MACHINE LEARNING-BASED INTRUSION DETECTION UNDER ADVERSARIAL INFLUENCE:APPLICATION TO MAC SPOOFING DETECTION IN IoT NETWORKS

POORIA MADANI

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (EECS)
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 2021

Abstract

Internet of things (IoT) has brought a greater prevalence of smart objects with higher connectivity between them. Today, there are millions of such smart devices controlling critical infrastructures, such as nuclear power plants, and have brought a new form of comfort to our home through smart appliances. As such, IoT devices have become valuable targets for (potentially state-sponsored) adversaries that are most often after cyberattacks with physical ramifications. Taking over a trusted node's identity (also known as identity spoofing) in an IoT network can enable more sophisticated multi-tier attacks against other nodes/resources in the same network. Thus, detecting identity spoofing attacks must be part of any sound defensive measure when protecting IoT networks. Several learning-based detection schemes have been proposed in the literature that attempt to detect identity attacks (i.e., MAC Spoofing) in wireless networks. However, the proposed learning-based methods are highly susceptible to adversarial evasion attacks - one of the main theme studied in this manuscript - and a

sophisticated adversary could circumvent detection by identifying "blind spots" in the learning algorithms of proposed approaches. In this dissertation, we have extensively studied the use of randomization (another major theme) both from defensive and offensive perspectives to add robustness to existing learning-based MAC spoofing detection methods. Specifically, we have proposed a randomization scheme that can be added to the existing learning-based detection approaches to increase the uncertainty of the adversary in the search of finding an optimal evasion strategy. Moreover, we have also proposed an adversarial search approach based on active learning that an adversary could use to mount an optimal evasion attack against detection classifiers that utilize randomization. Finally, we have proposed a novel multi-model MAC spoofing detection system based on deep autoencoders, which have been specifically designed and tested for IoT networks deployed in adversarial settings by taking into account environmental variabilities induced by moving objects.

To my Hannah, whose laughs kept me on track and gave me the courage to finish this work.

To my loving wife, Shabnam, who stood by my side through this long and difficult journey.

To my parents, whose words of encouragement will ring in my ears forever.

- Pooria, 28th of October 2021

Acknowledgements

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would like to thank my supervisor, Prof. Natalija Vlajic, for her consistent support and guidance during the course of completing this dissertation. Her insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to acknowledge members of my supervisory committee and members of Department of Electrical Engineering and Computer Science (EECS) at York University for their continual support and assistance that I have received during my Ph.D. study.

Table of Contents

List of Figures

List of Acronyms

RSSI Received Signal Strength Indicator

MAC Medium Access Control

IoT Internet of Things

LSTM Long-Short Term Memory

PCA Principle Component Analysis

WSN Wireless Sensor Network

OSI Open System Interconnection

WEP Wired Equivalent Privacy

SDR Software Defined Radio

DoS Denial of Service

AP Access Point

WLAN Wireless Local Area Network

SSID Share Service Set Identifier

GMM Gaussian Mixture Model

MLP Multi Layer Perceptron

MCMC Markov Chain Monte Carlo

RNN Recurrent Neural Network

RBM Restricted Boltzmann Machine

IDS Intrusion Detection System

DBN Deep Belief Network

SVM Support Vector Machine

RONI Reject On Negative Impact

ACRE Adversarial Classifier Reverse Engineering

IMAC Instance of Miinimal Adversarial Cost

MTD Moving Target Defense

GPR Gaussian Process Regression

RMTD Randomized Moving Target Defense

NIDS Network Intrusion Detection System

ANN Artificial Neural Networks

PCC Principle Component Classifier

DDoS Distributed Denial of Service

ROC Receiver Operating Characteristic

MSE Mean Squared Error

Chapter 1

Introduction

1.1 Motivations and Contributions

Machine learning is often used to detect malicious activities in a variety of cyber security applications (e.g., spam, intrusion, and fraud detections). Most commonly, in these applications, a learning algorithm attempts to discover a decision boundary that partitions (i.e., classify) the input instances into two group - malicious and normal. It has been demonstrated in [1] that convex-inducing classifiers (i.e., classifiers with convex decision boundaries, examples of which would be: linear classifiers, anomaly detection classifiers using bounded Principle Component Analysis (PCA) [2], anomaly detection algorithms that use hyper sphere boundaries, and

other more complicated convex bodies), are prone to optimal adversarial evasion. In other words, it has been demonstrated that through a search process sophisticated adversaries can find/discover evading instances (i.e., malicious instances that can pass detection) by issuing polynomially many queries against convex-inducing classifiers [1]. However, it remains an open-question whether and how *randomization* - which is an increasingly popular paradigm in the domain of cyber defence - could be used to improve the robustness of convex-including classifiers under similar adversarial conditions. Since convex-inducing classifiers are heavily used in cybersecurity for anomaly detection purposes, we have found this open-question extremely important and interesting to study. In this dissertation work, first by taking the perspective of an advanced adversary, we set to develop an optimal evasion algorithm that finds the most effective evading instances against a randomized convex-inducing classifier. Subsequently, by taking the perspective of the system's defender, we set to formalize a randomization defense scheme that can ensure improved robustness against the previously investigated optimal evasion attacks.

In addition to studying the aforementioned open-problem in theoretical settings [3], we also put our developed theories/methodologies into practice and utilize our findings within the realm of Internet of Things (IoT) application(s). IoT devices, in many deployment scenarios, do not enjoy similar physical security measures that other net

worked devices do. Therefore, sophisticated adversaries that are well-funded (e.g., nation sponsored) tend to go after cyber-physical systems that are controlled/monitored using IoT devices. Such adversaries have resources and know-hows to use machine learning and statistical techniques against learning-based classifiers. Specifically, we have discovered that in many seminal works on the use of Received Signal Strength Indicator (RSSI) values to detect Medium Access Control (MAC) Spoofing in wireless networks, the communicating devices are assumed to utilize convex-inducing classifiers which (as previously discussed) are prone to optimal evasion attacks. Thus, as part of this research work, we are also set to investigate: 1) the degree of susceptibility of RSSI-based MAC spoofing detection approaches in IoT environments to optimal evasion attacks, and 2) how our randomization defense scheme can improve such detection approaches.

The contributions of this dissertation revolve around two main themes: (a) ran- domization of convex-inducing classifiers in Internet of Things (IoT) Received Signal Strength Indicator (RSSI)-based MAC Spoofing Detection Systems, and (b) enhancing the state-of-the-art in MAC Spoofing detection in IoT applications using deep generative models.

Within the first theme, as depicted in Figure 1.1, we have investigated the issue of optimal evasion attack in randomized convex-inducing classifiers from attackers'

perspective and proposed a novel evasion algorithm in order to address some of the existing open research questions pertaining to optimal evasion attacks (Figure 1.1, Point 1). Subsequently, assuming the defenders' perspective, we have proposed a randomization defence scheme that can be adapted by the existing non-randomized convex-inducing classifiers to further enhance their robustness against optimal evasion attacks (Figure 1.1, Point 2). We have also studied the efficacy of our findings in IoT settings against identity spoofing attack (Figure 1.1, Point 3).

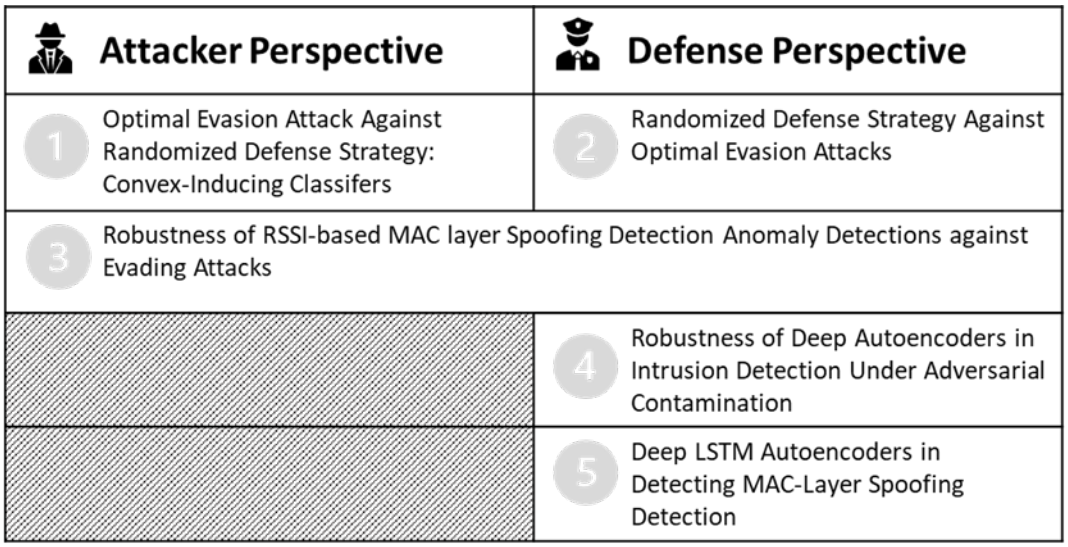


Figure 1.1: Contributions from attackers and defenders perspectives.

In the context of the second theme, we expand our investigation by studying the robustness of a deep generative model for anomaly detection using noisy datasets (Figure 1.1, Point 4). We than proceed by developing a novel non-convex RSSI-

based Medium Access Control (MAC) spoofing detection technique that addresses shortcomings of the existing methodologies (e.g., overlooking of variable environmental noise, assumption of unimodality, etc.), and is more robust against optimal evasion attacks (Figure 1.1, Point 5).

1.2 Dissertation Organization

The reminder of this dissertation is organized as follows. In Chapter 2, we introduce the key background concepts that are used throughout this manuscript, and we survey the main related previous works as well as the technical details of the existing approaches in intrusion detection in IoT systems and adversarial machine learning. In Chapter 3, we present a novel near-optimal semi-supervised machine learning based adversarial evasion approach against generic randomized classification systems with convex decision boundary. In Chapter 4, we first test the adversarial evasion approach developed in Chapter 3 against an existing MAC spoofing intrusion detection scheme. Subsequently, we propose a new robust randomization (defensive) technique as a way of enhancing the existing MAC spoofing detection approaches. In Chapter 5, we investigate the viability of deep autoencoders for anomaly detection in adversarial environments while particularly focusing on the effectiveness of using autoencoder reconstruction error as a means to measure anomaly. Extending on the idea from

the previous chapter, in Chapter 6, we propose the use of Long-Short Term Memory (LSTM) deep autoencoders for the purpose of robust multi model MAC spoofing detection in environments with variable environmental noise.

Chapter 2

Background and Related Works

In this chapter, we are going to review topics and research works that are essential to understanding the novel contributions of this dissertation. We begin by discussing some well-known datalink layer threats commonly faced by IoT devices (Section 2.1), and then we outline the key objectives of intrusion detection systems deployed on IoT nodes - concepts foundational for Chapter 4 and 6. In Section 2.2, we go into the details about the detection of MAC-Spoofing in IoT devices, and we specifically survey the works that focus on how such an activity can be detected using physical layer's received signal strength indicator (RSSI) – concepts also foundational for Chapter 4 and 6. In Section 2.3, we discuss the viability of deep generative models for the purpose of developing novel anomaly detection engines to be used in intrusion

detection systems, and we survey the existing cybersecurity works that utilize such deep learning models – concepts foundational for Chapter 5 and 6. In Section 2.4, we survey the existing body of works that have studied attacks against machine learning algorithms deployed in adversarial environments and primarily within the scope of modern intrusion detection systems. Then, in Section 2.5, we expand more on a specific form of attack against learning-based convex-inducing classifiers, also known as "Optimal Evasion" attacks, and we lay foundations for better understandings of the material in Chapters 3 and 4. We conclude this chapter by discussing the role of randomization to protect learning-based classifiers against adversarial influence – the concepts that also contribute to a better understanding of the material presented in Chapter 4.

2.1 Taxonomy of Data Link Layer Threats in IoT Systems

In this section, we are going to survey the most common datalink layer threats against IoT devices and we provide the necessary foundation for a better understanding of the overall contributions of this dissertation. The main motivation for focusing on the data-link threats is twofold: First, due to computational, storage and memory

limitations, many IoT and WSN devices refrain from implementing OSI protocols past layer 2, and only use datalink layer to exchange messages or sensor readings. Moreover, some of the most popular and commonly deployed IoT protocols (including IEEE 802.15.4 [4]) mandate only certain communication standards/functionality at the physical and data link layers, and the onus is on the device manufacturers and solution architectures to develop and utilize higher OSI layers, if needed.

Second, implementing upper layers of the OSI model which offer many complex security features (e.g., TLS) does not completely resolve security challenges present at the physical and datalink layer. For example, using physical and data link layer patterns extracted from captured traffic, an adversary can fingerprint and precisely identify individual IoT devices present on a premise. Following that, the adversary can mount some simple attacks on the fingerprinted devices, such as packet replay, packet flooding or battery exhaustion attacks. In other words, given the open-access nature of the wireless communication medium, adversaries can still threaten the security and reliability of wireless networks – even in cases when datalink or other upper layer security mechanisms are deployed (including encryption and authentication). Therefore, understanding the existing threats against lower layers of the protocol stack and possible defense mechanism against these threats can be crucial for secure and thus successful adaptation of IoT technologies, especially in case of critical

cyber-physical applications.

2.1.1 MAC Spoofing Attack

A Media Access Control (MAC) address is a hardware identification number that uniquely identifies each device in a network. Similar to other networks, in IoT systems a MAC address is used as part of device identification and packet routing procedures. From the communication perspective, MAC addresses appear as a field in communication frames, and they remain unencrypted even in case of secure communication channels. Therefore, simply by monitoring the traffic exchanged within the devices of an IoT network, a malicious observer can learn about the MAC addresses of the participating nodes. Subsequently, by using some readily available software libraries, the adversary could forge the identity of any legitimate (identified) device in this network simply by creating and transmitting frames with falsified (i.e., spoofed) MAC address of the given device – an action that is also known as 'masquerading attack'.

As previously indicated, given the inherent resource constraints of many IoT devices (e.g., a wireless sensor in a remote environment), sophisticated authentication and identity management – as a possible defence against MAC spoofing attack - is not always possible. For example, implementing a single cryptographic key on each device

for the purposes of authentication can be easily jeopardized should the adversary gain physical access to one of the devices. This is a very realistic threat to wireless sensor networks where sensors are implemented in an open environment lacking high physical security measures, such as farmlands, dams, or a country's border.

Furthermore, considering the lack of proper physical security, any form of key management can also be very challenging for many IoT networks. For example, with the adversary in control of one victim node, any subsequent key rotations involving this node can also be easily jeopardized. Therefore, one should consider segmenting IoT devices in multiple non-overlapping groups to avoid use of single encryption key for different critical environments/devices in a network; or even consider assigning a pairwise encryption keys between any two IoT devices that are in direct communication range. As an illustration of this, in a Zigbee sensor network with multiple edge sensor nodes and a few intermediary routing nodes, one can implement pair-wise network keys between each sensor and its corresponding router node and a separate key between routing nodes and the central sink node, in order to increase the security of the overall network and reduce the probability of MAC spoofing attack(s).

It should also be pointed out that physical access to IoT nodes, and the compromise to their encryption key(s) combined with MAC spoofing, can enable a number of

different more sophisticated attacks. For example, by extracting the network/session key from a jeopardized IoT node (in a network utilizing a single/master symmetric cryptographic key), the attacker can broadcast spoofed messages on behalf of other participating nodes in order to gain control over the information flow. An example of this would be a malicious node in a Zigbee network that advertises false routing path costs on behalf of other routing nodes (i.e., by falsifying these nodes' MAC address) in order to convince other nodes to forward their traffic through it – a malicious activity also known as Sybil attack [5]. As a result, the attacker ultimately gets in the position of being able to control significant routing traffic in the given network.

Since the datalink layer's encryption and authentication are optional features in some practical IoT protocols (e.g., IEEE 802.15.4), unfortunately many IoT device manufacturers produce and ship out their products without these features enabled or utilized. For example, it is discovered that Philips Hue light bulbs do not perform any form of authentication or encryption, and any adversary capable of broadcasting controlling commands with appropriate (spoofed) MAC addresses over the Zigbee protocol would be in the position to control these bulbs [6]. The most likely reason why an IoT manufacturer would omit the use of encryption and authentication in its devices are the difficulties pertaining to key management and pairing that an end-user might experience during installation and pairing of IoT devices.

It should also be pointed out that even in scenarios where IoT devices come with built-in/enabled encryption and authentication functionalities, not all of their communication procedures actually assume the use of these functionalities. For example, in de-authentication procedure specified in 802.11 Wi-Fi networks, a disassociation message/frame is generated by the access point in order to terminate its network connection with a particular node. However, it turns out that this particular management message/frame does not need to be encrypted even if the connection between the access point and the recipient node was previously protected (i.e., encrypted)using Wired Equivalent Privacy (WEP). Consequently, it has been demonstrated in [7] that a malicious adversary, by spoofing the MAC address of the access point, can evict victim devices from the network for short period of time and force them to reconnect to a rouge access point in their subsequent reconnecting attempts. Due to lack of frequent user interaction with many IoT/WSN devices (e.g., remote sensing applications) such MAC address spoofing could take longer to be detected by network operators. In other words, through rouge access point attack, an adversary could control or simply cut-off a victim IoT node from the rest of the network for an extended period of time.

2.1.2 Replay Attack

Capturing communication packets over a wireless channel is trivial. Replay attack is a network attack in which the adversary captures and replays an intercepted communication packet. For example, by intercepting an "unlock" command of a remote car keyset (not equipped with a replay defense mechanism), and then using a specialized transceivers such as software defined radio (SDR), the adversary can replay the intercepted packets at any time to unlock the victim's vehicle without requiring access to the car's original keyset [9, 8].

What makes the replay attack especially challenging to defend against is the fact that even encrypted payloads can be intercepted, retransmitted and then successfully accepted by a receiving node (unless the deployed communication protocol comes with anti-replay defences). Namely, since in a replay attack the adversary does not need to modify the content of the intercepted payload/frame, a simple retransmission of this frame with its original (i.e., unmodified) source and destination MAC addresses is all it takes for the attack to be successful. For example, in the car key unlock replay attack scenario, the attacker does not need to know the exact command codes or protocol details between the keyset and the vehicle – only a simple replay of the previous capture frame(s) could unlock the car door.

One of the reasons why it especially important to study replay attack within

the context of IoT networks, is the fact that many IoT devices are used for sensing and remote control applications in home or enterprise environments. For example, a remote sensor deployed in the uranium holding pool of a nuclear power plant may periodically report state of the pool such as the water level and/or temperature. By capturing the nominal state transmissions of this sensor, and then replaying them in the future, an adversary could create a critical state for the plant and/or ultimately endanger human life.

2.1.3 Denial of Service (DoS)

A denial of service (DoS) attack occurs when legitimate users/devices are prevented from accessing information systems or other network resources due to the action of malicious cyber threat actor. The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in a denial of service to additional requests by legitimate users/devices. There are several different forms of DoS attacks in IoT networks that we are going to discuss in this section.

Jamming wireless signals is one of the oldest and most trivial forms of DoS attack against any wireless communication network, and probably one of the hardest to defend against. In this form of DoS attack an adversary would continuously occupy the communication radio spectrum by producing electromagnetic noise and prevent

participating legitimate wireless devices to communicate with each other.

Another form of DoS attack that is unique to IoT networks, and generally does not apply to other types of networks, is energy resource depletion. As previously discussed, many IoT devices are battery operated with long periods of time between battery swaps (e.g., wireless sensor nodes deployed in remote and harsh environments), therefore any unnecessary transmission and computation can shorten life-time of the node and eventually result in the node become unavailable. For example, in networks where encryption is used for the purposes of sender authentication (e.g., only genuine nodes possess the right encryption/decryption key(s)), an adversary could send a flood of packets with a spoofed MAC address corresponding to a legitimate node and carrying dummy payloads encrypted using a random key. To validate these packets (i.e., ensure that they have been sent by the legitimate node whose MAC address is appearing in the packets' headers), the receiving/victim node(s) would need to decrypt each of them. This process would consume excessive amounts of energy only for the received spoofed packets to be discarded in the end (as they were encrypted with a wrong/random key).

Another variant of the energy exhaustion DoS attack, that does involve MAC spoofing, can be performed by actively replaying a broadcast massage of a controlling node (e.g., access point in Wi-Fi or coordinator node in Zigbee networks). Namely,

by doing so, the attacker can prevent nodes from going back to sleep mode by making them busy receiving and processing replayed broadcast packets ,which will result in a significant energy loss on each of the participating nodes [10, 11]. Note that even in cases when the participating (i.e., victim) nodes deploy upper layer guards against packet replay, this attack will still be effective, as the nodes will be engaged in the reception and processing of received replay packets on the lower (i.e., physical) layer.

2.1.4 Threat to Privacy

Passive monitoring of wireless-communication spectrum in an environment can be very revealing about the activities of devices (and residents) present in that environment. For example, large volume of continues Wi-Fi signal in a residential network could be due to the usage of online video streaming service such as NetFlix; thus, an adversary can infer that there are human occupants currently on the premise. Or, the similar occupancy information can be inferred by signal spikes generated by motion detectors in a smart home. Not surprisingly, researchers have demonstrated that wireless communication signal patterns can be used to identify type and roles of IoT devices (and/or activities of their users) even in case of an encrypted communication channel [12].

Additionally, radio signals bouncing off moving object can be used to develop

passive radar, which then can be used to detect and track objects in an environment. Researchers have demonstrated that motion detection based on passive-radar architecture using wireless signals such as Wi-Fi can achieve a remarkable detection accuracy [13]. Thus, deployment of IoT devices (i.e., active transmitters of wireless signal) in home and enterprise environment can help adversaries perform the so-called physical reconnaissance. For example, with large deployment of WSN nodes, adversaries can construct a passive radar for tracking security patrol vehicles of a high-security facility [14].

Although the above described privacy threats are not the result of flaw(s) in the design or utilization of IoT devices (but rather are an unavoidable consequence), they nevertheless raise significant privacy issues. Such issues can be precursor and enabler of physical security breaches.

2.2 MAC-Spoofing Detection using Received Signal Strength Indicator (RSSI)

As explained in the previous section, the proliferation of Internet of Things (IoT) and Wireless Sensor Network (WSN) networks has revived an old yet serious form of attack – MAC-layer Spoofing attack. In MAC address spoofing attack, as the name

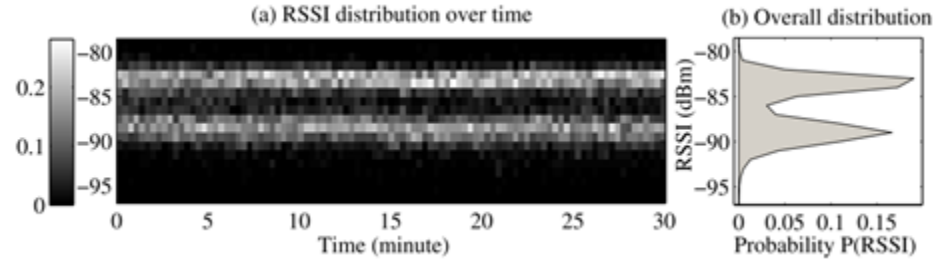


Figure 2.1: An example of multi-model RSSI distribution of an Zibee IoT Node.

suggest, a rouge wireless node masquerades as another legitimate device by cloning the legitimate device's MAC address. The most common way of defending against this form of attack is through the use of cryptographic techniques for MAC-address authentication[15]. However, due to the resource limitations that are inherently present in many IoT and WSN devices (e.g., low processing power, low memory, and limited battery life), many of these devices operate with very scaled-down (if any) versions of encryption and authentication protocols.

Clearly, in wireless systems with limited cryptographic and authentication protection, other alternative measures against MAC address spoofing are required. One such measure – which can also be used as an added layer of security even in wireless system with extensive cryptographic and authentication protection – is the utilization of physical layer (i.e., signal-level) variables. Received Signal Strength Indicator (RSSI) is a wireless communication variable that is directly influenced by the transmission power and the location of the transmitter as well as different environmental variables

such as obstacles. As suggested in a number of earlier research works [17, 16, 5], RSSI values can be used to create the fingerprint profile (refer to Figure 2.1) of each device in a wireless network and then deploy these profiles to do preliminary authenticity check against the MAC spoofing attack(s). Another point that makes RSSI profiling an attractive ally against MAC spoofing attacks is that the use of this single real-valued physical-layer variable is easy to implement, requires no modifications to the existing higher layer protocols and applications, and has a very small processing and memory footprint.

In the seminal paper [18], Faria and Cheriton were among the first ones to propose the use of RSSI values as a fingerprinting variable to detect MAC Spoofing attacks in a WLAN environment. As part of their detection model, it is assumed that there are multiple access points (APs) capable of receiving the wireless signals for all clients in the network, so the RSSI values measured at each AP's antenna and for each transmitted data frame are ultimately aggregated into a single profile. Consequently, a masquerading attack is detected by comparing the aggregated RSSI values of two consecutives data frames with the same MAC identifier. Also, they have demonstrated that using multi sensing APs, and assuming constant transmitting power, a physical node can be triangulated with an accuracy of 5 to 10 meters. Unfortunately, the practical merit of these findings is rather limited since the use of multiple overlapping

APs in many WSN and IoT network is not always possible.

Chen et al. [19] and Wu et al. [17] have both independently proposed the use of k-means clustering algorithm to detect signal/frame spoofing by a rogue access point (AP). Their work is grounded on the assumption that the sequence of last n RSSI values received from an AP would have minimum fluctuations around the mean in the absence of another rogue AP (i.e., an Evil Twin). Thus, when clustering the elements of a received RSSI sequence into two clusters using k-means algorithm in the absence of an Evil Twin, the distance between two formed centroids would be small (i.e., smaller than a threshold value). At the same time, large distance between the centroids of the two formed clusters would be indicative of the existence of an Evil Twin AP with its unique RSSI distribution. However, since their approach does not have any offline learning (i.e., previously trained model of what to be considered legitimate distribution), the MAC address spoofer and the legitimate node must transmit in a relatively close time intervals in order to be detected.

Sheng et al. [20] studied the effect of antenna diversity in 802.11 access points and their effect on RSSI device fingerprinting as well as spoofing detection. They demonstrated that RSSI values from a stationary receiver collected from a stationary transmitter form a mixture of two Gaussian distributions due to antenna diversity permitted under 802.11 protocol. As a result, they have trained a Gaussian mixture

model for each wireless node and access point pair in the network and used *log-likelihood ratio* test on the sequence of latest received RSSIs at each access point from a given MAC address. A transmitting node is ruled spoofed if the ratio test fails by more than n Gaussian mixture models – where n is smaller than the number of available access points in the network and needs to be set empirically. However, using available off-the-shelf hacking tools an adversary can easily manipulate its transmission power to evade detection by this model, as discussed in later sections.

Gonzales et al. [21] have developed a novel technique known as *context-leashing* for detection of public Evil Twin access points. They have argued that publicly available access points such as the ones available at franchise coffee shops (e.g., Starbucks) share service set identifier (SSID) across different locations and oftentimes lack any authentication. This provides an opportunity for adversaries to spoof such SSIDs and trick clients into associating with the rouge access point (e.g., after performing a disassociation attack). The defense against the Evil Twin APs proposed in [5] assumes the use of a so-called context-leashing engine. Upon association with a publicly available access point, the context-leashing engine would collect a list of contexts $C_i = \{(c_1, r_1), \ldots, (c_n, r_n)\}$, which contains the list of all visible SSIDs (denoted by c_j) and their corresponding average RSSI value (denoted by r_j) that are reachable at the time of association with a particular SSID in the environment. This

list is compiled for each associated SSID. For any future re-association with a given SSID, a new context list is constructed and compared to the previously stored one. If the similarity of available neighboring SSIDs and their average RSSI values does not have a significant (empirically defined) overlap with the historical context-list, then the associated SSID is deemed as Evil Twin and the connection should be terminated. The main drawback of their method is the assumption that the list of SSIDs (and their respective signal strength) in a given geolocation remains relatively unchanged over time, however, with today's tethering capabilities of cellphones this assumption is far from the truth.

2.3 Deep Generative Models

In the domain of machine learning, there are two main categories of models/algorithms: generative and discriminative models. Generative models learn the joint probability distributions $p(x, y)$ directly from the training dataset. In other words, they aim to model a joint probability distribution over both the observations and respective label sequences (i.e., full probabilistic model of all variables), in order to be able to categorize new (previously unseen) instances. On the other hand, discriminative models do not attempt to model the underlying probability distributions $p(x, y)$, but instead aim to directly estimate posterior probabilities $p(y|x)$. In other words,

discriminative models provide a model only for the target variables conditioned on the observed variable(s).

Generative models can be used to sample and generate instances from the modelled distribution. Such generation capability can be useful in many tasks, including the evaluation of another trained model, which is discussed in the following sections. In this section, we are going to explore the deep generative models that can be used for the construction of anomaly-based IDS.

2.3.1 Deep Autoencoders

An autoencoder is an unsupervised neural network that is trained to reconstruct (i.e., copy) its input to its output. Architecturally, an autoencoder is similar to the multilayer perceptron (MLP), having an input layer, an output layer, and one (or more) hidden layers connecting them (see Figure 5.1). The hidden layer h, also known as representation layer, describes a code that facilitates the desired input-output mapping. Alternatively, it can also be said that an autoencoder neural network consists of two parts: an encoder function $h = f(x)$ that encodes the input onto the representation layer h, and a decoder function $r = g(h)$ that reconstruct the coded representation h back into the original input. Hence, the main goal of this architecture, as depicted in Figure 5.1, is to learn an approximation of $g(f(x)) \approx x$.

Nevertheless, in order to achieve a necessary/required reconstruction ability, the model (i.e., this approximation) often ends up depicting the most useful properties of the input data.

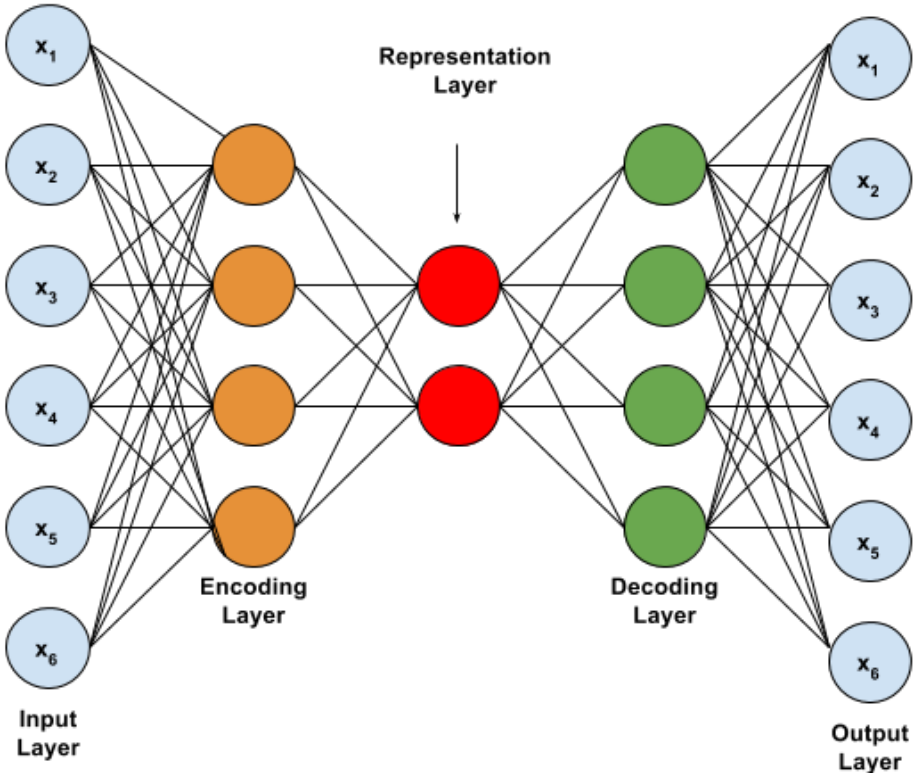


Figure 2.2: Anatomy of a generic autoencoder.

The original idea of autoencoders, initially introduced by LeCun [22], has been studied for decades. Traditionally, autoencoders have been used for the purpose of dimensionality reduction or feature extraction. Recently, as researchers have identified some inherent similarities between autoencoders and latent/hidden variable models,

autoencoders have also found their use in the domain of subspace analysis.

The learning process in autoencoders aims to minimize

$$L(x, g(f(x))), \tag{2.1}$$

where L is a loss function (e.g., mean squared error) penalizing the dissimilarity between $g(f(x))$ and x. When the activation function used in the decoder layer(s) are linear and L is the mean squared error, the autoencoder learns to span the same subspace as Principle Component Analysis (PCA). On the other hand, autoencoders with nonlinear *encoding* function f and nonlinear decoding function g can learn a more powerful nonlinear generalization of PCA [23].

Autoencoders are a special case of feedforward Artificial Neural Networks (ANN). Thus, they inherit all the properties and training procedures associated with ANN. More importantly, given nontrivial depth in an autoencoder, universal approximation theorem [24] guarantees that autoencoder can approximate any function to an arbitrary degree of accuracy.

Variational Autoencoders

Variational autoencoder [25] is a stochastic generative modification of regular autoencoder. The encoding function $f(x) = p(h|x)$ outputs parameters σ and μ of a normal distribution such that $h \sim \mathcal{N}(\mu, \sigma^2)$. In other words, output values of h stochastically

change for a given x but within the region of the described normal distribution by the output parameters of the encoding layers.

Under this new configuration, $g(h)$ has to be able to reconstruct the given x for any value represented by h that are drawn from $\mathcal{N}(\mu, \sigma^2)$. Thus, the decoding portion of the neural network estimates $g(h) = p(x|h)$. In other words, $g(h)$ is a generative model that is capable of generating samples similar to the training set for different values of h that are drawn from normal distributions.

Given that variational autoencoders represent the latent probability distributions of a given dataset, they can easily become a generative model by adapting Markov Chain Monte Carlo (MCMC) walk.

Anomaly Detection using Autoencoders

It is worth noting that, although autoencoders are mostly used for feature learning and dimensionality reduction, they can also be used for the purpose of anomaly detection. Namely, the underlying assumption is that the autoencoder can be trained on (a subspace of) the normal dataset, and any input that is substantially apart from the majority of the normal dataset will have a large reconstruction loss. In other words, as shown in Equation 2.2, *mean squared error* loss function L can be used for both the purpose of training, and subsequently (once the training is completed) for

the purpose of anomaly detection.

$$L(x, g(f(x))) = \frac{1}{n}\sum(x_i - g(f(x_i)))^2 \tag{2.2}$$

$$\alpha = P(L(x, g(f(x))) > C \mid \text{x is normal instance}) \tag{2.3}$$

In order to be able to approximate a general function, lossless reconstruction is not the ultimate objective in the training of an autoencoder. Instead, an allowance threshold C (appearing in Equation 2.3), can be set to differentiate between a normal and faulty reconstruction. Given that FAR is vital in the design and deployment of intrusion detection systems, as shown in Equation 2.3, the threshold value C can be calculated empirically given the desired rate of false-positive α.

2.3.2 Recurrent Neural Network (RNN)

Recurrent neural networks (RNN) [26] are a family of neural networks that are capable of processing and modelling sequential data. Unlike MLP where input variables x are fixed in length and order-independent, by the means of parameter sharing, RNN is capable of modelling variable-length sequence of values $x^{(1)}, \ldots, x^{(\tau)}$ while discovering order dependencies. In order to generalize, consider a dynamic system in which its

current state $h(t)$ (at time t) is driven by an external input signal x and the previous state $x^{(t)}$, such that

$$h(t) = f(h^{t-1}, x^t), \tag{2.4}$$

implying that the current hidden state $h(t)$ indirectly contains information about the whole past sequence. Figure 2.3 is the graphical representation of Equation 2.4 .

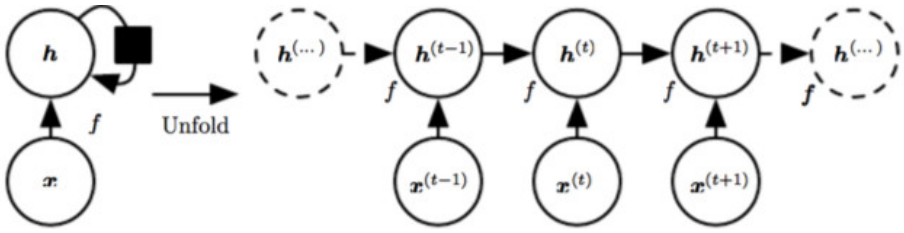


Figure 2.3: Visualization of Equation 2.4 unfolded in time t [23].

Different design patterns can be used in the construction of RNN. For example, a commonly used design pattern is the one where RNN produces an output at each time step and have recurrent connections between hidden units. Such RNNs are generally used for the purpose of sequence classification or regression.

Restricted Boltzman Machine RNN (i.e., RNN-RBM) [27] is a generative variation of RNN in which the hidden state $h(t)$ emits parameters of a restricted Boltzmann machine at each time step t. Such models can be easily adapted to capture a sequence of probability distributions that are responsible for the generation of the observed sequence.

2.3.3 Use of Deep Learning in IDS Development

An intrusion detection system (IDS) is a special software or hardware system designed to detect malicious activities on a stand-alone computer or a network of computers. Detecting intrusions can transpire at different levels and scales, ranging from antivirus software on a single workstation/device to a complex multi-device system that monitors the traffic of an entire network. The basic assumption of intrusion detection is that, on some level, intrusive activities are different from normal ones, and this difference can be successfully detected. In the domain of IDSs, there are two common approaches to the detection of intrusive activities: signature-based and anomaly-based. Signature-based techniques encode knowledge about patterns of malicious known activities and perform detection through pattern matching. On the other hand, anomaly-based intrusion detection techniques try to establish a model of what is considered normal behavior/profile and detect any deviation from the established model. In this section, we discuss these two different approaches in more detail and summarize some representative examples in each category.

Many recent research works have looked into deep learning techniques for construction of modern IDS. Alom et al. [28] have used a deep belief network (DBN), which consists of many layers of restricted Boltzmann machine, to classify network intrusions in NSL-KDD dataset. Since their model is trained using both malicious

and normal instances, it is classified as a signature-based detection technique. They have compared their approach with the one from [29] which used DBN as a feature extraction for Support Vector Machine (SVM) classifier, and showed that DBN classifier could outperform the hybrid SVM models.

Abolhasanzadeh et al. [30] have used deep autoencoder for the purpose of feature extraction and trained a special support vector machine (SVM) classifier to perform intrusion classification on features extracted from the deep autoencoder model. They have compared the classification accuracy of a model using the features extracted by an autoencoder and those extracted by PCA. Based on the obtained results they have concluded that deep autoencoders are a more promising feature extractor (i.e., better suited approach) for use in real-world intrusion detection systems.

Given the nature of information fed into an IDS (e.g., a sequence of network packets or sequence of system-calls), RNN is an excellent candidate for the use in anomaly-based IDSs that are required to deal with sequential data directly. Jihyun et. al. [31] have trained a RNN for classification of network attacks in KDD Cup'99 dataset. Although this dataset is not inherently sequential, disjoints instances are concatenated to synthesize normal and malicious traffic sequences. Despite of the questionable approach to the construction of their training dataset, the experiments described in [31] have yielded rather favourable result. Thus, further research in the

use of RNN for development of anomaly-based IDS using real sequential datasets seems like a valid and important research direction.

2.4 Survey of Adversarial Machine Learning (AML)

Adversarial machine learning is a recently emerged interdisciplinary field of machine learning (ML) and cybersecurity that studies the challenges faced by machine learning techniques under adversarial influence. In other words, AML is a collection of techniques to build more robust models against adversarial attempts to evade detection by targeting blind spots of classical machine learning techniques. Because of its interdisciplinary nature, this field operates under a unique set of assumptions, which are rather different from the classical assumptions of the traditional ML community. For examples, in the traditional ML, the training and the testing dataset are drawn from the same (possibly unknown and most often stationary) source distribution. Thus, it is the task of the learning model to approximate the characteristics of the given source distribution. However, in the world of cybersecurity, sophisticated adversaries could change their behaviour after the deployment of one or a set of (targeted) defensive measures, and thus render these originally developed/trained models (and respective intrusion-detection defence measure) ineffective. Such a possibility of dealing with non-stationary adversarial datasets requires that the adversarial learning

techniques be able to adapt to any significant change in the input data, in a timely manner. Unfortunately, the very ability to adapt (i.e., change) their criterion of what constitutes (new) normal behaviour may put these techniques at the mercy of sophisticated adversaries, as they could introduce intentional change in the input data with the sole purpose of destabilizing the defence system [34, 33, 32].

In this chapter, different attacks against existing machine learning-based detection schemes are surveyed. Moreover, some potential defensive measures as well as the robustness of detection algorithms against the outline attacks are discussed in details.

2.4.1 Attack Taxonomy

Before diving into the details of different attacks against ML-based detection models, it is important to understand the nature of alerts generated by an intrusion detection system. As depicted in Figure 2.4, not all malicious instances get captured by a given detection model and not all generated alerts are worthy of investigation. In Figure 2.4, let A denotes the set of all generated alerts and M denotes the set of all malicious instances. In general, the goal of a defender is to develop a model that is capable of achieving $max(A \cap M)$; in other words, all and only malicious instances should be flagged.

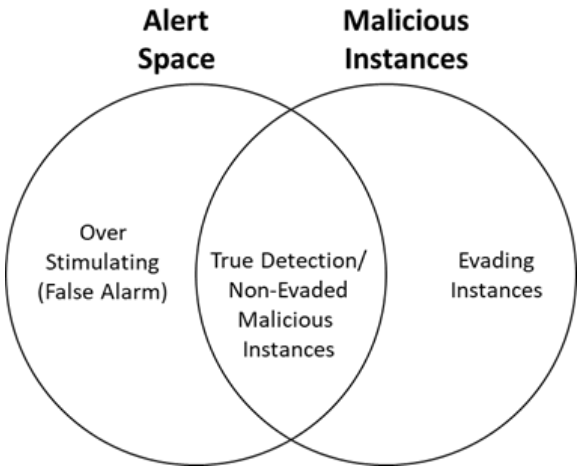


Figure 2.4: Alert space of a typical detection schema.

It is the goal of the adversary, however, to devise strategies to either reduce the $A \cap M$ space and/or find instances that are not covered by it. Namely, alerts that are not result of a valid detection (hence not within $A \cap M$) are considered false alarms, which in large volumes can overwhelm the system operators. Put another way, in case of a large number of false alarms, the detection system falls victim to its own misdetections, which can overwhelm the system to the point of causing effects similar to those of denial of service (DoS) attack.

Table 2.1: Taxonomy of attacks against machine learning systems [35]

		Integrity	Availability
Causative			
	Targeted	Poisoning training data for a particular attack family to evade detection.	Poisoning training data to block certain legitimate instances.
	Indiscriminate	Poisoning training data for range of attacks and introducing noise to evade detection.	Poisoning training data to cause the classifier to block broad range of legitimate instances.
Exploratory			
	Targeted	Camouflage (i.e., obfuscating) a chosen attack to evade detection.	Overstimulation to block specific legitimate activity.
	Indiscriminate	Camouflage any attack family to evade detection.	Overstimulation to block range of legitimate activities.

Now, in systems that deploy learning-based intrusion detection, there are multiple ways of how an adversary could go about increasing the number of false alarms. In [35] Nelson et al. have proposed a comprehensive taxonomy of attacks against machine learning systems, which is provided in Table 2.1. In their taxonomy, the adversary can either: a) Cause malfunctioning in the detection system by manipulating training instances with the primary aim to camouflage future attacks as normal in order to evade detection. Or, b) Explore blind spots of a trained classifier to find evading malicious instances at runtime. In the rest of this section, we are going to provide a comprehensive review of the two main attack categories from Table 2.1 - Causative

and Exploratory.

Causative Attacks

Causative attacks (also known as poisoning attack) are a set of techniques that an adversary could employ in order to cause the creation of a faulty or incomplete classification model, as described in Table 2.1. Huang et al. [33] have proposed a game-theoretic approach between an adversary and a defender - where the adversary manipulates data to poison or evade a learning algorithm chosen by the defender. Their proposed framework can be summarized as follows:

- Defender generates a hypothesis H from the observed data (i.e., training dataset) using a chosen machine learning algorithm.
- Based on some knowledge of H, the attacker assembles some attack procedures A.
- Attack on Learning:
  - Contaminate training dataset D^{train} with contamination from A^{train}.
  - Learn hypothesis H using the contaminated D^{train}.
- Attack On Evaluation:
  - Contaminate evaluation dataset D^{eval} with contamination from A^{eval}.

– Compare predictions $f(x)$ to y for each data point

$(x, y) \in contaminatedD^{eval}$

For an integrity attack, the attacker who desires false negative will use A to create or discover instances that result in no alarms. For attack against the availability, the attacker will try to discover points that result in false positives.

Poisoning attacks are not just meant to result in the creation of faulty models so that adversary evades detection. For example, Nelson et al. [35] demonstrated a powerful denial of service attack against SpamBayes [36], a statistical spam filter. In their first attack scenario, the adversary would send a very large number of obvious spam emails containing a very large set of legitimate and often used vocabularies. After retraining of SpamBayes on the newly received spam emails, the victim's spam filter will have a higher spam score for every word occurred in the vocabulary of the spam emails. And since most of the used vocabularies are legitimate words (e.g., name of companies, people, etc.) the resultant spam filter would end up filtering many legitimate emails. In their papers, they have shown that such online systems can effectively be poisoned by controlling only a small percentage (e.g., 2%-5%) of the entire training dataset.

Red Herring attack is a particular case of poisoning attack where the *feature space* becomes the main target of the adversary. In this case, initial malicious samples

advertise highly correlated and discriminatory set of features (i.e., spurious features) in order to be picked by the learning algorithms. In subsequent generations, emerging malicious samples will refrain from employing the spurious features in order to evade detection. Newsome et al. [37] have shown how red herring attack is possible against Polygraph worm detection.

Rubinstein *et al.* [38] have demonstrated a stealth poisoning technique against the so-called subspace anomaly detectors, by means of injecting crafty chaff data points into the normal flow of traffic instances. Specifically, they have demonstrated that injection of chaff traffic during the training phase of an IDS can be effective in increasing the success of subsequent DoS attacks. Principle Component Analysis (PCA) is an example of subspace analysis techniques that can be used for the purpose of adaptive DoS detection. In [38], the authors have specifically demonstrated that a proactive adversary that wishes to launch DoS traffics can slowly introduce irregularities into the traffic pattern over time, and by doing so the adversary can distort the set of principle components used by the ISP's detection engine. Such proactive stealthy poisoning of training dataset is known as *Boiling Frog*.

In a recent work, Xiao et al. [39] have theoretically investigated the auto-discovery poisoning sets aimed at maximizing classification error in detection systems deploying SVM. By arguing that large datasets which are manually labeled by multiple parties

(and possibly through user feedback) are prone to label flipping, and in cases when adversaries have full knowledge of features used by the learning algorithm, it is possible to discover the set of minimum L labels that maximize the test error of the SVM. Although their method requires full knowledge of features used by the training algorithm, they have indicated that development of partial knowledge attacks is part of their future work.

In many cases the effectiveness of poisoning attacks is related to the robustness of learning algorithms to data's noise as well as the data sensitization techniques employed. Reject On Negative Impact (RONI) is one of the data sensitizations defenses that measure the empirical effect of removing each new training instance that have a substantial negative impact on classification accuracy [3]. After training on the initial training dataset, which is presumably empty of any malicious noise, the defender will use RONI for future iterative training. Although RONI is found to be an effective cleansing procedure against dictionary attack scenarios such as the ones executed on SpamBayes [35], it can be ineffective against red herring or boiling frog attacks. On the other hand, Biggio et al. [40] have demonstrated that weighted bagging (i.e., associating different weights to trained classifiers) can be an effective approach for handling 20% of poisoning attacks, without compromising the accuracy of the respective detection model.

Exploratory Attacks

Exploratory attacks, as categorized in Table 2.1, are attempts by the adversary to passively circumvent an already trained detection model and exploit some of its blind spots. In contrast to causative attacks which aim to influence learning by controlling the training data, in exploratory attacks the adversary does not have any control over the training data but instead aims to exploit potential misclassifications of the detection system (e.g., by camouflaging intrusion traffic as normal traffic in order to avoid detection). Different parameters can result in a successful exploratory attack, many of which have been studied extensively by the community [32, 33, 34]. The most common cause of successful exploratory attacks is the incompleteness of the initial training datasets.

In many detection scenarios, the detection model will provide a binary feedback (e.g., whether a input transaction has been accepted or not) that adversary can utilize to reverse engineer valuable information regarding the decision boundaries of the detection model. For example, Cumming et al. [41] have described a spam detection scenario where the adversary can freely experiment with the detection system and discover vocabularies that successfully evade detection. In particular, they have presented an automatic evading system that employs Naive Bayes learning and is capable of estimating the core parameters of the detection model by observing the

relevance feedbacks provided by the system.

Kantchelian et al. [42] have described a technique based on Mixed Integer Linear programming that is capable of finding evasion examples for tree-based ensemble classifiers. In their seminal work, they have demonstrated that tree ensembles are particularly susceptible to exploratory evasions, and as a solution they have proposed a dataset augmentation technique using the discovered evasion instances.

Xu et al. [43] have proposed a method based on Genetic Programming (GP) to discover variations of a PDF Malware that retain their original behavior while resulting in different signatures (as in the case of polymorphic malware that can effectively evade detection). Although they do not propose any robust detection model against these types of attacks, they do propose a general robustness test of malware detection models. Their work also highlights the general importance of frameworks for testing the robustness of models which operate in adversarial settings [44].

Querying and observing the behaviour of a detection model, as in the case of spam detectors, is not always without cost. In some cases, the adversary has to query the target system a polynomial number of times in order to be able to approximate the decision boundaries and discover the suitable evading examples. *Cost-based evasion* is a subcategory of research in exploratory attacks that models the costs of attack evasions, and studies their overall implementation difficulty.

In a pioneering work, Low and Meek [34] have introduced adversarial classifier reverse engineering (ACRE) learning problem. Although, in this work, these authors focus on reverse engineering learning algorithms and discover near-optimal evasion instances with respect to a cost function a, it should be noted that optimality is not always the key objective of an adversary. In other words, a classifier that has a high optimal-attack cost might still be evaded using relatively trivial queries that are non-optimal but are cheap with respect to adversary's budget.

Violation of privacy is another issue of learning models operating in an adversarial environment. Through interaction with classification models, indirectly, adversaries may be able to build surrogate models [3] that reveal partial or full reconstruction of the original training dataset, which in some cases could result in violation of user privacy. This problem has been addressed in [45], which also discussed the idea of privacy-preserving learning algorithms. These algorithms would allow the release of aggregate statistics on a (training) dataset, but without disclosing the information about individual elements of that dataset.

Overstimulation, as shown in Table 2.1, is an exploratory attack on availability of detection systems. Overstimulation of a detection model (i.e., an IDS) is the case when adversaries craft patterns that result in a large number of false alarms. Since in many systems human operators need to review many of the security alerts before

deciding on the right course of action, large volume of false alerts can delay the response time in case of an actual/true incident. Overstimulation is also a very serious problem in intrusion detection systems that implement immediate mitigation plan as a response to a generated alert. For example, blocking a particular connection, port or IP address can be executed automatically by a firewall of a network as a result of generated alerts by the network's NIDS [46]. Such operation can lead to denial of service (DoS) to legitimate users if enforced based on a false alarm.

Unfortunately, there is no straightforward solution to overstimulation problem of ML-based detection models. Although cost-sensitive learning techniques is an alternative approach to address the cost of accidentally introduced DoSs, such detection models are at the mercy of the employed training algorithms and quality of the training dataset.

2.5 Near-Optimal Evasion Attacks in Convex Classifiers

Adversarial Classifier Reverse Engineering (ACRE), first defined by Lowd and Meek [34], is an effort by the adversary to find an instance that is classified as non-malicious (negative) by the defending classifier while it is very similar to the malicious (positive)

instance of interest to the adversary - a type of exploratory attack discussed in 2.4.1. For example, consider a spam e-mail message that is successfully flagged by a spam detector of a public e-mail service provider (this can be checked by sending a test email and confirming it is labeled as spam - we refer to this interaction as issuing membership query by the adversary against the defending classifier). Then, the adversary is interested in finding an e-mail composition similar to its own original message (with minimal changes) that would be labeled as non-spam. To present this concept more formally, let (2.5) define a weighted adversarial ℓ_1 norm cost function, where x^A is the instance of interest to the adversary (e.g., the original message that was labeled spam), x is an arbitrary instance represented in a D dimensional feature space, c_d is the cost associated with each feature.

$$A(x) = \sum_{d=1}^{D} c_d |x_d - x_d^A| \tag{2.5}$$

Then, minimal adversarial cost (MAC) of a classifier f is

$$MAC(f, A) \triangleq \inf_{x \in X_f^-} [A(x)], \tag{2.6}$$

where a non-malicious instance x with a cost equal to MAC is called an instance of minimal adversarial cos t ($IMAC$). (In (2.6), X_f^- denotes set of instances that are considered normal under a trained classifier f.) It should be noted here that an adversary is not always interested in finding an instance x with the exact cost of

MAC, as in many cases an instance with a cost close to MAC may be sufficient.

Let us further define $\epsilon - IMAC$ to be a non-malicious data point with a cost no more than a factor $(1 + \epsilon)$ of the MAC. According to [34], a classifier is *ACRE learnable* if there exists an algorithm that could find an $\epsilon - IMAC$ by issuing only polynomial-many (in respect to the size of the feature space) membership queries.

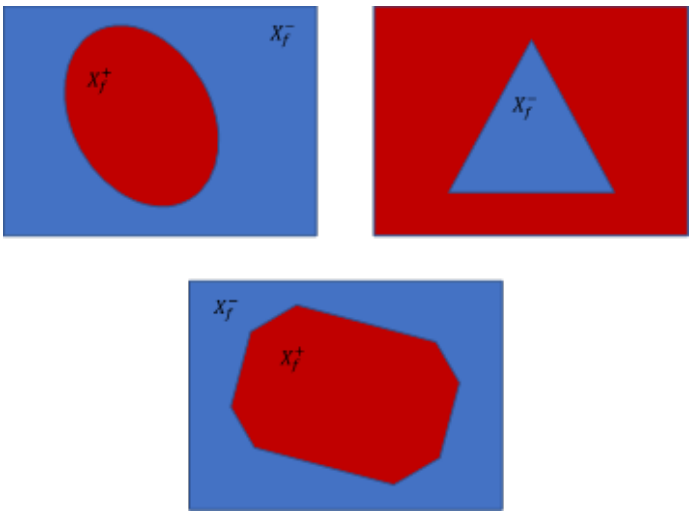


Figure 2.5: Example of partitioned feature spaces where one of the classes are convex in 2D.

The complexity of ACRE for a given defending classifier is directly related to the decision boundaries formed by the defending learning algorithm and the adversarial cost function. In a seminal work, Nelson et al. [1] proposed two algorithms that can efficiently solve ACRE learning problem when the defender uses a convex-inducing

classifier (i.e., learned convex decision boundaries), and where the adversarial cost function is $\ell_1 - norm$. Otherwise, they have proved that the problem becomes intractable. As depicted in Figure 2.5, Convex-inducing classifiers are set of classifiers that partition their input feature space into a positive (i.e., malicious) and negative (i.e., normal), one of which is convex. The class of convex-inducing classifiers include the linear classifiers (e.g., logistic regression or SVM), one-class classifiers that predict anomalies by thresholding the log-likelihood of a log-concave (or uni-modal) density function, and quadratic classifiers. The convex-inducing classifiers also include complicated bodies such as any intersections of a countable number of half-spaces, cones, or balls.

The query strategies and results presented by Nelson et al. [1] proved to be important to the community of cybersecurity practitioners given that many important anomaly detection algorithms used in the field are from the family of convex-inducing classifiers. After tractability study of [1] the community posed a new open problem, namely: how would randomization of decision boundaries affect the theoretical results obtained? (for more see [3] - a question which we have investigated as part of this thesis research. Randomization of decision boundaries refer to the concept of utilizing multiple decision boundaries in a classification (e.g., training multiple classifiers) task that are selected non-deterministically at runtime when handling incoming

classification requests.

2.6 Randomized Defense: a form of Moving Target Defense

Moving target defense are strategies that continually change over time to increase complexity and cost for attackers and increase system resiliency. This philosophy can be adopted against optimal ACRE. The main intuition is that the effect of a randomization scheme would increase the uncertainty that an adversary experiences during reverse engineering of the decision boundaries and/or finding of an optimal evading instance. And, subsequently, this extra uncertainty would reduce the effectiveness and increase the cost of attacks mounted by the adversaries. Defensive randomization schemes can be divided into three categories, namely Algorithm Parameters, Feature Space, and Instance Space.

In Algorithm Parameter randomization schemes, multiple classifiers are trained by set of different parameters of a chosen learning algorithm (or different algorithms); then, at the runtime, the defender would select one of the trained classifiers at random to handle an incoming instance (e.g., e-mail, network traffic, credit card transaction.) Alabdulmohsin et al. [47] have proposed a robust (against adversarial

reverse engineering) Support Vector Machine (SVM) multi classification system by training multiple SVM models with slightly different objective function parameters and randomly selecting among them for handling each observed query. Their approach is grounded in the fact that by slightly modifying the training parameters, the decision boundaries learned by each SVM model will be slightly different. This difference would result in non-consistent/variable labeling of the points closer to the set of learned hyperplanes (that represent different decision boundaries) and consistent labeling for points that are further away. Thus, their approach, as verified by experiments, would obscure exact location of the decision boundaries and make reverse engineering and/or discovery of optimal evading instances (ACRE) more challenging.

In Feature Space randomization schemes, the feature space is divided into multiple (possibly overlapping) sub-feature sets before training a classifier on each of the subsets. Then, at runtime, one of the classifiers will be selected at random to handle an observed query. Colbaugh and Glass [48] have proposed such randomization defense scheme against evolving spammers. They have argued that spammers use feedback they receive from e-mail servers to change the content of their emails for possible evasion. But, such modifications might not have significant effect if the revised words are not part of the feature space of the randomly selected classifier.

In Instance Space randomization scheme, the training instances are divided up into

multiple (possible overlapping) subsets, and each set is used to train a classification model. Then, at runtime, one of the classifiers will be selected at random to handle an observed query. Alternatively, the classifiers can be mixed in order to develop a more hybrid approach. However, it is important to consider under any randomized scheme adversarial evasion and reverse engineering of decision boundaries can still be very successful if all the trained models accurately predict the target class. It should also be noted that in cases when at least some of the trained classifiers are not very accurate, randomization can cause the attacker to incur an error, which suggests there is a pronounced tradeoff between classification accuracy and susceptibility of being reverse engineered [50, 49].

Chapter 3

Near-Optimal Evasion of Randomized Convex-Inducing Classifiers in Adversarial Environments

3.1 Introduction

Consider a black-box program that implements some function $f : \{x \in \mathbb{R}^n\} \rightarrow \{+, -\}$ with the goal of differentiating between two sets of inputs. Such black-box can be used in many applications where discrimination between sets of inputs, such as spam detection or intrusion detection, is desired. Although many security practitioners

utilize statistical analysis and machine learning techniques to discover the optimal parameters of the discriminating function f, others can rely on domain expert knowledge to manually set the function parameters to some predetermined values. Nevertheless, these parameters form the decision boundaries according to which discrimination between two or more classes of instances can be made.

By submitting queries to f and observing its decision response (e.g., '+' or '-'), an adversary could deduce the decision boundary parameters governing the classification results for some class of functions this is known as adversarial classifier reverse engineering (ACRE) [34] (discussed in details in Section 2.5). Upon discovering the information about f's parameters, sophisticated adversaries may morph their malicious input instances in order to evade detection. For example, by creating two auxiliary email accounts, a spammer can send spam emails from one to the other email address and observe what word distributions are getting flagged as spam and use the discovered knowledge to rewrite its previously flagged spam messages.

Effectiveness and security of learning algorithms are extensively studied in the general context [51, 52, 3], as well as in the context of different cybersecurity application such as biometric authentication systems [53, 54, 55], spam detection [57, 56], and malware detection [58]. Randomization of decision boundary parameters (e.g., utilization of multiple classification/detection functions and then selecting them

at random) has been one of the primary defenses against ACRE attacks. Namely, it is commonly assumed by security defenders that discovering *randomized* decision boundaries is more resource intensive for adversaries [59, 3].

The family of convex inducing classifiers that are capable of partitioning feature space into two sets, one of which is convex, is extensively used in many cybersecurity applications. This family includes linear classifiers, hyper-sphere boundary anomaly detectors, one-class anomaly detectors, just to name a few. Therefore, improved understanding of attack complexities against randomized convex-inducing classifiers is of uttermost importance for the development of next-generation defense techniques. In this chapter we propose a novel algorithm through which an adversary could initiate ACRE attack against randomized convex inducing classifiers and study the complexity of this attack as an extension to the complexity work conducted by Nelson et al. [1] and the open question that they proposed in [3].

3.2 Problem Preliminaries

We are going to extend and build upon the problem setup proposed by Nelson et al. [1] for near optimal evasion of convex-inducing classifiers. In particular, we assume that the set of all possible input instances $X \subseteq \mathbb{R}^D$ is represented in a D-dimensional feature space and each component of an instance $x \in X$ is a feature denoted by x_d

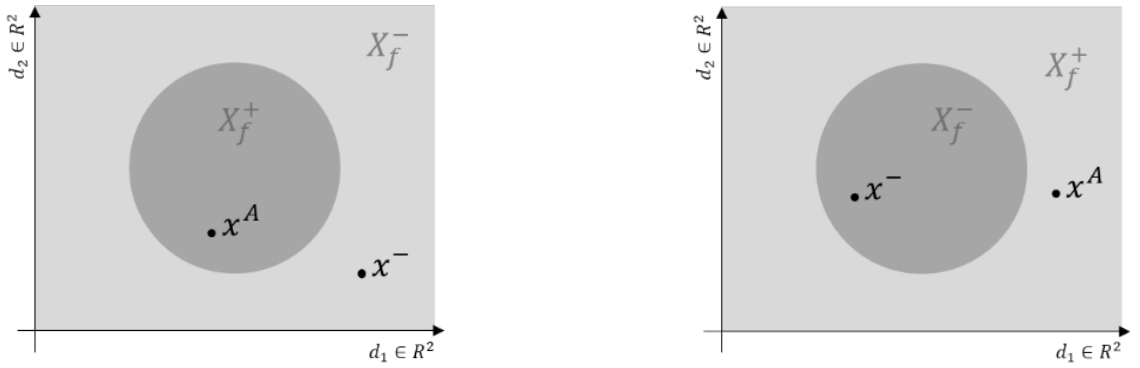
where $d = 1 \ldots D$. Let us use $\delta_d = [0, \ldots, 1, \ldots, 0]$ to denote a coordinate vector in R^D with 1 only at the d^{th} feature.

Furthermore, we assume that the target classifier (i.e., black-box function) is a mapping function $F : X \to Y$, from instance space X to its response space Y. We assume that adversary is not aware of parameters of F, and for simplicity we consider binary classification where $Y = \{+, -\}$.

Nelson et al. [1] have considered the case where F is a single convex-inducing classifier partitioning the feature space into two sets, one of which is convex (refer to Figure 3.1). Furthermore, they define $X^+ = \{x \in X | F(x) = +\}$ denote the positive class (i.e., set of instances classified as malicious by F) and $X^- = \{x \in X | F(x) = -\}$ denote the negative class (i.e., set of instances classified as normal by F) as partitioned by the classification function F. Moreover, they assume that the adversary is aware of at least one instance $x^A \in X^+$ (referred to as the starting instance) with ultimate objective to replace this instance with a closest instance in the instance space that evades detection.

Let $A : X \to R^{0+}$ denotes a cost function, which measures the distance to the starting instance $\mathbf{x}^A \in X^+$. The cost function relative to the $\mathbf{x}^A$ is defined as a weighted Minkowski (ℓ_p) distance

$$A_{p,c}(x - \mathbf{x}^A) = \left(\sum_{d=1}^{D} c_d |x_d - \mathbf{x}_d^A|^p \right)^{1/p}, \tag{3.1}$$



(a) an example where malicious (X^+) class is convex.

(b) an example where benign (X^-) class is convex.

Figure 3.1: Possible binary classification scenarios using convex inducing classifier.

where $0 < c_d < \infty$ is the cost/weight that the adversary associates with changing d^{th} feature; and $A_{p,c=1} = A_p$ represent a cost function when the relative costs are uniform for all features (i.e., $c_d = 1, \vee d \in D$). Let

$B^C\left(A_p; \mathbf{x}^A\right) = \left\{x \ \in X \mid A_p(x - x^A) \ \leq \ C\right\}$ denote the cost ball centered at $\mathbf{x}^A$ containing all instances that do not exceed an ℓ_p (e.g., $A_{p=1}$) cost of C.

Let *Minimal adversarial cost* (MAC) [34] of a classifier F be defined as

$$MAC(F, A_p) \triangleq inf_{x \in X^-}\left[A_p(x - \mathbf{x}^A)\right], \tag{3.2}$$

which is the lower bound on the cost obtained by any negative instance. Additionally, let *ε-instance of minimal adversarial cost* (ε-IMAC) be any data point from X^- with a cost no more than a factor of $1 + \varepsilon$ of the MAC. The challenge with near-optimal

evasion problem is when an adversary is set to find an ε-IMAC, preferably with as few queries as possible. For example, given a spam message created by an adversary and represented as a word-to-vector instance x^A, the adversary is interested in changing as few words as possible in this instance vector in order to evade spam detector deployed at the victim's e-mail server.

3.2.1 Moving Target Defense: Randomization of Classification Boundary

It has been suggested by a number of previous research studies [48, 50, 49] (also surveyed in Section 2.6) that randomization of decision boundaries is an effective way to defend against ACRE attack by making it more difficult for the adversary to find $\varepsilon - IMAC$. (I.e., by requiring a larger number of search queries to be issued in order to find $\varepsilon - IMAC$.) Now, the objective of this chapter is to explore the actual effectiveness of such an approach (in which the system deploys multiple and randomly selected classification boundaries) which is also commonly referred to as the moving target defense (MTD) approach.

Let $F_\gamma = \{f_1, f_2, f_3, \ldots, f_n\}$ denote a set of convex-inducing classifiers trained on instance space X, let γ be a selection mechanism that draws a classifier at random (i.i.d.) from F_γ, and the adversary does not have any knowledge about γ. Let

$X_\gamma^+ = \{x \in X |\ p(F(x) = +) > 0\}$ denote the set of instances that have a chance of being classified/labeled as positive and $X_\gamma^- = \{x \in X \mid p(F(x) = -) > 0\}$ denote the the set of instances that have a chance of being classified/labeled negative by the classification function F_γ governed by some probability function p. It should be noted that due to the nature of randomness introduced at the classification level, X_γ^+ and X_γ^- are not mutually exclusive and some instances from the instance space can be part of both sets with different probabilities. We also assume that the adversary is aware of at least one instance from each set - e.g., $x^- \in X_\gamma^-$ and $x^A \in X_\gamma^+$. The ultimate objective of the adversary is to replace the start instance x^A with a closest instance in the instance space that evades detection with some probability P (the evasion probability P is selected by the adversary). Or, put another way, the adversary's aim is to identify/discover x such that in $p(F(x) = -) > P$ (for some threshold value P) that is closest to x^A. For example, suppose an adversary is aware that a given malware detector is utilizing an unknown randomization classification scheme and his goal is to find a modified version of its malware's binary that evades detection n times out of N classification trials when analyzed by the randomized malware detector (where $P = n/N$).

The introduction of multiple classifiers and γ implies that the adversary must be able to discover information about more than one classifier and also approximate

the governing selection parameters of γ in order to find an $\varepsilon - IMAC$. Figure 3.2 is depicting an example of $|F_\gamma| = 2$ convex inducing classifiers for when either X_γ^+ is convex or X_γ^- is convex (for other possible geometries of randomized convex inducing classifiers refer to Appendix A).

3.2.2 Probable Near-optimal Evasion

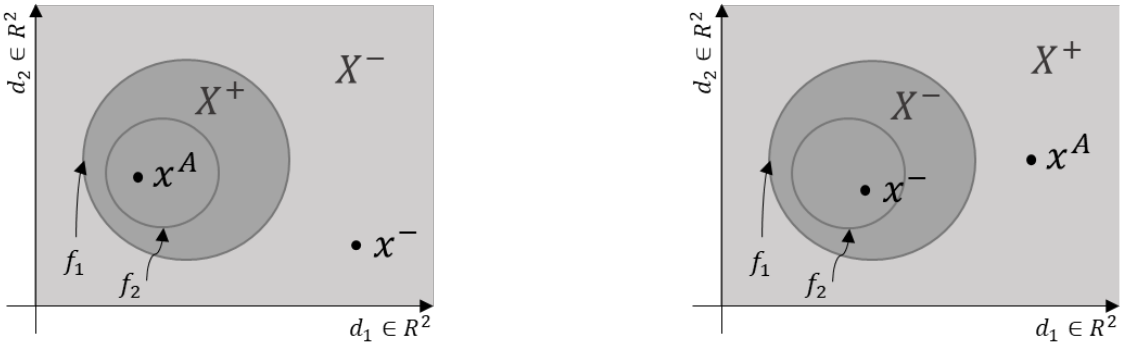
The problem of finding a $\varepsilon - IMAC$ in a non-randomized case with one single decision boundary (non 'moving-target' scenario) is relatively simple, since the MAC value for any potential $\varepsilon - IMAC$ point does not change. (This problem has been investigated in details by Nelson et al. [1].) However, in the moving-target scenario, the MAC value for any potential $\varepsilon - IMAC$ will/may change in each trial as new decision boundary is selected. Moreover, a point that is a potential $\varepsilon - IMAC$ under one decision boundary may not remain $\varepsilon - IMAC$ under another decision boundary and thus not even qualify to be the ultimate $\varepsilon - IMAC$. Therefore, in the light of randomized classification systems, the problem of near-optimal evasion by finding an $\varepsilon - IMAC$ requires new considerations. Specifically, we will define an instance to be a probable evading instance with minimal adversarial cost ($\varepsilon p - IMAC$) if the probability of being classified as malicious (under the random changing of classifiers from F) is less than a given threshold P . In other words, all eligible $\varepsilon p - IMAC$

instances are now members of the set

$$\varepsilon p - IMAC(f_\gamma, A, P) \triangleq \{x \ \in X_\gamma^- \mid A_p(x - \mathbf{x}^A) \leq (1+\varepsilon) \cdot MAC(f_\gamma, A)$$

$$s.t.\ p(F_\gamma(x) = +) \leq P\},$$

and it is the goal of the adversary to find an $\hat{x} \in \varepsilon p - IMAC$ efficiently (i.e., using polynomial-many queries).



(a) a collection of convex decision boundaries probabilistically defining the class of X^+.

(b) a collection of convex decision boundaries probabilistically defining the class of X^-.

Figure 3.2: Example scenarios implementing MTD.

We are going to consider two cases, namely (a) when X_γ^- is defined using a set of convex inducing classifiers and (b) when X_γ^+ is defined using a set of convex inducing classifiers. As depicted in Figure 3.2a, when union of decision boundaries that define X_γ^+ is convex, the adversarial search is translated into a convex optimization problem

(for other cases refer to Appendix A). In the following sections, we have proposed and studied a novel approach to finding $\varepsilon p - IMAC$ when X_γ^+ is defined by a collection of randomized convex decision boundaries. However, the problem of finding $\varepsilon p - IMAC$, when X_γ^- is defined using a collection of convex decision boundaries (refer to Figure 3.2b), is an optimization problem over a non-convex set (recall, that the adversary start the search at point x^A which is in a space with a convex cutout - therefore the outer set becomes non-convex). Thus, providing a general evasion method for this configuration is a form of non-convex optimization where there is no known polynomial algorithm for finding the exact solution (although our proposed method can be used to find *approximate* solution to such non-convex settings) and therefore is not further considered in our work.

3.3 Methodology

The generalized optimal query strategy for ACRE of non-randomized convex-inducing classifiers, proposed by Nelson et al. [1], uses ℓ_1 adversarial cost (i.e., $A_{p=1}$) and performs binary search along each feature vectors (i.e., multi-line search) in order to find a ball-cost $B^C(A; \mathbf{x}^A)$ containing at least an element of $\varepsilon - IMAC$ (refer to Figure 3.3). However, under the discussed MTD scheme (Section 3.1), the multi-line search method [1] does not guarantee to return $\varepsilon - IMAC$ with certainty. Under

Algorithm 1: Algorithm to ep-IMAC.

Data: x^A, x_-, C,ε,T_σ,P
Result: x^* $p\varepsilon$-IMAC instance

$\overrightarrow{\theta} \leftarrow [1, 0]$
$\overrightarrow{X} \leftarrow [x^A, x^-]$
$x^* \leftarrow x^-$
do
  construct GPR (MEAN($\overrightarrow{\theta}$), KERNEL($\overrightarrow{X}$))
  foreach $d_i \in D$ **do**
    foreach $\{x \ | \mathbf{x}^A \pm \ C_d \cdot \delta_d$ *that are* ε *spaced*$\}$ **do**
      $\theta_i, var_i \leftarrow \hat{f}(x^i)$; `// GPR prediction at` x^i
      if var_i *is maximum* **then**
        $max_i \leftarrow x^i$
      end
      if $\theta_i < P$ **and** $var_i \approx 0$ **and** $|x^A_d - x^i_d| \leq C_d$ **then**
        $C_d \leftarrow |x^A_d - x^i_d|$
        if $||x^A - x^i||_1 \leq ||x^A - x^*||_1$ **then**
          $x^* \leftarrow x^i$
        end
      end
    end
  end
  query the oracle and find θ_i and $\sigma(\theta_i)$ for max_i
  $\overrightarrow{X} \leftarrow \overrightarrow{X} \bigcup max_i$
  $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} \bigcup \theta_i$
  update KERNEL and noise $\sigma(\theta_i)$ based on Equation 3.4
while *maximum variance is above* T_σ;
return x^*

the MTD scheme, the adversary must perform a stochastic search for discovering information about classifiers in F_γ and its randomized selection mechanism γ in order to find an $\varepsilon - IMAC$.

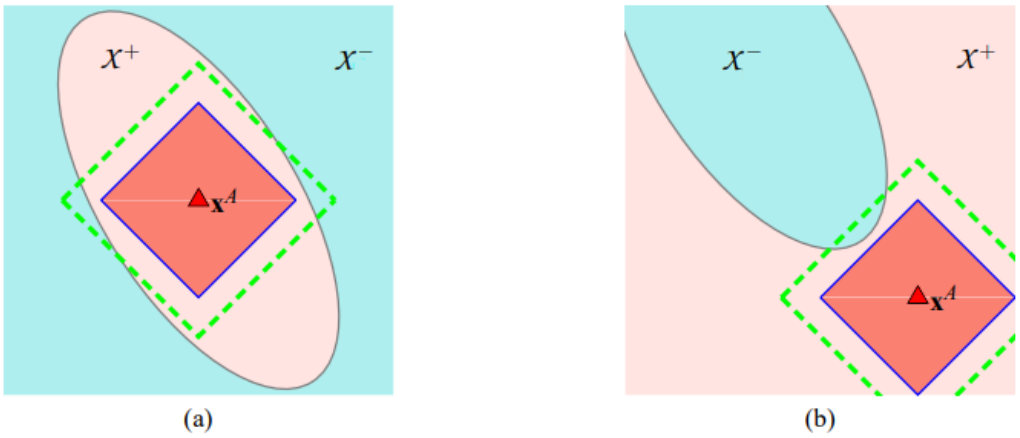


Figure 3.3: Geometry of convex sets and ℓ_1 balls [1] (a) If the positive set is convex, finding an ℓ_1 ball contained within X^+ establishes a lower bound on the cost. (b) If the negative set X^- is convex, we can establish upper bound on the cost by determining whether or not an ℓ_1 ball intersects with X^-

3.3.1 $\varepsilon p - IMAC$ Search for Convex X_γ^+ and ℓ_1 Cost Function

Searching for $\varepsilon p - IMAC$ when X^+ decision boundaries are convex is proven to be infeasible [1] for the adversarial cost function A_p where $p \neq 1$. (I.e., the problem is feasible only for $p = 1$.) Therefore, by respecting this constraint and setting the

initial cost to be of the form $C = ||\mathbf{x}^A - x^-||_1$, we search for an instance x^* that is: a) different from x^A in exactly one feature, b) satisfies the requirement that $||x^A - x^*||_1 + \varepsilon < C$, and c) has an associated probability of being classified as benign (i.e., negative) greater than or equal to a predefined threshold P. Our multi-line search algorithm attempts to learn the coordinates where the decision boundaries intercept along each feature dimension in order facilitate the search for finding an $\varepsilon p - IMAC$ under a given P .

3.3.2 Active Multi-Line Search

The search for an $\varepsilon p - IMAC$ with the adversarial cost of $A_{p=1}$ where X_γ^+ is convex, is done by querying points that are different from x^A in exactly one feature dimension. (Refer to Section 2.5 for a more detailed discussion on finding an $\varepsilon - IMAC$.) From the adversary perspective, F_γ (i.e., the oracle) is a mapping function from instance space X into a real number range between 0 and 1 denoting probability of classifying the input as malicious. Let $Q \subseteq X$ denote set of instances that adversary may choose when interacting with the randomized classifier (F_γ), then, the problem of finding $\varepsilon p - IMAC$ can be turned into estimating the mapping function F_γ denoted as $\hat{F}_\gamma$

such that

$$\forall d \in X \; and \; \forall Q \in [\mathbf{x}^A, \mathbf{x}^A \pm C \cdot \delta_d]$$

$$\hat{\mathbf{F}}_\gamma : Q \rightarrow p(F_\gamma(Q) = + \,),$$

along each feature dimension d centered at $\mathbf{x}^A$ and bounded by a constant cost $C \in MAC$ using as few queries as possible.

Approximation of F_γ (i.e., $\hat{\mathbf{F}}_\gamma$) can be formulated as a regression learning problem by the adversary. Therefore, we submit that the adversary must intelligently select multiple $q^i \in Q$ and query the oracle (F_γ) in order to reduce the approximation error of $\hat{\mathbf{F}}_\gamma$. (Recall, learning by strategically interacting with an oracle is known as *active learning*.)

Now, for $\forall q^i \in Q$, there exist a Bernoulli process $\theta_i \in \theta$ where its probability of success $\theta_i = p(\mathbf{F}_\gamma\,(q^i) = +)$. *Gaussian process regression* (GPR), is a nonparametric kernel-based approach,that models θ as a multivariate Gaussian distribution $P\left(p(\hat{\mathbf{F}}_\gamma\,(q^1) = +), p(\hat{\mathbf{F}}_\gamma\,(q^2) = +), \ldots, p(\hat{\mathbf{F}}_\gamma\left(q^N\right) = +)\right)$ with some mean function $\mu\,(Q)$ and covariance matrix $\Sigma\,(Q)$ given by $\Sigma_{ij} = \kappa\,(q^i, q^j)$, where κ is a positive definite kernel function. The main idea of GPR is that if q^i and q^j are deemed by the kernel to be similar, then the output of $\hat{\mathbf{F}}_\gamma$ at those points must be similar (refer to Equation 3.3).

Not only the trained GPR model can be used to estimate $p(F(q^i) = +)$ for all

$q^i \in Q$, also, the uncertainty of the estimated values are reported by the model. This ability of quantifying uncertainty makes GPR a great tool to help the adversary in selecting sequence of points to be queried from the oracle that can further reduce the GPR model's overall uncertainty about its estimations. Let θ denotes a set of values resulted from the evaluation of the training set $Q^\circ \subset Q$ on $p(\mathbf{F}_\gamma(\cdot) = +)$[1] and θ^* denotes a set of values from evaluation of $Q^\dagger \subset Q$ on $p(\hat{\mathbf{F}}_\gamma(\cdot) = +)$ where $Q^\dagger \cap Q^\circ = \emptyset$. In other words, $Q^\dagger$ are the potential points that the adversary might be interested in and still has not evaluated them against the oracle, but, he can evaluate them against the trained GPR to find their approximate probability of malicious classification according to his trained model. In that case, Equation 3.3 describes how GPR is constructed and used for inference.

[1]Training set $Q\circ$ is not the training set used by defense to build F_γ. This is a training set that the adversary assembles from his interaction with the oracle.

$$\theta^* \sim GPR(\mu, \Sigma),$$

$$\mu = E[\theta],$$

$$\Sigma = \begin{pmatrix} \kappa + \sigma(\theta) I & \kappa_* \\ \kappa_*^T & \kappa_{**} \end{pmatrix}$$

$$\kappa = \kappa(Q^\circ, Q^\circ), \quad \kappa_* = \kappa\left(Q^\circ, Q^\dagger\right), \quad \kappa_{**} = \kappa\left(Q^\dagger, Q^\dagger\right) \tag{3.3}$$

$$\kappa\left(q^i, q^j\right) = exp\left(-\frac{1}{2\ell^2}\left(q^i - q^j\right)^2\right)$$

$$\theta^* = \kappa_*^T(\kappa + \sigma I)^{-1}\theta$$

$$Var[\theta^*] = \kappa_* - \kappa_*^T(\kappa + \sigma I)^{-1} \kappa_*$$

The search on a given dimension d, as described in Algorithm 1, starts by constructing a GPR model using the vector of the known initial points $\theta = \left[p(\mathbf{F}_\gamma\left(\mathbf{x}^A\right) = +), p(\mathbf{F}_\gamma(\mathbf{x}^-) = +)\right]$ and $Q^\circ = \left[\mathbf{x}^A, \mathbf{x}^-\right]$ by the adversary. The model constructs a mean function with $\mu\left(\mathbf{x}^A\right) = 1$ with $Var\left[\mathbf{x}^A\right] = 0$ point and $\mu(\mathbf{x}^-) = 0$ with $Var[\mathbf{x}^-] = 0$ and a smooth interpolation of the mean and the variance drawn from the multivariate Gaussian distribution for points in $Q^\dagger \approx X - Q^\circ$. Then, at each iteration:

1. the process selects a data point $q^i \in Q^\dagger$ with the highest variance tested against the currently trained GPR (variance value will be collected from current iteration's GPR trained model for all the points in $Q^\dagger$),

2. acquires $p(F_\gamma(q^i) = +)$ by binomial sampling of $F_\gamma(q^i)$ (using N trials),

3. and finally update the GPR model by adding queried point q^i to Q° and $p(F_\gamma(q^i) = +)$ to θ^* and retrain the GPR.

Setting $P = 0$ (in Algorithm 1) will result in discovery of $\varepsilon - IMAC$. Reducing the upper bound of the cost ball (C) reduces number of queries in the subsequent iterations and helps to find $\varepsilon p - IMAC$ efficiently. This process is continued until the variance of the GPR model for all points in Q falls below a set threshold T_σ. Since the oracle (F_γ) is a sign function with a binary range of $\{+, -\}$, one must issue multiple queries in order to approximate θ_i at a given q^i and add approximated value to the training dataset (θ^* and Q°) before retraining the GPR. Algorithm 1 uses Beta distribution to find the mean and variance of observed θ_i as described in Equation 3.4, where N denotes the number of the total queries issued with value q^i and N^+ denotes the number of times F_γ returned malicious label (i.e., $+$). The parameters (α and β) of the Beta distribution controls our belief about the range of the values that θ_i can take on. Since the adversary is completely uncertain about F_γ, setting $\alpha = \beta = 1$ establishes a uniform probability for all θ_i between 0 and 1. The uncertainty about value of $\theta^i \in \theta^*$ represented by $\sigma(\theta_i)$ in Equation 3.4 is incorporated as a sampling noise into the covariance matrix Σ in Equation 3.3. This allows to issue fewer queries per each q^i in the beginning and refine their accuracy (reducing sampling variance of the point q^i) if needed in later iterations by issuing more sampling queries for q^i

against the oracle and update corresponding θ_i and $\sigma(\theta_i)$.

$$\begin{aligned} \theta_i &= \frac{\alpha \ + \ N^+}{\alpha \ + \ \beta \ + \ N} \\ \sigma\left(\theta_i\right) &= \sqrt{\frac{\theta_i\left(1-\theta_i\right)}{N}} \end{aligned} \tag{3.4}$$

Our method, similar to the proof provided by Nelson et al. [1], searches along each feature dimension by utilizing ℓ_1-norm cost function. Therefore, the overall complexity of Algorithm 1 is $O^*(2D \cdot K)$, where D is the number of dimensions of the feature space and K is a derived valued from chosen kernel function κ and is bounded by a constant.

3.4 Experimental Analysis

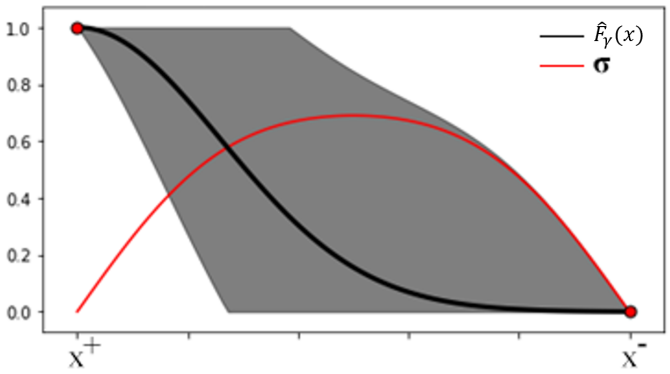
To illustrate iterations of our proposed method, we have created two toy examples. The first example is the classification of a single feature (i.e., single dimension) with three decision points $F_\gamma = \{3, 4, 5\}$, where each $f_i \in F_\gamma$ makes a binary classification decision based on the value it represents. Namely, if the input to a selected classifier is smaller than the function value, the input will be labeled as malicious, otherwise labeled as normal. Additionally, randomization selection function γ uniformly chooses $f_i \in F_\gamma$, and $x^A = 0$ and $x^- = 10$ are the (initial) input points/configuration for Algorithm 1. Figure 3.4.a is depicting the initialization of the system with the two known points to the adversary and interpolation of other points in between using the

GPR trained on the two known initial points. At later iterations (Figures 3.4.b and 3.4.c), points with larger variance are selected for binomial sampling from oracle and to retrain the GPR on the newly acquired information (i.e., system feedback).

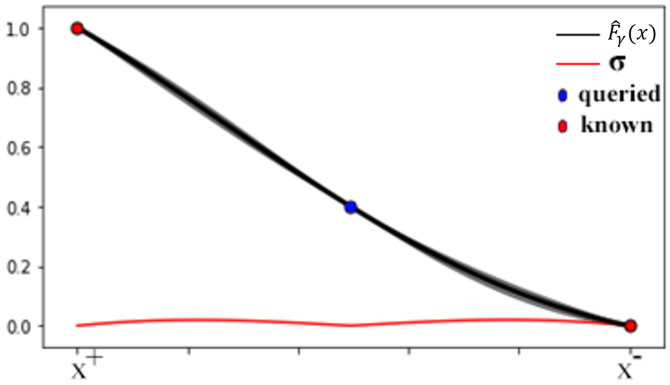
The algorithm is terminated when the overall variance of the latest GPR model became negligible. The resulting graph depicted in Figure 3.4.c is a visualization of conditional probability function $p(F_\gamma(x) = +)$. Finding an instance of $\varepsilon p - IMAC$ is accomplished by performing a walk along this curve searching for $\hat{x}$ with desired value P closet to $\mathbf{x^A}$.

The second toy example consists of three two-dimensional convex (circular) decision boundaries around the origin with radiuses $F_\gamma = \{5, 6, 7\}$ (i.e., the three classifiers are represented using circular decision boundaries, with the specified radius modelling malicious instance space). Function γ is uniformly choosing from F_γ, and $x^A = \{0, 0\}$ and $x^- = \{10, 10\}$ are the input points/configuration into Algorithm 1, as depicted in Figure 3.5a. The initialization of the system is shown in Figure 3.5b where $p\left(F_\gamma(x^A) = +\right) = 1$ and a smooth interpolation of other points everywhere else.

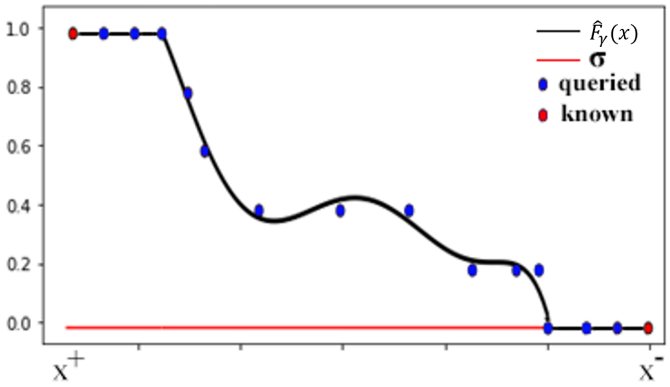
As $\varepsilon - IMAC$ is proved [1] to be found by search along feature dimensions (i.e., $\varepsilon - IMAC$ is different from x^A in only one feature value), Algorithm 1 also models $p(F_\gamma(x) = +)$ only along each feature dimension by iteratively selection points with large variances along unit vectors. As is shown in 3.5c, the algorithm has expanded



(a) The initial GPR with the two known points to the adversary.

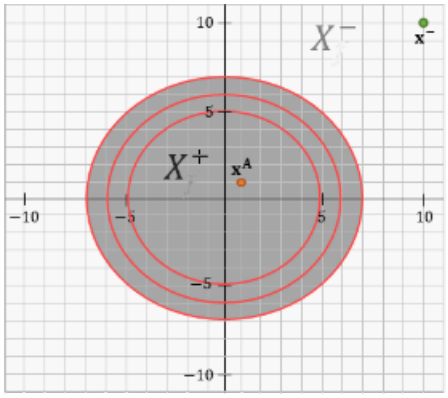


(b) First queried point x from the black box and subsequently revised model. As can be seen from (a), point x was chosen from the region with highest uncertainty (where σ is at its peak).

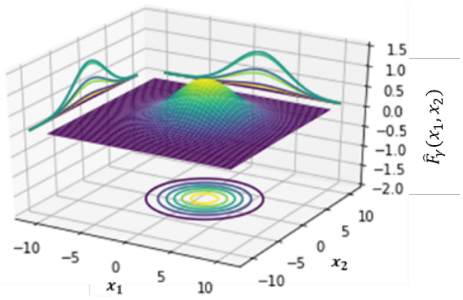


(c) After 14 queries from $f(x)$ of uncertain regions.

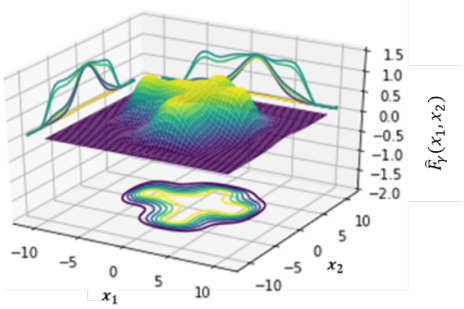
Figure 3.4: The evolution of $\hat{F}_\gamma(x)$ modelled using Algorithm 1 in 1D space.



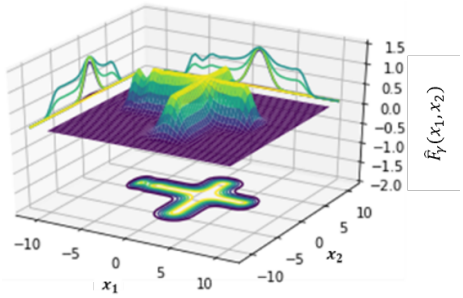
(a) The initial configuration of the black box with the two known points to the adversary.



(b) The initial GPR with the two known points to the adversary.



(c) The modelled GPR after few iterations.



(d) The converged GPR along orthogonal feature dimensions.

Figure 3.5: The evolution of $\hat{f}(x)$ modelled using Algorithm 1 in 2D space.

along unit vectors in the feature space and terminated when the maximum variance of the model becomes insignificant.

Similar to the previous example, a simple walk along the unit feature vectors and querying the GPR model to find an instance $\hat{x}$ for the desired P can reveal $\varepsilon p - IMAC$. It should be noted that Algorithm 1 is approximating $p(F_\gamma(x) = +)$ only along the unit vectors in the feature space for ℓ_1-norm adversarial cost function, and for all other ℓ_p-norms where $p \neq 1$ the search problem is NP-Complete [1].

For completeness, we have compared the accuracy of our method against the one proposed in [1] (refer to Table 3.1). Although the method proposed in [1] was not intended to discover $IMAC$ in randomized convex-inducing classifiers, we still want to demonstrate the level of error an adversary must endure if she decides to ignore the randomness and uses the query strategies described in [1]. Moreover, due to the binomial sampling in Algorithm 1, the overall query counts in our method are higher compared to the method proposed in [1]. It is conclusive that the introduced randomness necessitates significantly more queries to be issued by the adversary. But, since the number of binomial sampling is bounded by a constant, the order of queries issued by our method is just comparatively more by a constant factor.

Table 3.1: Comparison of accuracy of our proposed method against classical ACRE search for non-randomized classifiers.

Method	To Example	P	#Queries	Avg Error relative to MAC
Nelson et al. [1]	1-D	1.0	**34**	-0.6183020
		0.5	N/A	N/A
	2-D	1.0	**57**	-1.5764189
		0.5	N/A	N/A
Active Multiline Search	1-D	1.0	122	**0.0022201**
		0.5	122	**0.0175023**
	2-D	1.0	254	**0.5510204**
		0.5	254	**0.5306122**

3.5 Conclusions

Randomization of decision boundaries is considered to be an effective defensive measure against ACRE attack. In this chapter we presented a novel adversarial approach for probabilistic ACRE attack against randomized convex-inducing classifiers for X_γ^+ set. We have demonstrated that although randomization could result in more queries for finding $\varepsilon p - IMAC$, still adversary can find the optimal evading instance bounded

by polynomial-many queries.

In this work, we have only considered selection function γ with a uniform probability distribution over elements of F_γ. However, in Chapter 4 we are going to study effect of different randomization techniques on the number of queries required to be executed by the adversary in the context of randomized MAC spoofing classifiers.

Chapter 4

A Randomized Moving Target Approach for Detecting MAC-Layer Spoofing Detection

4.1 Introduction

There have been many research works [17, 16, 5] in the past investigating the use of RSSI profiling for the purpose of MAC spoofing detection (many of which are surveyed in Section 2.2). Most of these works assume an adversary that keeps both, his location within the victim network as well as the RSSI of the transmitted/spoofed signal

unchanged. However, one can argue that with the current state of technology and the widespread availability of off-the-shelf hacking tools, the modification/adjustment of the transmitting power by the adversary - in order to evade detection - is not only possible but extremely likely.

In this chapter, we first investigate how difficult it is for an adversary capable of modifying and adjusting the transmitting power of his device to evade detection that is based on some commonly deployed schemes for RSSI profiling of legitimate devices. Next, as a countermeasure against such adversarial transmission power modifications, in this chapter we also present our novel adaptive intrusion detection strategy based on randomization of RSSI-based decision boundaries (a concept previously discussed in Chapter 3). This strategy, as our experiments demonstrate, can greatly improve the defender's chances to detect the adversary's presence and to respond accordingly.

4.2 Problem Setup and Threat Model

In this section, we introduce the main annotation and assumptions of this Chapter. First, let $s_{i|i=1,\ldots,n} \in S$ denotes the set of legitimate senders and $r_{i|i=1,\ldots,n} \in R$ the set of legitimate receivers communicating over a wireless communication channel. Also, we assume that each legitimate receiver utilizes a *blackbox* algorithm to profile legitimate senders (i.e., their respective RSSI characteristics), and use the established

profile at runtime to differentiate between (received) data frames that are legitimate and those that spoofed.

Furthermore, let ρ denote an arbitrary unicast (i.e., point-to-point) communication protocol that implements a feedback mechanism. In other words, for every successful transmission of a sequence of data frames by sender s_i, receiver r_j provides feedback (e.g., r_j transmits an acknowledgment packet back to s_i) according to the protocol specification.

Finally, let α denote the adversary with the following characteristics:

- Adversary is situated at a location from which it can observe/receive signals transmitted by all legitimate senders, as well as the acknowledgments returned by the respective receivers.

- Besides being able to monitor the wireless spectrum, adversary can also read and interpret unencrypted part of captured data frames, including those containing the sender's and receiver's MAC address.

- Adversary is aware of the type/model of each legitimate sender device and their respective transmission power setting (P_{Tx}), which is not a substantial assumption as system information about most IoT/WSN devices are publicly accessible on the Internet. If the information about the type/model of legitimate sender devices is not readily available, the adversary can utilize sophisticated

methods proposed by Maiti et al. [60] to discover device types by observing wireless traffics.

- Adversary is *not* aware of the actual physical/geographic locations of other (legitimate) nodes in the network.

- Network participants, including the adversary, are equipped with common omnidirectional antennas, and are not capable of detecting the positional angle of (other) transmitting nodes. However, the adversary can move-about in order to triangulate other nodes' locations based on the strength of the signal received from those nodes.

- Adversary is capable of adjusting its transmission power.

- Adversary is also capable of altering (i.e., spoofing) its MAC address value – i.e., it can generate data frames that carry MAC addresses of other legitimate nodes from this particular network.

The ultimate goal of adversary α is to impersonate a particular sender $s_i \in S$ by transmitting frames with s_i' spoofed MAC address. The spoofed frame is assumed to be specifically intended for one particular receiver $r_i \in R$, as illustrated in Figure 4.1. To achieve this goal, and ensure that the spoofed frames are actually accepted by r_i as fully authentic, the adversary needs to additionally discover/adjust its transmission

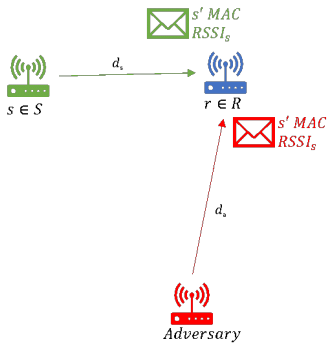


Figure 4.1: Example: the goal of adversary A is to transmit a frame with s' MAC address and signal power that results in RSSI values observed by r comparable to the RSSI values corresponding to the actual s.

power (P_{Tx}) so that the signal 'carrying' the spoofed frames closely matches the RSSI-profile of s_i upon their reception by the victim receiver r_i (i.e., the adversary's P_{Tx} must ensure some desired probability of evasion at r_i). Clearly, the spoofed frames that are sent with adequate levels of P_{Tx} and end up being accepted as 'authentic' will trigger the transmission of an acknowledgment by r_i, while the spoofed frames sent with inadequate levels of P_{Tx} will produce no response by r_i. The problem of how to discover/adjust the adversary's transmission power so as to achieve a certain evasion probability is closely related to the optimal adversarial evasion problem introduced by Nelson et al. [1].

It is important to emphasize that in this work we are going to consider the *number of transmission queries* issued by the adversary while searching for its most

adequate P_{Tx} (we denote this number by Q) as one of the attack objectives. More specifically, we are assuming that in addition to successfully impersonating sender s_i, the adversary's goal is to also try to preserve its stealthiness by minimizing the number of spoofed frames that need to be transmitted while searching for (i.e., trying to match) the RSSI characteristics of sender s_i. Clearly, fewer issued exploratory frames during the signal reconnaissance phase will imply better attack sealthiness. Nonetheless, fewer issued exploratory frames are also likely to result in a less accurate estimate of the required transmission power and, consequently, reduced effectiveness of the spoofing attack following the exploratory phase (i.e., a reduced chance of avoiding detection). Thus, the ultimate goal of the attacker is to find the best tradeoff between the number of issued transmission queries Q (which ideally would be as small as possible) and the probability of finding P_{Tx} that is most effective for evading detection (which ideally would be as high as possible).

4.3 Novel Attack Strategy Against Stationary RSSI-based MAC-Layer Spoofing Detection

4.3.1 Assumed Intrusion Detection System Operation

The spoofing detection method proposed by Sheng et al. [20] is an approach based on the statistical analysis of RSSI values observed by the receiving node. Namely, it is demonstrated in [20] that the signal generated by a stationary wireless node, when measured at a receiving node, results in RSSI values that form a *Gaussian distribution*. Any deviation from this particular RSSI distribution would be a reasonable indicator of a potential spoofing attack.

Building on this fact, in our work we adopt (i.e., assume) the following three-phase procedure for detection of MAC-Layer spoofing based on RSSI measurements by the defending system[2]. (Without loss of generality, we are considering an environment that consists of three actors: the adversary denoted by A, the legitimate receiver denoted by r, and the legitimate sender s.)

[2]Every defending node must be able to perform the discussed three-steps locally and should not be dependent on another service in the network. We acknowledge that one could design a more collaborative/distributed way to detect MAC spoofing, but any extra wireless communication in a wireless sensor or IoT network result in more power consumption that could become unrealistic in power-constrained applications.

1. **Profiling Phase**: The receiving node r begins its operation by collecting samples of RSSI values from a given trusted sending node s denoted by $\{x_i\}$ and fitting these samples into a Gaussian mixture model with computed parameters θ_0. It is assumed that there is no adversarial activity occurring during this phase.

2. **Active Monitoring Phase**[3]: Then, at runtime, the receiving node r continues to collect RSSI samples corresponding to the data-link frames that carry node i's MAC address as the frame's source address, which we annotate $\{\overline{x_i}\}$. Clearly, these frames could be legitimate node s' frames or could be spoofed frames generated by an adversary A. Every n of the collected samples (in a manner of a sliding window) are then fitted into another Gaussian mixture model with computed parameters θ_1.

3. **Decision Phase**: In order to determine whether the signal samples observed during Phase 2 are a cause for concern, the system conducts *Likelihood Ratio Test* by calculating the likelihood ratio of observing $\{\overline{x_i}\}$ under θ_0 versus θ_1(refer

[3]Many works [16, 20] in the literature assume that the adversarial transmission occur while the legitimate transmitter is quiet. The power conservation state of many WSN nodes and IoT devices makes this assumption relevant and compatible with real-world examples (e.g., a smart smoker detector that transmits only when smoke detected.)

to (4.1)). If λ is greater than a set threshold c, then a spoofing attack is detected.

$$\lambda(x) = \frac{\mathcal{L}(\theta_0|\{\overline{x_i}\})}{\mathcal{L}(\theta_1|\{\overline{x_i}\})} \tag{4.1}$$

It should be noted here that the result of the likelihood ratio test will ultimately depend on the nature/distribution of the collected RSSI samples (i.e., $\{\overline{x_i}\}$), which are in turn a function of the transmitter's signal power (P_{Tx} in (4.2)) and the transmitter's distance to the receiving node r.

From the adversary's perspective, to be able to evade detection by a system that deploys the above outlined procedure, the adversary should ensure that the strength of his spoofed signal, when measured by r, produces RSSI samples that closely match the RSSI samples of the legitimate sender s. However, given that the actual location of r and s (i.e., the distance between the two nodes) are not known to the adversary, determining what those RSSI values are (i.e., should be) is a challenging task, in spite of the fact that the adversary is assumed to have readily available knowledge of both - r's and s's P_{Tx}. The exact steps taken by the adversary to determine what the transmission power of his spoofed signal should be - so as to ensure that the RSSI values of the spoofed signal observed by r match the RSSI values of the legitimate signal generated by s - are described in the next subsections.

4.3.2 Novel Attack Strategy

Mathematical Preliminaries

In this subsection, we discuss the physical/mathematical models that help the adversary estimate the required strength of the spoofed signal, and we defined some required notations that are extensively used in the reminder of this chapter.

First, let d_0 denote the distance between r and s, and let d_r and d_s denote the distances of adversary to legitimate sender and receiver respectively. Let P^s_{Rx} and P^A_{Rx} denote the distributions of RSSI values measured by r as a result of node s and the adversary's transmissions, respectively. Let $E[P^s_{Rx}]$ and $E[P^A_{Rx}]$ denote the expected values of the corresponding distributions. (Please observe that throughout the text we use P_{Rx} to denote the received signal strength, and P_{Tx} denote the transmissions power at the source.)

Wireless signals experience reduction in power density (also known as *signal loss* or *attenuation*) as they propagate through space which is expressed as: (4.2)

$$[P_L(d)]dB = 10\log_{10}\frac{P_{Tx}}{1mW} - 10\log_{10}\frac{P_{Rx}}{1mW} \tag{4.2}$$

Wireless signal attenuation can be caused by various factors, including: free-space loss, refraction, diffraction, reflection, etc. [61]. There are different mathematical models, also known as *path loss* models, that can be used to predict the loss of signal

power at a given distance from the signal source. In this work, we use *log distance* path loss model (see (4.3)), since it can predict the propagation loss for a wide range of environments, including environments with obstacles.

$$[P_L(d)]dB = [P_L(d_{ref})]dB + 10\gamma \log 10\frac{d}{d_{ref}} + \chi \tag{4.3}$$

$$[P_L(d_{ref})]dB = -10 \log 10\frac{\lambda^2}{(4\pi)^2 d_{ref}^2} \tag{4.4}$$

In (4.3), γ is the path loss exponent which takes different values for different propagation environments (e.g., $\gamma = 2.2$ for indoor office or $\gamma = 1.7$ for commercial places), and χ is a Gaussian random variable with zero mean representing the signal attenuation caused by slow fading. The actual variance of the slow fading Gaussian distribution χ is also different in different propagation environments (e.g., $\sigma = 7$ offices with hard partitions or $\sigma = 7.9$ for commercial places). $P_L(d_{ref})$ in (4.3) is the path loss experienced at a fix reference distance $d_{ref} = 1$ meter as per *Free Space Loss model* [61]. The expression for $P_L(d_{ref})$ is given in (4.4), where λ is the signal wave-length of the assumed communication channel.

By substituting the left-hand side of (4.3) with (4.2), we obtain

$$10 \log 10\frac{P_{Tx}}{1mW} - 10 \log 10\frac{P_{Rx}}{1mW} = [P_L(d_{ref})]dB + 10\gamma \log 10\frac{d}{d_{ref}}. \tag{4.5}$$

Now, (4.5) can be solved for d, as shown in (4.6). From the practical point of view, d in (4.6) is the distance between the signal source that transmits with power P_{Tx} and the location where the strength of the signal is measured to be P_{Rx}.

$$d = 10^{(\frac{10\log 10\frac{P_{Tx}}{1mW} - 10\log 10\frac{P_{Rx}}{1mW} - P_L(d_{ref}) - \chi}{10\gamma})} \tag{4.6}$$

Alternatively, (4.5) can be solved for P_{Rx}, as shown in (4.7). From the practical point of view, P_{Rx} in (4.7) is the strength of the signal measured at distance d and generated by a source that transmits with power P_{Tx}.

$$[P_{Rx}]mW = 10^{\frac{P_L(d_{ref}) + 10\gamma\log 10\frac{d}{d_{ref}} + \chi - 10\log 10\frac{P_{Tx}}{1mW}}{-10}} \tag{4.7}$$

Attack Strategy

As explained in Section 4.2, in order to conduct a successful MAC-spoofing attack against s, adversary A needs to estimate what distribution of RSSI values the receiving node r registers upon receiving data frames from the legitimate senders s (i.e., A needs to estimate P^s_{Rx}). To achieve this objective, the adversary first must discover the exact coordinates/location of s (the legitimate sender) as well as of r (the legitimate receiver), as depicted in Figure 4.1. Namely, the knowledge of these coordinates will further allow the adversary to estimate the distance between the two nodes, d_0, also

depicted in Figure 4.1. (The steps involved in estimating d_0 are referred to as Attack Reconnaissance Phase in Figure 4.2.) Once estimated, d_0 in combination with the assumed knowledge of s' transmission power, will ultimately allow the adversary to mathematically derive a preliminary estimate of $\widehat{P_{Rx}} \approx E[P^s_{Rx}]$ using (4.7). (In the following paragraph we elaborate on why the initial estimate using 4.7 is preliminary - i.e., is likely to contain a certain error.) Finally, using $\widehat{P_{Rx}}$, the adversary can begin the search for (i.e., the fine tuning of) its own optimal evading transmission power, which we refer to as Active Attack Phase in Figure 4.2.

Now, a few points should be clarified about the assumed attack strategy outlined in Figure 4.2:

Firstly, in our work we assume that during the *attack's reconnaissance phase*, and while located at one specific geographic location, the adversary receives/observes a few data frames from each s and r, which allows the adversary to measure the corresponding RSSI values (i.e., P_{Rx} for original-packet signal sent by s as well as P_{Rx} for signal of the acknowledgement-packet sent by r). Combining these values with the knowledge of P_{Tx} for s and P_{Tx} for r (assumption made in Section 3.2), the adversary is further able to compute its own distance from s and r - d_s and d_r, respectively - simply by using (4.6)[4]. By repeating this process from four distinct

[4]We have assumed that the adversary is mobile and can move around and collect signal strengths from three different locations in order to trilaterate. In the case of static (non-mobile) adversary,

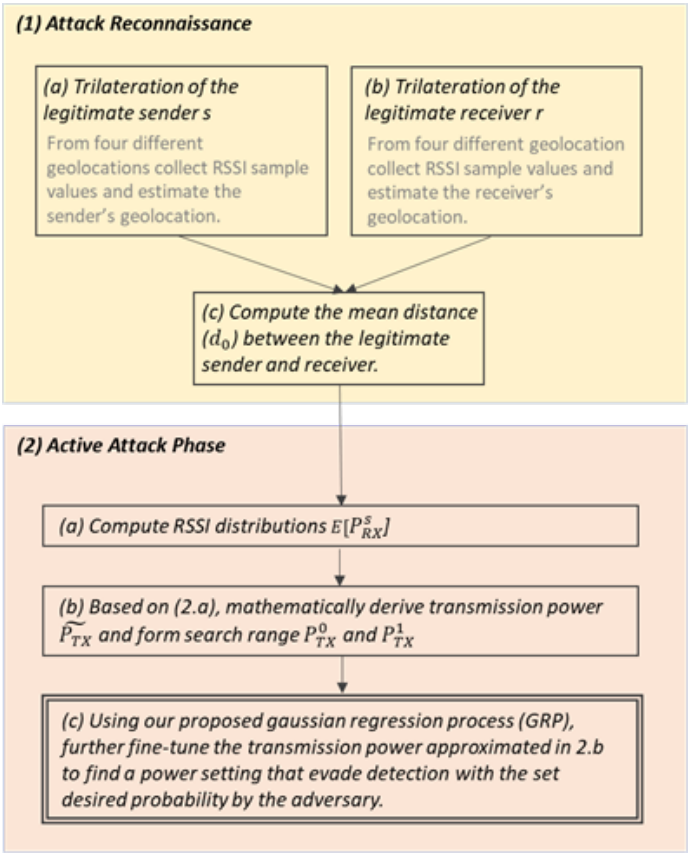


Figure 4.2: Attack steps

geographic locations[5], and deploying the well-known trilateration methods [64, 63, 62], the adversary is ultimately able to estimate the geographic locations of r and s – steps 1.a and 1.b in Figure 4.2.

Secondly, it is important to note that due to the signal variability caused by χ, individual measurements of RSSI taken by the adversary during the attack reconnaissance phase at any of the four locations, are likely to contain certain error relative to the true P_{Tx}, and as such are likely to cause an error in the respective d as calculated using (4.6). One way to account for the signal variability caused by χ is to take multiple measurements of RSSI, and then consider the mean of all RSSI measurements as the actual P_{Rx}. And, of course, one way to attempt to improve the accuracy of the finally computed P_{Rx} (as well as the accuracy of the distance d computed from $\hat{p}_{Rx}$) is dependent on the number of collected samples. Unfortunately, in many real-world applications, IoT/WSN devices are set/programmed to make only occasional transmission of small amounts of data. Hence, when dealing with such devices, it may be very challenging if impossible for the adversary to collect a large number of signal (i.e., RSSI) samples in a limited interval of time. Consequently,

without loss of generality, one could compensate the non-mobile scenario with multiple static adversary nodes in the environment that collaborate together to achieve trilateration.

[5]We are considering a 3D scenario, and therefore, we are required to have four distinct measurements from different locations.

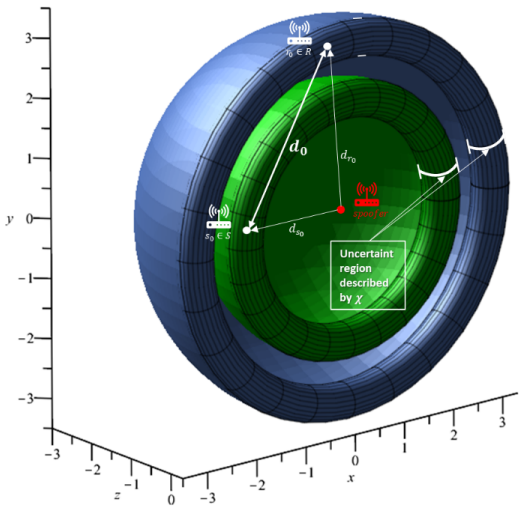


Figure 4.3: Computing d_0 requires exact coordinates of s_0 and r_0

due to a limited number of measured signal (i.e., RSSI) samples, the adversary's estimates of both d_s and d_r, at each of the four trilateration locations of the attack's reconnaissance phase, are likely to carry a degree of inaccuracy. To illustrate the above point, let us assume a typical WSN and IoT utilizing a simple *Isotropic Antenna*. By measuring the RSSI values corresponding to this node, and then deploying (4.6), the adversary can determine its distance to the given node, but not the node's exact geolocation. In other words, the calculated value of d using (4.6) only implies that the given node is situated somewhere on a surface of a sphere centered at adversary's location and with the radius of d, as illustrated in Figure 4.3. Since log-normal path loss model uses Gaussian distribution to model signal fading, the variance of the distribution describes the uncertainty of the calculated radius.

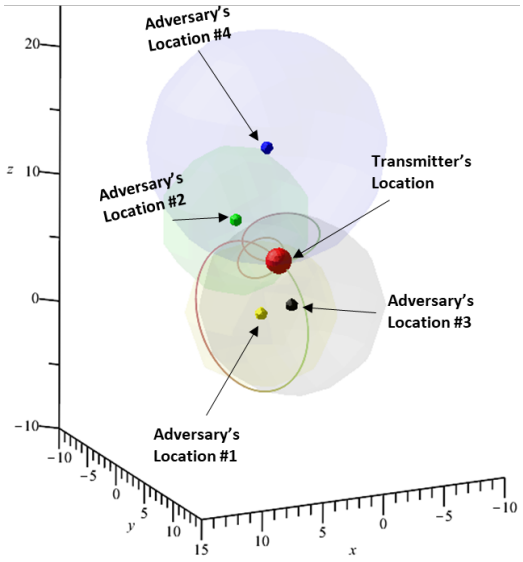


Figure 4.4: Using three different sniffing locations to find directional vector towards the transmitting node.

Thirdly, once the adversary estimates the distance between the two legitimate nodes d_0 (see Figure 4.3), he can further utilize (4.7) to come up with a mathematical estimate of the strength of s' signal when measured by r - $\widehat{P_{Rx}} \approx E[P^s_{Rx}]$. And then, by rearranging (4.7) and deploying the previously determined d_r, the adversary can also *derive* its own transmission power (denoted by $\widetilde{P^A_{Tx}}$) which can ensure that the adversary's signal, when measured by the receiver r, satisfies: $E[P^s_{Rx}] \approx E[P_{Rx}]$.

Due to the uncertainties that exist in the trilateration step (also depicted in Figure 4.5) and unknown unique obstacles between r and s, the derived transmission power $\widetilde{P_{Tx}}$ by the adversary can result in $|E[P^s_{Rx}] - E[P^A_{Rx}]| \gg \epsilon$ when measured by

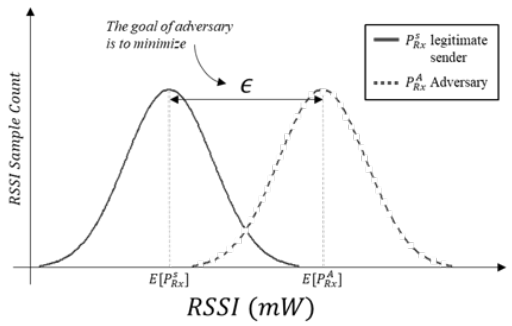


Figure 4.5: Large ϵ result in failure of attack by reducing the probability of evasion. The adversary reduces ϵ by investigating different $\widetilde{P_{Tx}}$ – where this investigation requires more query and probing by the adversary.

the receiver r. Note that even in the rare case of P^A_{Tx} being very accurately estimated, the obstacles between adversary and r could additionally (and very likely) cause $E[P^A_{Rx}]$ to deviate from the intended value - $E[P^s_{Rx}]$. Therefore, the adversary must explore a range of transmission powers centered at the estimated $\widetilde{P_{Tx}}$ in order to find a transmission power setting that evades detection with the probability of ϑ (which we refer to as *desired evasion probability*, and consider to be chosen by the adversary) when the receiving node r performs likelihood ratio test.

Let $P^0_{Tx} = \widetilde{P_{Tx}} - (3 \cdot \sigma[\chi])$ be the minimum and $P^1_{Tx} = \widetilde{P_{Tx}} + (3 \cdot \sigma[\chi])$ be the maximum transmission power settings range the adversary needs to search to find the appropriate evading transmission power. ($\sigma[\chi]$ is the standard deviation used in log *normal pathloss* model.) Clearly, different transmission power settings utilized

by the adversary in this range will result in different probability of evasion, and the goal of the adversary is to explore the given range so as find the transmission power setting that results in a desired evasion probability (ϑ).

A simple brute force method in which many different power settings between P^0_{Tx} and P^1_{Tx} (i.e., the search range) are tried and the respective evasion probability is computed could help the adversary to find the optimal power setting for evasion. But a large search range would likely result in a large volume of failing queries which in turn could trigger the defense mechanisms implemented by the defender. Thus, using a minimum number of queries for the discovery of the desired evading transmission power is more advantageous to the adversary.

In Chapter 3, we have proposed a framework to help the adversary with a general search scenarios that require the discovery of evasion probabilities in a range of feature values using a minimum number of queries and assisted by data interpolation. Here, we propose the use of the same technique, this time assuming the adversary's transmission power to be the feature/search value. The adversary should begin by finding the probability of evasion of the two initial (i.e., the minimum and the maximum) power settings P^0_{Tx} and P^1_{Tx} - by issuing at least n data frames with each of the transmission powers and counting the number of successful feedbacks from r. (Recall: feedback generated by r is indicative that the transmitted frame

is accepted as genuine, and n is dependent of the application protocol.) Then, the adversary would form a *Gaussian Regression Process* (GRP) [65] with the two initial power settings (refer to Figure 4.6 (a)) as values for the independent variable and their corresponding ratio of successful evasion as values for the dependent variable. Based on these values the GRP model can interpolate the probability of evasion for all the power settings in the search range and the uncertainty of the model for this interpolation will also be accessible to the adversary. Given that the goal of adversary is to reduce its uncertainty about this search range, adversary would select a transmission power setting from the interpolation list with highest uncertainty and submit at least n data frame using this specific transmission power to find its true evasion probability and update the GRP with this new found information (refer to Figure 4.6 (b)). The adversary would continue its exploration (selecting new power settings in the search space) and exploitation (issuing more queries to reduce its uncertainties about previously queried points) of the search space until it has confidently found the most probable evading transmission power setting (refer to Figure 4.6 (c)). (Here, we assume the most probable evading transmission is found when the variance across the GRP model domain values falls below a threshold value set by the adversary - refer to [66].)

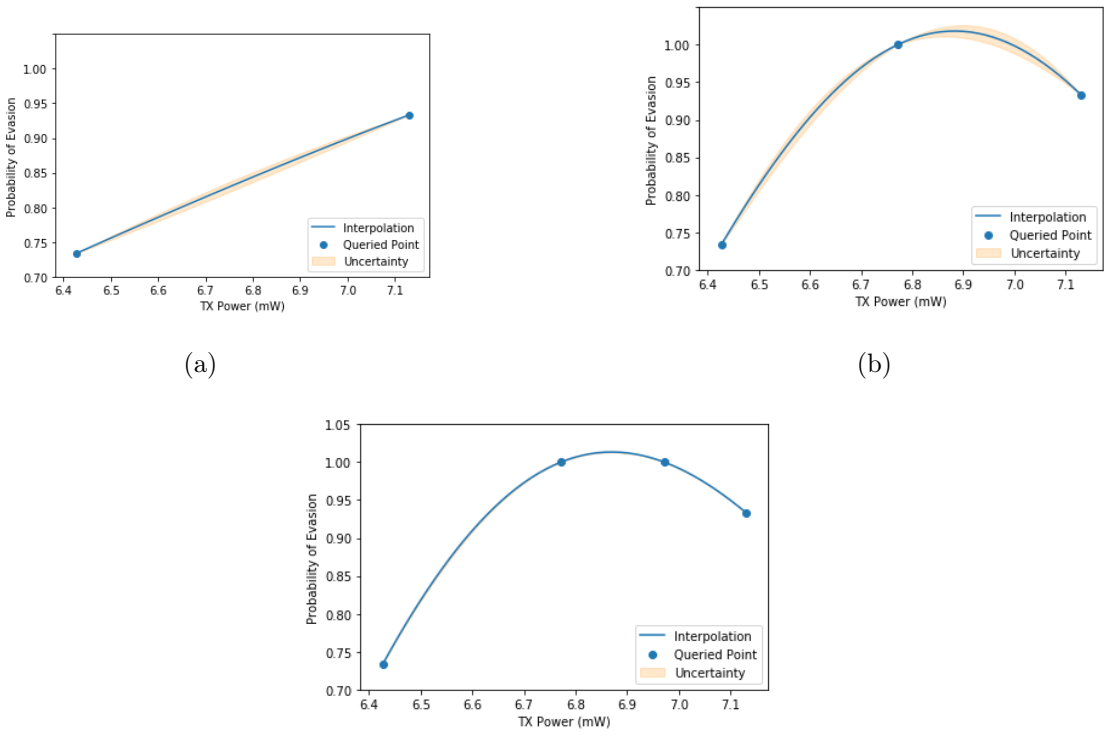


Figure 4.6: (a) Initial GPR interpolation using the initial minimum and maximum power settings. (b) First iteration, querying the most uncertain point along the interpolated graph. (c) Second iteration, querying the second most uncertain point.

4.4 Overview of Randomized Defense Strategy

A randomized defense strategy, also known as moving target defense, is a strategy that stochastically reacts to inputs in order to impede the discovery of some important information and/or parameter(s) of the target system. In the MAC-spoofing detection system discussed in Section 4.3, the adversary's main objective is to discover the RSSI-related decision parameters that assist the defending system in distinguishing between the genuine and the spoofed/attacker's signal. Also, in Section 4.3, these critical RSSI-related decision parameters of the defense system are assumed to be static (i.e., not subject to deliberate change). In this section, by taking the opposite perspective from the one in Section 4.3 (i.e., by placing us on the defender's side), we explore the idea of randomized change of RSSI-related decision parameter as a possible approach to increasing the robustness of the assumed detection system against MAC-spoofing attacks described in Section 4.3. In particular, we propose the use of multiple RSSI-related decision boundaries (i.e., intervals), with one particular interval being stochastically selected and deployed by the system at any given time. The specific contributions of our novel method include: (a) an adaptive parametric approach for the creation of changing different-sized acceptable-RSSI-value intervals , and (b) an adaptive parametric approach for probabilistic selection of individual acceptable-RSSI-value intervals.

4.4.1 Interval Creation and Spacing

Many studies, including [16], [5] and [20], have shown that RSSI values corresponding to a single stationary transmitter s, and recorded at a single stationary receiver r - in both IEEE 802.11 and 802.15.4 - form a Gaussian distributions, as also depicted in Figure 4.7. Namely, after collecting and processing RSSI samples corresponding to such a scenario, it is possible to identify a minimum and a maximum RSSI value pair (i.e., the statistical range of recorded RSSI values), a mean, and a standard deviation that would describe the given distribution with sufficient accuracy.

Our randomized defense strategy begins by creating a set of encompassing acceptable-RSSI-values intervals around the mean of the identified Gaussian distribution. In particular, let x_{max} and x_{min} denote the maximum and minimum RSSI values of the identified Gaussian mixture model corresponding to node s (i.e., its MAC address), then we use

$$I = \Bigg\{ [-\frac{tanh(\frac{i}{n.r})(x_{max} - x_{min})}{tan(\frac{1}{r})} + x_{min}, \frac{tanh(\frac{i}{n.r})(x_{max} - x_{min})}{tan(\frac{1}{r})} + x_{min}] \Bigg| i \in Z, i = -n, \ldots, -1, 1, \ldots, n \Bigg\}, \tag{4.8}$$

to define an ordered set of n encompassing acceptable-RSSI-value intervals associated with node s, where $r \in [0 - 1]$ allows us to controls/adjust how intervals are spaced.

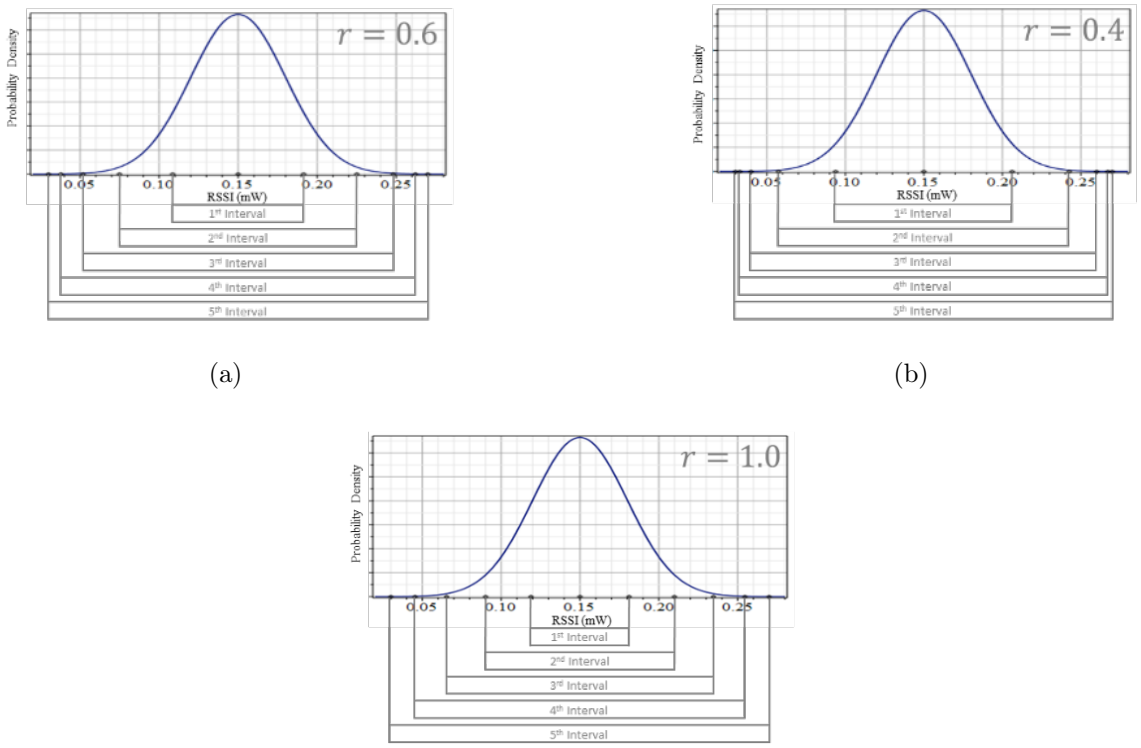


Figure 4.7: Normal distribution of RSSI values: (a) 5 intervals with $r = 0.6$, (b) 5 intervals with $r = 0.4$, (c) 5 intervals with $r = 1.0$.

Once the n intervals are created, a randomization function (described next in Section 4.4.2) is further deployed to select one of the intervals to make decisions about all new signals (i.e., frames) that appear to be coming from node s. (Recall, in reality, these could be genuine frames generated by node s, or spoofed frames generated by adversary A.) The decision about these frames could be as naïve as simply checking if their respective RSSI values fall within the selected decision interval. Or, as a more involved approach, the decision could be based on the procedure previously described in Section 4.3: 1) build a Gaussian mixture model of the originally observed (i.e., training) RSSI values that fall within the given interval, 2) build another Gaussian mixture model of the most recently received RSSI values, and 3) perform the Likelihood Ratio Test using these two distributions.

4.4.2 Randomization Function Selection

Selecting randomly from the decision intervals defined in Section 4.4.1 is at the core of our randomized defense strategy. As previously indicated, each of the decision intervals implies different pros and cons for the system. For example, by selecting the largest interval - the probability of false negative is likely to be very high. (Recall, *false negative* refers to be the case when a malicious signal/frame is flagged as genuine.) In contrast, by selecting the smallest interval around the mean, the probability of false

positive (i.e., probability that genuine data points be falsely flagged as malicious) is likely to be very high. Now, in some systems, the goal may be to balance these two probabilities, which potentially could be achieved by deploying a uniform probability of interval selection. However, there are many systems in which one of the two probabilities (false positive vs. false negative) could be more important than the other. In order to allow the end users to choose and adjust the probability of interval selection to the specific needs of their own systems, we introduce the following parametrized probability function. Namely, provided $i \in I$ denotes the index of a decision interval, Equation 4.9 represent the parametrized beta-binomial distribution that quantifies the probability of (the i^{th}) interval actually being selected to classify an incoming data frame.

$$p(i) = \binom{n}{i} \frac{Beta(i+\alpha, n-i+\beta)}{Beta(\alpha, \beta)} \tag{4.9}$$

In (4.9), α and β denotes the shape parameters of the beta-binomial distribution and are used to manage the degree of accuracy and robustness of our system. For example, $\alpha = \beta = 1$ would create uniform probability selection over all the intervals. On the other hand, $\alpha \geq 1$ and $\beta < 1$ would places larger probability of selection over bigger intervals while $\alpha < 1$ and $\beta \geq 1$ would do the opposite - refer to Figure 4.8.

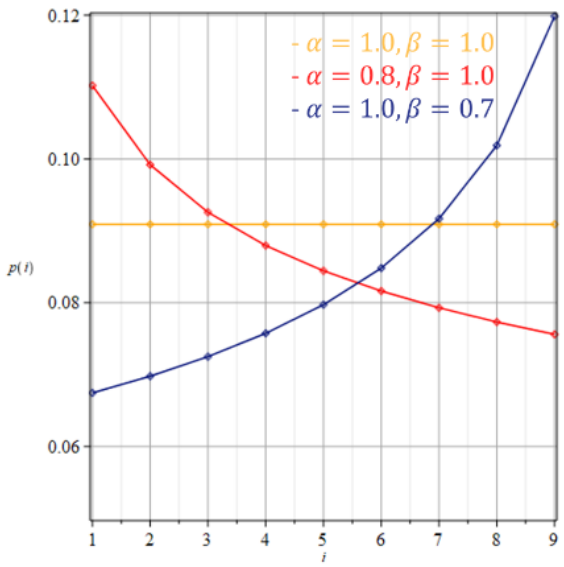


Figure 4.8: Probability of a given interval (among 9) getting selected with different parameterization Beta-binomial distribution.

4.5 Experiment

The experiments described in this section are conducted using *Omnet++ simulation software*[6][67], and are intended to demonstrate *proof of concept* as well as study the effect of different parameters of our proposed model in isolation. There are two main motivating reasons for using *Omnet++* simulation software:

1. Provides a comprehensive library for describing the experimental environment (e.g., exterior walls) that could affect the received signal strength,

2. Also, comes with pre-built libraries for supporting IEEE 802.14.5 protocol that is mainly used in IoT and WSN networks.

4.5.1 Simulation Setup

The experiment assumes two legitimate nodes $s \in S$ and $r \in R$ that communicate on a regular basis and an adversary that aims to impersonate s by generating spoofed frames with s's MAC address. During the first hour of the simulated time, s transmits a data frame to r every $T = 1$ [seconds]. Whenever a frame is successfully received by r an acknowledgement is sent back to s. Also, during this first simulation hour, all RSSI values recorded by r (as a result of successful data frame transmission by s)

[6]Relevant project configurations and source files are hosted on https://github.com/pooria121/RSSIRandomizedDefence

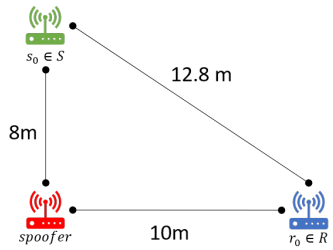


Figure 4.9: The simulation's initial node positions

are used for training two different MAC spoofing detection models – one static/non-randomized (as per current state-of-the-art in the field described in [20]) and one randomized (as per our solution described in Section 4.4 of this chapter).

During the same first hour of simulation - in order to estimate the distance between the two legitimate nodes as discussed in Section 4.3 - the adversary moves to 4 different location and (passively) collects as many RSSI values as it can. Once the first hour of simulation is elapsed, by using the collected RSSI values (as explained in Section 4.3.2), the adversary calculates the range of its own transmission-power settings which will be explored/deployed during the active phase of the attack.

All three nodes (s, r, A) are equipped with IEEE 802.11 WLAN card and an omnidirectional antenna and the legitimate nodes' transmit power is set to $15mW$ communicating in 2.4 GHz bandwidth. The pathloss is modeled using Log-distance path loss with $\gamma = 2.2$ and $\sigma = 3$ to emulate an office space environment.

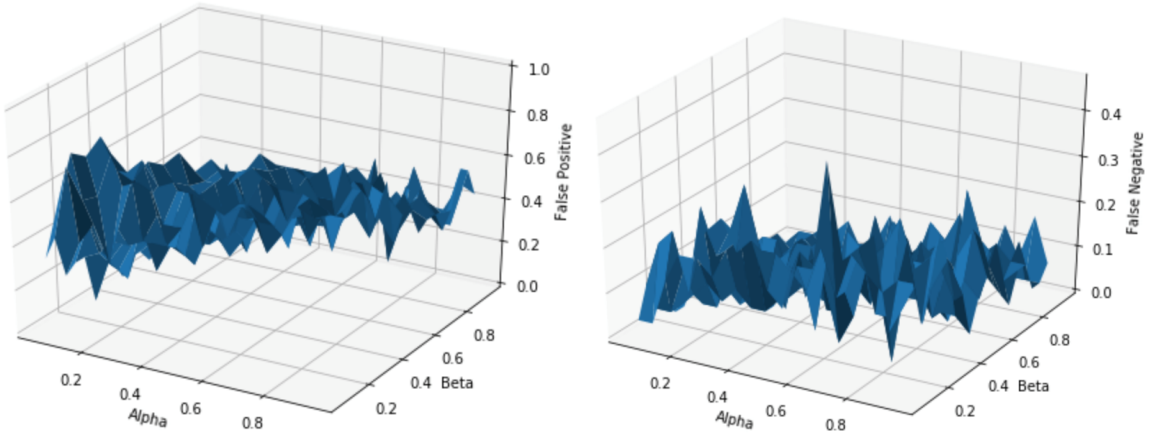
4.5.2 MAC Spoofing Detection Engines

We implemented the likelihood-ratio test proposed by Sheng et al. [20] as the baseline for the non-randomized statistical approach to detection of spoofed frames. On the other hand, the implemented randomized strategy was based on our solution proposed in Section 4.4 with 10 encompassing intervals. (Note, at runtime, the same approach as proposed by Sheng et al. [20] is applied using the training data points that fall within the selected interval.)

As part of the experiment, we have studied the effect of different parameter settings in the beta-binomial distribution. Our obtained results are depicted in Figure 4.10. As a general observation, as we decreased α the false-positive rate increased. This in part is due to the fact that decreasing α would increases the probability of selecting smaller (closer to the mean) detection intervals (refer to Figure 4.7), which in turn makes the defense mechanism more aggressive in rejecting suspicious instances. In our experiment we have set $\alpha = 1$ and $\beta = 0.5$ to achieve false positive rate of about 10%.

4.5.3 Results

After setting up the aforementioned simulation environment, we compared the original likelihood-ratio-test based MAC spoofing detection proposed by Sheng et al. [20] with



(a) (b)

Figure 4.10: The values of α and β in Beta-binomial distribution and their effects on the rate of false positive (a) and false negative (b) in our experimental model.

the same model complemented by our randomized RSSI decision-boundary strategy. In our experiment, we have considered two main criteria for the comparison of the two evaluated approaches:

- Criteria A: the adversary's success of achieving the desired evasion probability assuming he is willing to transmit/issue as few queries as possible. (Note, a lower adversary's success would be an indication of more effective system defense.)

- Criteria B: the number of queries the adversary is required to transmit/issue in order to attain some desired probability of evasion. (Note, a higher number of

required queries would be an indication of more effective system defense.)

As depicted in Figure 4.11, the adversary is capable of discovering transmission power settings that attain the desired evasion probability with a great accuracy when the detection model is not randomized – the line depicting this fact is almost linear with very small standard deviation. Also, as shown in Figure 4.12, the number of queries that needs to be issued by the adversary in order to discover the required transmission power remains almost unchanged irrespective of the level of desired evasion probability.

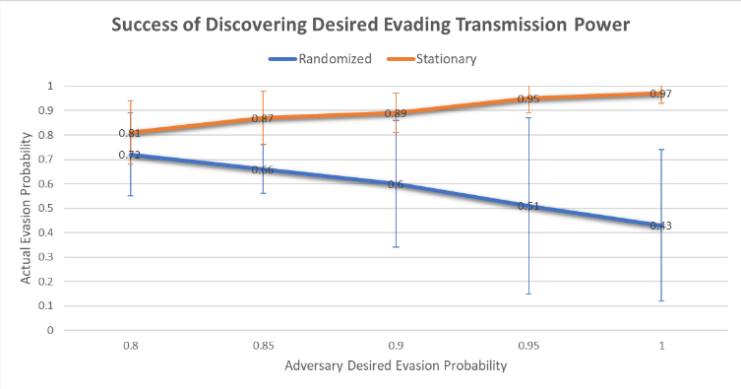


Figure 4.11: Desired adversarial evasion probability vs achieved evasion probability.

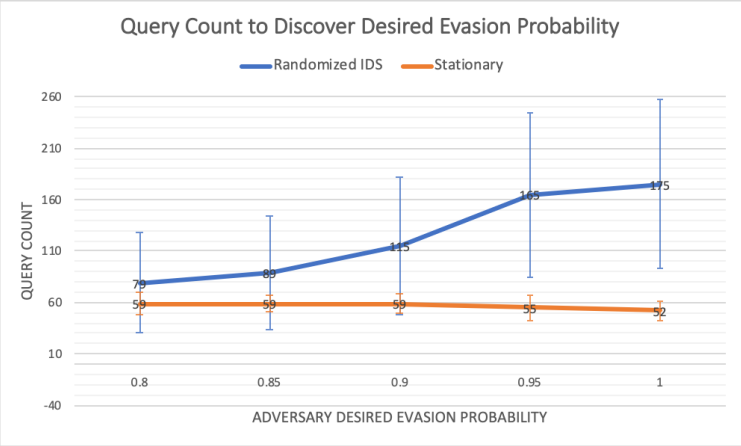


Figure 4.12: The number of queries required to discover the transmission power setting which achieves a desired evasion probability .

On the other hand, our novel randomized strategy is inducing great uncertainty to the adversary's search accuracy – observe the graphed standard deviations in Figure 4.11. Moreover, the adversary's overall search accuracy diminishes as the desired evasion probability increases. In other words, in the randomized defense system, adversary's attempts to achieve higher evasion probabilities result in higher overall uncertainty and progressively lower average of the actual evasion probability. Moreover, the superior performance of the randomized defence strategy is additionally demonstrated by the fact that in order to achieve a higher desired evasion probability the adversary needs to increase the number of submitted/transmitted queries, as illustrated in Figure 4.12.

It should be noted that our randomization scheme requires multiple versions of a detection model[7] to be trained/deployed. Consequently, the overall construction time complexity of our proposed model increases by a factor equal to the number of intervals chosen by the defender. At run-time (i.e., during the models actual deployment), however, the complexity of the classification using our randomization scheme is comparable to that of the stationary scheme given that the time and computational overhead of selecting a different acceptable-RSSI-value intervals is rather negligible.

4.6 Real-world Implementation Considerations and Concept Solutions

In this section, we are presenting some conceptual recommendations (depicted in Figure 4.13) for more successfully deployment of our proposed solution in the real world. In particular, we propose the use of an adversarial generative process to emulate different system conditions (e.g., generative model of adversarial behaviours). Namely, as discussed in detail in previous sections, active adversaries may intelligently modify their transmission power in order to evade detection. Thus, ideally, such

[7]Using training data points that fall within each decision interval.

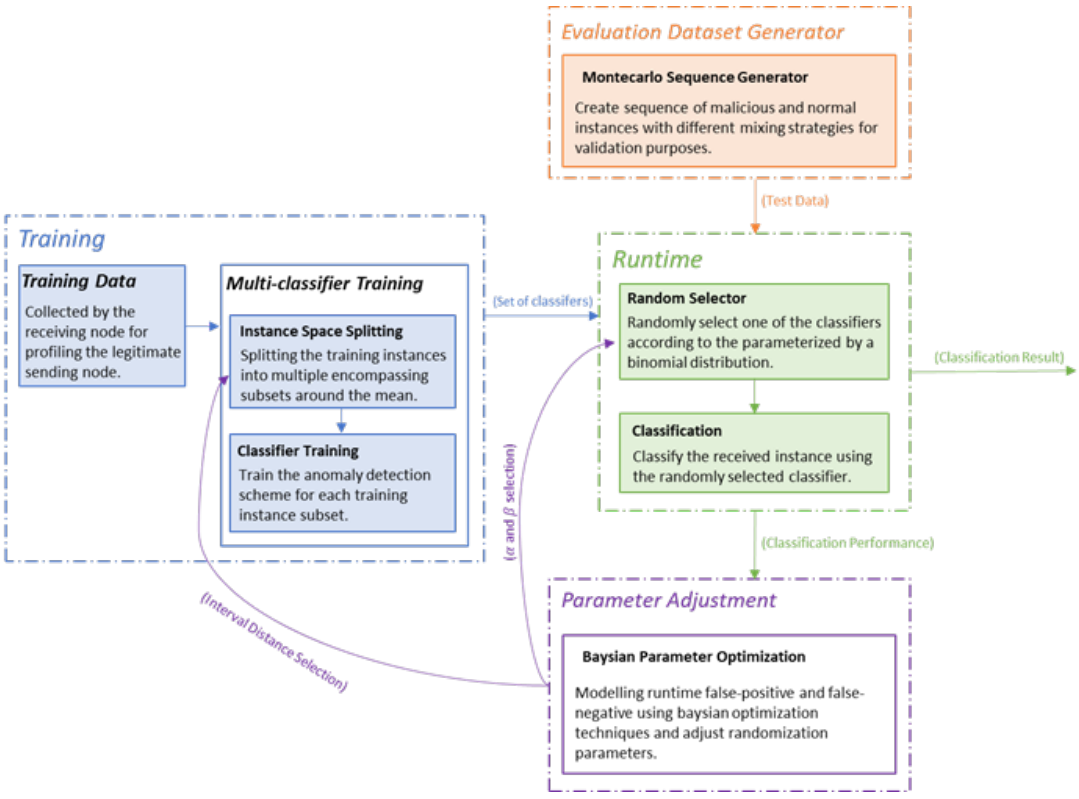


Figure 4.13: Overview of parameter optimization process for fine-tunning the proposed randomization defense.

exemplary behavioural data points would be part of the testing/tuning dataset in order to ensure a more robust performance of the proposed defense system.

By using the training data gathered from the legitimate sender, the defender can compute the mean and variance of the normal distribution representing the legitimate sender's RSSI profile. At the same time, generating unseen adversarial samples in order to fine-tune system parameters is critical. Generating adversarial samples is

(a) (b)

Figure 4.14: Two special cases of adversarial sample generation: (a) Adversarial and legitimate transmissions overlap, (b) Adversarial transmission occur right after legitimate transmissions concluded. (Adversarial transmissions are denoted by strip pattern background.)

accomplished by drawing RSSI values from Gaussian distributions whose means and variances are different from those of the legitimate sender's RSSI profile. By varying the closeness of these means to the legitimate sender's profile would allow us to *emulate* different attackers, with different aggressiveness capabilities.

Due to the nature of the windowing mechanism used by the classifiers, there could be situations where, as depicted in Figure 4.14, the adversary(s) and the legitimate sender could transmit in overlapping or close time intervals. Such corner cases could result in False-Positive or False-Negative classification errors. Depending on the operational conditions the system operator might choose to create and include such overlapping sequences in the evaluation dataset generation to further test the robustness of the defending classifier.

4.7 Conclusions and Discussions

In this chapter we have studied the problem of identity spoofing from the perspective of the adversary as well as from the perspective of the defense system. On the adversary's side, we have proposed an advanced technique which allows the adversary to discover his optimal transmission-power that maximizes his probability of evading detection. As a countermeasure against such an attack, on the defense side we have proposed and evaluated a novel strategy that combines the principles of Randomized Moving Target Defense (RMTD) with RSSI-based MAC spoofing detection (i.e., signal-level device fingerprinting).

The experimental results have shown that our newly propose RMTD-enhanced technique offers far superior performance over simple RSSI-based MAC spoofing detection approaches previously studied in the literature. It is also worth mentioning that the adaptability and modular design make our technique deployable in a wide range of wireless systems, though it is particularly suitable for sensor-based IoT systems.

The adaptability and modular design of our approach allow that our randomization strategy gets adopted by many different RSSI-based MAC spoofing detection systems as we make minimum number of assumptions about the implementation details of detectors. We have successfully demonstrated the extra burden/challenge imposed

on the adversary (query count and actual evasion probability) as a result of the randomized strategy.

Our parameterized approach to randomization is a way to achieve a better control over detection performance and robustness of the detection engines under adversarial search attack. However, to strike a balance between the detection performance and robustness, it is necessary to fine tune the system parameters. In this work we assumed that adversaries are equipped with a special hardware with tunable transmission power settings in order to discover optimal evading transmission power. However, adversaries can attain similar results by being able to move about the perimeters in a controlled fashion using a robotic mobile platform (i.e., drone and/or wheeled platforms) and emulate the effect of variable power settings.

Chapter 5

Robustness of Deep Autoencoders in Intrusion Detection Under Adversarial Contamination

5.1 Motivation

In Chapter 4 we investigated the robustness of the state-of-the-art RSSI-based MAC Spoofing Detection (a form of intrusion detection) systems against exploratory evasion attacks. Moreover, we proposed a novel randomized scheme to further enhance the robustness of such systems/models. However, the assumed convexity of the decision

boundaries of the proposed models (surveyed in Chapter 2 and discussed in Chapter 4) is one of the major contributing factors for potential failing of these systems in some real world scenarios (where the assumptions of convexity may not necessarily hold) to the success of discussed attacks. As such, in order to deal with this problem and build a more robust anomaly-based IDS, in this chapter we propose and examine the of non-convex classifier based on deep autoencoders. After the theoretical discussion of this chapter, in Chapter 6, we examine the use of the proposed autoencoder-based solution in the context of MAC Spoofing detection.

5.2 Introduction

Deep neural networks, thanks to their use of cascaded layers of nonlinear processing units, have shown promising results in many hard computational tasks and have revolutionized many research fields such as computer vision. Recently, the IDS research community has started to adopt deep learning models in the construction of anomaly-based IDSs. (Many of these works have been surveyed in Chapter 2.) But, the use of deep learning models in an adversarial setting with potentially contaminated (i.e., some degree of mislabeled training instances) training datasets requires further investigation before being adopted in cybersecurity applications.

Due to lack of stationarity in the world of cybersecurity (e.g., frequent changes in

normal and malicious behaviors), strategies adopted by IDSs (including the strategies of the MAC Spoofing Detection system discussed in the earlier chapters) should ideally incorporate frequent updates in order to compensate for these continual changes. In other words, in order to successfully deal with the non-stationary conditions of their environment, modern-day IDSs should be inherently adaptable/adaptive[8]. However, by incorporating adaptivity into their operation, IDSs run a risk of potentially becoming a target of the so-called *adversarial attacks* aiming to poison their modeling and/or decision process [33]. (The concepts of adversarial learning and poisoning have been introduced in Section 2.4.) For that very reason, achieving robustness under (potential) adversarial contamination is one of the most desired properties of adaptive IDSs.

In this chapter, we have investigated the viability of using deep autoencoders (a subcategory of deep neural network models) for the construction of a *generic* anomaly-based IDS. In particular, we have proposed a novel framework for testing the robustness of adaptive anomaly-based IDSs under adversarial contamination (contaminated training dataset using mislabeled training instances), and have studied their viability under the adversarial influence. Moreover, we have compared the

[8]Systems that allow the user to change certain system parameters and adapt their behavior accordingly are called adaptable. Systems that adapt to the users automatically based on the system's assumptions about user needs are called adaptive.

performance of our proposed deep autoencoder-based IDS to that of an IDS based on the other well-known subspace analysis method - Principle Component Analysis (PCA) - in terms of their ability to capture the normality of the input data as well as to detect their anomalies.

5.3 Background and Related Works

Intrusion detection systems are classified into two distinct categories based on the employed detection techniques: (1) signature-based IDSs, and (2) anomaly-based IDSs. Signature- based IDSs identify attacks by looking for *known* malicious patterns in incoming data, also referred to as signatures. However, new attacks, for which there are no identified signatures, can evade such detection systems. In contrast to signature-based, anomaly-based IDSs are capable of capturing unknown malicious activities by measuring the degree of deviation of incoming data streams from what is considered to be the *normal data profile*. There are many machine learning approaches [69, 68, 70] that are developed to detect outliers or anomalies in data and as such can be deployed in anomaly-based IDSs. Unfortunately, in intrusion detection systems, due to the inherent absence of stationarity, non-adaptive anomaly-based techniques often result in high rate of *false alarm* (i.e., false positive) during runtime. As a result, these techniques have not been widely used in practice [71].

Shyu *et al.* [72] were one of the first adopters of Principle Component Analysis (PCA) within the domain of anomaly-based IDSs. They have assumed that anomalies are qualitatively different from the normal instances and this difference can be detected by measuring the deviation from the established *normal* (i.e., non-malicious) datasets. Their proposed Principle Component Classifier (PCC) consists of two functions of principle component scores: (a) major components $\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i}$, and (b) minor components $\sum_{i=p-q+1}^{p} \frac{y_i^2}{\lambda_i}$, where λ_i is sample variance of feature i in the dataset and the top q principle components form the major components. Due to the simplicity of PCA, many anomaly detection-based IDSs have adopted a variation of PCC in detecting malicious activities. However, PCC in its original form suffers from lack of robustness against adversarial noise in the training dataset. In other words, the training data must be free of any noise which in many real-world situation is an unrealistic assumption. Ringberg *et al.* [73] have studied the lack of robustness in PCA models for detecting anomalous traffic in IP networks. Specifically, they have shown that the false alarm rate is very sensitive to small differences in the number of principle components, and furthermore a relatively small number of anomalies in the training data can easily pollute the normal subspace.

Rubinstein *et al.* [74] have demonstrated an attack against a typical adaptive PCA-based IDS by injecting malicious points and perturbing the principle components

gradually. Furthermore, they have developed ANTIDOTE [75] as a framework for testing and enhancing the robustness of anomaly detection techniques in network-based (i.e., flow-based) intrusion detectors. It should be noted that our work differs from [75] in terms of the nature of the attack initiated against the model. In ANTIDOTE, the PCA engine is being tested against network traffic flow-based attacks (e.g., DoS) whereas in our proposed framework we study the robustness of anomaly detectors against training data contamination (i.e., classification problem when training data label is partially influenced by the adversary).

Xiao *et al.* [39] have studied the robustness of another machine learning model - Support Vector Machine (SVM) - against adversarial training data contamination. In their work, they theoretically analyzed and proposed an algorithm that can manipulate a set of labels to maximize the empirical loss in accuracy of the original classifier on a trained dataset. Although our work is inspired by the work of Xiao *et al.* [39], the difference comes from the fact that our proposed simulation is closer to an actual deployment scenario of a real-world adaptive IDS.

Niyaz *et al.* [76] have used deep autoencoder for feature extraction and feature reduction before using a feedforward neural network to perform the classification. Our approach is significantly different from that of [76] as we utilize the reconstruction error of the autoencoder as the measure of input data anomaly.

Hawkins *et al.* [77] used reconstruction error of the trained autoencoder to detect outliers which provided the basis for our research. Although, Hawkins *et al.* [77] used this method for outlier detection, the sensitivity of their method under adversarial setting (e.g., intrusion detection system) has not been studied extensively. Moreover, their technique was used to detect generic outliers outside of the context of building an adaptive IDS.

5.4 Approach

In Section 2.3.1 we have introduced and discussed the anatomy and details of deep autoencoders. In this section, we are going to elaborate more on the use of deep autoencoders for the purpose of anomaly-based intrusion detection.

5.4.1 Anomaly Detection using Autoencoders

Functionally, a deep autoencoder consists of two parts: an encoder function $h = f(x)$ that encodes the input onto the representation layer h (refer to Figure 5.1), and a decoder function $r = g(h)$ that reconstruct the coded representation h back into the original input. The main goal of this architecture is to learn an approximation of $g(f(x)) \approx x$.

The learning process in autoencoders is driven by minimization of a loss function

$$L(x, g(f(x))), \tag{5.1}$$

where L(e.g., mean squared error) is intended to penalize the dissimilarity between $g(f(x))$ and x. When the activation functions used in the decoder layers are linear and L is the mean squared error, the autoencoder learns to span the same subspace as Principle Component Analysis (PCA). Autoencoders with nonlinear encoding function f and nonlinear decoding function g can learn a more powerful nonlinear generalization of PCA [23].

Although deep autoencoders are mostly used for feature learning and dimensionality reduction, we are going to use its reconstruction loss (i.e., error) as a measure of anomaly in our IDS system. The underlying assumption (which is verified by our experiments) is that a deep autoencoder trained on normal (i.e., non-malicious) dataset will have a large reconstruction error when is used to reconstruct non-normal (i.e., attack) data samples. This is due to the fact that latent variables (such as environmental factors, learned by the representation h of the deep autoencoder) that exist in non-malicious data points are substantially different from those of malicious data points. (Note, in this work, we consider an autoencoder for which mean squared error loss function L, as given in (5.2), is used for both training and anomaly detection purposes.)

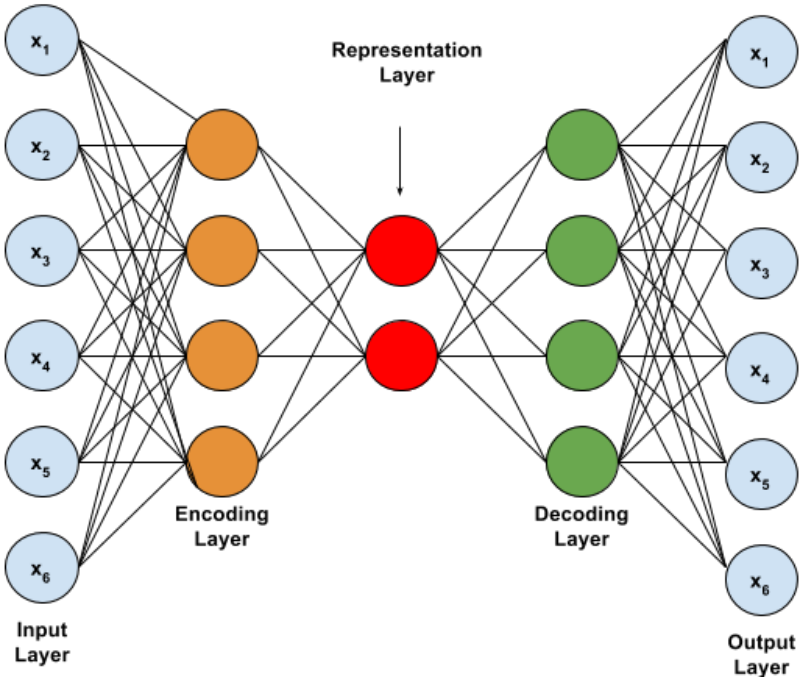


Figure 5.1: Anatomy of a generic autoencoder.

$$L(x, g(f(x))) = \frac{1}{n}\sum(x_i - g(f(x_i)))^2 \tag{5.2}$$

$$\alpha = P(L(x, g(f(x))) > C \mid \text{x is normal instance}) \tag{5.3}$$

In order to be able to approximate a general function, perfect (i.e., lossless) reconstruction is not the main objective of the autoencoder training process. Instead, an allowance threshold is (i.e., can be) introduced to help differentiate between a normal and faulty reconstruction - threshold C in Equation 5.3. Obviously, during

the autoencoder post-training deployment, the value of this parameter will ultimately control the rate of false-positives (i.e., normal instance mistakenly flagged as malicious) – a parameter which is considered to be vital in the design and deployment of IDSs. To minimize the rate of false positives, the value of threshold C can be calculated empirically before the actual system deployment.

5.4.2 The Framework for Adversarial Drift Simulation

In this section we are going to describe details of our framework for testing the robustness of deep autoencoders for anomaly detection purposes in an adversarial setting. It is well known that in some real-world anomaly detection based IDSs the number of normal training instance can be overwhelming. (For example, RSSI values collected from operating IoT nodes in a nuclear powerplant.) Now, in IDS systems additionally characterized by non-stationary nature of their input dataset (a phenomenon also known as concept-drift), one reasonable approach to reducing the overall number of training instances is based on the elimination of old/stale input samples. In other words, in such systems, only the subset of the most recent input instances should be kept and deployed in the training of the anomaly detection engine. However, even such strategically reduced training pools are not immune to the general problems of noise, outliers and adversarial contamination [75, 39]. To date, there

have been several research efforts to test and/or make anomaly detection techniques robust to the presence of noise and outliers [75, 39]. This work is one of the first known studies to specifically test the robustness of an autoencoder-based anomaly detection system against adversarial contamination, and compare the performance of this system to that of a PCA based counterpart under the same environmental settings. Figure 6.2 depicts the specific system framework assumed in our work. Under the assumed framework, the training data consists of three particular subsets:

- Initial set T which is comprised of the initial benign training samples,
- benign set B which contains the benign instances that are statistically drifted in distribution parameters relative to T , and
- malicious set M that contains malicious instances.

As for the anomaly detection engine, the initial detection model $MODEL_i = 0$ is trained entirely on set T. This model is then periodically presented with a batch of new input instances, and only the instances that are labeled as sufficiently normal (i.e., benign) are used to further refine (i.e., retrain) the model. Or more specifically, from the simulation perspective, at each iteration i, a batch of new input points are created by sampling (with replacement) n_{benign} instances from set B and sampling (without replacement) $n_{malicious}$ instances from set M. (The proportion of n_{benign} and

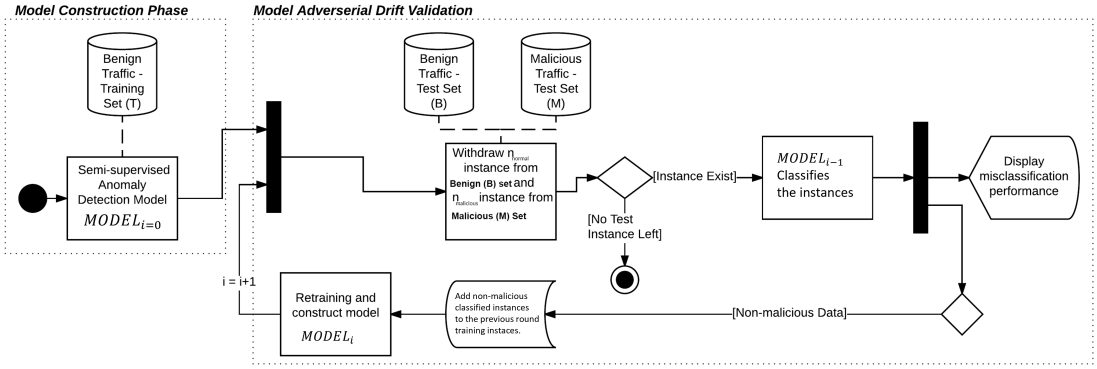


Figure 5.2: General overview of adversarial drift emulation

$n_{malicious}$ should resemble the reality that the simulation tries to capture.) Out of this batch, only those points that are labeled as normal are added to the training set T_i [9] in order to be used during the retraining (i.e., refinement) of model $MODEL_i$. Since the assembled test batch contains both benign and malicious data points, it is possible for malicious points that are not detected (i.e., false-negative points) to get added and contaminate set T_i. At each iteration, before retraining of the IDS, the performance of the detection model $MODEL_{i-1}$ is obtained on the assembled test batch in order to construct the performance trend at the end of the simulation. This process continues until $M = \{\emptyset\}$. Equation 5.4 is the formula for calculating cumulative contamination at iteration i.

[9] T_i contains all the initial training instances and everything added from the previous iterations.

$$\text{Cont}\%(i) = \frac{\sum_{j=0}^{i} P\big(L(x, D_j(x)) \leq C \mid \text{x is malicious}\big) \times n_{malicious}}{|T_i|} \times 100 \quad (5.4)$$

Note that since each detection model is treated as a black-box in our proposed evaluation framework, our main objective is to measure the trend of the system's detection performance by calculating cumulative contamination percentage.

Although this method of contaminating training dataset is very simple, it is very realistic when considering deployment of an adaptive anomaly detection-based IDS in practice. After the initial training phase, the decision boundaries of the latest detection model will be used to select from the new incoming data points to enrich the existing training dataset T_i. Any false-negative classification of new data points can result in contamination of the subsequent retraining dataset, while any false-positive classification can result in loss of a valuable non-malicious training instance.

5.5 Experiments

We have implemented a deep autoencoder with the proposed anomaly detection scheme in order to assess its robustness against adversarial label contamination. As a baseline for comparison, we have used a widely adapted anomaly detection scheme based on Principle Component Analysis proposed by Shyu *et al.* [72]. Initially,

both models are trained with no adversarial drift emulation in order to test their comparative performance under the stationarity assumption. Following that, we assess the performance of the two models under a simulated adversarial drift.

Note, before utilizing deep autoencoders in the context of MAC Spoofing detection using RSSI (presented in Chapter 6), the experiments of this chapter are performed on a different dataset so as to facilitate a fair comparison with the performance of a PCA-based anomaly detector as presented in *et al.* [72]. In particular, NSL-KDD dataset is used for this comparative analysis. Initiated by DARPA in 1998, NSL-KDD-KDDCUP'99 [78] became the mainstream dataset to test and evaluate research works in the field of intrusion detection by containing normal and malicious traffics on a typical U.S. Air Force LAN.

NSL-KDD contains about 4 gigabytes of compressed raw (binary) tcp-dump data of 7 weeks of network traffic, which translates into about 5 million connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records. NSL-KDD training dataset consists of approximately $4,900,000$ single connection vectors each of which contains 41 features (refer to Table 7.1 in Appendix B) and is labeled as either *normal* or an *attack*, with exactly one specific attack type. There are four categories of attacks in the dataset, namely, (1) Denial of Service, (2) User to Root, (3) Root to Local, and (4) Probing [79]. Since our attempt

is to distinguish between malicious and normal traffics, we have replaced the label of all non-normal traffic instances with 'malicious'. Moreover, similar to PCC, our proposed autoencoder IDS model uses only 'normal' training instances in NSL-KDD as the initial training set T. All the 'malicious' instances from the NSL-KDD training as well as NSL-KDD testing dataset are merged in order to form the malicious set M, which is then used for the purposes of robustness test and poisoning simulations. 'Normal' instances left in the NSL-KDD testing set constitute the instances in B that are continually sampled in order to build a simulation of novel normal traffic batches in each iteration.

NSL-KDD contains both nominal and numerical features (refer to Table 7.1 in Appendix B), and for implementation convenience, many researchers exclude nominal-valued features. Moreover, some research works that are focused on model development for classification of malicious instances, tend to exclude some of the features based on their statistical relevance to their classification task [79]. In our work, however, we could not make any argument about unseen normalities, and since we acknowledge that real-world normal profiles are generally non-stationary, we have decided to retain all the features and rely on dynamic representation learning capability of the autoencoder to discover and update the appropriate representations in different layers. As a result, we have transformed all the nominal features using

one-hot encoding and normalized numerical-based features before using them as inputs to the anomaly detection schemes.

5.5.1 Implementation

For the purposes of baseline comparison, we have implemented Principle Component Classifier (PCC) proposed by Shy et al. [72]. In their proposed anomaly detections scheme, top K principle components are considered to be the major components that can capture anomalies with extreme values. Additionally, the remaining principle components are used to detect anomalies that do not have same correlation structure similar to the majority of the normal instances.

As previously described, the desired false-positive rate dictates the selected value of the classification threshold C. (Recall, this threshold is used to differentiate between a normal and faulty/malicious instance.) The false-positive rate of 2%, frequently used as the target rate in the literature on IDS design, is selected to define the threshold values for both PCC and our proposed autoencoder-based IDS.

The architecture of the implemented autoencoder is similar to the one shown in Figure 5.1. The input and output layers each contain 122 sigmoid nodes, encoding and decoding layers each contain 50 sigmoid nodes, and the representation layer contains 10 sigmoid nodes.

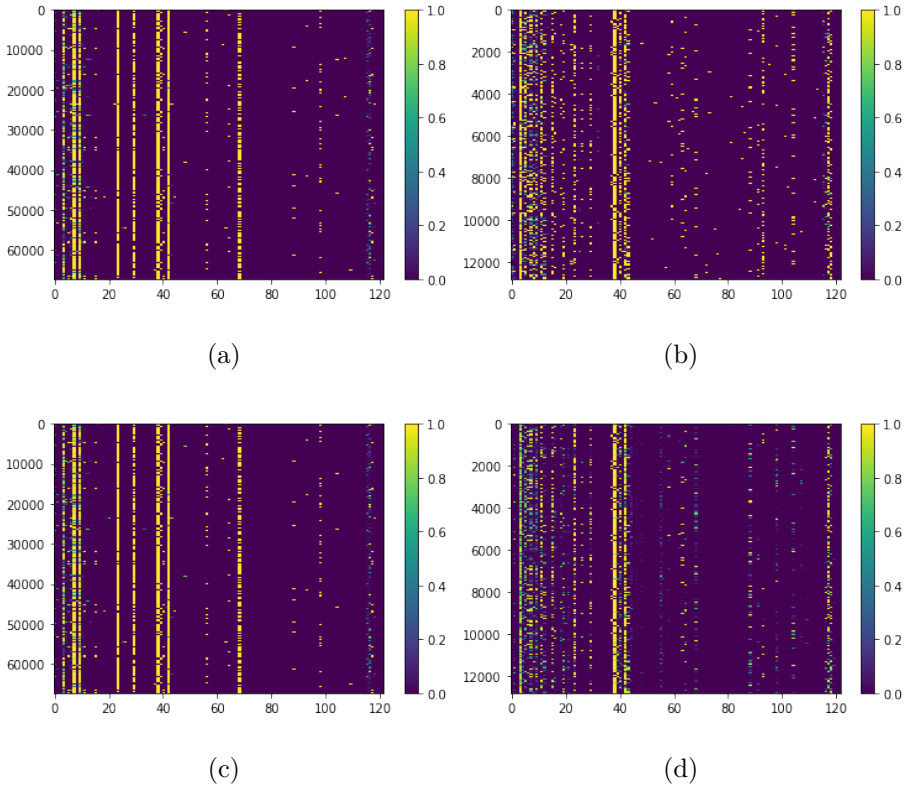


Figure 5.3: Dataset reconstruction visualization, x-axis represent the feature index while y-axis represent instance index - (a) normal training dataset, (b) anomaly test dataset, (c) reconstruction of normal training dataset using the trained autoencoder, and (d) reconstruction of malicious test dataset using the trained autoencoder.

$$S(t) = \frac{1}{1 + e^{-t}} \tag{5.5}$$

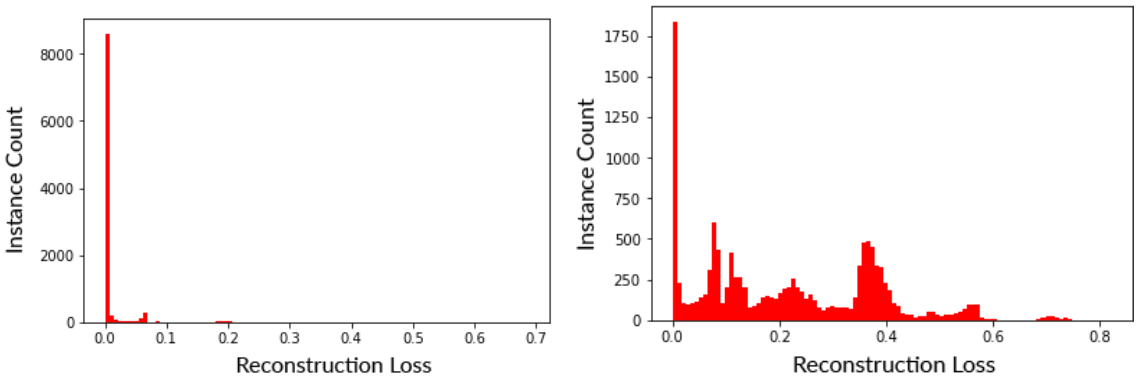
Sigmoid (Equation 5.5) activation function, in our experiments, outperformed other activation functions such as $ReLU$ or $tanh$. Moreover, the lack of smoothness in the error surface (refer to Equation 5.2), had trapped many deterministic optimization methods such as *normal gradient descent* in different local minima, which resulted in a poor performance. In contrast, Adam Optimizer by Kingma and Ba [80] was used to stochastically compute the optimal *gradient* during the training which resulted in an acceptable detection performance.

Algorithm 2 used to iteratively simulate the adversarial drift for each of the trained models (Algorithm 2 in the subsequent page) is the pseudocode realization of the process depicted in Figure 6.2. Note that in our experiments, at each iteration i, T_{test} (i.e., the set of newly introduced/tested points) contained $n_{benign} = 500$ benign samples and $n_{malicious} = 50$. Although in some attacks (e.g., distributed denial of service (DDoS) attacks) number of malicious samples could be exponentially larger than normal instances, we believe that 10% malicious instances of the total sample size (as used in our experimentation) is a realistic configuration for most other types of attacks including low-rate DDoS. Furthermore, this ratio is treated as a parameter in our proposed framework that could be tuned to emulate any attack scenario.

Algorithm 2: Adversarial Label Contamination Robustness Test

Data: T : Initial benign traffic training sample
B : Benign traffic sample to be used for testing and re-training
M : Malicious traffic sample to be used for testing and poisoning
Result: $R = \{(I, C\%, D\%)\}$ is a set of tuples with I representing the iteration number, $C\%$ representing the contamination percentage, and $D\%$ representing the detection rate of the model interval order.

```
1  begin
2      Model ⟵ Construct anomaly detection model using T
3      I ⟵ Initial iteration count to 0
4      C ⟵ Initial contamination count to 0
5      while M ≠ ∅ do
6          I ⟵ I + 1
7          T_test[] ⟵ ∅
8          Add n_benign random samples from B (by selection) to T_test[]
9          Add n_malicious random samples from M (by removal) to T_test[]
10         D ⟵ Initialize detection count to 0
11         for x ∈ T_test do
12             if Model.classify(x) as MALICIOUS and x is MALICIOUS
                then
13                 D ⟵ D + 1
14             end
15             else if Model.classify(x) as BENIGN and x is MALICIOUS
                then
16                 False-negative has occurred, poisoning will take place
17                 Add x to T
18                 C ⟵ C + 1
19             end
20             else if Model.classify(x) as MALICIOUS and x is BENIGN
                then
21                 Update false-positive statistics
22             end
23             else
24                 Add true BENIGN classified traffic to training data to emulate
                    online adaptation
25                 Add x to T
26             end
27             Add (I, (C/|T|)%, (D/n_malicious)%) to R
28         end
29         M ⟵ re-train anomaly detection mode using updated T
30     end
31 end
```



(a) (b)

Figure 5.4: Autoencoder reconstruction errors: (a) distribution of normal, and (b) malicious datasets.

5.5.2 Result

The performance of an arbitrary deep autoencoder is measured by its ability to reconstruct presented inputs with a minimum loss. Figure 5.3 depicts a visualization of NSL-KDD datasets (malicious and non-malicious sets) and their corresponding reconstructions using the trained deep autoencoder. Figure 5.3(a) shows the heat-map visualization of the normal test instances. (It is important to note that all the feature values are normalized between 0 and 1. Each horizontal pixel-line represents an instance data and the main goal of this visualization is to represent an overall view of value distributions in the dataset.) Figure 5.3(b) is the corresponding visualization for malicious instances that are used during the robustness test.

Once the autoencoder is trained using normal training instances its reconstruction on malicious and normal training datasets is tested. Our proposed use of autoencoder's reconstruction error as an anomaly detection measure is validated by means of the obtained results shown in Figures 5.4 and 5.3c. These results clearly show that the autoencoder trained on normal instance dataset can poorly reconstruct malicious instances, while normal test instances are almost perfectly reconstructed. The actual reconstruction error (i.e., square mean error) corresponding to Figure 5.3c is presented in Figure 5.4a and confirms near-perfect reconstruction. On the other hand, malicious instances shown in Figure 5.3b are poorly reconstructed, as shown in Figure 5.3d. The actual reconstruction error corresponding to Figure 5.3d is also shown in Figure 5.4b and confirms inadequate performance.

The *receiver operating characteristic* (ROC) curve depicted in Figure 5.5 shows superiority of the autoencoder in detecting network anomalies compared to PCC for a broad range of examined rates of 'false positive' and under non-poisoning simulation scenarios. Note, for this specific set of experiments, all the training instances used to train the PCC and the autoencoder for construction of the ROC curve were empty of any malicious data point. Specifically, the proposed autoencoder-based IDS has outperformed PCC by approximately 15% in detection rate while keeping the rate of reported false alarms specifically at/around 1%.

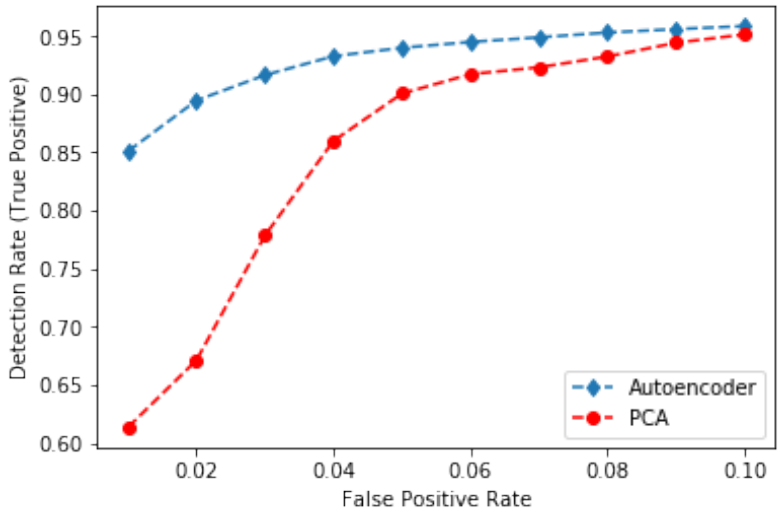


Figure 5.5: ROC comparison of PCA versus Autoencoder

The testing of both anomaly detection techniques for robustness under adversarial contamination (i.e., poisoning) was performed using the procedure described in Algorithm 2. Figure 5.6 shows the performance of each of the methods at different contamination percentage. It should be noted and stressed that contamination that are added into the training datasets (at each iteration) are based on the actual false negative classifications (i.e., labeling) performed by the model under test.

As shown in Figure 5.6, the autoencoder IDS is generally more robust to contamination (compared to PCC), as the percentage of contaminated instances that make it into the training set never exceeds 2%. Moreover, under the comparable observed percentages of input data contamination (i.e., under the same/comparable rates of

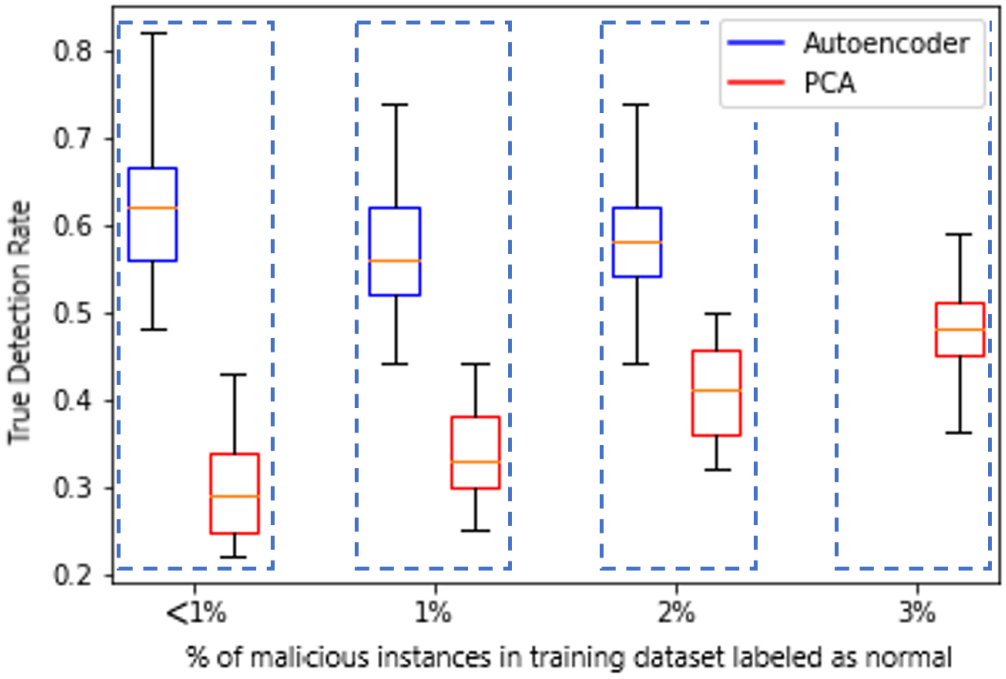


Figure 5.6: Detection rate of the deep autoencoder vs. PCA models under different observed percentages of the training dataset contamination.

false negatives), the detection rates of the autoencoder model are persistently better than those of PCC.

5.6 Discussion and Conclusion

In most anomaly-based IDSs, high rates of successful attack detection are often achieved at the expense of high rates of false alarm (i.e., false positives). Unfortunately, since most alarms need to be investigated by the human operator (regardless of whether

in the end the alarm turns out to be true or false), high rate of false-positives can quickly and unjustifiably overwhelm the operator and render the given detection scheme impractical. The problem of high false-positive rates is especially pronounced in IDSs dealing with non-stationary input datasets (i.e., datasets with concept drift). The only way to reduce the false-positive rates in such systems is by making them adaptive to the changes in the input dataset; or, more specifically, by performing regular retraining/refining of their underlying anomaly-detection engine.

Our experimental evaluation has demonstrated that, if an adaptive IDS gets exposed to deliberate adversarial contamination, the appearance of contaminating points in the retraining datasets is inevitable. However, we have also demonstrated that IDSs with autoencoder-based anomaly detection are generally less susceptible to input dataset contamination compared to IDSs with PCA-based anomaly detection. Moreover, through a comparative simulation study, we have also shown that the deep autoencoder IDSs maintain a more stable rate of detection compared to PCA-based IDSs, even under comparable level of contamination in their training datasets. Given that experimental results obtained in this chapter alluding to robustness of deep autoencoders as anomaly detectors in adversarial settings, in the next chapter we study more advanced deep autoencoder models that can be used to detect anomalies in time series for detecting MAC spoofing attacks using RSSI timeseries data.

Chapter 6

Deep LSTM Autoencoders in Detecting MAC-Layer Spoofing Detection

6.1 Introduction

There have been many research works in the past (that are surveyed in Section 2.2 and further studied in Chapter 4) investigating the use of RSSI profiling for the purpose of MAC spoofing detection. Most of these works implicitly assume that: 1) RSSI samples received from a non-moving transmitting device forms a stationary

time series with normally distributed variance, and 2) these samples are *independent and identically distributed* (i.i.d.) from an unknown normal distribution. Moreover, in the given works, the act of profiling a wireless device based on its RSSI values strongly relies on these two very assumptions. However, in our field study presented in this chapter, the two assumptions (RSSI samples are stationary and i.i.d.) have come under scrutiny. Namely, through our extensive real-world experimentation, we have observed that RSSI values measured by a receiving node are highly affected by the changes (e.g., moving objects) in their operating environments. In particular, we have observed that moving human bodies (and their absence) have a noticeable effect on RSSI values of IoT devices deployed in a residential environment, and as a result the variance of the RSSI time series changes significantly when occupants are present and move around the property - we call this effect *time series clustering* [81] (refer to Figure 6.1 where there are two different cluster, one with lower volatility than the other). Furthermore, it is clear from the depicted figure that there is a correlation between neighboring RSSI values; therefore, it would be hard to justify the claim that neighboring RSSI values are independent (as presumed by previous works [17, 16, 5]).

Except in few use cases where there are no moving objects in the environment (e.g., farm land monitoring), most real-world IoT networks deploy computing/sensing nodes in environments with some number of movable objects. Thus, in order to

account for changes in RSSI values due to the above described clustering effect, it is necessary to have an adaptive and/or multi-model RSSI-based profiling scheme which will be able to improve/reduce the rates of false-positives. (In our previous work [82] presented in Chapter 4, we have demonstrated how i.i.d. assumption pertaining to RSSI values can lead to probable evasion of detection systems that rely on RSSI-based profiling.) In this chapter, we have proposed and studied a multi-classifier system based on deep LSTM autoencoders (the effectiveness of deep autoencoers used in adversarial setting as anomaly detector has been studied in Chapter 5) to profile IoT devices based on their RSSI values under two different conditions (presence vs absence of objects in the surrounding environment). Also, our profiling approach takes into considerations the relationship between neighboring RSSI values in the time-series to further improve the accuracy and robustness of IoT node profiles.

6.2 Threat Model and Assumptions

In this section, we introduce the main annotations and assumptions of our work, which are also illustrated in Figure 6.2. First, consider a simple setup where there are a legitimate transmitting node (e.g., a temperature sensor) denoted by s and a legitimate receiving node (e.g., an IoT hub) denoted by r that communicating over a wireless channel. Also, we assume that r utilizes an arbitrary approach (including

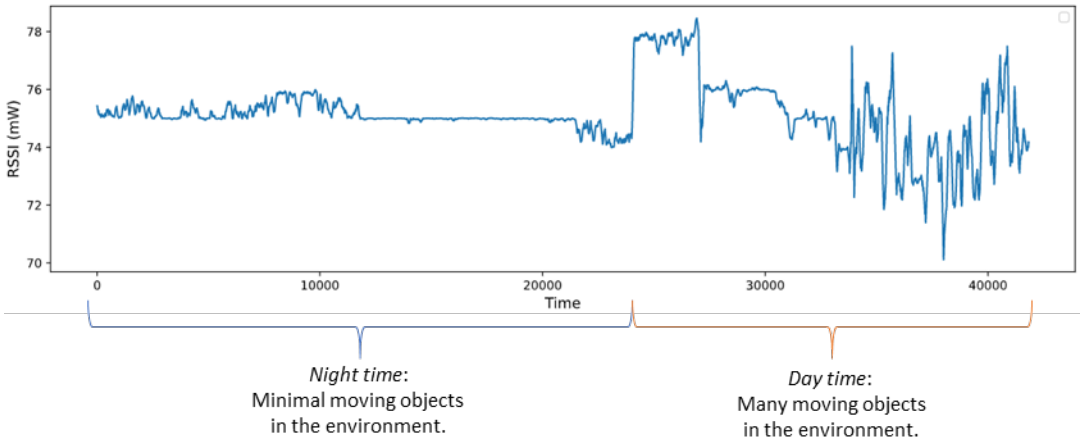


Figure 6.1: RSSI values of an IoT device deployed in a residential property with routine movements of occupants in a 24hours period.

what we propose in this work) to profile s based on RSSI samples it has received in a period absent of any adversary, and then uses this profile at runtime to differentiate between received data frames that carry s' MAC address (legitimate vs. spoofed ones).

The ultimate goal of adversary a is to impersonate a particular s by transmitting frames with s' spoofed MAC address. The spoofed frames are specifically intended for a particular r.

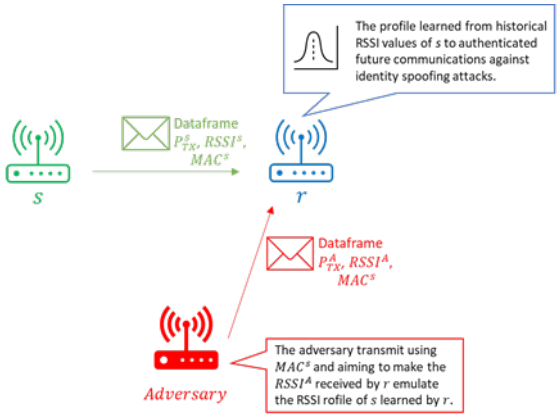


Figure 6.2: Overview of the threat model: the goal of the receiving node is to use historical (clean) RSSI values from the legitimate sender to learn a robust profile to use in future against identity attacks; while the goal of the adversary is to get past the established profile by taking over s identity.

6.3 Detection Approach: Deep Authentication

As demonstrated in Figure 6.1 (and argued in Section 6.1), given that RSSI time-series values of a wireless IoT device are not i.i.d., one could incorporate dependencies amongst neighboring RSSI values to build more robust and accurate predictive models for the purpose of device authentication. Deep autoencoders are deep generative neural networks that have demonstrated strong capability of modelling latent variables in anomaly detection and authentication datasets [83]. LSTM Autoencoders [23], in particular, are known for their generative modelling capabilities on time-series

data. In this section we present our novel technique for authentication of legitimate IoT nodes using RSSI-based anomaly detectors deploying LSTM Autoencoders. In addition, expanding on our argument from Section 6.1 with respect to *time-series clustering-effect* of RSSI values in dynamic environments, we also discuss how our novel multi LSTM Autoencoder architecture is able to switch between multiple trained LSTM models at runtime. Such a multi-LSTM Autoencoder architecture would help with addressing the clustering effect of RSSI time-series.

6.3.1 LSTM Autoencoder Anomaly Detector

In the context of our work, let $X =< x_1, x_2, \ldots, x_n >$ denote an ordered sequence of n RSSI values received by node s. Then, the LSTM Autoencoder is trained to learn two functions, namely, encoder $E(.)$ and decoder $D(.)$ such that $X \approx D\left(E\left(X\right)\right)$. In other words, as depicted in Figure 6.3, the LSTM Autoencoder learns an encoding state that best describes the structure of the training/input data and a decoding function that reconstruct the input sequence given the encoding state with minimal error. In general, a large reconstruction errors occur when the input does not conform to the structure previously learned by the LSTM Autoencoder. As such, a large reconstruction error can be used as a measure of input anomaly [83, 31, 84, 85].

In order to build an RRSI profile of s (through the use of LSTM Autoencoder),

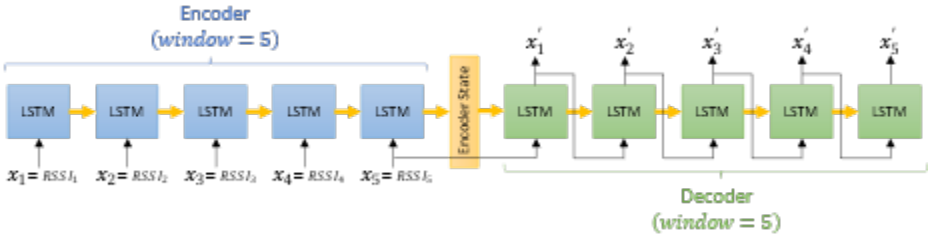


Figure 6.3: Anatomy of the LSTM Autoencoder.

the receiving node r begins the process of collecting and assembling a time-series of RSSI values extracted from the data frames transmitted by s. Then, using a *rolling* window of size n, the time series is segmented into m different overlapping sequences (where the extend of the overlap is controlled by the shift constant of the rolling window), which are further used to train the LSTM Autoencoder. Since the LSTM Autoencoder is supposed to learn the reconstruction of the input sequences, the m training inputs are also supplied as the expected outputs to the training algorithm with the *mean squared error* (MSE) as the loss function.

At runtime (i.e., during the actual use of the trained LSTM Autoencoder for the purpose of attack/anomaly detection), n most recently observed RSSI samples are supplied into the trained LSTM Autoencoder, and then the MSE of the reconstructed sequence (relative to the provided input) is computed. Our experimental investigations (as described in Section 6.4) have demonstrated that the MSEs of the training data, in the absence of attack/spoofed instances, form a normal distribution. Therefore,

our system uses Z-Score to measure deviation from the expected MSE as the decision function to differentiate between the spoofed and the normal traffic. Specifically, for Z-score $\geq l$ the system declares the inspected RSSI window as malicious, where l can be computed experimentally and set for desired *false-positive rate*.

6.3.2 Multiclassifer and Model Switching

As discussed in the Section 6.1, in IoT environments with moving objects (e.g., residential, or commercial premises), RSSI time-series of a transmitting node can be divided into two significantly different time-series with substantially different volatility (i.e., time-series with clustering effect). Using the entirety of such a time-series (refer to Figure 6.1) for the training of our system's LSTM Autoencoder would result in a less sensitive - if not outright ineffective - anomaly detection model. Thus, we propose to deploy/train two independent LSTM Autoencoders – one for the volatile period of the observed time-series when moving objects are present, and one for the relatively calm period when the relative volatility is at its minimum.

Now, one obvious issues that would have to be adequately addressed in an anomaly detection system with two LSTM Autoencoders is the issue of their scheduling. As one possible approach, the system operator could manually set the exact time when each of the trained LSTM Autoencoders is to be deployed according to his/her knowledge

of the environment. However, in such a system with manually determined 'switch times', a number of potential problems could arise. For example, an employee of a factory showing up earlier than usual could significantly affect the RSSI time-series of the nearby sensors/transmitters, which as a result could trigger a false positive alert (provided the detection model corresponding to the non-volatile conditions is still active).

One way to resolve the above challenges is by simultaneous monitoring MSE Z-Scores outputted by the two models at runtime, and looking for the point in time when the Z-score of one of the models crosses another. For example, as show in our experimentation and depicted in Figure 6.4, at night where there are fewer moving objects in the environment, the night's LSTM Autoencoder model is reconstructing RSSI time-series perfectly as reflected by its low Z-Score, while at the same time the day's LSTM autoencoder does a poor job in reconstructing the same RSSI time-series. However, during the transition period when moving objects start to appear in the environment, the night's LSTM Autoencoder performance starts to decline, while the performance of the day's LSTM Autoencoder (which is trained to cope with day-time volatility) starts to exhibit noticeable improvement with respect to the reconstruction MSE. Thus, the moment when the two Z-Score time-series crossover each other would be the optimal point in time when the system should switch from

using the night-time to using the day-time LSTM Autoencoder model. This suggest that by simply monitoring the output of both trained LSTM Autoencoder models, it is possible to determine the optimal 'switch time' in an adaptable and automated manner.

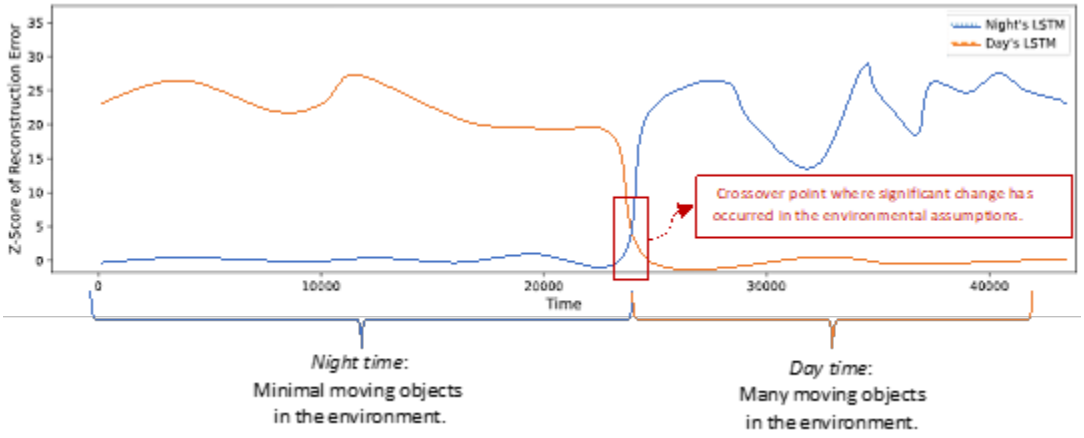


Figure 6.4: Starting at midnight, the Z-Scores of reconstructed RSSI values corresponding to the transmitting node s using the two trained models (for day and night) is tracked. At about dawn, when occupants started to wake up and move about, the error rate of the night model significantly increases while the day model's error rate drops significantly.

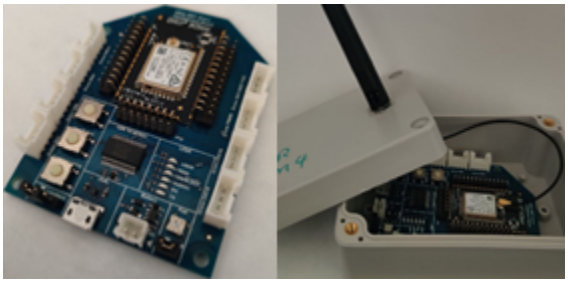


Figure 6.5: Digi XBee 3 Series programmable module implementing IEEE 802.15.4. [86] in a weatherproof secure enclosure protecting the devices from elements when deployed.

6.4 Experiments and Results

6.4.1 Environment Setup

We have designed two experiments involving different form of obstacles and moving objects to best collect the noise and other disturbances that IoT devices may face when attempting to profile their neighboring nodes using RSSI observations. In our experiments we have used three Digi XBee 3 Series programmable modules implementing IEEE 802.15.4 [86] (as depicted in Figures 6.5), where one device acts as the legitimate temperature reading sensor (denoted by s) transmitting its reading to the legitimate receiver (denoted by r) and the adversary (denoted by a) who spoofs s' MAC address in the hope to provide false temperature readings to r.

In the first experiment (refer to Figure 6.6), s is situated in a waterproof container

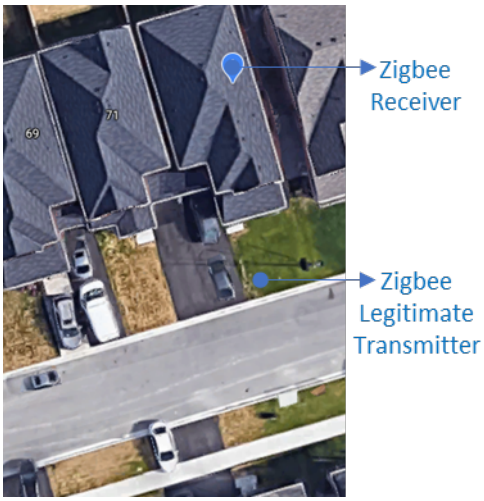


Figure 6.6: The legitimate transmitter is situated outdoor in the lawn transmitting temperature readings and the receiver is situated in the bedroom of the second floor separated by exterior building walls. The pedestrians and motor vehicles in the nearby residential area as well as the 5 occupants in the property are considered to be the influencing moving objects.

on the lawn outside the house, while r is situated in the second-floor bedroom. Aside from 5 occupants living on the property that move about the house during the day, outside pedestrians and moving vehicles effect s' RSSI values observed by r. The adversary is free to move about both - inside the property and outside - to carry its spoofing attack[10]. This is an ideal experiment for resembling scenarios where IoT devices are separated by exterior walls and experience some degree of moving objects during the course of their daily operations.

In the second experiment both r and s are situated in the property separated by a floor/ceiling and interior walls (depicted in Figure 6.7) while the adversary is allowed to move about inside and outside of the property. Similar to the first experiment the house occupants have their routine daily schedule of moving around the property during the day and resting (i.e., minimal movement) at night.

In both experimental setups, r starts its training phase by collecting RSSI samples from s (refer to Figure 6.9 & 6.8) both during hours of minimal and significant movements (24 hours of capture of RSSI at the sample rate of 1 frame/sec) – where these hours are assumed to be empty of any adversarial presence to perturb the training dataset. Once the training stage is completed, r starts using its two trained

[10]This is a very generous assumption to highlight a worse-case scenario and superiority of our approach. In most settings, there are some degree of physical security that constraint adversaries in their physical positioning.

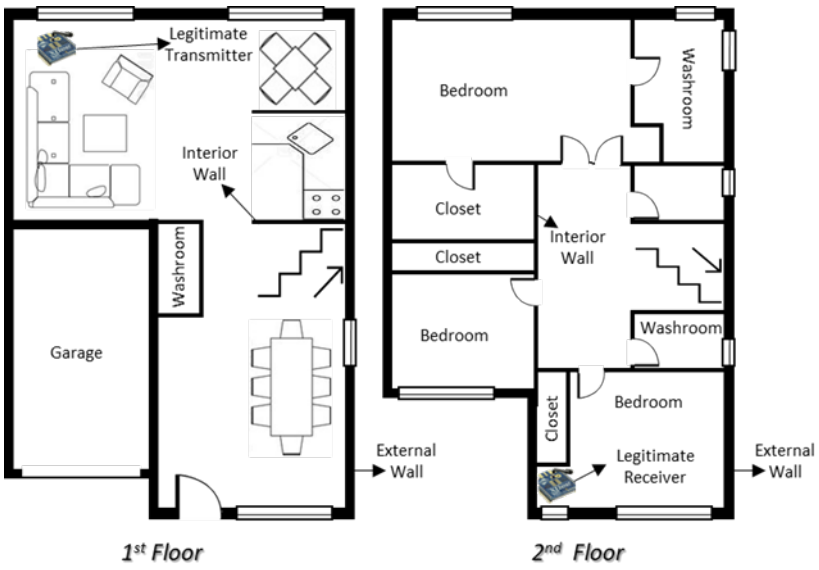


Figure 6.7: The legitimate transmitter is situated at the first floor family room while the legitimate receiver is situated in the second floor's bedroom separated by interior walls and an interior floor. The 5 occupants in the property are considered to be the influencing moving objects.

LSTM Autoencoder models to authenticate received signals and detect MAC-spoofed frames. (Each LSTM Autoencoder has 2 LSTM layers with 20 nodes in each layer and a final dense lake of size 1 and using Adam [87] optimizer for training.)

6.4.2 Note on Special Spoofed Traffic Mix

All the surveyed works pertaining to MAC Spoofing detection using RSSI in Chapter 2 that use a rolling window on collected RSSI stream(s) for the purposes of signal

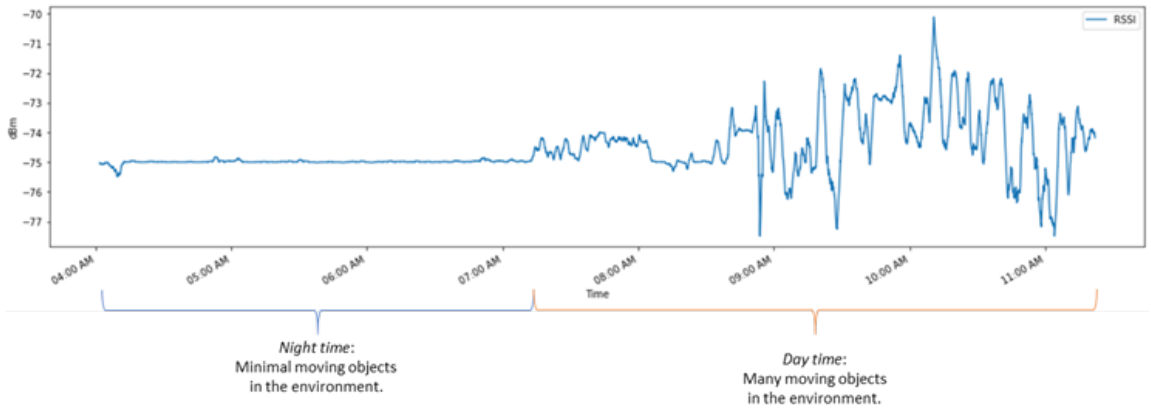


Figure 6.8: s' RSSI stream received by r during s' deployment outside of the property.

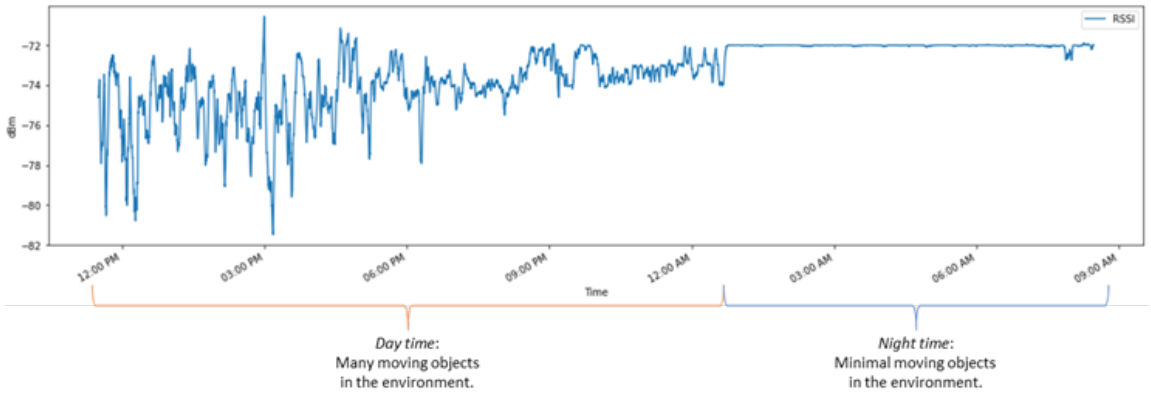


Figure 6.9: s' RSSI stream as received by r during s' deployment inside of the property

classification (i.e., authentication) have implicitly assumed that each window of length n may fully consist of RSSI values from either an attacker or a legitimate device. However, this is not a realistic assumption given unknown motivation and capabilities of adversaries. Moreover, many modern-day IoT devices (e.g., especially those used in home automation) are not battery operated and/or are not much concerned with energy preservation, and as a result may be in frequent communication with other nearby devices. Consequently, in any given window of length n (used by the classification engine) there may exist some mix of the legitimate node's and the adversary's RSSI values as depicted in Figure 6.10. This particular real-world possibility has been taken into account in our experimentation, as described in the subsequent section.

6.4.3 Model Classification Performance

We have evaluated our novel LSTM Autoencoder-based spoofing detection approach against SVM-based anomaly detection technique described in [88] (as a baseline detection model) and also against state-of-the-art Log-likelihood ratio test approach proposed in [20]. We have evaluated all three approaches against two real-world datasets (refer to Section 6.4.1) using 10-fold cross validation by leaving out 10% of the dataset (selected at random) at each run for model testing purposes. The average

(a)

(b)

Figure 6.10: (a) Case where the adversary starts transmitting right after the legitimate node terminated its transmission. (b) the adversary gain access to the channel while the legitimate node have not finished transmitting all of its frames.

classification/detection performance is reported in Table 1 (Table 1 is also visualized in Figure 6.11 and 6.12).

We have trained two classifiers for our Autoencoder as well as each of the other two approaches (SVM [88] and Log-likelihood [20]): one for period of high volatility (e.g., environmental moving objects – daytime) and another for period of low volatility (e.g., minimal environmental moving objects – nighttime) as reported in Table 1. All three classifiers perform relatively better during low volatility period (i.e., nighttime) than high volatility period - with our approach performing the best in both categories significantly.

We have also evaluated classification performance of the three models against adversarial traffic mix (as explained in Section 6.4.2)[11]. We can observe in Table 1, that our approach slightly loses classification accuracy (by 1%) when 20% of RSSI values in a given window is generated by an adversary while the performance of the other two classifiers deteriorates significantly. This can partly be explained by the fact that our LSTM Autoencoder approach takes into consideration the order in which RSSI samples appear (i.e., are collected), while the other two approaches treat RSSI values in a window as independent data points. It is clear from the obtained results that our approach is well equipped to deal with an active adversary that transmit during transmission period of legitimate while such overlap of traffic is not well protected using existing approaches.

[11]We are not making any specific assumption about the ability of the adversary when generating the window contamination. As a result, we have compiled different contamination percentages to evaluate the robustness of the models under the study in order to emulate both passive and aggressive adversaries.

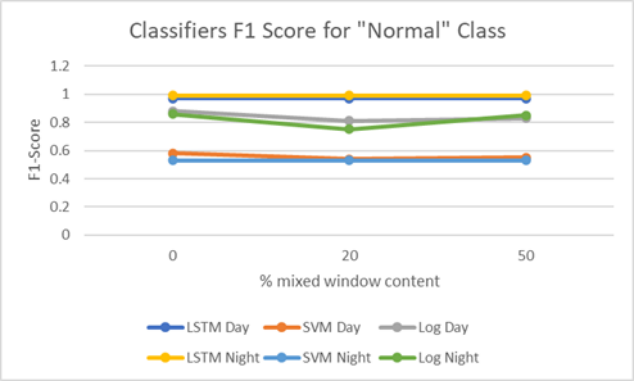


Figure 6.11: Comparison of our novel detection method compared with two other [88, 20] state of the art approaches proposed in the literature in terms of "Normal" class detection.

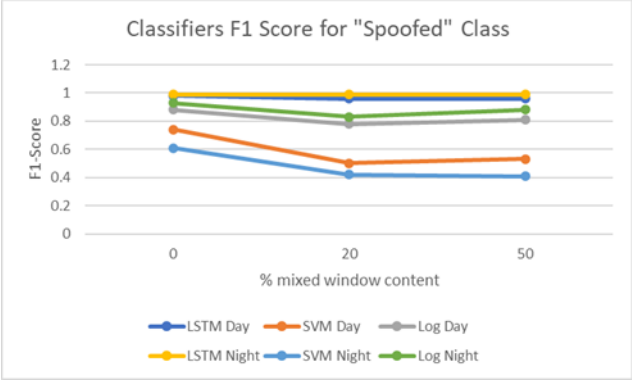


Figure 6.12: Comparison of our novel detection method compared with two other [88, 20] state of the art approaches proposed in the literature in terms of "Spoofed" class detection.

6.4.4 Model Switching at Runtime

In Section 6.3.2 we have explained the need for a bi-modal LSTM Autoencoder classifier, and we have proposed a fully automated and adaptive approach to switching between the two train models/classifiers at runtime. Using the collected real-world datasets we have put this idea to test by continuously monitoring reconstruction error of the two train models at runtime. As depicted in Figure 6.13, at night when RSSI stream had relatively lower volatility the night's model (the blue line) has resulted in low reconstruction error while the day's model (the orange line) has resulted in high reconstruction error - as to be expected. However, at the point in time when the volatility was about to pick up, we can observe a sudden jump in the night's model reconstruction error accompanied by significant improvement in the day's model reconstruction error, ultimately resulting in a crossover between the two error lines (orange and blue). This is a clear indication that the night model could be retired, and the day model could be activated for detection. Clearly, this demonstrates the viability of *c*rossover indicator to facilitate automated and adaptive switching schedule between the two trained LSTM autoencoder models.

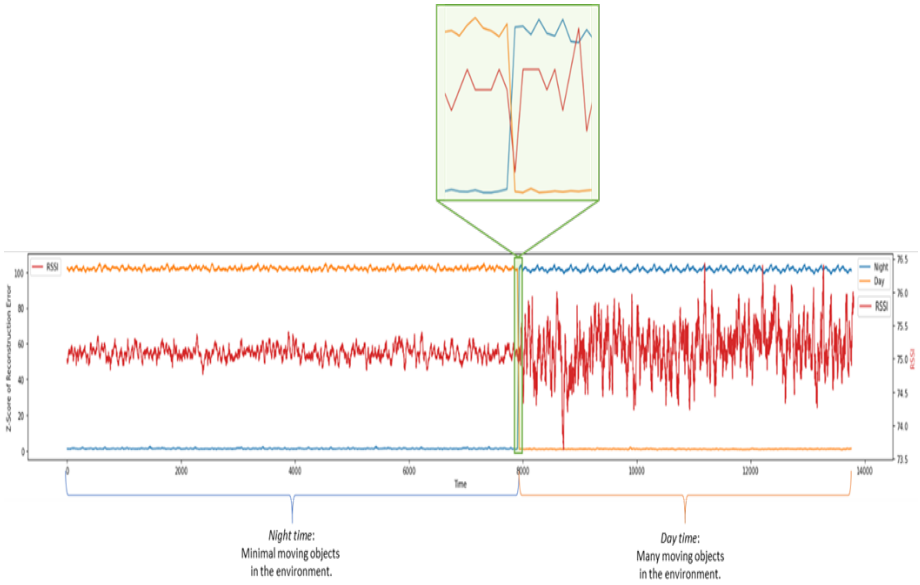


Figure 6.13: Tracking reconstruction error of two trained models during an entire day. The crossover point between the two reconstruction error lines (orange and blue) coincide with increase in volatility of RSSI stream (the red line) – a clear indicator to be used to switch between trained models.

6.4.5 Experimental Considerations

It must be noted that the actual training of the LSTM models (in our experiments) took place on a conventional personal computer and not directly on the IoT nodes. Training deep LSTM autoencoder requires special programming libraries that are not readily available (at the moment) when one is programming an embedded device

(e.g., IoT node). Since in our proposed scheme we are using a single numerical value as an input feature (i.e., RSSI) training the deep LSTM autoencoders is not a memory-intensive task and with proper support of relevant programming libraries is a feasible task to be accomplished on an IoT node directly.

Using pre-trained deep LSTM autoencoders effectively boils to extracting tensors learned during the training phase and transferring them into IoT devices. After all, at the classification stage, tensor multiplication is effectively the main mathematical operation when using the trained model. It is important to note that the tensor multiplication (given our feature space) in our scheme is very fast and does not introduce any significant latency to the overall operation of an IoT node.

6.5 Conclusions and Discussions

In this chapter we have proposed a novel RSSI-based MAC spoofing detection approach using a multi model LSTM Autoencoder classifier. The advantages of our approach over earlier works in this field are twofold. First, our approach is capable of coping with periodic environmental (i.e., signal) disturbances caused by moving objects. Second, our approach can tolerate and detect presence of an adversary that transmits in close time intervals as legitimate network devices.

As part of this chapter, we have also studied the variability of RSSI streams in a

real-world residential area, and (from the collected measurements) we have confirmed the existence of two very distinct periods in the observed RSSI streams (i.e., day vs night). These observations provide real-world justification for the use of a bi-modal LSTM Autoencoder, with one Autoencoder being trained for each variability period. In addition, we have proposed an automated and adaptive technique for determining the optimal point in time to switch between the two train model.

Table 1 - Passive Adversary, *who assumes a single spot in the environment and does not adjust its transmission power.*

		0% Mixed Window Content						20% Mixed Window Content						50% Mixed Window Content					
		Day Classifier			Night Classifier			Day Classifier			Night Classifier			Day Classifier			Night Classifier		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Multi Model LSTM Autoencoder *	Normal	1.0	0.95	0.97	1.0	0.99	0.99	1.0	0.93	0.97	1.0	0.99	0.99	1.0	0.93	0.97	1.0	0.99	0.99
	Spoofed	0.97	1.0	0.98	0.99	1.00	0.99	0.93	1.0	0.96	0.98	1.00	0.99	0.93	1.0	0.96	0.98	1.00	0.99
One-Class SVM [23] (baseline)	Normal	0.66	0.52	0.58	0.73	0.42	0.53	0.56	0.52	0.54	0.60	0.48	0.53	0.58	0.52	0.55	0.59	0.48	0.53
	Spoofed	0.69	0.80	0.74	0.50	0.79	0.61	0.48	0.52	0.50	0.37	0.49	0.42	0.50	0.56	0.53	0.36	0.47	0.41
Log-likelihood ratio [12]	Normal	0.85	0.92	0.88	0.83	0.89	0.86	0.75	0.89	0.81	0.73	0.78	0.75	0.77	0.91	0.83	0.81	0.89	0.85
	Spoofed	0.87	0.90	0.88	0.92	0.95	0.93	0.76	0.81	0.78	0.84	0.83	0.83	0.80	0.83	0.81	0.85	0.92	0.88

Chapter 7

Conclusions and Future Works

In this manuscript, we have investigated the use of learning algorithms in detecting MAC Spoofing attack in adversarial environments.

In Chapter 3, we demonstrated that randomization of decision boundaries in convex-inducing classifiers is considered to be an effective defensive measure against ACRE attack. We have demonstrated that although randomization could result in more queries for finding $\varepsilon - IMAC$, still adversary can find the optimal evading instance using polynomial-many queries. Although finding $\varepsilon p - IMAC$ when X^- is modelled using series of convex-inducing randomized classifiers is believed to be infeasible, we firmly believe that Monte-Carlo techniques discussed in [1, 90, 89] can help with finding an approximation solution to this problem and should be studied as

a potential future work. Moreover, adversary aware randomized classification models that dynamically adapt their randomized selection function γ to the sequence of queries submitted by the adversary can also be considered as an important future-work extension of the work presented in Chapter 3.

In Chapter 4 we have studied the problem of identity spoofing from the perspective of the adversary as well as from the perspective of the defense system. On the adversary's side, we have proposed an advanced technique which allows the adversary to discover his optimal transmission-power that maximizes his probability of evading detection. As a countermeasure against such an attack, on the defense side we have proposed and evaluated a novel strategy that combines the principles of Randomized Moving Target Defense (RMTD) with RSSI-based MAC spoofing detection (i.e., signal-level device fingerprinting). The experimental results have shown that our propose RMTD-enhanced technique offers far superior performance over simple RSSI-based MAC spoofing detection approaches previously studied in the literature. It is also worth mentioning that the adaptability, modular design and modest implementation requirements make our technique deployable in a wide range of wireless systems, though it is particularly suitable for sensor-based IoT systems. We have successfully demonstrated the extra burden/challenge imposed on the adversary (query count and actual evasion probability) as a result of the randomized strategy.

In Chapter 4, we assumed that adversaries are equipped with a special hardware with tunable transmission power settings in order to discover optimal evading transmission power. However, adversaries can attain similar results by being able to move about the perimeters in a controlled fashion using a robotic mobile platform (i.e., drone and/or wheeled platforms) and emulate the effect of variable power settings. The viability of using moving platform to launch optimal evasion attacks has a great potential as a future work.

In Chapter 5, we studied the effectiveness of deep autoencoders acting as anomaly detection engines in adversarial settings under noisy/contaminated training datasets. Also, we proposed a model agnostic framework for assessing robustness of learning-based models under contaminated training datasets. In the context of this proposed novel framework , we have furthered compared the robustness of the deep autoencoder-based IDS against the IDS based on PCA which is well adopted by the industry. Through our comparative study, the deep autoencoder IDS maintained a more stable rate of detection even in the presence of contamination in its training dataset.

In Chapter 6, we have proposed a novel RSSI-based MAC spoofing detection approach using a multi model LSTM Autoencoder classifier. The advantages of our approach over earlier works in this field are twofold. First, our approach is capable of coping with periodic environmental (i.e., signal) disturbances caused by moving

objects. Second, our approach can tolerate and detect presence of an adversary that transmits in close time intervals as legitimate network devices. As part of this research, we have also studied the variability of RSSI streams in a real-world residential area, and (from the collected measurements) we have confirmed the existence of two very distinct periods in the observed RSSI streams (i.e., day vs night). These observations provide real-world justification for the use of a bi-modal LSTM Autoencoder, with one Autoencoder being trained for each variability period. In addition, we have proposed an automated and adaptive technique for determining the optimal point in time to switch between the two train model.

By moving the training process into individual IoT nodes, it is natural to wonder how individual node could adaptively retrain their internal deep LSTM autoencoders. Retraining models at runtime in an adversarial environment could potentially open the training process to different adversarial poisoning attacks (surveyed in Chapter 2). Although we have demonstrated (Chapter 5) that deep autoencoders are generally robust against adversarial noise introduced into their training instance, one must also consider and develop proper techniques to such adversarial noise from the training dataset before retraining the models.

Given that most IoT networks have multiple participants, it is natural to wonder how our proposed method could be further expanded should participating nodes be

capable and/or willing to cooperate with each other in order to detect an ongoing MAC spoofing attack. Although such an idea could likely enhance the overall detection and network performance, it also requires careful consideration and engineering in order to ensure robustness against (e.g.) potential byzantine nodes. An in-depth investigation of such cooperative multi-node approach is one the future research directions of our work.

Also, in Chapter 6, we have assumed that the system operator is in charge of detecting low vs high volatility periods in the training RSSI time-series and dividing the training set into two subsets for training the proposed bi-modal LSTM Autocoders. However, one could argue that due to variabilities of RSSIs during presence or absence of moving objects, it is possible to detect two periods without human intervention and by using unsupervised clustering approaches such as k-means. This is certainly an interesting future work that could further enhance our proposed *crossover model switching indicator*.

Bibliography

)

[1] Blaine Nelson, Benjamin I P Rubinstein, Ling Huang, Anthony D Joseph, Steven J Lee, Satish Rao, and J D Tygar. "Query strategies for evading convex-inducing classifiers". In: *Journal of Machine Learning Research* 13.May (2012), pp. 1293–1332.

[2] Anukool Lakhina, Mark Crovella, and Christiphe Diot. "Characterization of network-wide anomalies in traffic flows". In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement - IMC '04* 6 (2004), p. 201. ISSN: 00283940. DOI: 10.1145/1028788.1028813.

[3] Blaine Nelson, Benjamin I P Rubinstein, Ling Huang, Anthony D Joseph, and J D Tygar. "Classifier evasion: Models and open problems". In: *International Workshop on Privacy and Security Issues in Data Mining and Machine Learning*. Springer. 2010, pp. 92–98.

[4] Nurzaman Ahmed, Hafizur Rahman, and Md I Hussain. "A comparison of 802.11 ah and 802.15. 4 for IoT". In: *Ict Express* 2.3 (2016), pp. 100–102.

[5] Murat Demirbas and Youngwhan Song. "An RSSI-based scheme for sybil attack detection in wireless sensor networks". In: *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*. ieee. 2006, 5–pp.

[6] The Independent IT-Security Institute. "AX Test https://www.iot-tests.org/2017/06/hue-let-there-be-light/". In: 2017. URL: https://www.iot-tests.org/2017/06/hue-let-there-be-light/.

[7] Kevin Curran and Elaine Smyth. "Demonstrating the wired equivalent privacy (WEP) weaknesses inherent in Wi-Fi networks". In: *Inf. Secur. J. A Glob. Perspect.* 15.4 (2006), pp. 17–38.

[8] Ryad Benadjila, Mathieu Renard, José Lopes-Esteves, and Chaouki Kasmi. "One car, two frames: attacks on hitag-2 remote keyless entry systems revisited". In: *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*. 2017.

[9] Kyle Greene, Deven Rodgers, Henry Dykhuizen, Kyle McNeil, Quamar Niyaz, and Khair Al Shamaileh. "Timestamp-based defense mechanism against replay attack in remote keyless entry systems". In: *2020 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE. 2020, pp. 1–4.

[10] Vasily Desnitsky and Igor Kotenko. "Modeling and analysis of IoT energy resource exhaustion attacks". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2017, pp. 263–270.

[11] Tapalina Bhattasali, Rituparna Chaki, and Sugata Sanyal. "Sleep deprivation attack detection in wireless sensor network". In: *arXiv preprint arXiv:1203.0231* (2012).

[12] Hamid Tahaei, Firdaus Afifi, Adeleh Asemi, Faiz Zaki, and Nor Badrul Anuar. "The rise of traffic classification in IoT networks: A survey". In: *Journal of Network and Computer Applications* 154 (2020), p. 102538.

[13] Paolo Falcone, Fabiola Colone, Antonio Macera, and Pierfrancesco Lombardo. "Localization and tracking of moving targets with WiFi-based passive radar". In: *2012 IEEE Radar Conference*. IEEE. 2012, pp. 0705–0709.

[14] Wenda Li, Robert J Piechocki, Karl Woodbridge, Chong Tang, and Kevin Chetty. "Passive WiFi radar for human sensing using a stand-alone access point". In: *IEEE Transactions on Geoscience and Remote Sensing* 59.3 (2020), pp. 1986–1998.

[15] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and Behrang Samadi. "A survey on wireless security protocols (WEP, WPA and WPA2/802.11 i)". In: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. IEEE. 2009, pp. 48–52.

[16] S Mobarakeh Moosavirad, Peyman Kabiri, and Hamidreza Mahini. "RSSAT: A Wireless Intrusion Detection System Based on Received Signal Strength Acceptance Test". In: *Journal of Advances in Computer Research* 4.1 (2013), pp. 65–80.

[17] Wenjia Wu, Xiaolin Gu, Kai Dong, Xiaomin Shi, and Ming Yang. "PRAPD: A novel received signal strength–based approach for practical rogue access point detection". In: *International Journal of Distributed Sensor Networks* 14.8 (2018), p. 1550147718795838.

[18] Daniel B Faria and David R Cheriton. "Detecting identity-based attacks in wireless networks using signalprints". In: *Proceedings of the 5th ACM workshop on Wireless security*. 2006, pp. 43–52.

[19] Yingying Chen, Wade Trappe, and Richard P Martin. "Detecting and localizing wireless spoofing attacks". In: *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE. 2007, pp. 193–202.

[20] Yong Sheng, Keren Tan, Guanling Chen, David Kotz, and Andrew Campbell. "Detecting 802.11 MAC layer spoofing using received signal strength". In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 1768–1776.

[21] Harold Gonzales, Kevin Bauer, Janne Lindqvist, Damon McCoy, and Douglas Sicker. "Practical defenses for evil twin attacks in 802.11". In: *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE. 2010, pp. 1–6.

[22] F Fogelman Soulié, Patrick Gallinari, Yann Lecun, and Sylvie Thiria. "Automata networks and artificial intelligence". In: *Automata networks in computer science, theory and applications*. Princeton University Press, 1987.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[24] Balázs Csanád Csáji. "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24 (2001), p. 48.

[25] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[26] David E Rumelhart, Paul Smolensky, James L McClelland, and G Hinton. "Sequential thought processes in PDP models". In: *Parallel distributed processing: explorations in the microstructures of cognition* 2 (1986), pp. 3–57.

[27] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *arXiv preprint arXiv:1206.6392* (2012).

[28] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. "Intrusion detection using deep belief networks". In: *Aerospace and Electronics Conference (NAECON), 2015 National*. IEEE. 2015, pp. 339–344.

[29] Mostafa A Salama, Heba F Eid, Rabie A Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. "Hybrid intelligent intrusion detection scheme". In: *Soft computing in industrial applications*. Springer, 2011, pp. 293–303.

[30] Bahareh Abolhasanzadeh. "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features". In: *2015 7th Conference on Information and Knowledge Technology (IKT)*. IEEE. 2015, pp. 1–5.

[31] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. "Long short term memory recurrent neural network classifier for intrusion detection". In: *2016 International Conference on Platform Technology and Service (PlatCon)*. IEEE. 2016, pp. 1–5.

[32] Igino Corona, Giorgio Giacinto, and Fabio Roli. "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues". In: *Information Sciences* 239 (2013), pp. 201–225.

[33] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin I P Rubinstein, and J D Tygar. "Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM. 2011, pp. 43–58.

[34] Daniel Lowd and Christopher Meek. "Adversarial learning". In: (2005), p. 641. DOI: 10.1145/1081870.1081950.

[35] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J D Tygar. "The security of machine learning". In: *Machine Learning* 81.2 (2010), pp. 121–148.

[36] Tony A Meyer and Brendon Whateley. "SpamBayes: Effective open-source, Bayesian based, email classification system." In: *CEAS*. Citeseer. 2004.

[37] James Newsome, Brad Karp, and Dawn Song. "Polygraph: Automatically generating signatures for polymorphic worms". In: *Security and Privacy, 2005 IEEE Symposium on*. IEEE. 2005, pp. 226–241.

[38] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. "Antidote: understanding and defending against poisoning of anomaly detectors". In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 2009, pp. 1–14.

[39] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. "Support vector machines under adversarial label contamination". In: *Neurocomputing* 160 (2015), pp. 53–62. ISSN: 18728286. DOI: 10.1016/j.neucom.2014.08.081.

[40] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks". In: *International Workshop on Multiple Classifier Systems*. Springer. 2011, pp. 350–359.

[41] John Graham-Cumming. "How to beat an adaptive spam filter". In: *Presentation at the MIT Spam Conference*. 2004.

[42] Alex Kantchelian, J D Tygar, and Anthony Joseph. "Evasion and hardening of tree ensemble classifiers". In: *International Conference on Machine Learning*. 2016, pp. 2387–2396.

[43] Weilin Xu, Yanjun Qi, and David Evans. "Automatically evading classifiers". In: *Proceedings of the 2016 Network and Distributed Systems Symposium*. 2016.

[44] Pavel Laskov et al. "Practical evasion of a learning-based classifier: A case study". In: *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE. 2014, pp. 197–211.

[45] Cynthia Dwork. "Differential privacy: A survey of results". In: *International Conference on Theory and Applications of Models of Computation*. Springer. 2008, pp. 1–19.

[46] Natalia Stakhanova, Samik Basu, and Johnny Wong. "A taxonomy of intrusion response systems". In: *International Journal of Information and Computer Security* 1.1-2 (2007), pp. 169–184.

[47] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. "Adding robustness to support vector machines against adversarial reverse engineering". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 2014, pp. 231–240.

[48] Richard Colbaugh and Kristin Glass. "Moving target defense for adaptive adversaries". In: *2013 IEEE International Conference on Intelligence and Security Informatics*. IEEE. 2013, pp. 50–55.

[49] Yan Chen, Wei Wang, and Xiangliang Zhang. "Randomizing svm against adversarial attacks under uncertainty". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2018, pp. 556–568.

[50] Yevgeniy Vorobeychik and Bo Li. "Optimal randomized classification in adversarial settings." In: *AAMAS*. 2014, pp. 485–492.

[51] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. "Can machine learning be secure?" In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM. 2006, pp. 16–25.

[52] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2013, pp. 387–402.

[53] Battista Biggio, Paolo Russu, Luca Didaci, Fabio Roli, et al. "Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective". In: *IEEE Signal Processing Magazine* 32.5 (2015), pp. 31–41.

[54] Emanuele Maiorana, Gabriel Emile Hine, and Patrizio Campisi. "Hill-climbing attacks on multibiometrics recognition systems". In: *IEEE Transactions on Information Forensics and Security* 10.5 (2015), pp. 900–915. ISSN: 15566013. DOI: 10.1109/TIFS.2014.2384735.

[55] M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-Garcia, and J. A. Siguenza. "Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification". In: *Proceedings - International Carnahan Conference on Security Technology* (2006), pp. 151–159. ISSN: 10716572. DOI: 10.1109/CCST.2006.313444.

[56] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I.P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. "Exploiting machine learning to subvert your spam filter". In: *LEET 2008 - 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More* (2008).

[57] Gregory L Wittel and S Felix Wu. "On Attacking Statistical Spam Filters". In: *CEAS: First Conference on Email and Anti-Spam* (2004).

[58] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. "Adversarial perturbations against deep neural networks for malware classification". In: *arXiv preprint arXiv:1606.04435* (2016).

[59] Battista Biggio, Giorgio Fumera, and Fabio Roli. *Multiple classifier systems under attack*. Tech. rep. 2010, pp. 74–83. DOI: 10.1007/978-3-642-12127-2-8.

[60] Rajib Ranjan Maiti, Sandra Siby, Ragav Sridharan, and Nils Ole Tippenhauer. "Link-layer device type classification on encrypted wireless traffic with cots radios". In: *European Symposium on Research in Computer Security*. Springer. 2017, pp. 247–264.

[61] TS Rappaport. *Wireless communications: principles and practice*. 1996.

[62] M Ganesh Madhan, U Susaritha, R Ragupathi, and Ashita Priya Thomas. "RSSI based location estimation in A Wi-Fi environment: An experimental study". In: *ICTACT Journal on Communication Technology* 5.4 (2014), pp. 1015–1018.

[63] O Oguejiofor, V Okorogu, Abe Adewale, and B Osuesu. "Outdoor localization system using RSSI measurement of wireless sensor network". In: *International Journal of Innovative Technology and Exploring Engineering* 2.2 (2013), pp. 1–6.

[64] Zheng Yang, Yunhao Liu, and X-Y Li. "Beyond trilateration: On the localizability of wireless ad-hoc networks". In: *IEEE INFOCOM 2009*. IEEE. 2009, pp. 2392–2400.

[65] Christian Robert. *Machine learning, a probabilistic perspective*. 2014.

[66] P. Madani and N. Vlajic. "Near-optimal evasion of randomized convex-inducing classifiers in adversarial environments". In: *ACM International Conference Proceeding Series*. 2019. ISBN: 9781450371643. DOI: 10.1145/3339252.3340520.

[67] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[68] A Buczak and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection". In: *IEEE Communications Surveys & Tutorials* PP.99 (2015), p. 1. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2494502.

[69] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. "Anomaly-based network intrusion detection: Techniques, systems and challenges". In: *Computers and Security* 28.1-2 (2009), pp. 18–28. ISSN: 01674048. DOI: 10.1016/j.cose.2008.08.003.

[70] Rupali Kandhari, Varun Chandola, Arindam Banerjee, Vipin Kumar, and Rupali Kandhari. "Anomaly detection". In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–6. ISSN: 03600300. DOI: 10.1145/1541880.1541882. arXiv: arXiv:1011.1669v3.

[71] Robin Sommer and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection". In: *Proceedings - IEEE Symposium on Security and Privacy* 2.1 (2010), pp. 305–316. ISSN: 10816011. DOI: 10.1109/SP.2010.25.

[72] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. *A novel anomaly detection scheme based on principal component classifier*. Tech. rep. DTIC Document, 2003.

[73] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. "Sensitivity of PCA for traffic anomaly detection". In: *ACM SIGMETRICS Performance Evaluation Review* 35.1 (2007), p. 109. ISSN: 01635999. DOI: 10.1145/1269899.1254895.

[74] Benjamin I P Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J D Tygar. "Stealthy poisoning attacks on PCA-based anomaly detectors". In: *ACM SIGMETRICS Perform. Eval. Rev.* 37.October (2009), p. 73. ISSN: 01635999. DOI: 10.1145/1639562.1639592.

[75] Anthony D Joseph and Nina Taft. "ANTIDOTE : Understanding and Defending against Poisoning of Anomaly Detectors". In: *ACM SIGMOD Conference on Management of Data* November (2009), pp. 1–14. ISSN: 0885-579X. DOI: 10.1145/1644893.1644895.

[76] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. "A Deep Learning Approach for Network Intrusion Detection System". In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ACM, 2016, pp. 21–26. ISBN: 978-1-63190-100-3. DOI: 10.4108/eai.3-12-2015.2262516.

[77] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. "Outlier detection using replicator neural networks". In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer. 2002, pp. 170–180.

[78] K D D Cup. "Dataset". In: *available at the following website http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html* 72 (1999).

[79] Ali A Ghorbani, Wei Lu, and Mahbod Tavallaee. *Network intrusion detection and prevention: concepts and techniques*. Vol. 47. Springer Science & Business Media, 2009.

[80] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[81] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering - A decade review". In: *Information Systems* 53 (May 2015), pp. 16–38. ISSN: 03064379. DOI: 10.1016/j.is.2015.04.007.

[82] Pooria Madani, Natalija Vlajic, and Shadi Sadeghpour. "MAC-Layer Spoofing Detection and Prevention in IoT Systems: Randomized Moving Target Approach". In: *CPSIOTSEC 2020 - Proceedings of the 2020 Joint Workshop on CPS and IoT Security and Privacy*. Association for Computing Machinery, Inc, Nov. 2020, pp. 71–80. ISBN: 9781450380874. DOI: 10.1145/3411498.3419968.

[83] Pooria Madani and Natalija Vlajic. "Robustness of deep autoencoder in intrusion detection under adversarial contamination". In: *ACM International Conference Proceeding Series*. 2018, pp. 1–8. ISBN: 9781450364553. DOI: 10.1145/3190619.3190637.

[84] W Luo, W Liu, S Gao - 2017 IEEE International Conference On, and undefined 2017. "Remembering history with convolutional lstm for anomaly detection". In: *ieeexplore.ieee.org* ().

[85] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. "LSTM-based encoder-decoder for multi-sensor anomaly detection". In: *arXiv preprint arXiv:1607.00148* (2016).

[86] Stanislav Safaric and Kresimir Malaric. "ZigBee wireless standard". In: *Proceedings ELMAR 2006*. IEEE. 2006, pp. 259–262.

[87] Zijun Zhang. "Improved adam optimizer for deep neural networks". In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE. 2018, pp. 1–2.

[88] Rikard Laxhammar. "Conformal anomaly detection: Detecting abnormal trajectories in surveillance applications". PhD thesis. University of Skövde, 2014.

[89] Dimitris Bertsimas and Santosh Vempala. "Solving convex programs by random walks". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM. 2002, pp. 109–115.

[90] Robert L. Smith. *Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed Over Bounded Regions.* Tech. rep. 6. 1984, pp. 1296–1308. DOI: 10.1287/opre.32.6.1296.

[91] Bahareh Abolhasanzadeh. "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features". In: *2015 7th Conference on Information and Knowledge Technology, IKT 2015*. IEEE, May 2015, pp. 1–5. ISBN: 9781467374859. DOI: 10.1109/IKT.2015.7288799.

[92] Md Sohail Ahmad and Shashank Tadakamadla. "Short paper: security evaluation of IEEE 802.11 w specification". In: *Proceedings of the fourth ACM conference on Wireless network security*. 2011, pp. 53–58.

[93] Parisa Alaei and Fakhroddin Noorbehbahani. "Incremental anomaly-based intrusion detection system using limited labeled data". In: *Web Research (ICWR), 2017 3th International Conference on*. IEEE. 2017, pp. 178–184.

[94] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Analysis of classifiers' robustness to adversarial perturbations". In: 2014 (2015), pp. 1–14. arXiv: 1502.02590v1.

[95] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. "Mutual information-based feature selection for intrusion detection systems". In: *Journal of Network and Computer Applications* 34.4 (2011), pp. 1184–1199.

[96] Fabio Anselmi, Joel Z Leibo, Lorenzo Rosasco, Jim Mutch, Andrea Tacchetti, and Tomaso Poggio. "Unsupervised learning of invariant representations". In: *Theoretical Computer Science* 633 (2016), pp. 112–121. ISSN: 03043975. DOI: 10.1016/j.tcs.2015.06.048. arXiv: 1311.4158.

[97] Daniele Apiletti, Elena Baralis, Tania Cerquitelli, and Vincenzo D'Elia. "Characterizing network traffic by means of the netmine framework". In: *Computer Networks* 53.6 (2009), pp. 774–789.

[98] P Armitage. "Sequential analysis with more than two alternative hypotheses, and its relation to discriminant function analysis". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 12.1 (1950), pp. 137–144.

[99] Karel Bartos, Michal Sofka, and Vojtech Franc. "Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants". In: *USENIX Security Symposium* (2016), pp. 807–822.

[100] Battista Biggio and Fabio Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Pattern Recognition* 84 (2018), pp. 317–331.

[101] Misty Blowers and Jonathan Williams. "Machine learning applied to cyber operations". In: *Network Science and Cybersecurity*. Springer, 2014, pp. 155–175.

[102] Imen Brahmi, Sadok Ben Yahia, and Pascal Poncelet. "MAD-IDS: novel intrusion detection system using mobile agents and data mining approaches". In: *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer. 2010, pp. 73–76.

[103] Daniela Brauckhoff, Kave Salamatian, and Martin May. "Applying PCA for traffic anomaly detection: Problems and solutions". In: *INFOCOM 2009, IEEE*. IEEE. 2009, pp. 2866–2870.

[104] James Cannady. "Artificial neural networks for misuse detection". In: *National information systems security conference*. 1998, pp. 368–381.

[105] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[106] Yingying Chen, Wade Trappe, and Richard P. Martin. *Detecting and localizing wireless spoofing attacks*. Tech. rep. 2007, pp. 193–202. DOI: 10.1109/SAHCN.2007.4292831.

[107] Tsung Huan Cheng, Ying Dar Lin, Yuan Cheng Lai, and Po Ching Lin. "Evasion techniques: Sneaking through your intrusion detection/prevention systems". In: *IEEE Commun. Surv. Tutorials* 14.4 (2012), pp. 1011–1020. ISSN: 1553877X. DOI: 10.1109/SURV.2011.092311.00082.

[108] Richard Colbaugh and Kristin Glass. "Predictive defense against evolving adversaries". In: *2012 IEEE International Conference on Intelligence and Security Informatics*. IEEE. 2012, pp. 18–23.

[109] Gregory F Cooper. "The computational complexity of probabilistic inference using Bayesian belief networks". In: *Artificial intelligence* 42.2-3 (1990), pp. 393–405.

[110] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[111] Diazde Leon and Luis Enrique Sucar. "Progress in Pattern Recognition, Image Analysis and Applications". In: *Lect. Notes Comput. Sci. Prog. Pattern Recognition, Image Anal. Comput. Vision, Appl.* 3287.October 2005 (2005), pp. 350–357. ISSN: 1875-7855. DOI: 10.1007/b101756.

[112] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. *Adversarial classification*. Tech. rep. 2004, pp. 99–108. DOI: 10.1145/1014052.1014066.

[113] Anton Dries and Ulrich Rückert. "Adaptive concept drift detection". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 2.5-6 (2009), pp. 311–327.

[114] Rüdiger Ehlers. "Computing the Complete Pareto Front". In: *arXiv preprint arXiv:1512.05207* (2015).

[115] ENISA. *ENISA Threat Taxonomy*. URL: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/enisa-threat-landscape/etl2015/enisa-threat-taxonomy-a-tool-for-structuring-threat-information.

[116] Mustafa Amir Faisal, Zeyar Aung, John R Williams, and Abel Sanchez. "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study". In: *IEEE Systems journal* 9.1 (2015), pp. 31–44.

[117] Ugo Fiore, Francesco Palmieri, Aniello Castiglione, and Alfredo De Santis. "Network anomaly detection with the restricted Boltzmann machine". In: *Neurocomputing* 122 (2013), pp. 13–23. ISSN: 09252312. DOI: 10.1016/j.neucom.2012.11.050.

[118] Chapman Flack and Mikhail J Atallah. "Better logging through formality". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2000, pp. 1–16.

[119] F Fogelman Soulie, P Gallinari, Yann Lecun, and S Thiria. "Automata networks and artificial intelligence". In: *Automata networks in computer science, theory and applications*. Princeton University Press, 1987, pp. 133–186.

[120] Ethod For. "H OST -B ASED I NTRUSION D ETECTION S YSTEMS". In: (2011), pp. 1–12. arXiv: arXiv:1611.01726v1.

[121] João Gama, Indr\.e Žliobait\.e, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. "A survey on concept drift adaptation". In: *ACM Computing Surveys (CSUR)* 46.4 (2014), p. 44.

[122] Ali A Ghorbani, Wei Lu, and Mahbod Tavallaee. *Network Intrusion Detection and Prevention*. 2010. DOI: 10.1007/978-0-387-88771-5.

[123] Amir Globerson and Sam Roweis. "Nightmare at test time: robust learning by feature deletion". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 353–360.

[124] Vladimir A Golovko, Leanid U Vaitsekhovich, Pavel A Kochurko, and Uladzimir S Rubanau. "Dimensionality Reduction and Attack Recognition using Neural Network Approaches". In: *2007 International Joint Conference on Neural Networks* (Aug. 2007), pp. 2734–2739. ISSN: 1098-7576. DOI: 10.1109/IJCNN.2007.4371391.

[125] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems 27* (2014), pp. 2672–2680. ISSN: 10495258. arXiv: 1406.2661.

[126] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. "Active learning for network intrusion detection". In: *Proceedings of the 2nd ACM workshop on Security and artificial intelligence AISec 09* (2009), p. 47. ISSN: 15437221. DOI: 10.1145/1654988.1655002.

[127] Simon Hansman and Ray Hunt. "A taxonomy of network and computer attacks". In: *Computers and Security* 24.1 (2005), pp. 31–43. ISSN: 01674048. DOI: 10.1016/j.cose.2004.06.011.

[128] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[129] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Christos Tachtatzis, and Robert Atkinson. "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey". In: (Jan. 2017), pp. 1–43. arXiv: 1701.02145.

[130] Peter J Huber. *Robust statistics*. Springer, 2011.

[131] Farah Jemili, Montaceur Zaghdoud, and Mohamed Ben Ahmed. "A framework for an adaptive intrusion detection system using bayesian network". In: *Intelligence and Security Informatics, 2007 IEEE*. IEEE. 2007, pp. 66–70.

[132] Anthony D Joseph, Blaine Nelson, Benjamin I P Rubinstein, and J D Tygar. *Adversarial Machine Learning*. Cambridge University Press, 2018.

[133] Shrijit S Joshi and Vir V Phoha. "Investigating hidden Markov models capabilities in anomaly detection". In: *Proceedings of the 43rd annual southeast regional conference on - ACM-SE 43* 1 (2005), p. 98. DOI: 10.1145/1167350.1167387.

[134] Hanna Kamyshanska and Roland Memisevic. "On autoencoder scoring". In: *Proceedings of The 30th International Conference on Machine Learning* 28 (2013), pp. 1757–1765.

[135] Alex Kantchelian, Sadia Afroz, Ling Huang, Aylin Caliskan Islam, Brad Miller, Michael Carl Tschantz, Rachel Greenstadt, Anthony D Joseph, and J D Tygar. "Approaches to Adversarial Drift". In: *AISec* (2013), pp. 99–109. ISSN: 15437221. DOI: 10.1145/2517312.2517320.

[136] Alex Kantchelian, J D Tygar, and Anthony D Joseph. "Evasion and Hardening of Tree Ensemble Classifiers". In: 48 (Sept. 2015), pp. 1–9. arXiv: 1509.07892.

[137] Michael Kearns and Ming Li. "Learning in the presence of malicious errors". In: *SIAM Journal on Computing* 22.4 (1993), pp. 807–837. ISSN: 00975397. DOI: 10.1137/0222052.

[138] M Sadiq Ali Khan. "Rule based network intrusion detection using genetic algorithm". In: *International Journal of Computer Applications* 18.8 (2011), pp. 26–29.

[139] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection". In: *2016 International Conference on Platform Technology and Service, PlatCon 2016 - Proceedings*. IEEE, Feb. 2016, pp. 1–5. ISBN: 9781467386852. DOI: 10.1109/PlatCon.2016.7456805.

[140] Ralf Klinkenberg. "Learning drifting concepts: Example selection vs. example weighting". In: *Intelligent Data Analysis* 8.3 (2004), pp. 281–300.

[141] Ivan Koychev and Ingo Schwab. "Adaptation to drifting user?s interests". In: *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age*. 2000, pp. 39–46.

[142] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. "Bayesian event classification for intrusion detection". In: *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE. 2003, pp. 14–23.

[143] Christopher Kruegel and Thomas Toth. "Distributed pattern detection for intrusion detection". In: *In Proceedings of the Network and Distributed System Security Symposium. Internet Society, Internet Society*. Citeseer. 2002.

[144] Christopher Kruegel and Thomas Toth. "Using decision trees to improve signature-based intrusion detection". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2003, pp. 173–191.

[145] Anukool Lakhina, Mark Crovella, and Christophe Diot. "Diagnosing network-wide traffic anomalies". In: *ACM SIGCOMM Computer Communication Review* 34.4 (2004), p. 219. ISSN: 01464833. DOI: 10.1145/1030194.1015492.

[146] Anukool Lakhina, Mark Crovella, and Christophe Diot. "Mining anomalies using traffic feature distributions". In: *ACM SIGCOMM Computer Communication Review* 35.4 (2005), p. 217. ISSN: 01464833. DOI: 10.1145/1090191.1080118.

[147] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D Kolaczyk, and Nina Taft. "Structural analysis of network traffic flows". In: *ACM SIGMETRICS Performance Evaluation Review* 32.1 (2004), p. 61. ISSN: 01635999. DOI: 10.1145/1012888.1005697.

[148] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. "Differential privacy: From theory to practice". In: *Synthesis Lectures on Information Security, Privacy, & Trust* 8.4 (2016), pp. 1–138.

[149] Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. "An efficient intrusion detection system based on support vector machines and gradually feature removal method". In: *Expert Systems with Applications* 39.1 (2012), pp. 424–430.

[150] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review". In: *Journal of Network and Computer Applications* 36.1 (2013), pp. 16–24. ISSN: 10848045. DOI: 10.1016/j.jnca.2012.09.004. arXiv: arXiv:1401.1637v1.

[151] Richard P Lippmann and Robert K Cunningham. "Improving intrusion detection performance using keyword selection and neural networks". In: *Computer Networks* 34.4 (2000), pp. 597–603.

[152] Yunlong Ma, Peng Zhang, Yanan Cao, and Li Guo. "Parallel auto-encoder for efficient outlier detection". In: *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*. Vol. 2. 3. 2013, pp. 15–17. ISBN: 9781479912926. DOI: 10.1109/BigData.2013.6691791.

[153] Federico Maggi, William Robertson, Christopher Kruegel, and Giovanni Vigna. "Protecting a moving target: Addressing web application concept drift". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5758 LNCS. 2009, pp. 21–40. ISBN: 3642043410. DOI: 10.1007/978-3-642-04342-0_2.

[154] John McHugh. "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory". In: *ACM Transactions on Information and System Security* 3.4 (2000), pp. 262–294. ISSN: 10949224. DOI: 10.1145/382912.382923.

[155] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. "Is deep learning safe for robot vision? adversarial examples against the icub humanoid". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 751–759.

[156] Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan D Payne. "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices". In: *ACM Comput. Surv.* 48.1 (Sept. 2015), pp. 1–41. ISSN: 03600300. DOI: 10.1145/2808691.

[157] Brad Miller, Alex Kantchelian, Sadia Afroz, Rekha Bachwani, Edwin Dauber, Ling Huang, Michael Carl Tschantz, Anthony D Joseph, and J Doug Tygar. "Adversarial active learning". In: *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. ACM. 2014, pp. 3–14.

[158] Leandro L Minku, Allan P White, and Xin Yao. "The impact of diversity on online ensemble learning in the presence of concept drift". In: *IEEE Transactions on Knowledge and Data Engineering* 22.5 (2010), pp. 730–742.

[159] S Mobarakeh Moosavirad, Peyman Kabiri, and Hamidreza Mahini. *RSSAT: A Wireless Intrusion Detection System Based on Received Signal Strength Acceptance Test*. Tech. rep. 1. 2013, pp. 65–80.

[160] Nedim and Pavel Laskov. "Practical evasion of a learning-based classifier: A case study". In: *Proc. - IEEE Symp. Secur. Priv.* IEEE, May 2014, pp. 197–211. ISBN: 9781479946860. DOI: 10.1109/SP.2014.20.

[161] Blaine Nelson and Anthony D Joseph. "Bounding an attack?s complexity for a simple learning model". In: *Proc. of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, France*. 2006.

[162] Yury Nikulin, Kaisa Miettinen, and Marko M Mäkelä. "A new achievement scalarizing function based on parameterization in multiobjective optimization". In: *OR spectrum* 34.1 (2012), pp. 69–87.

[163] C NITRD. "IWG: Cybersecurity game-change research and development recommendations". In: (2013).

[164] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*. Nov. 2016, pp. 582–597. ISBN: 9781509008247. DOI: 10.1109/SP.2016.41. arXiv: 1511.04508.

[165] V Pareto. "Manuel d'économie politique". In: (1909).

[166] Nalini K Ratha, Jonathan H Connell, and Ruud M Bolle. "Enhancing security and privacy in biometrics-based authentication systems". In: *IBM systems Journal* 40.3 (2001), pp. 614–634.

[167] Martin Roesch et al. "Snort: Lightweight intrusion detection for networks." In: *Lisa*. Vol. 99. 1999, pp. 229–238.

[168] B I P Rubinstein, B Nelson, L Huang, A D Joseph, S Lau, Nina Taft, and J D Tygar. "Compromising PCA-based anomaly detectors for network-wide traffic". In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-73, May* (2008), pp. 2008–2073.

[169] Y Sawaragi, H NAKAYAMA, and T TANINO. *Theory of multiobjective optimization*. 1985.

[170] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. "Support vector method for novelty detection". In: *Advances in neural information processing systems*. 2000, pp. 582–588.

[171] Christoph Scholz, Stephan Doerfel, Martin Atzmueller, Andreas Hotho, Gerd Stumme, Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis. *Machine Learning and Knowledge Discovery in Databases*. Ed. by Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný. Vol. 6913. Lecture Notes in Computer Science PART 3. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 129–144. ISBN: 978-3-642-23807-9. DOI: 10.1007/978-3-642-23808-6. arXiv: arXiv:1207.6324.

[172] D Sculley, Matthew Eric Otey, Michael Pohl, and Spitznagel. "Detecting adversarial advertisements in the wild". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 274–282.

[173] Chris Simmons and Dipankar Dasgupta. *AVOIDIT: A cyber attack taxonomy*.

[174] Khomdram Jolson Singh, K L Rita Kho, Sapam Jitu Singh, Yengkhom Chandrika Devi, N Basanta Singh, and S K Sarkar. "Artificial Neural Network Approach for More Accurate Solar Cell Electrical Circuit Model". In: *Int. J. Comput. Appl.* 4.3 (2014), pp. 101–116. ISSN: 22000011. DOI: 10.5121/ijcsa.2014.4310. arXiv: arXiv:1412.5068v3.

[175] Paul Smolensky. *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. DTIC Document, 1986.

[176] Dusan Stevanovic and Natalija Vlajic. "Next generation application-layer DDoS defences: applying the concepts of outlier detection in data streams with concept drift". In: *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*. IEEE. 2014, pp. 456–462.

[177] David Stevens and Daniel Lowd. "On the hardness of evading combinations of linear classifiers". In: *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*. ACM. 2013, pp. 77–86.

[178] W Nick Street and YongSeog Kim. "A streaming ensemble algorithm (SEA) for large-scale classification". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2001, pp. 377–382.

[179] Christian Szegedy, W Zaremba, and I Sutskever. "Intriguing properties of neural networks". In: *arXiv Prepr. arXiv ...* (Dec. 2013), pp. 1–10. ISSN: 15499618. DOI: 10.1021/ct2009208. arXiv: arXiv:1312.6199v4.

[180] Kai Tao, Jing Li, and Srinivas Sampalli. "Detection of spoofed MAC addresses in 802.11 wireless networks". In: *International Conference on E-Business and Telecommunications*. Springer. 2007, pp. 201–213.

[181] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. "A detailed analysis of the KDD CUP 99 data set". In: *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE. 2009, pp. 1–6.

[182] Vladimir N Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[183] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Muehlhaeuser, and Mathias Fischer. "Taxonomy and Survey of Collaborative Intrusion Detection". In: *ACM Computing Surveys (CSUR)* 47.4 (May 2015), pp. 1–35. ISSN: 15577341. DOI: 10.1145/2716260.

[184] Cynthia Wagner, Jérôme François, Thomas Engel, et al. "Machine learning approach for IP-flow record anomaly detection". In: *International Conference on Research in Networking*. Springer. 2011, pp. 28–39.

[185] Zhanyi Wang. "The Applications of Deep Learning on Traffic Identification". In: *Black Hat USA* (2015).

[186] David Warde-Farley and Ian Goodfellow. "Adversarial Perturbations of Deep Neural Networks". In: *Perturbation, Optim. Stat.*

[187] Gerhard Widmer and Miroslav Kubat. “Learning in the presence of concept drift and hidden contexts”. In: *Machine learning* 23.1 (1996), pp. 69–101.

[188] Andrzej P. Wierzbicki. “The Use of Reference Objectives in Multiobjective Optimization”. In: 1980, pp. 468–486. DOI: 10.1007/978-3-642-48782-8_32.

[189] Graham Williams and Lifang Gu. “" for Outlier Detection in Data Mining”. In: (2002), pp. 709–712.

[190] Anthony D. Wood and John A. Stankovic. “Denial of service in sensor networks”. In: *Computer* 35.10 (Oct. 2002), pp. 54–62. ISSN: 00189162. DOI: 10.1109/MC.2002.1039518.

[191] Anthony D Wood and John A Stankovic. “Denial of service in sensor networks”. In: *computer* 35.10 (2002), pp. 54–62.

[192] Wenjia Wu, Xiaolin Gu, Kai Dong, Xiaomin Shi, and Ming Yang. “PRAPD: A novel received signal strength–based approach for practical rogue access point detection”. In: *International Journal of Distributed Sensor Networks* 14.8 (2018). ISSN: 15501477. DOI: 10.1177/1550147718795838.

[193] Jing Hao Xue and D. Michael Titterington. “Comment on "on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes"”. In: *Neural Processing Letters* 28.3 (2008), pp. 169–187. ISSN: 13704621. DOI: 10.1007/s11063-008-9088-7. arXiv: /dx.doi.org/10.1007/s11063-008-9088-7 [http:].

[194] Liu Yang. *Active Learning with a Drifting Distribution*. 2011.

[195] Yang Yi, Jiansheng Wu, and Wei Xu. “Incremental SVM based on reserved set for network intrusion detection”. In: (2011). DOI: 10.1016/j.eswa.2010.12.141.

[196] Gil Zeira, Oded Maimon, Mark Last, and Lior Rokach. “Change detection in classification models induced from time series data”. In: *Data mining in time series databases* (2004), p. 101.

[197] Wentao Zhao, Jun Long, Jianping Yin, Zhiping Cai, and Geming Xia. “Sampling attack against active learning in adversarial environment”. In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*. Vol. 7647 LNAI. Springer, Berlin, Heidelberg, 2012, pp. 222–223. ISBN: 9783642346194. DOI: 10.1007/978-3-642-34620-0_21.

[198] Li Zhuowei, Amitabha Das, and Sukumar Nandi. “Utilizing statistical characteristics of N-grams for intrusion detection”. In: *Cyberworlds, 2003. Proceedings. 2003 International Conference on*. IEEE. 2003, pp. 486–493.

[199] H.-J. Zimmermann. *Fuzzy Set Theory—and Its Applications*. 2001, pp. 1–525. ISBN: 978-94-010-3870-6. DOI: 10.1007/978-94-010-0646-0.

Appendix

Appendix A

In a seminal work by Nelson et al. [1], authors have provided a strategy for an adversary to grow a searching ball around a given instance of interest confined inside a convex decision boundary in order to find an instance of minimum adversarial cost (IMAC) (refer to Chapter 3 for definition) with only polynomial many queries issued against the oracle. In their work, authors have proved that the cost ball must be in norm ℓ_1, otherwise, the time complexity of their proposed query strategy is no longer polynomial and that there is no strategy to solve this problem for $\ell_{p\neq 1}$. They have demonstrated that when growing a ℓ_1 norm cost ball centered around the instance of interest, at least one of the corners of the ball would cross the decision boundaries first and the coordinate of the crossing corner would point to IMAC. Since the corners are situated along the feature axes, the adversary needs to conduct its search along

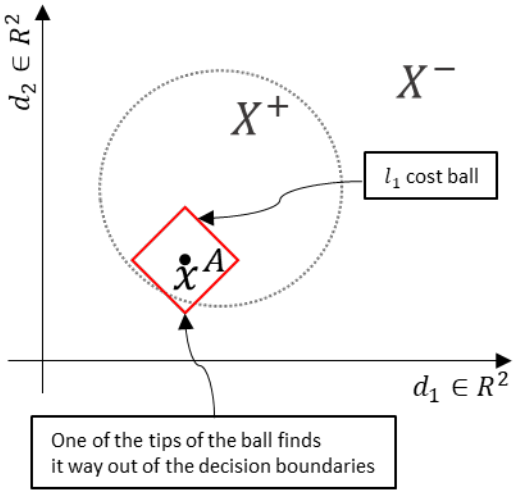


Figure 7.1: Anatomy of cost ball search.

feature axes individually and not their permutations (refer to Figure 7.1).

In Chapter 3, where we have extended Nelson et al. [1] work to randomized convex-inducing classification settings, we made an important observation about the anatomy of multiple convex-inducing decision boundaries that are selected at random in runtime. Specifically, we noticed that if the union of decision boundaries is convex, then the problem of finding can be solved using polynomial many queries using our methodology in Chapter 3. However, as depicted in Figure 7.2, if the union of decision boundaries is a non-convex shape, then proving MAC property for the discovered instance using our method (or Nelson et al. [1]) is not possible and the problem becomes a nonconvex optimization.

Although our proposed search method in Chapter 3 utilizes elements from Bayesian

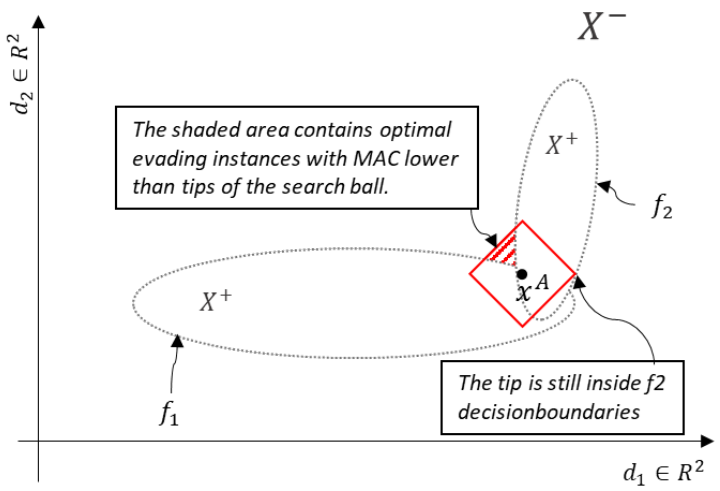


Figure 7.2: Two convex inducing classifiers chosen randomly where their union does not form a convex shape in the feature space.

optimization techniques that are heavily used in nonconvex optimization problems, one might be still interested in quantifying the error in the estimation of IMAC (the shaded area in Figure 7.2) in cases when the union of classifiers is not convex. Quantifying the upper bound of error that one would experience in such settings using our methodology proposed in Chapter 3 remains an open problem.

Appendix B

Table 7.1: NSL-KDD's Feature set with description.

Feature Number	**Feature Name**	**Description**	**Feature Type**
1	duration	Length of time duration of the connection	Basic
2	protocol_type	Protocol used in the connection	Basic
3	service	Destination network service used	Basic
4	flag	Status of the connection – Normal or Error	Basic
5	src_bytes	Number of data bytes transferred from source to destination in a single connection	Basic
6	dst_bytes	Number of data bytes transferred from destination to source in a single connection	Basic
7	land	If the source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	Basic
8	wrong_fragment	Total number of wrong fragments in this connection	Basic
9	urgent	The number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	Basic
10	hot	Number of "hot" indicators in the content such as: entering a system directory, creating programs, and executing programs	Content

11	num_failed _logins	Count of failed login attempts	Content
12	logged_in	Login Status: 1 if successfully logged in; 0 otherwise	Content
13	num_compromised	Number of “compromised’ ’ conditions	Content
14	root_shell	1 if root shell is obtained; 0 otherwise	Content
15	su_attempted	1 if “su root” command attempted or used; 0 otherwise	Content
16	num_root	Number of “root” accesses or number of operations performed as a root in the connection	Content
17	num_file_creations	Number of file creation operations in the connection	Content
18	num_shells	Number of shell prompts	Content
19	num_access_files	Number of operations on access control files	Content
20	num_outbound _cmds	Number of outbound commands in an ftp session	Content
21	is_hot_login	1 if the login belongs to the “hot” list i.e., root or admin; else 0	Content
22	is_guest_login	1 if the login is a “guest” login; 0 otherwise	Content
23	count	Number of connections to the same destination host as the current connection in the past two seconds	Traffic
24	srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds	Traffic

25	serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	Traffic
26	srv_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	Traffic
27	rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	Traffic
28	srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	Traffic
29	same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	Traffic
30	diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	Traffic
31	srv_diff_host _rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	Traffic

32	dst_host_count	Number of connections having the same destination host IP address	Host-based
33	dst_host_srv _count	Number of connections having the same port number	Host-based
34	dst_host_same _srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32)	Host-based
35	dst_host_diff _srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Host-based
36	dst_host_same _src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_c ount (33)	Host-based
37	dst_host_srv _diff_host_rate	The percentage of connections that were to different destination machines, among the connections, aggregated in dst_host_srv_c	Host-based
38	dst_host_serror _rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)	Host-based
39	dst_host_srv _serror_rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_c ount (33)	Host-based

40	dst_host _rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	Host-based
41	dst_host_srv _rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_c ount (33)	Host-based