9000 bytes. The assumption for the maximum upper boundary of the frame length will be 1500 bytes. An example can of a packet that violates these rules can be seen in Figure 6.6.

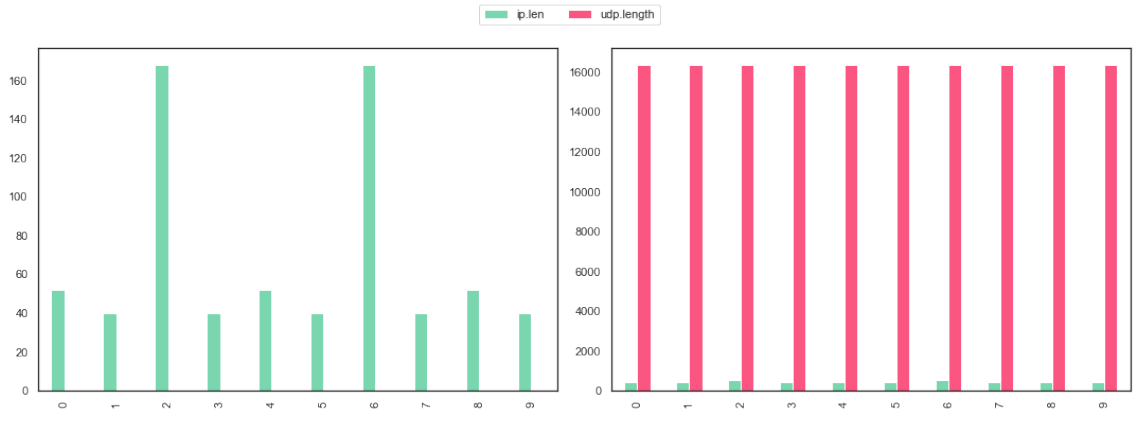


Figure 6.6: IP and UDP length values for FGSM with $\epsilon = 0.25$

Additionally, values for the TCP attributed and the UDP attributes cannot coexist in the data. This means that when a packet has the TCP protocol set, features for UDP such as source, destination, port and length should be 0. An example can be seen in Figure 6.7.

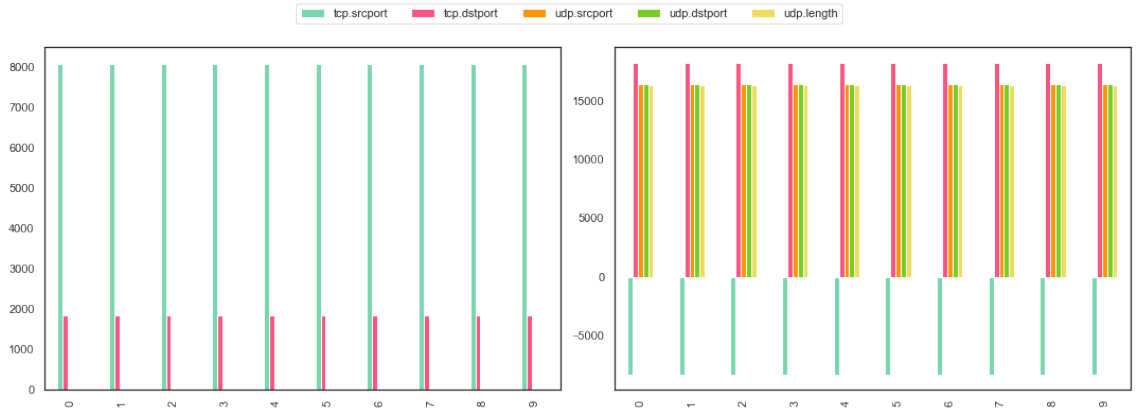


Figure 6.7: Protocol values for FGSM with $\epsilon = 0.25$

TCP Attributes

The TCP protocol has a set of flags included in its header. These flags represent different functionalities in the TCP protocol, but not all of them can be changed without affecting the validity of the packet. An example of illegitimate flag combinations can be observed in Figure 6.8

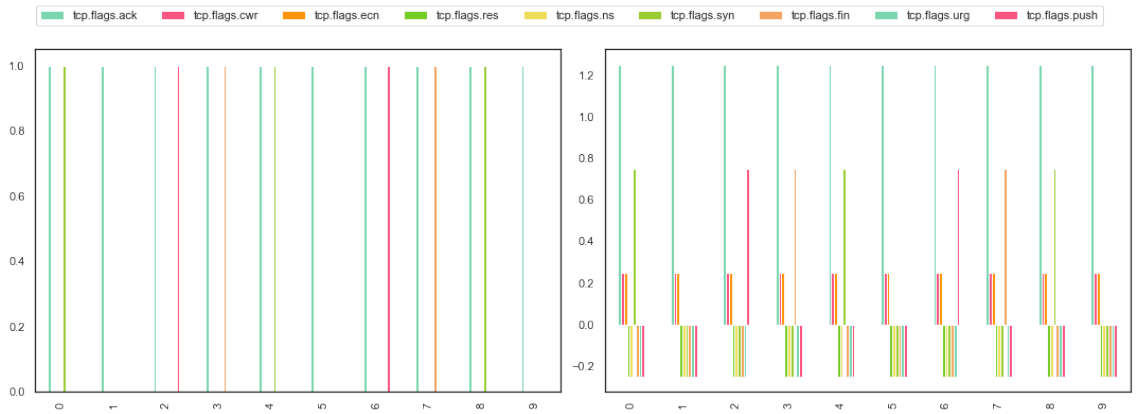


Figure 6.8: TCP Flags set for FGSM with $\epsilon = 0.25$

The TCP sequence number is incremented for each packet in the traffic flow and bounded by the TCP window. This is also the case for the TCP acknowledgement number [2]. The TCP window size is bounded by the receivers window limit which cannot be known beforehand. The TCP acknowledgement number is also constrained by the TCP acknowledgement flag, The TCP acknowledgement is considered valid only when this flag is set to 1 . An example can be seen in Figure 6.9.

6.0.3. Conclusion

Adversarial modifications designed for the unconstrained domain are generating unrealistic values for the network packets. This can be seen from the decimal numbers which cannot be set for real packets as well as the negative or very high positive values that are crossing the imposed boundaries. Additionally, the dependencies between the different components of a packet are not considered by these adversarial methods which leads to impossible scenarios

[2] https://datatracker.ietf.org/doc/html/rfc4413section-4.2.1

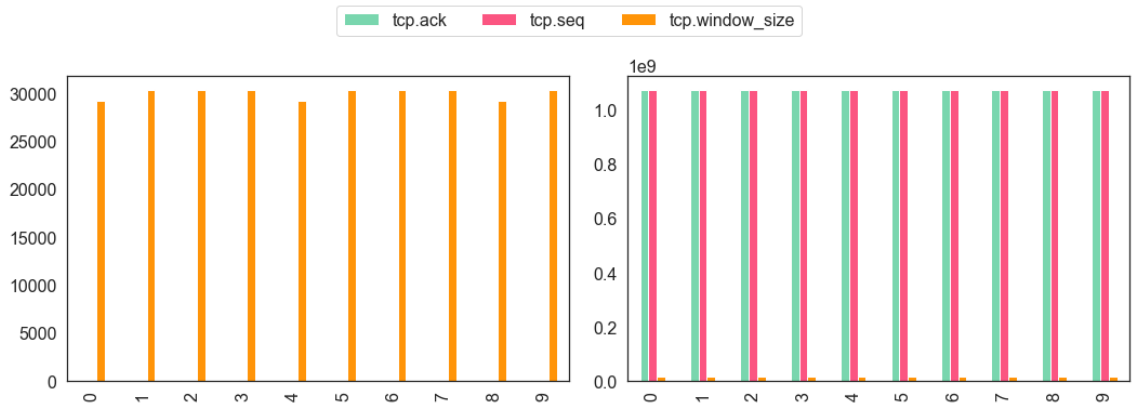


Figure 6.9: TCP connection values for FGSM $\epsilon = 0.25$

such as packets with a negative time-to-live or TCP packets where UDP options are also set. The results from this exploration confirm the assumption that the adversarial modifications that are initially designed for unconstrained domains cannot realistically be used by adversaries for performing practical attacks.

From the official network traffic definitions and experiments conducted in this research, a set of constraints has been defined for the packet header features. These constraints are boundaries, dependencies between the different components of a packet and also a set that cannot be altered while performing practical traffic modifications. The obtained set contains both equality and inequality constraints. Due to this, it is a complex optimization problem where both binary and continuous variables are included. A method to translate the constrained problem to an unconstrained one is by using penalty functions.

7

Constrained Gradient Sign Method

In this chapter, the optimization objective and optimization function will be stated. The objective of the attacker is to alter the malicious packets in such a way that they preserve their functionality, but get classified as benign samples by the target classifier. In order to generate evasive packets in a controlled manner a technique needs to be used which allows the use of network traffic constraints in the process of generating adversarial examples. Additionally, the modifications need to be performed in feature space to assess the success of the attack and the validness of the samples. Adversarial techniques using the gradient loss allow for such modifications with embedded constraints and boundaries.

7.1. Objective

The problem that is considered in this work is a binary classification problem. A classification model can only choose between two classes 0 and 1. The benign samples belong to class 0 and the malicious samples belong to class 1. The goal is to make the classification model predict class 0 for the malicious packets that originally belong to class 1. This can be achieved by ensuring that the combination of selected features of a network packet evaluate to class 0. To achieve this, the values of the packet features are modified according to the sign of the gradient.

The number of incorrectly classified points or true positives needs to be maximized. For each malicious packet vector v_i that belongs to class $c_i = 1$, the model should output $c_i = 0$. Therefore, the objective function can be defined as maximizing the sum of the points in the data set which represent the total number of network packets sent by the adversary. The function that can be used for this purpose is a binary loss function. For the attack, the gradients of the loss function with respect to the input points are calculated. The function that is used for the optimization of adversarial values is given in equation 7.1. This function is calculated subject to the set of constraints that were defined in Chapter 6, to make sure that the generated adversarial network packets remain as close to the original packets as possible and retain their original functionality. Consider a network packet x with label Y. The proposed method solves the maximization problem by computing:

$$\max_x f(x) = -\frac{1}{n}\sum_{i=1}^{n} Y_i log(Y_i') + (1-Y) log(1-Y_i')$$

$$\text{subject to} \begin{cases} 46 \leq x\_i \geq 1500, & \text{x\_i} \in \text{frame.len} \\ 20 \leq x\_i \geq 65507, & \text{x\_i} \in \text{ip.len} \\ x\_i + x\_j \leq 1500, & \text{x\_i} \in \text{frame.len, x\_j} \in \text{ip.len} \\ 20 \leq x\_i \geq 65535, & \text{x\_i} \in \text{tcp.len} \\ 8 \leq x\_i \geq 6553520, & \text{x\_i} \in \text{udp.len} \\ x\_i + x\_j \leq 65507, & \text{x\_i} \in \text{ip.len, x\_j} \in \text{tcp.len} \\ x\_i + x\_j \leq 65507, & \text{x\_i} \in \text{ip.len, x\_j} \in \text{udp.len} \\ x\_i + x\_j + x\_k \leq 1500, & \text{x\_i} \in \text{frame.len, x\_j} \in \text{ip.len, x\_k} \in \text{tcp.len} \\ x\_i + x\_j + x\_k \leq 1500, & \text{x\_i} \in \text{frame.len, x\_j} \in \text{ip.len, x\_k} \in \text{udp.len} \\ x\_i = 0 \vee x\_i = 1, & \text{x\_i} \in \text{ip.flags} \\ x\_i = 0 \vee x\_i = 1, & \text{x\_i} \in \text{tcp.flags} \\ 0 \leq x\_i \geq 65535, & \text{x\_i} \in \text{ports} \\ 0 \leq x\_i \geq 255, & \text{x\_i} \in \text{ip.ttl} \\ x\_i = 1 \wedge x\_j = 1, & \text{x\_i} \in \text{tcp.flags.ack, x\_j} \in \text{tcp.flags(urg, rst, fin)} \\ x\_i \leq x\_j, & \text{x\_i} \in \text{tcp.ack, x\_j} \in \text{tcp.window\_size} \\ x\_i \leq x\_j, & \text{x\_i} \in \text{tcp.seq, x\_j} \in \text{tcp.window\_size} \end{cases} \tag{7.1}$$

The FGSM attack works by taking the sign of the gradient of the loss function multiplied by some perturbation magnitude epsilon and adds this value to the original input point x to create the adversarial point x'. The loss function is thus computed on the original and predicted labels with respect to the value of x. A widely applied technique to integrate constraints with functions is to translate the constraints into penalties. In the proposed solution, a penalty function is used to encode the different feature constraints and interfeature relationships. The penalty is then added to the final loss function that is used for the attack. The new gradients are computed based on the values of the previous data points and these are then used to lead the perturbations in the direction that reduces the number of violated constraints.

7.1.1. Adversarial Loss Function

The objective for the adversarial attack is the maximization of the binary cross entropy loss that is used by the classification model to learn the correct classes for the points. The second penalty term, is the L_2 norm, which computes the distance between the original and modified adversarial points. The goal of adding this term is to minimize the distance such that the adversarial points do not deviate too much from the original points. This term is subtracted from the loss function to affect the perturbation direction. The last term of the adversarial loss consists of the quadratic penalty function. This penalty is chosen because it is a differentiable function and is only applied if a constraint is violated. The quadratic penalty is used for both the equality and inequality constraints, and it works by adding a penalty greater than zero if the constraint is violated or adding no penalty when there is no violation. In terms of the adversarial loss function, both the L_2 norm and the quadratic penalty will be substracted from the cross entropy loss since the goal of these terms is to penalize the maximization objetive.

$$L(x, y, x', y') = (-\frac{1}{n}\sum_{i=1}^{n} y_i log(y'_i) + (1-y) log(1-y'_i))$$
$$- \lambda_1 * (\sqrt{\sum_{i=1}^{n}(|x_i - x'_i|^2)})$$
$$- \lambda_2 * (\sum_{i=1}^{n}(max[0, c(x'_i)])^2))$$

The point x_i represents an input point that belongs to class y_i and $c(x'_i)$ is one constraint of the constraint set C that applies to adversarial point x'_i. The features can be either exclusive or inclusive, additionally, the features are continuous and bounded by an inequality relationship. The class for input point x as predicted by the classification model is represented as y'. The variable n is the number of input points and the λ_1 and λ_2 variables are the weights that are added to the regularization terms to specify the magnitude of their effect on the adversarial loss function.

7.1.2. Constrained Gradient Sign Method

The proposed method is a modified version of the existing IFGSM attack and is called the Constrained Iterative Fast Gradient Sign Method (CIFGSM). The CIFGSM method is a gradient-based method and uses the loss function of a gradient based classifier for the calculation of the perturbation direction. Additionally, clipping is applied to ensure that the values remain within the predefined boundaries. The main difference with the existing method is the custom adversarial loss function, where a regularization term and the penalties for the set of constraints are added. Furthermore, weights are used for the penalty terms to control the magnitude of the impact these terms can have on the loss. To calculate the cross entropy loss, the Logistic Regression model is used. The proposed method is defined as:

Algorithm 1 *Constrained Iterative Fast Gradient Sign Method*

Input: model, x, y, epsilon, epochs, clip_values
alpha := $\frac{epsilon}{epochs}$
x' = x
y' := model(x)
loss := L(x, y, x', y')
for i ∈ epochs **do**
    gradient:= grad(loss, x')
    x' := x' + alpha * sign(gradient)
    x' := clip(x', clip_values)
end for
return x'

The input of this method is the surrogate model based on which the points are optimized, the data point x, the label y, the perturbation magnitude epsilon, the number of epochs and the boundaries in the constrained space. The model makes a prediction on the original input and calculates the initial loss. Subsequently, for each epoch, the gradient of the loss is calculated and the new adversarial point x' is crafted by adding the product of epsilon and the sign of the

gradient with respect to x. Then a check is performed to verify that x' is within the boundaries, if this is not the case x' is clipped and mapped to the closest value that falls within the boundaries. The adversarial point x' is the input for the next iteration. The method returns the adversarial point x' of the last iteration.

7.1.3. Results

In this section, the results will be presented and discussed that are obtained through the experiments on the processed network traffic data. The selection of a suitable perturbation value is discussed as well as the effect of the penalty function on the attack's success rate and how this is represented through the features. Furthermore, the resulting adversarial values obtained through the application of the IFGSM and CIFGSM methods are compared and discussed with respect to the original data. Based on these results, the best settings for the practical attack are selected.

7.1.3.1 Selection of loss weights

Since the new loss function consists of three different terms, it is important to select suitable weights for each of the terms in order to get a balanced loss function that will both perturb the points enough while also comply with the imposed distance and traffic constraints. The first term that is subtracted from the original loss is the L_2 distance norm. This term ensures that the data points remain as similar as possible to the original data by changing the direction of the perturbation. Setting the weight for this term to 1 heavily decreases the success rate of the attack. Therefore a smaller value is more suitable. The second weight is the weight of the penalty term which ensures that the perturbations do not violate the constraints. Setting this term to 1 does not affect the success rate much but ensures that most of the violated data points are changed such that they remain within the constrained space. In Figure 7.1 a matrix is shown

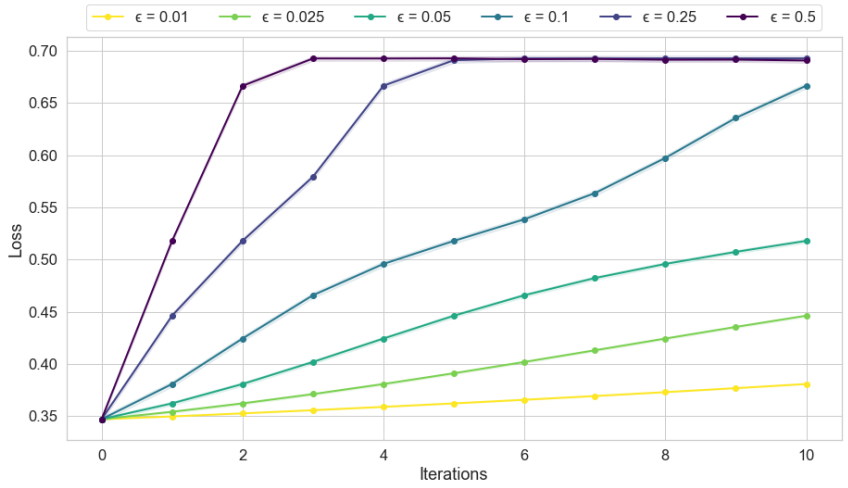
Figure 7.1: The success rate of the classification model with different combinations of the perturbation magnitude ϵ and the penalty magnitude λ

with the success rates obtained by experimenting with different values of ϵ and the weights for

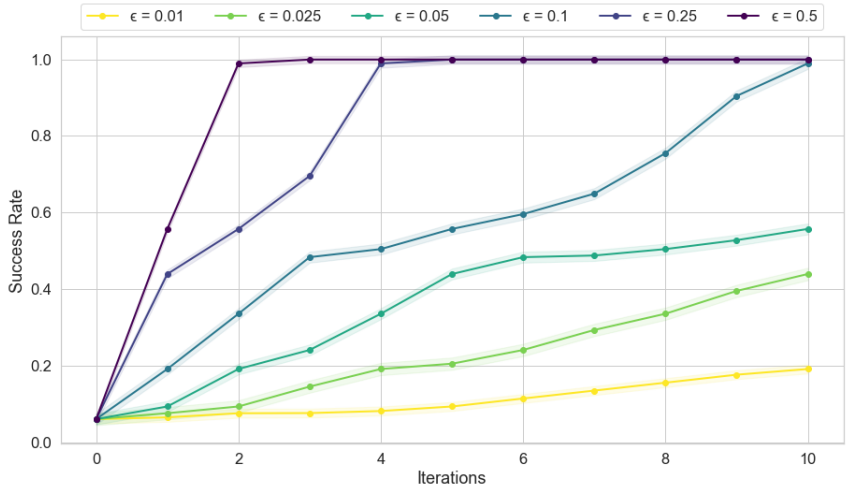
the penalty terms. In this example, the weight for both the L_2 norm and the constraint penalty is chosen to have the same value. However, it is evident that the L_2 norm is overpowering both the cross entropy and the constraint penalty.

7.1.3.2 Selection of perturbation magnitude ϵ

For the purpose of selecting a perturbation size that will create successful evasion packets, the CIFGSM method is tested with different values for epsilon and constant values for the loss function weights. Based on the results obtained from the experiment, as can be seen in Figure 7.2b, for the further generation of the practical attacks $\epsilon = 0.5$ will be used because the CIFGSM method with this value was the most successful at evading the target classification model. Larger values than 0.5 were also tested, however since the maximum success rate was reached with $\epsilon = 0.5$ at the second iteration, these are not visualized in this report. It should be noted that larger perturbation values would also increase the loss and might be necessary if the penalty and L_2 norm outweigh the cross entropy loss, so this will heavily depend on how the loss weights are combined.



(a) Loss for the constrained iterative fast gradient sign method with different values of ϵ



(b) Success Rate for the constrained iterative fast gradient sign method with different values of ϵ

Figure 7.2: Experiments with constrained iterative fast gradient sign method for different perturbation magnitudes.

7.1.3.3 Penalty Evaluation

In Figure 7.3, examples are shows of different feature pairs for which a relation was defined. The first 10 iterations of the IFGSM and CIFGSM methods are plotted and each step represents the value of the point for the respective iteration. On the first subfigure, the frame and IP lengths are shown. As we can see these become very similar at the sixth iteration of the method and when the penalty is applied, it redirects the value of the IP length downwards. On the second plot the fragment flags are shown which are mutually exclusive. However, the IFGSM directs these flags to the same point, which means that the flags will have the same value. The CIFGSM solves this issue by penalizing them and leading them to opposite directions. This is also the case for the TCP SYN and TCP FIN flags shown in the last subfigure. The effect of the penalty function can also be observed in the third subfigure where the TCP source port becomes a negative value whereas the packet represents a TCP connection and therefore both the TCP source and destination port should be positive values. After applying the penalty, the direction of the perturbation is changed and the resulting values for the TCP source port are legitimate. However, in some cases, when a feature has multiple constraints and multiple penalties are applied for that feature, this can lead to conflicting results and make the feature values valid for one constraint but invalid for the other. For this reason, it is still possible to have a certain percentage of invalid packets or packets with warnings. Also because network traffic is multidimensional it is challenging to combine all these constraints into one penalty term. Additionally, groups of packets represent connections and on this level there are also rules that the traffic needs to adhere to like the sequence and acknowledgement numbers derived from the previous packet or some pairs of flags that need to be both set at the beginning or ending of a connection, but not during the connection, like the SYN and FIN flags.

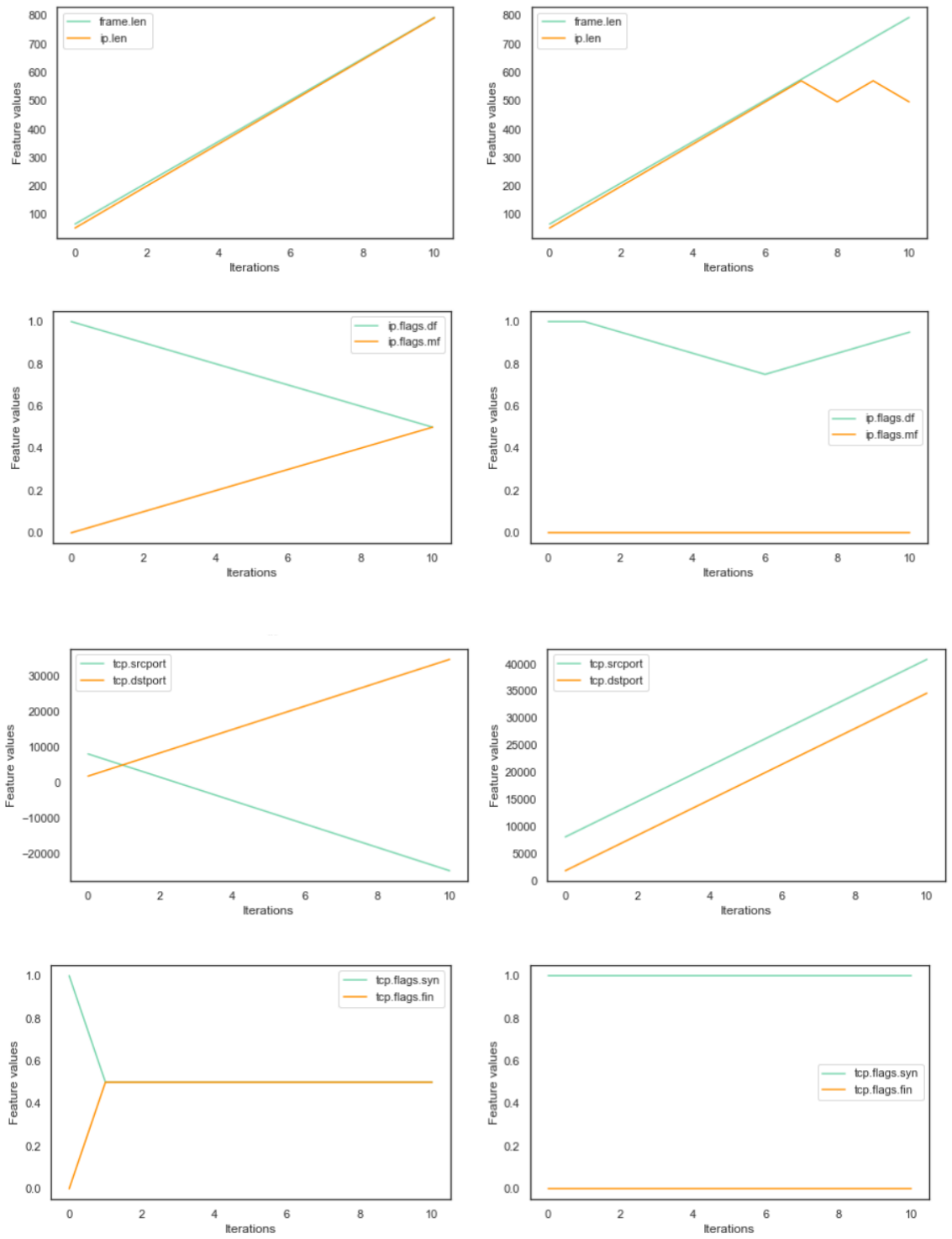


Figure 7.3: In this figure a visualisation is shown of how the different network traffic features change direction whenever a constraint is violated. In the first column the values of one point are shown through 10 iterations for a perturbation magnitude $\epsilon = 0.5$. The second column shows the values with the penalty function added to the loss.

7.2. Practical Traffic-space Attack

In this section, the results of several experiments performed with the proposed method are presented and discussed. The performance of this method is evaluated on five classification models and a comparison with other adversarial crafting methods is given. The validity of the adversarial network packets is evaluated as well as the differences between the original and adversarial examples.

7.2.1. Adversarial Packet Modifications

The practical modifications are performed using the library Scapy, which provides extensive functionality for manipulating network packets. The datasets used for evaluation, consist of packet capture files which represent previously captured sets of packets. Each packet capture is given as an input of the method, as described in Algorithm 2, and each network packet is modified according to the previously obtained adversarial values for that packet.

Algorithm 2 *Practical Adversarial Modifications*

```
1: Input: adversarial_values, pcap
2: adv_pcap = []
3: for p ∈ pcap do
4:    p_val:= adversarial_values[p]
5:    p' := modify_packet(p, pval)
6:    adv_pcap.insert(p')
7: end for
8: return adv_pcaps
```

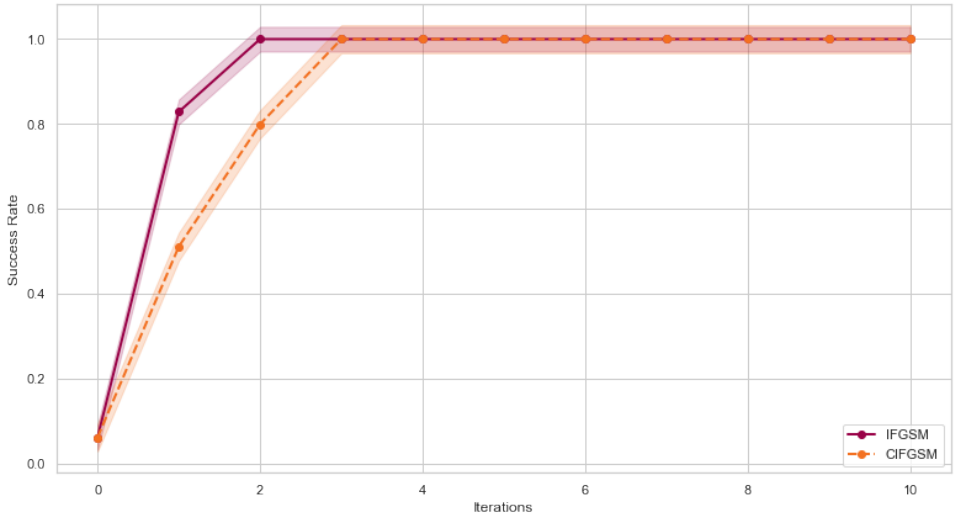
The resulting packets are written to pcap files and the features are extracted and processed for classification and further evaluation. All modifications are performed with the functionality that was available in Scapy. It is possible to modify the packets per layer, where each layer consists of a set of elements that accept any value within the allowed byte range. To apply the modifications, several functions are defined that target single features, a subset of these functions is described in Chapter 5. This was required because of the complexity of the network packets. For changing the IP length as outputted by the adversarial attack, it is necessary to perform a chain of modifications starting from rounding the adversarial value, changing the length of selected packets based on the TCP handshake and recalculating the checksum. Additionally, the TCP acknowledgement and sequence values could not be changed randomly as these are set based on the size of the data and the previous values. Other values were more straightforward to change such as the frame length, time-to-live, source port, and the different flags. Furthermore, there were some parts of the packet that could not be altered at all during this experiment because of limitations in the reconstruction of the modified packets.

7.2.2. Results

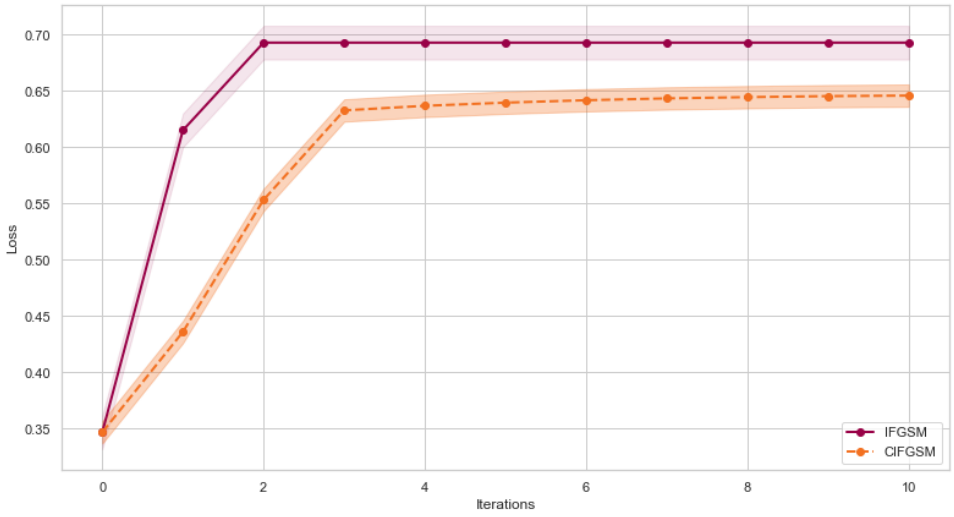
In this section, the results obtained from the practical experiment are presented and analyzed. For crafting the adversarial examples with the CIFGSM method, based on the previous experiments the following parameter values are chosen: the perturbation magnitude $\epsilon = .5$, the weight $\lambda_1 = 10^{-7}$, and the weight $\lambda_2 = .01$ where λ_1 is the weight applied to the L_2 norm and λ_2 is the weight applied to the penalty P.

7.2.2.1 Performance

In this section the performance of the proposed method is evaluated on the surrogate model and four other classification models. For this purpose the success rate and the accuracy of the classification model are chosen. These are considered representative metrics for measuring the extent to which the target classification models can be evaded by the adversarial network packets, which are crafted with the proposed method.



(a) The success rate of the IFGSM and CIFGSM methods during 10 iterations for $\epsilon = 0.5$



(b) The loss of the IFGSM and CIFGSM methods during 10 iterations for $\epsilon = 0.5$

Figure 7.4: Success rate and loss

In Figure 7.4 the success rate and loss of both the IFGSM and CIFGSM methods are shown. From the success rate plot it can be seen that the success of the attack increases very rapidly. The difference between the original IFGSM method and the proposed method where the network traffic constraints are incorporated is shown through the slightly lower success rate at the first two iterations. The loss is also increasing at each iteration which is expected with the increasing success rate. However, for the constrained method, the loss does not increase more than approximately 0.6 which is enough to achieve a high success rate. The reason for the loss

not increasing at the same rate as the loss of the IFGSM are the extra terms that are subtracted from the adversarial loss function. These are preventing a further increase as there is a certain value extracted for the distance between the points and a certain value for the subset of violated constraints. The violations are not enough to decrease the loss, however, they do ensure that the direction of the gradient remains unchanged to prevent a further increase of the loss.

According to the feature space experiments with the chosen set of values, the adversarial attack, when performed in this setting reaches a high evasion rate. Therefore, the obtained adversarial values are used for the practical attack. The results of both the feature space and traffic space attacks are presented in Table 7.1 for the CICIDS2017 dataset as well as the classification of traffic flows that are obtained from the packet captures.
For the original data, it can be seen that the performance of the classification models is good because all models have an accuracy greater than 90%, and the success rate of the attack is very low, even lower then 1%, meaning that the models do classify the majority of benign and malicious packets correctly. For the second method, IFGSM, it can be seen that it is very successful at evading the classifiers since the success rate of this attack reaches above 95%, for some models, it even reaches 100%. The accuracy of the different models for the adversarial traffic generated with the IFGSM, decreases to below 12%. For the CIFGSM the same applies, as this method is very successful at evading both the surrogate and the target classification models with an accuracy of maximum 12% and a success rate of at least 93%.

	Org		**IFGSM**		**CIFGSM**		**TA**		**Org_f**		**TA_f**	
Model	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.
LR	.93	.06	.10	1.	.10	.99	.34	**.73**	.98	.03	.62	**.96**
NN	.97	.0	.09	1.	.09	1.	.40	**.65**	.98	.03	.60	**1.**
SVC	.98	.0	.10	1.	.10	1.	.38	**.68**	.99	.02	.99	**.0**
RF	.93	.07	.12	1.	.12	1.	.16	**.94**	1.	.0	.36	**.60**
DT	.97	.03	.11	.94	.11	.93	.22	**.87**	1.	.0	.19	**.80**

Table 7.1: In this table the results of the accuracy and success rate for the different attacks with $\epsilon = 0.5$ on the packet based classification models are shown. It can be seen that the traffic-space attack (TA) is less successful than the feature-space attacks for all classification models. However it still is successful at reducing the detection rate significantly.

The success rate of the traffic space attack (TA) varies for the different classification models. From Table 7.1 it can be seen that the surrogate model LR (73%), the NN (65%) and the SVC (68%) models are more robust to the adversarial attack than the RF (94%) and DT (87%) models. However, even though the traffic space attack is less successful than the feature space attack, it succeeds in reducing the detection rate of the classifiers to at least 65%.

From the generated adversarial packet captures, flow features were extracted and the classifiers were trained on these flow features. The reason to also evaluate the proposed method for flow classification is because many solutions are trained to recognize malicious flows rather than specific packets, and thus it is of importance to verify that by generating adversarial packets, the flows become evasive as well. From the results, it can be concluded that the classification models correctly classify the original flow data, however, these are less successful at correctly classifying the adversarial flow data. The success rate of the attack is at least 60% for all the models except the SVC model. For this model, the attack is not at all successful as it correctly predicts all class labels. This can be explained by observing the feature importance of the SVC

model. This model gives a higher importance to the RST flag, which cannot be modified, because if it is set to 1, it will break the connection. In contrast to the other models that have a lower importance for this feature, and consider the backward header length and the number of forwarded packets as more important. This is an example of a robust classifier against this attack and shows that feature selection techniques could be employed as part of a defense strategy against such adversarial modifications.
Also the decrease in success for the traffic space attack can be a result of the limitations in the practical packet modifications. Firstly, most of the features are rounded before they are applied in the practical attack, since decimal values are not accepted for the flags or other settings of the network packet. The second practical limitation is modifying the sequence and acknowledgement numbers. This is a very complex modification due to the TCP connection requirements about the relation between the previous and next sequence and acknowledgement numbers, and the segment size.

To verify if the proposed method can also be generalized for other network traffic data, it is tested on the CTU-13 dataset. The results obtained for this dataset can be observed in Table 7.2. From these results, it can be seen that the method is also effective on other network traffic datasets with an accuracy of at most 42% and a success rate greater than 80%. For this particular dataset, the traffic space attack is even more successful than on the initial dataset. This is also confirmed with the flow classification results, where all models are evaded with a success rate of at least 90%.

	Org		**IFGSM**		**CIFGSM**		**TA**		**Org_f**		**TA_f**	
Model	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.	Acc.	Succ.
LR	.89	.07	.24	1.	.24	1.	.24	**.99**	.97	.05	.35	**.97**
NN	.93	.05	.27	.99	.27	1.	.27	**.99**	.98	.03	.35	**.96**
SVC	.93	.04	.26	1.	.26	1.	.39	**.81**	.98	.02	.40	**.90**
RF	.90	.14	.30	1.	.30	1.	.42	**.81**	1.	.0	.36	**.95**
DT	.87	.18	.30	1.	.30	1.	.30	**.99**	.99	.0	.36	**.95**

Table 7.2: In this table the results of the accuracy and success rate for the different attacks with $\epsilon = 0.5$ on the packet based classification models are shown. It can be seen that the traffic-space attack (TA) is less successful than the feature-space attacks for all classification models but still reducing the detection rate significantly.

7.2.2.2 Transferability

The results in Table 7.1 show that both the IFGSM and CIFGSM attacks performed with the LR classification model are transferable to other classification models. The traffic space attack that is performed based on the values obtained from the CIFGSM attack is also transferable. This means that the proposed method could also be used in a black/grey box scenario with a pre-trained classifier on publicly available data without having information about the target system. The results for the transferability of the attack on the different classification models for the CICIDS2017 and CTU-13 datasets can be observed in Figure 7.5 and Figure 7.6 respectively.

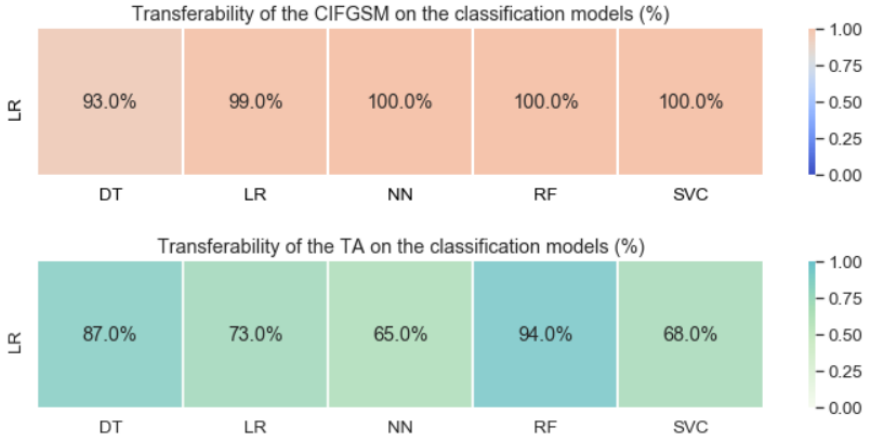


Figure 7.5: Transferability of the attack applied to the different target classification models for the CICIDS2017 dataset

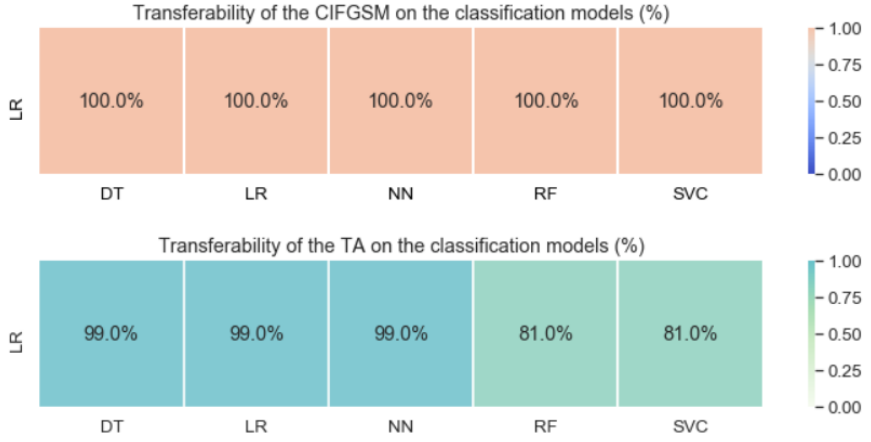


Figure 7.6: Transferability of the attack applied to the different target classification models for the CTU-13 dataset

7.2.3. Traffic Analysis

The validity of the data that is generated according to the adversarial values, is evaluated for success but needs to also be evaluated for its practical applicability. It is not straightforward to do so since the traffic that is used during this research is artificially generated for research purposes, consequently, it might not be a realistic enough representation of malicious and benign traffic. Therefore, the options to evaluate the real effectiveness of the included attacks are limited.

Validity Verification

To verify the validity of the generated adversarial network traffic, several different metrics and tools are used. The expert info of Wireshark is used to check the adversarial packets. An example is shown in Appendix B.1. This tool examines the packet capture, finds anomalies and different unusual network behaviors, and provides a summary of the discovered issues grouped by severity level. The errors have the highest severity and are marked with a red color. An error message means that the examined packet is malformed. The warnings, marked with a yellow color, have a lower severity level than the errors, and list issues related to connection problems.

Te TCP replay tool is also used for the validity verification of the network traffic, an example of how this tool is used is shown in Appendix B.2. The metrics that are used to quantify the validity verification are the DPR, VCR, and IPR metric.

For determining whether the adversarial packets will be received by the target host, the TCPReplay tool is used and the DPR is computed as the amount of dropped packets on the network. When many packets are dropped, it can be concluded that the evasion traffic is less effective since the original attack functionality will not work properly if packets are not received, and the data flows cannot be completed. However, the TCPReplay tool may not be completely reliable in terms of counting the number of dropped packets, since often incorrect packets are still replayed and seem to be received, while if sent to a real network, these might not be accepted. As no better option was available, the TCPReplay tool is chosen for this purpose. Another metric that is used to check whether the generated packets conform to the network traffic constraints, the VCR metric, which represents the fraction of packets with at least one violated constraint. Lastly, the IPR is calculated as the sum of errors and warnings of the pcap files, obtained through the expert information of Wireshark. Here each packet is analysed with built-in techniques that check every aspect of the packet header and its contents. When a packet is marked with an error, the probability is high that the functionality of this packet is broken, and thus the evasion attack will also be less successful in a practical scenario.

	DPR	**VCR**	**IPR**	**Errors**	**Warnings**
Original Traffic	0%	0.04%	6%	0%	6%
Adversarial Traffic IFGSM	1.9%	1.1%	26.6%	0.6%	26%
Adversarial Traffic CIFGSM	**0.002 %**	**0.08 %**	**24 %**	**0.04%**	**24%**

Table 7.3: Results of the original and adversarial network traffic on the CIC2017 dataset.

From the results in Table 7.3 it can be noticed that although higher than the DPR of the original network (0%) traffic, the DPR of the CIFGSM (0.002 %) is very low for the adversarial packets. This means that the majority of packets have been processed by the TCPReplay tool. The DPR of the adversarial packets generated with the IFGSM is 1.9% which is very high in comparison to the DPR of the CIFGSM.The VCR of the CIFGSM is also very low, however, there is still a small fraction of packets (0.08%) that could affect the performance of the attack. In comparison with the VCR of the IFGSM (1.1%), the VCR of the CIFGSM is 13 times lower. The errors, as obtained from Wireshark for the CIFGSM, are slightly higher (0.04%). The majority of these errors are malformed packets where the contents of the packet cannot be dissected, therefore, the packet will probably be dropped by the receiving host. The IFGSM has a percentage of 0.6% errors, which is approximately 15 times higher than the errors resulting from the CIFGSM attack.

The warnings generated for the adversarial network traffic with the CIFGSM (24%), are higher than those for the original network traffic (6%), and they are very similar to the warnings generated for the IFGSM (26%). This does not necessarily mean that the packets are malformed but that the features are set to unexpected values, and therefore, warnings are generated. The majority of warnings encountered are related to out-of-order packets, retransmissions and previous uncaptured segments which are a resulting from the changed ports and flag and length values. These warnings are all related to TCP flows where an unusual connection behavior is noticed. Whether these warnings will have an impact on the functionality of the traffic, and how large this impact will be, is very dependent on the implementation of the target network.

In Table 7.4 the validity analysis for the CTU-13 data can be observed. The results are similar to the results obtained for the CICIDS2017 dataset, except in this case, there are slightly more

	DPR	VCR	IPR	Errors	Warnings
Original Traffic	1.02%	0.0003%	12%	0.002%	12%
Adversarial Traffic IFGSM	4.8%	6.5%	28.14%	0.14%	28%
Adversarial Traffic CIFGSM	**4%**	**0.05%**	**17%**	**0.003%**	**17%**

Table 7.4: Results of the original and adversarial network traffic on the CTU-13 dataset.

errors (0.14%) generated for the IFGSM and (0.003%) for the CIFGSM in comparison with the original data (0.002%). When comaring the IFGSM and CIFGSM, the IFGSM generates approximately 50 times more errors than the CIFGSM for this dataset. The same can be concluded for the warnings, which are 12% for the original data, 28% for adversarial data generated with the IFGSM and 17% for the CIFGSM. This could be due to the type of attacks in the CTU-13 dataset, which are botnets that use the TCP protocol. The VCR of the IFGSM (6.5%) is also higher than the VCR of the CIFGSM which is 0.05%. Furthermore, the same can be noticed for the DRP and IPR of the IFGSM and CIFGSM attacks.
From the results in Table 7.3 and Table 7.4 it can be concluded that the performance of the CIFSM attack is better than the performance of the IFGSM attack in terms of generating valid adversarial network packets.

Similarity with original data

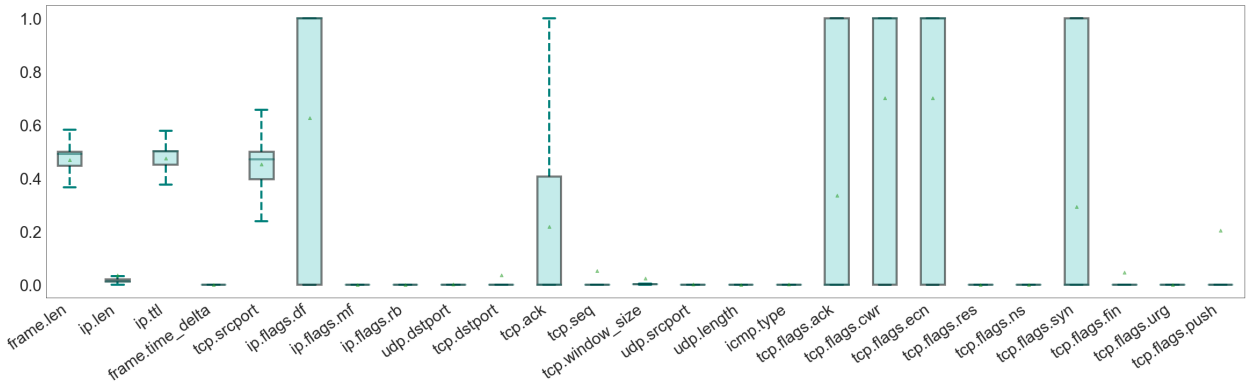


Figure 7.7: Boxplot of the mean euclidean distances between the original network traffic samples and the adversarial network traffic samples generated with the traffic space attack (TA).

From Figure 7.8 and Figure 7.11 it is evident that the original and adversarial traffic are similar, however, in Figure 7.10 it can be seen that CIFGSM has generated higher values for most of the features. This is expected since there is a subset of values that are changed by CIFGSM but are immutable in the current practical implementation, and therefore, are not transferred to the real traffic. This can be seen very clearly in Figure 7.7 where the distance between the features of the original data and the unmodified features of the adversarial data is zero. The reason that some flags have a very small or zero distance from the original data points is the rounding that is applied before the practical modifications. Furthermore, it can be noticed that, the data points of the other features are very similar to the adversarial points generated with the CIFGSM attack. The CIFGSM attack succeeds in finding adversarial points that are similar to the IFGSM attack but also belong to the valid region. This can be seen in the heatmaps through the distribution of different feature values. The distance between the features that were modified is also visibe in the box plot shown in Figure 7.7. The figure shows that the packet length features have a noticeably higher mean distance. Furthermore, the distance between the

original and adversarial flag values is very mixed, covering the whole range from being set to the same bit value to being set to the opposite bit value. Generally, the values of the adversarial packets are higher but the majority of the relations between the features are not violated. The feature-space heatmaps from the IFGSM and CIFGSM are very similar. This is also the case for the traffic space heatmaps of the original and practical attack and by comparing these two sets, it can be clearly observed which of the packet attributes are easier to manipulate and which of them are immutable, or have other more complex requirements that cannot be included as constraints. The same conclusion can be drawn for the CTU-13 dataset. The figures for this datset are included in Appendix C.

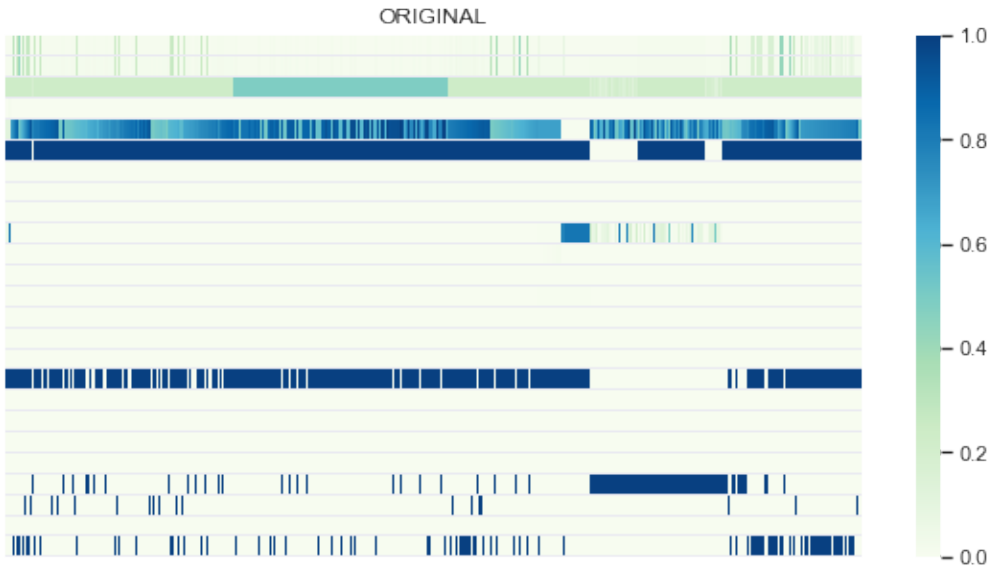


Figure 7.8: Heatmap of features values for the original dataset

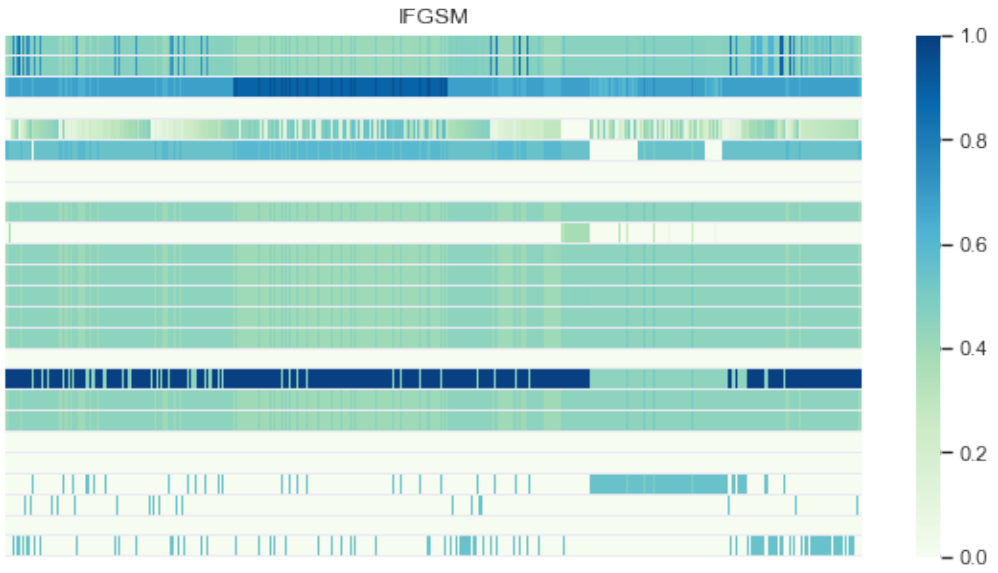


Figure 7.9: Heatmap of features values for the adversarial examples generated with the IFGSM on the CICIDS2017 dataset.

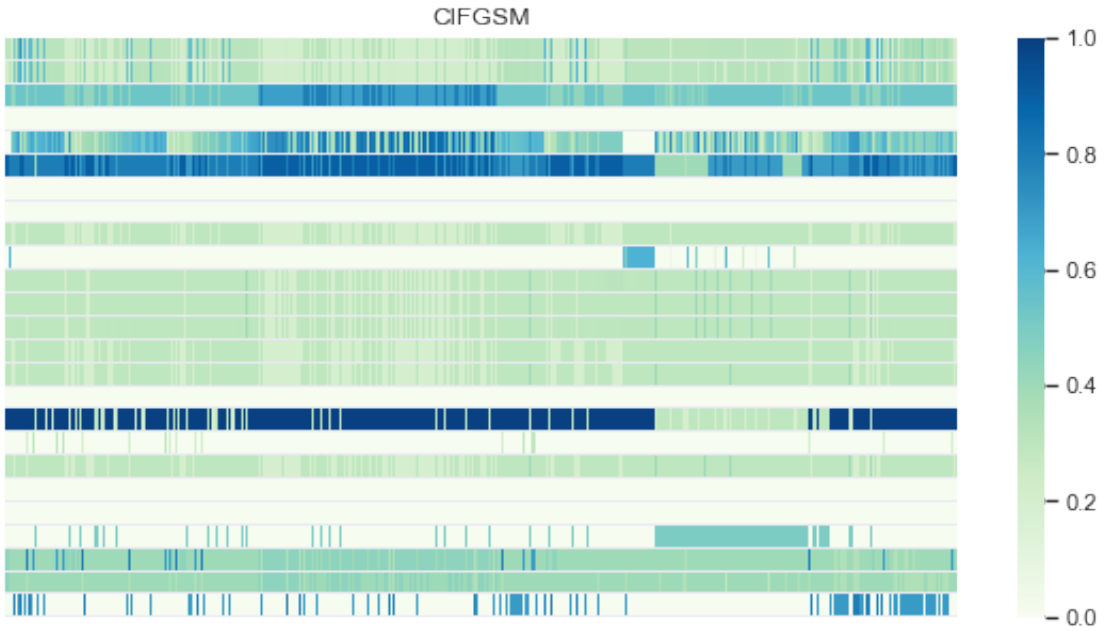


Figure 7.10: Heatmap of features values for the adversarial examples generated with the CIFGSM on the CICIDS2017 dataset.

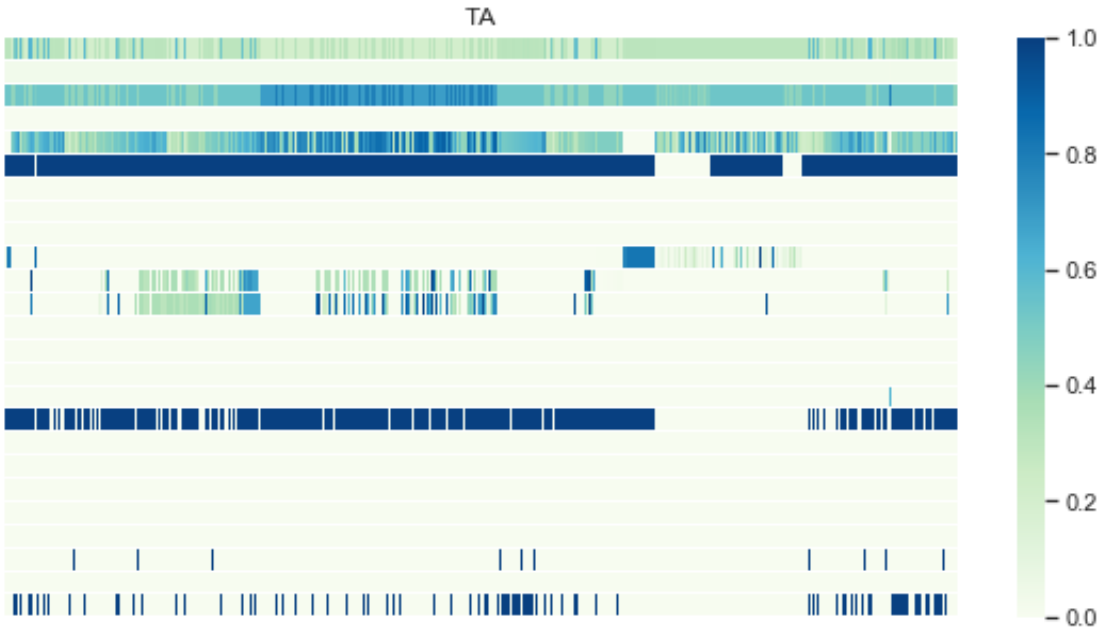


Figure 7.11: Heatmap of features values for the adversarial examples generated with the traffic-space attack (TA) on the CICIDS2017 dataset.

Time

The CIFGSM attack is slower than the FGSM and IFGSM attacks because the penalty calculation adds to the computational complexity. This is caused by the larger number of operations that need to be computed on the data points at each iteration. The runtime of the different attack models can be observed in Figure 7.12. The FGSM attack is performed in one step, while the final result of the IFGSM attack is obtained after 10 iterations. This explains the slightly higher runtime between these two methods. The time increase of the CIFGSM attack is evidently higher than the other two existing methods and is also increasing faster with the size of the data. All these methods are applied to 1054561 number of malicious network packets. This is an important factor to consider when generating adversarial values for a large number of network packets, that will be at the cost of the optimized attack.

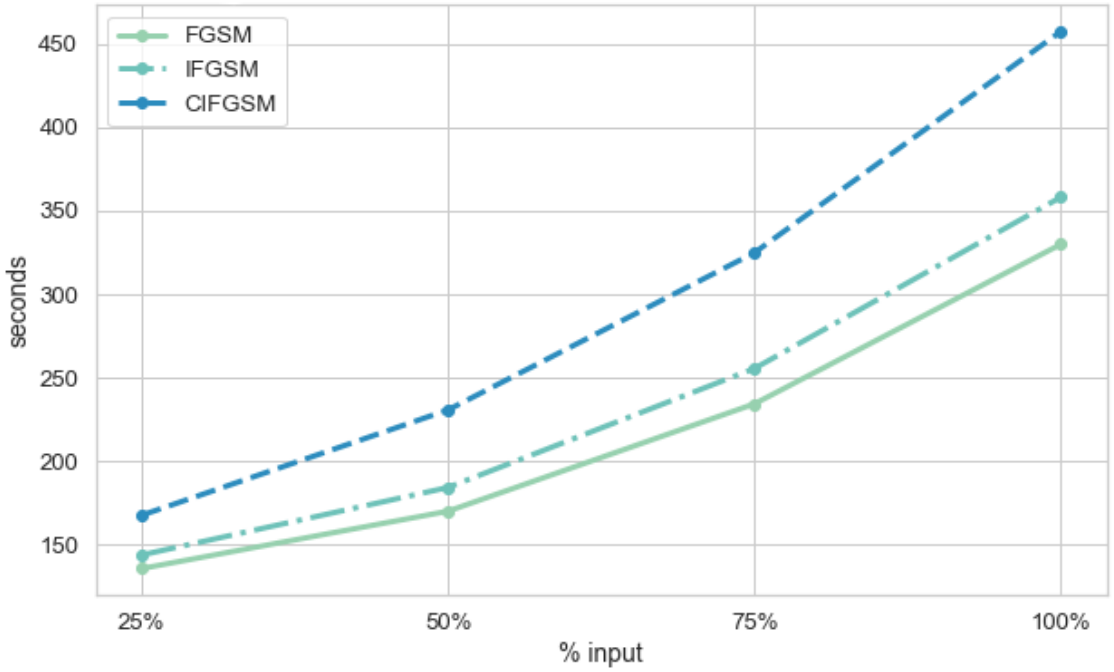


Figure 7.12: Runtime of the FGSM, IFGSM and CIFGSM attacks on the CICIDS2017 dataset.

7.3. Conclusion

By creating a custom adversarial loss function that includes a penalty function for the optimization of the adversarial modifications, and a regularization term for maintaining the similarity between the original and adversarial points, the validity of the network traffic can be preserved. However, more research and evaluation needs to be done to confirm that the attack functionality of the original packets remains unchanged. The penalty function is chosen to be the quadratic penalty, which is suitable for both the constraints that were previously defined. Additionally, the boundary constraints imposed by the protocol implementation can be captured by clipping the values on their upper and lower boundaries.

This method still does not ensure that the crafted packets will be fully legitimate, the results are very dependent on the selection of the perturbation magnitude, the number of iterations, and the penalty weigths. Because the constraints are applied on a per-packet level and there

is no guarantee that the optimal values will be found with the applied penalty, packet loss can still occur.

What can be concluded from the obtained results is that the penalty succeeds at minimizing the packet loss while also finding adversarial values for the features. These adversarial values are able to successfully fool the target classification model. The effect of the constrained method for finding adversarial examples, as applied in feature space, is very similar to the one without constraints. Which implies that this method still succeeds in maximizing the success rate of the adversarial network packets. The effect of the constrained method, when translated to a practical attack, also succeeds in reducing the detection rate, however, the success rate is evidently lower compared to the feature space attack. In consideration of the previously discussed practical limitations, this is expected.

The proposed method is also evaluated on flow data where the evasion rate is shown to be high as well. From these experiments, it can be concluded that the packet level modifications can also be effective on flow-based NIDS. To validate the proposed method, the feature space and traffic space practical attack are performed on an additional data set consisting of botnet traffic where similar results are obtained.

Compared to the initial experiments of Chapter 5 where practical attacks were applied to the packets with randomly chosen values, the optimized adversarial values are clearly more successful at evading the classifier. In the initial experiments, the success rates of the attacks are very variable per classifier. The lowest success rate for the packet classification is obtained from the delay attack, whereas the highest success rates are obtained from the fragmentation attack and the combined attack. When comparing to the most successful attacks, these achieve high evasion rates for the SVC and LR models, while they are almost not successful for the DT and RF models. In contrast to the proposed method, that has a similar rate for all the evaluated models. From the practical modifications, some limitations have been discovered which prevent to use all adversarial values obtained from the adversarial optimization method. There are several reasons for this, these are explained in more detail in the previous sections.

In summary, the main limitations are the level at which the constraints are applied. Since some of the packet-level features are directly dependent on the flow of data between the hosts, these could not be included as constraints to the current objective function, however, this could be a possibility for future work. Additionally, some of the features are simply immutable and cannot be changed in a real packet, or modifying these would require a complex reconstruction of the whole flow of packets, and this is not straightforward to do with the open-source packet manipulation tools. Thus, it is considered infeasible in the scope of this work. This is also no efficient and feasible solution, both in terms of time and required resources, for an adversary who's goal is to compromise a target system.

8

Discussion

In this chapter, the main limitations encountered during this research will be discussed as well as how these limitations have affected the obtained results. Ideas for future work are also proposed.

8.1. Limitations

One of the main limitations that were encountered during the experiments was the inclusion of all constraints that are applicable to network traffic that exist on different levels. In scope were only the constraints that are applicable to one single packet, however, there exist more complicated relationships between several network traffic features that can only be considered at flow level. One such case is the three-way handshake where the SYN, ACK, and FIN flags are combined with the ACK and SEQ. These constraints cannot be encapsulated on a per packet level because this relationship is visible only on a connection level between several packets. Here, it was difficult to combine relations that are contradicting for some features. For instance, both the SYN and FIN flags can be set in combination with the ACK flag, however, the SYN flag should be set at the start of the connection while the FIN flag at the end, and the other packets of the connection should not have the SYN and FIN flags set.

Another limitation is that a lot of parameters have to be specified. For the attack, these are the perturbation value and the number of iterations, for the loss function, these are different weights. Finding the optimal combination of weights, that will both increase the success rate and decrease the number of invalid packets, needs to be done by trial and error. In case of a real scenario, this way of selecting the method's parameters is not efficient for an adversary. Additionally, the proposed method should be evaluated on more datasets. The difficulty here is that there exist very few publicly available intrusion detection datasets, and those that are available are mainly created for research purposes. Thus, they do not provide a complete and accurate representation of real network traffic in a large organisation, for instance. The success rate of the proposed method will also be very dependent on the type of classification model and the configuration of the target NIDS.

Additionally, the proposed attack can be less effective on a target model, if it is set to block all incoming traffic that generates a warning or has certain white listing rules for specific source ports. Furthermore, the proposed attack was generated using the Logistic Regression classifier, other more sophisticated classifiers can be used as surrogate models, this could result in higher transferability and evasion rates. Finally, another limitation of this works is the actual working of the attacks. It has not been shown that the attacks have preserved their functionality in terms of conducting a port scan or performing actions of a botnet program. This is a very

important aspect of such modifications and should be considered in future research on this topic. This work provides a starting point for further research and discussion on the topic of crafting realistic adversarial network packets that can be used in a practical setting.

8.2. Future Work

Further exploration is necessary to validate the conclusions that are drawn from this work. This work demonstrates that it is possible to manipulate network traffic with adversarial machine learning techniques, for the purpose of evasion. Future studies should aim to extend and apply the proposed method to more intrusion detection datasets and different target networks, to evaluate the extent to which this method can be generalized. Additionally, in this work, only the packet-level constraints are taken into account. In order to have complete possibilities in the modification of packet values, several flow-level constraints and dependencies need also be accounted for and added to the existing constraint set. Furthermore, the surrogate model with which the proposed method is evaluated, is a Logistic Regression model. To further evaluate the possibilities of finding optimal adversarial values, Neural Networks and other more sophisticated models should be explored, since these might achieve higher success and transferability rates. Another recommendation for future work is extending the different types of malicious traffic that is modified, in this work, a limited set of attacks is used. More types of attacks should be evaluated to determine the level of difficulty of crafting adversarial examples for specific attack. Lastly, the robustness of the models should be evaluated with existing defense techniques, additional targeted defense mechanisms could be explored for this purpose.

9

Conclusion

RQ1: Which adversarial traffic evasion techniques exist for network traffic and what is the effect of applying these on a target system with limited knowledge ?

- **How can network traffic be manipulated to create adversarial network packets in a practical setting?**

  Several possible network traffic modifications can be realistically applied to the network packets without breaking the validity of a network packet. These are padding, splitting, modification of header values such as ttl, ip flags and some tcp flags. For this experiment the modifications are tested with randomly selected values. The attacks can be practically applied using traffic modification tools such as Scapy. There are several features of the packet header that are immutable and cannot be changed. The modifications can be effective against ML-based NIDS if the right modification values are chosen. At this moment, there exists no other technique except querying the target NIDS and measuring the effectiveness of the attack by trial and error. This approach is not feasible for an adversary, hence it is not a realistic method in practice.

- **To what extent do the crafted adversarial packets succeed at evading the target classification model?**

  From the experiments that are conducted with a chosen set of adversarial samples crafted for network traffic, it is evident that such modifications can have a great effect for reducing the detection rate of different types of classification models. It is important to note that the values for the attacks are chosen randomly, independent of the benign network traffic or classification model settings. The tested data contains different types of attacks (DDoS, Botnet, Port Scans) and some of the adversarial modifications are effective for the majority of malicious samples. However, in some cases it can be concluded that the attacks do not have a significant effect. There are several reasons for this. The first reason are the arbitrary values that are chosen for this experiment. The same attacks with values optimized against some classification models are likely to have greater success. The second reason why some of these attacks do not work is related to the feature set that the classifiers are trained on. It can be seen that the packet-based classifier is generally more robust than the flow based classifier. For an adversary who has no knowledge of the target classifier, the used feature set, or benign traffic on the target network, it can be concluded that it is challenging to perform successful evasion attacks. Therefore, it is interesting to explore the possibilities for finding optimal values using adversarial optimization techniques.

RQ2: How are adversarial modifications represented through the network traffic features?

- **What is the effect of applying adversarial modifications to the extracted network traffic features?**

  Adversarial modifications designed for the unconstrained domain are generating unrealistic values for the network packets. This can be seen from the decimal numbers which cannot be set for real packets as well as the negative or very high positive values that are crossing the possible boundaries. Additionally, the dependencies between the different components of a packet are not considered by these adversarial methods, which leads to impossible scenarios, such as packets with a negative time-to-live value or TCP packets where UDP options are also set. The results from this exploration confirm the assumption that the adversarial modifications initially designed for unconstrained domains cannot directly be applied and realistically used by adversaries for performing practical attacks.

- **Which constraints and feature relations need to be considered for crafting valid adversarial packets ?**

  A set of constraints is defined for the packet header features based on the network traffic definitions and experiments conducted in this work. These constraints are the boundaries and dependencies between the different components of a packet. The obtained set contains both equality and inequality constraints. Due to this, crafting adversarial network packets is a complex optimization problem in which both binary and continuous variables are included. A method to translate the constrained problem to an unconstrained one is by using penalty functions.

RQ3: To what extent can practical adversarial network packets be crafted using adversarial optimization techniques, while also considering the existing domain constraints?

- **How can network traffic domain constraints be included in the optimization function that is used for the generation of adversarial examples?**

  By creating a custom adversarial loss function, that includes a penalty function for the optimization of the adversarial modifications and a regularization term to maintain the similarity with the original input points, the original network packets can preserve their validity. The penalty function is composed of several different constraints that are suitable for both the equality and inequality terms that were previously defined. Additionally, the boundary constraints imposed by the protocol implementation can be captured by clipping the values on their upper and lower boundaries. However, this method does not ensure that the crafted packets will be fully legitimate, and when the adversarial values are applied to the actual packet captures, there is a percentage of packet loss, due to the inability to model and encapsulate the detailed content of a network packet. There remain protocol-specific attributes and properties which are difficult to include in the proposed method. Because the constraints are applied on a per-packet basis and there is no guarantee that the optimal values will be found with the applied penalty, a certain percentage of network packet loss can still occur. Conclusively, the adversarial loss function succeeds at minimizing the number of invalid network packets while also finding legitimate adversarial values for the majority of packet features. The adversarial network packets are able to successfully evade the target classification model after only a few iterations and are also transferable to other models.

- **To what extent does the proposed method succeed at evading the target classification model and are the adversarial network packets transferable to other models?**

The effect of the proposed method for crafting adversarial network packets, as applied in feature space is demonstrated to succeed at evading the target model. The effect of the constrained method, when translated to a practical scenario, does also succeed in reducing the detection rate. However, the success rate of the traffic space attack is lower compared to the success rate of the feature space attack. From the experiments on the validation dataset, it can be concluded that the traffic on which the surrogate model is trained, is not required to be collected from the target network to achieve high evasion rates. This means that the method could also be relevant for grey or black box scenarios where the attack has no knowledge of the targeted network but generates adversarial examples based on publicly available datasets. Nontheless, more experiments need to be conducted to evaluate this observation.

Main RQ: To what extent can adversarial optimization techniques be used for the generation of adversarial network traffic in a constrained domain while preserving the validity of the original traffic?

From the conclusions obtained for each of the three subquestions, the answer to the main research question is that adversarial optimization techniques can be used for the generation of realistic and highly evasive network traffic. Several limitations are discussed concerning the flow-based dependencies, but even in this setting, the obtained results are satisfying and can be used as a basis for further research and improvements on this topic. In essence, the constrained network traffic domain is not robust to adversarial examples even though it is more difficult to craft these examples in a practical setting. The proposed Constrained Iterative Fast Gradient Sign Method succeeds at finding adversarial examples that evade both the surrogate and target classification models and do not compromise the validity of the original network traffic.

A

Appendix A

A.1. Packet-based features

- frame.len
- frame.time_delta
- ip.len
- ip.ttl
- ip.flags.df
- ip.flags.mf
- ip.flags.rb
- tcp.dstport
- tcp.srcport
- tcp.ack
- tcp.seq
- tcp.window_size
- tcp.flags.ack
- tcp.flags.cwr
- tcp.flags.ecn
- tcp.flags.res
- tcp.flags.ns
- tcp.flags.syn
- tcp.flags.urg
- tcp.flags.push
- tcp.flags.fin
- udp.dstport
- udp.srcport
- udp.length
- icmp.type

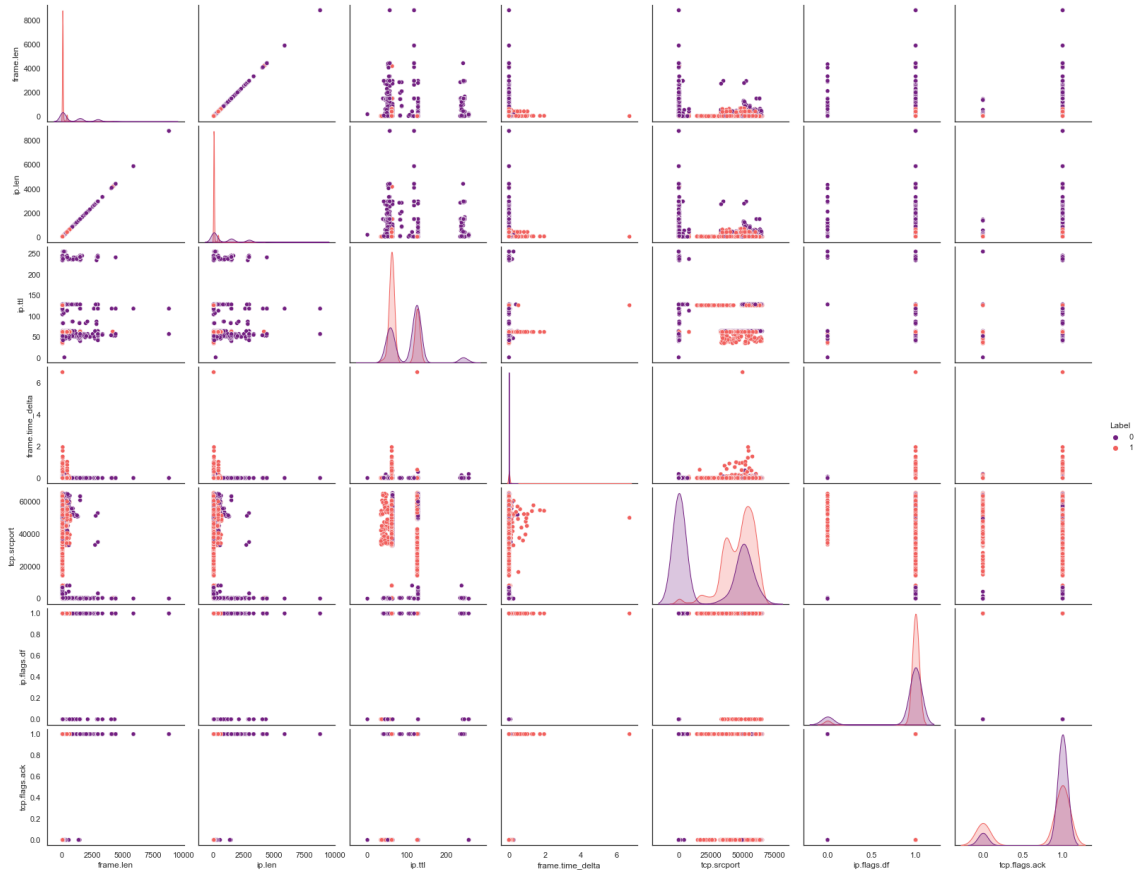


Figure A.1: Pairplot of the CICIDS2017 network traffic dataset.

A.2. Flow-based features

- Src Port
- Dst Port
- Protocol
- Flow Duration
- Tot Fwd Pkt
- Tot Bwd Pkt
- Tot Length of Fwd Pkt
- Tot Length of Bwd Pkt
- Fwd Pkt Length Max
- Fwd Pkt Length Min
- Fwd Pkt Length Mean
- Fwd Pkt Length Std
- Bwd Pkt Length Max
- Bwd Pkt Length Min
- Bwd Pkt Length Mean
- Bwd Pkt Length Std
- Flow Bytes/s
- Flow Packets/s
- Flow IAT Mean
- Flow IAT Std
- Flow IAT Max
- Flow IAT Min
- Fwd IAT Total
- Fwd IAT Mean
- Fwd IAT Std
- Fwd IAT Max
- Fwd IAT Min
- Bwd IAT Total
- Bwd IAT Mean
- Bwd IAT Std
- Bwd IAT Max
- Bwd IAT Min
- Fwd PSH Flags
- Bwd PSH Flags
- Fwd URG Flags
- Bwd URG Flags
- Fwd Header Length
- Bwd Header Length
- Fwd Packets/s
- Bwd Packets/s
- Pkt Length Min
- Pkt Length Max
- Pkt Length Mean
- Pkt Length Std
- Pkt Length Variance
- FIN Flag Count
- SYN Flag Count
- RST Flag Count
- PSH Flag Count
- ACK Flag Count
- URG Flag Count
- CWR Flag Count
- ECE Flag Count
- Down/Up Ratio
- Avg Packet Size
- Fwd Seg Size Avg
- Bwd Seg Size Avg
- Fwd Bytes/Bulk Avg
- Fwd Packet/Bulk Avg
- Fwd Bulk Rate Avg
- Bwd Bytes/Bulk Avg
- Bwd Packet/Bulk Avg
- Bwd Bulk Rate Avg
- Subflow Fwd Packets
- Subflow Fwd Bytes
- Subflow Bwd Packets
- Subflow Bwd Bytes
- FWD Init Win Bytes
- Bwd Init Win Bytes
- Fwd Act Data Pkts
- Fwd Seg Size Min
- Active Mean
- Active Std
- Active Max
- Active Min

B

Appendix B

B.1. Wireshark

Wireshark · Expert Information · brute_force.pcap

Severity	Summary	Group	Protocol	Count
Error	Malformed Packet (Exception occurred)	Malformed	TCP	2
Error	Malformed Packet (Exception occurred)	Malformed	WINS-Replic...	217
Error	Malformed Packet (Exception occurred)	Malformed	NDPS	434
Error	Malformed Packet (Exception occurred)	Malformed	RSL	3
Error	Malformed Packet (Exception occurred)	Malformed	MySQL	217
Error	Malformed Packet (Exception occurred)	Malformed	DNS	434
Warning	BER Error: Empty choice was found	Malformed	KRB5	217
Warning	Invalid frame type	Protocol	Gryphon	108
Warning	BER Error: Sequence expected but class:UNIVER...	Malformed	SNMP	434
Warning	Bogus request length (0)	Protocol	X11	11
Warning	Incorrect Magic Cookie	Protocol	PPTP	1
Warning	D-SACK Sequence	Sequence	TCP	11
Warning	No response seen to ICMP request	Sequence	ICMP	38
Warning	Connection reset (RST)	Sequence	TCP	25
Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	1154
Note	Didn't find padding of zeros, and an undecoded...	Protocol	Ethertype	2
Note	Unknown type (0x0000) packets not supported yet	Undecoded	NCP	1
Note	A new tcp session is started with the same ports ...	Sequence	TCP	62
Note	This frame is a (suspected) retransmission	Sequence	TCP	23465
Chat	Connection finish (FIN)	Sequence	TCP	2124
Chat	Connection establish request (SYN): server port 80	Sequence	TCP	3140

Figure B.1: Example of the Wireshark expert information for the brute force attack. Errors are malformed packets which will probably be dropped by the receiving

B.2. TCPReplay

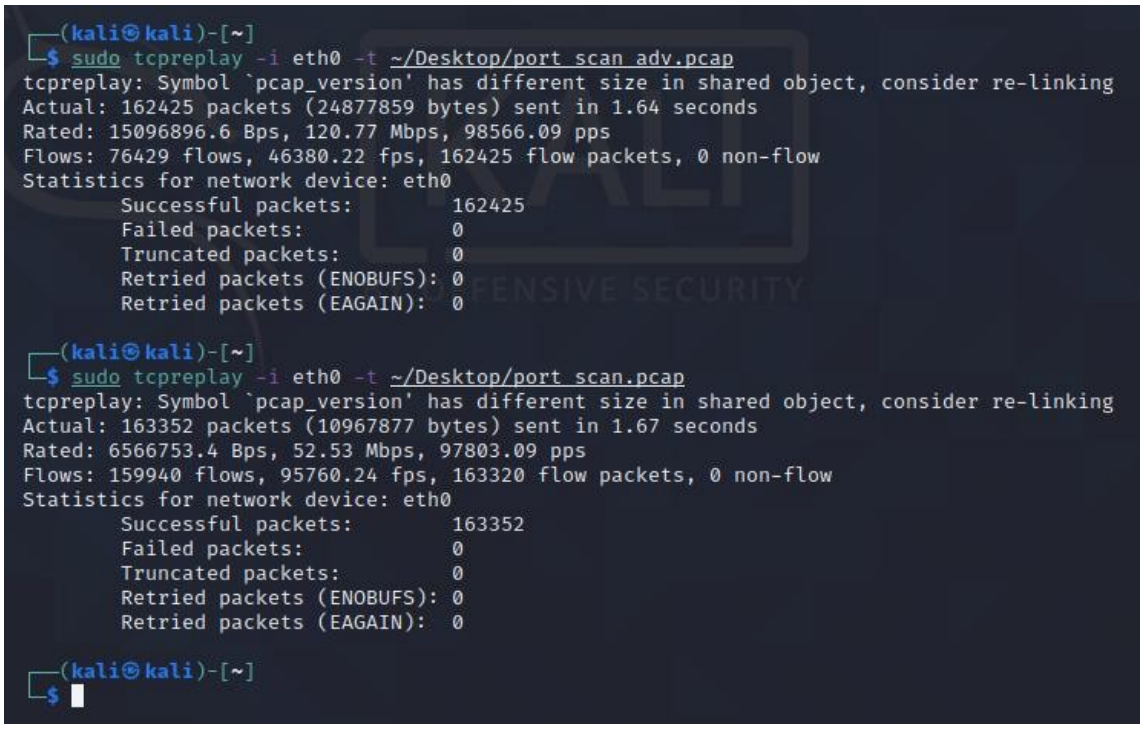


Figure B.2: Example of the tcp replay information for the port scan attack.

C

Appendix C

C.1. Heatmaps of CTU-13 dataset

Figure C.1: Heatmaps of the features values for the original data points of the CTU-13 dataset.

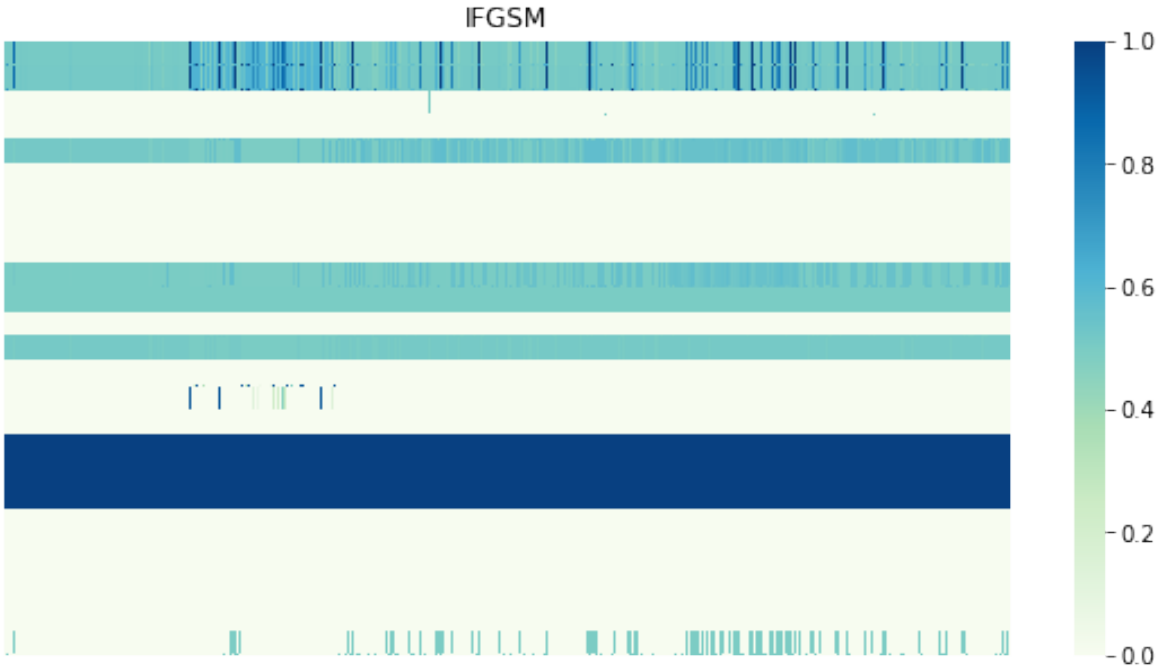


Figure C.2: Heatmaps of the features values for the data points generated with the IFGSM method for the CTU-13 dataset.

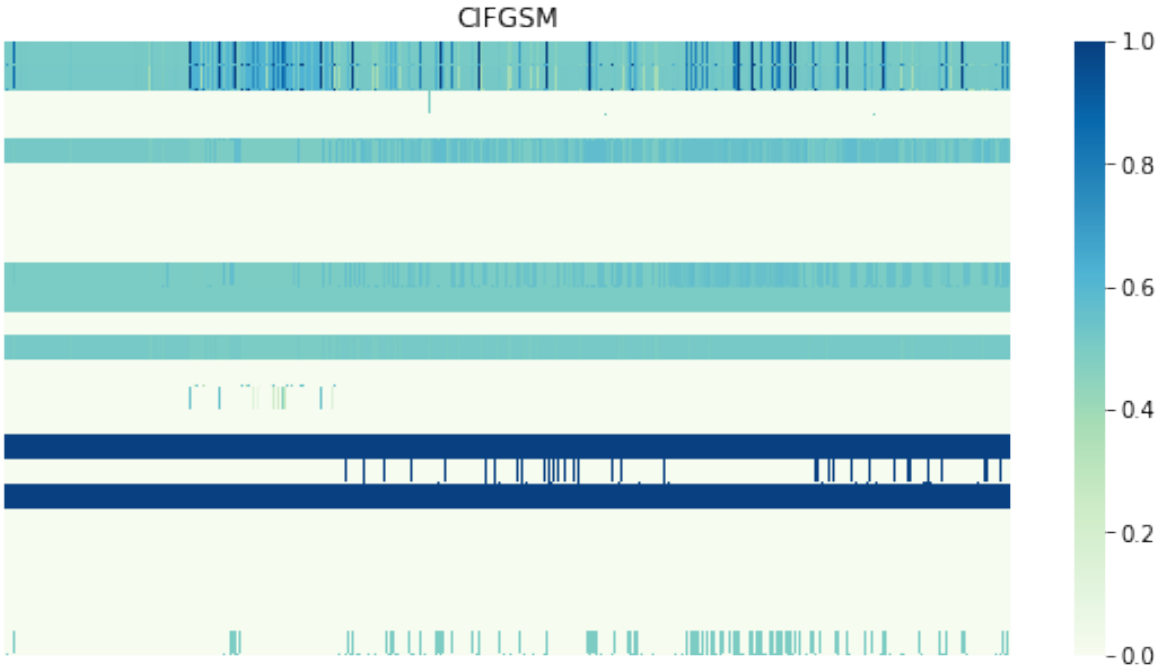


Figure C.3: Heatmap of the features values for the data points generated with the CIFGSM method for the CTU-13 dataset.

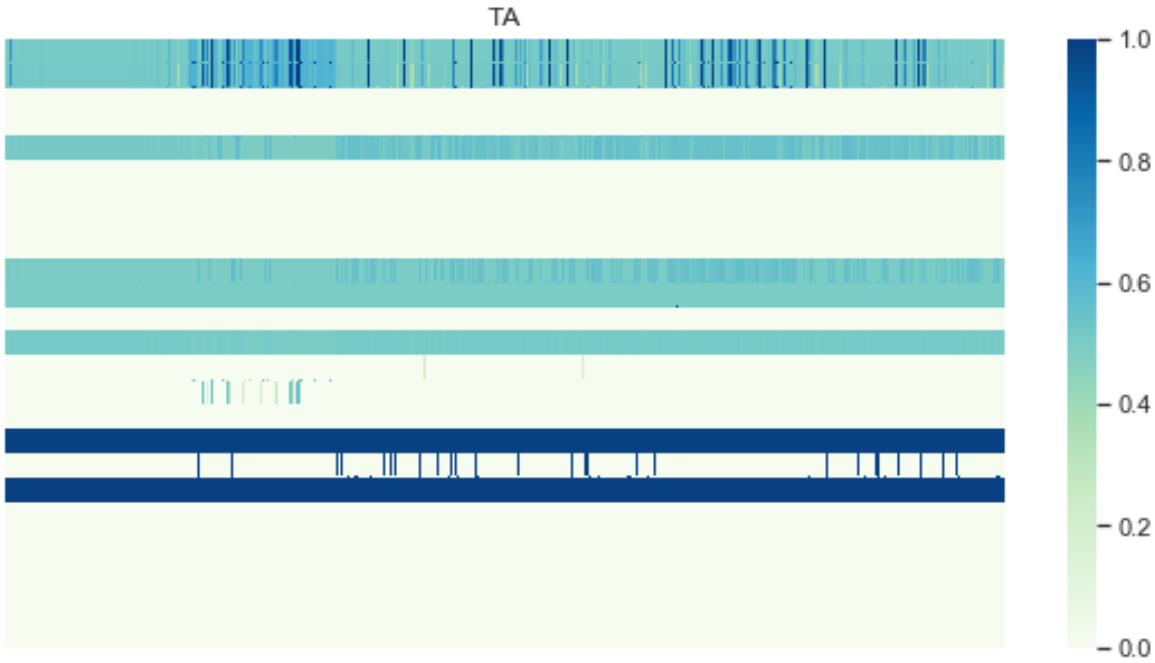


Figure C.4: Heatmap of the features values for the data points generated with the Traffic-space attack (TA) for the CTU-13 dataset.

Bibliography

[1] E. Alhajjar, P. Maxwell, and N. D. Bastian. Adversarial machine learning in network intrusion detection systems. *arXiv preprint arXiv:2004.11898*, 2020.

[2] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh. Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 000277–000282, 2017. doi: 10.1109/SISY.2017.8080566.

[3] G. Apruzzese and M. Colajanni. Evading botnet detectors based on flows and random forest with adversarial samples. *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8, 2018.

[4] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj. Model evasion attack on intrusion detection systems using adversarial machine learning. *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2020.

[5] A. S. A. Aziz, E. Sanaa, and A. E. Hassanien. Comparison of classification techniques applied for network intrusion detection and classification. *Journal of Applied Logic*, 24:109–118, 2017.

[6] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 387–402, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40994-3.

[7] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[9] D. J. Chaboya, R. A. Raines, R. O. Baldwin, and B. E. Mullins. Network intrusion detection: automated and manual methods prone to attack and evasion. *IEEE security & privacy*, 4 (6):36–43, 2006.

[10] S. Chan, 2021. URL `https://engineering.purdue.edu/ChanGroup/eBook.html`.

[11] T.-H. Cheng, Y.-D. Lin, Y.-C. Lai, and P.-C. Lin. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys & Tutorials*, 14(4): 1011–1020, 2011.

[12] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security*, 2021.

[13] L. Dhanabal and S. Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(6):446–452, 2015.

[14] Flickr. *Australian Box Jellyfish*. 2021.

[15] P. Fogla and W. Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, 2006.

[16] P. Fogla, M. I. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *USENIX security symposium*, pages 241–256, 2006.

[17] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan. Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):1–16, 2013.

[18] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial networks. *ArXiv*, abs/1406.2661, 2014.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[20] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

[21] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial examples for malware detection. In *European symposium on research in computer security*, pages 62–79. Springer, 2017.

[22] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin. Practical Traffic-space Adversarial Attacks on Learning-based NIDSs. Technical report.

[23] M. J. Hashemi and E. Keller. Enhancing Robustness Against Adversarial Examples in Network Intrusion Detection Systems. Technical report.

[24] M. J. Hashemi, G. Cusack, and E. Keller. Towards Evaluation of NIDSs in Adversarial Setting. page 14–21, 2019. doi: 10.1145/3359992.3366642. URL `https://doi.org/10.1145/3359992.3366642`.

[25] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek. Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach. *arXiv preprint arXiv:1805.02684*, 2018.

[26] M. H. Kamarudin, C. Maple, T. Watson, and H. Sofian. Packet header intrusion detection with binary logistic regression approach in detecting r2l and u2r attacks. *2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec)*, pages 101–106, 2015.

[27] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *ArXiv*, abs/1607.02533, 2017.

[28] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert systems with applications*, 39(1):424–430, 2012.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

[30] M. V. Mahoney and P. K. Chan. Phad: Packet header anomaly detection for identifying hostile network traffic. Technical report, 2001.

[31] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 2, pages 1702–1707. IEEE, 2002.

[32] C. Novo and R. Morla. Flow-based detection and proxy-based evasion of encrypted malware c2 traffic. *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, 2020.

[33] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016.

[34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Y. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *2016 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 372–387, 2016.

[35] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[36] Y. Peng, J. Su, X. Shi, and B. Zhao. Evaluating deep learning based network intrusion detection system in adversarial environment. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 61–66. IEEE, 2019.

[37] J. Postel. User datagram protocol. RFC 768, September 1980. URL `https://rfc-editor.org/rfc/rfc768.txt`.

[38] J. Postel. Internet control message protocol. RFC 792, September 1981. URL `https://rfc-editor.org/rfc/rfc792.txt`.

[39] J. Postel. Internet protocol - darpa internet program protocol specification. RFC 791, September 1981. URL `https://rfc-editor.org/rfc/rfc791.txt`.

[40] J. Postel. Transmission control protocol - darpa internet program protocol specification. RFC 793, September 1981. URL `https://rfc-editor.org/rfc/rfc793.txt`.

[41] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks inc Calgary Alberta, 1998.

[42] A.-U.-H. Qureshi, H. Larijani, N. Mtetwa, M. Yousefi, and A. Javed. An adversarial attack detection paradigm with swarm optimization. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[43] G. T. Rohrmair and G. Lowe. Using csp to detect insertion and evasion possibilities within the intrusion detection area. In *Formal Aspects of Security*, pages 205–220. Springer, 2002.

[44] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[45] R. Sheatsley, N. Papernot, M. Weisman, G. Verma, and P. McDaniel. Adversarial examples in constrained domains. *arXiv preprint arXiv:2011.01183*, 2020.

[46] E. Stinson and J. C. Mitchell. Towards systematic evaluation of the evadability of bot/botnet detection methods. *WOOT*, 8:1–9, 2008.

[47] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[48] A. Tamilarasan, S. Mukkamala, A. H. Sung, and K. Yendrapalli. Feature ranking and selection for intrusion detection using artificial neural networks and statistical methods. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4754–4761. IEEE, 2006.

[49] G. Vigna, W. Robertson, and D. Balzarotti. Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 21–30, 2004.

[50] L. Yu, J. Dong, L. Chen, M. Li, B. Xu, Z. Li, L. Qiao, L. Liu, B. Zhao, and C. Zhang. Pbcnn: Packet bytes-based convolutional neural network for network intrusion detection. *Computer Networks*, page 108117, 2021. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2021.108117. URL https://www.sciencedirect.com/science/article/pii/S1389128621001948.