A MACHINE LEARNING FRAMEWORK FOR HOST BASED INTRUSION DETECTION USING SYSTEM CALL ABSTRACTION

by

Reetam Taj

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2020

*This thesis is dedicated to the medical community, volunteers who are working 24*7 to save the humanity from COVID-19*

Table of Contents

List of Tables

List of Figures

Abstract

The number of cyber threats is increasing faster than the number of defensive strategies deployed to tackle those threats. An automated Intrusion Detection System (IDS) has the capability to detect, classify, and predict cyber intrusions.

To protect an individual host from low-footprint, new generation attacks, I propose a machine learning framework for Host-based Intrusion Detection using system calls identifiers. I chose ADFA-LD12 dataset to evaluate the framework. I developed a hybrid feature retrieval technique combining Integer Data Zero Watermark method and Frequency-based System Call modeling. I applied dimensionality reduction techniques to represent the retrieved features into lower-dimensional space. I finally trained several machine learning and neural network-based classifiers. I evaluated the efficiency of the proposed framework by comparing it with previously proposed approaches. Experimental results indicate that the proposed approach outperforms most of the existing methods in reducing false alarm rate, increasing detection rate, and reducing training time.

List of Abbreviations and Symbols Used

FV	Denotes the feature vectors retrieved by IDZW method
$\wedge$	Diagonal Covariance Matrix
f_i	Number of occurrence of an individual system call indexed by i
$z_{min-max}$	Min-Max scaler transformed value
z_{robust}	Robust scaler transformed value
$z_{standard}$	Standard scaler transformed value
ADS	Anomaly Detection System
AUC	Area Under Curve
CIDS	Collaborative Intrusion Detection System
CNN	Convolutional Neural Network
DARPA	Defense Advanced Research Projects Agency
DNN	Dense Neural Network
DR	Detection Rate
ELM	Extreme Learning Machine
FAR	False Alarm Rate
FNR	False Negative Rate
FPR	False Positive Rate
HADS	Host-based Anomaly Detection System
HIDS	Host-based Intrusion Detection System
HMM	Hidden Markov Model

IDS	Intrusion Detection System
IDZW	Integer Data Zero Watermark
KDE	Kernel Density Estimate
kMC	k-Means Clustering
kNN	k-Nearest neighbour
LSTM	Long Short-Term Memory
MDS	Misuse Detection System
NIDS	Network Intrusion Detection System
NLP	Natural Language Processing
OCSVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
RF	Random Forest
RF-RFE	Recursive Feature Elimination with Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
SPA	Stateful Protocol Analysis
STIDE	Sequence Time Delay Embedding
SVM	Support Vector Machine
XG-Boost	Extreme Gradient Boosting

Acknowledgements

I want to thank my supervisor, **Dr. Srinivas Sampalli**, who has been a constant source of support and encouragement. He motivated me to explore new dimensions of my research work. He helped me to sharpen my skills at identifying, contributing to, and producing research that is aligned to my domain of interest.

I would also like to thank **Dr. Kirstie Hawkey** for being patient while discussing the research ideas. I am also thankful to her for helping me to enrich my research work by teaching the importance of adhering to defined standards, the need for thorough ethical review, and the significance of statistical literacy in conducting valuable research.

I want to thank my parents, **Taj Mohammad** and **Sweta Sarkar**, and my and sister, **Reetoja Taj** for being there as my constant source of inspiration and support. Their rationality, kindness, farsightedness has inspired me to work hard and conduct productive research work. This thesis is dedicated to my parents, both of whom have made huge sacrifices to get me here.

I want to thank my partner, **Oyshee**, for being there as a friend, philosopher, and guide. She helped and inspired me throughout the journey of the development of my life, as well as the thesis and taught me the Latex tool. She also guided me to build my thesis in an organized approach.

I want to thank my dear friend **Mir Masood Ali** who always inspired me to conduct productive research work and stood beside me during hard times. I would also thank my friend **Aditi** for being supportive and a great friend, especially during this COVID crisis. I want to thank **Dinesh** and **Preethi** for "breaks," food, and for being there as a friend throughout my journey. They did not let me miss my home. I thank **Robbie MacGregor** for always being there as a constant inspiration like a big brother. I am very thankful to my best friend **Soumya Mitra**, for tolerating me since 2011 and keeping me energetic throughout the transition of life.

Finally, I would like to thank all the members of MyTech lab, especially **Junhong** and **Amin**, for their strong support, kindness, and productive suggestions.

Chapter 1

Introduction

The Internet has become one of the essential components of our daily lives. As the involvement and usage of the Internet is increasing rapidly, the number of cyber threats are also increasing significantly [1]. As a result, the confidentiality, integrity, and availability of critical information are heavily compromised [2]. The number of cyber attacks is growing faster than the development of defensive strategies to prevent these attacks. Some recent techniques to defending the cyber attacks involve machine learning approaches, policy-driven approaches, and dynamic, rule-based approaches [3]. These techniques have helped to classify and predict different cyber threats efficiently to take necessary precautions by system administrators. As the firewall system does not ensure protection at the open ports of a communication system, an independent intrusion detection system is required to detect and classify different cyber attacks. Intrusion Detection Systems can be classified into two categories according to the data they analyze: 1) Host-based Intrusion Detection System (HIDS) and 2) Network-based Intrusion Detection System (NIDS).

Network-based techniques collect and analyze network data in the affected system. In contrast, host-based detection techniques collect and analyze the data in servers and individual hosts. Even if the detection strategies to defend new generation attacks are improving significantly, the formidable number of widespread hacking mechanisms and a combination of the vulnerabilities of software and operating systems make it impossible to detect all possible modern attacks such as zero-day low-footprint attacks [4, 5]. The new generation low-footprint attacks can penetrate network defense measures such as firewalls and NIDSs. Even anomaly detection systems may not be able to recognize these activities because of the intelligent usage of automated tools such as Metasploit [6]. These tools are capable of misguiding anomaly detection systems by mixing normal with malicious behaviour. The low-footprint attacks contain a very small portion of malicious behaviour, which makes it difficult for classifiers to

detect. These types of attacks are only visible at the host level during routine system analysis or when malicious activities are detected [7]. The targets of these attacks are usually critical machines such as storage and information processing servers as these are the backbone of any business and corporate enterprise. During the operation of these servers, any trace of illegitimate activities in the operating system can be traced at the kernel level with the help of system call traces. As system calls are communication channels between processes and the operating system, they efficiently represent the running activities in an operating system [8].

Forrest et al. [9–11] introduced an approach in 1996 to detect malicious behaviour in Linux environments using a short sequence of system calls. It has been recently observed that short sequences are not efficient in detecting low-footprint attacks, and they do not represent the nature of the whole process. During the last 20 years, researchers have developed many techniques such as frequency domain-based feature retrieval, bag of system calls, and sequence-based feature retrieval to extract the representative features from the system calls. Alongside this research, there have been numerous innovations in the domain of decision engine building to classify system calls using enumerating sequence, machine learning, deep learning, Hidden Markov Models, and rule-based approaches [3, 7, 12, 13]. In the next section, I will present an overview of my research idea and briefly describe the development of the framework for HIDS using system call identifiers.

1.1 Overview

The thesis aims to design and evaluate a Host-based Intrusion Detection System (HIDS) based on System Calls. Figure 1.1 depicts an overview of my research work.

During the initial stage of the research work, I explored different data sources such as system calls, system logs, application logs, and process logs to evaluate the HIDS. I finally chose system call traces to evaluate the proposed framework because of the information richness and authenticity of the data source. I conducted an in-depth literature survey and compared the pros and cons of publicly available system call based datasets. In this process, a comparative study was conducted among four datasets :**ADFA-LD12 [14], UNM [15], DARPA [16], Firefox-DS [17]**. During this comparative study, three different factors were considered: 1) level of realism,

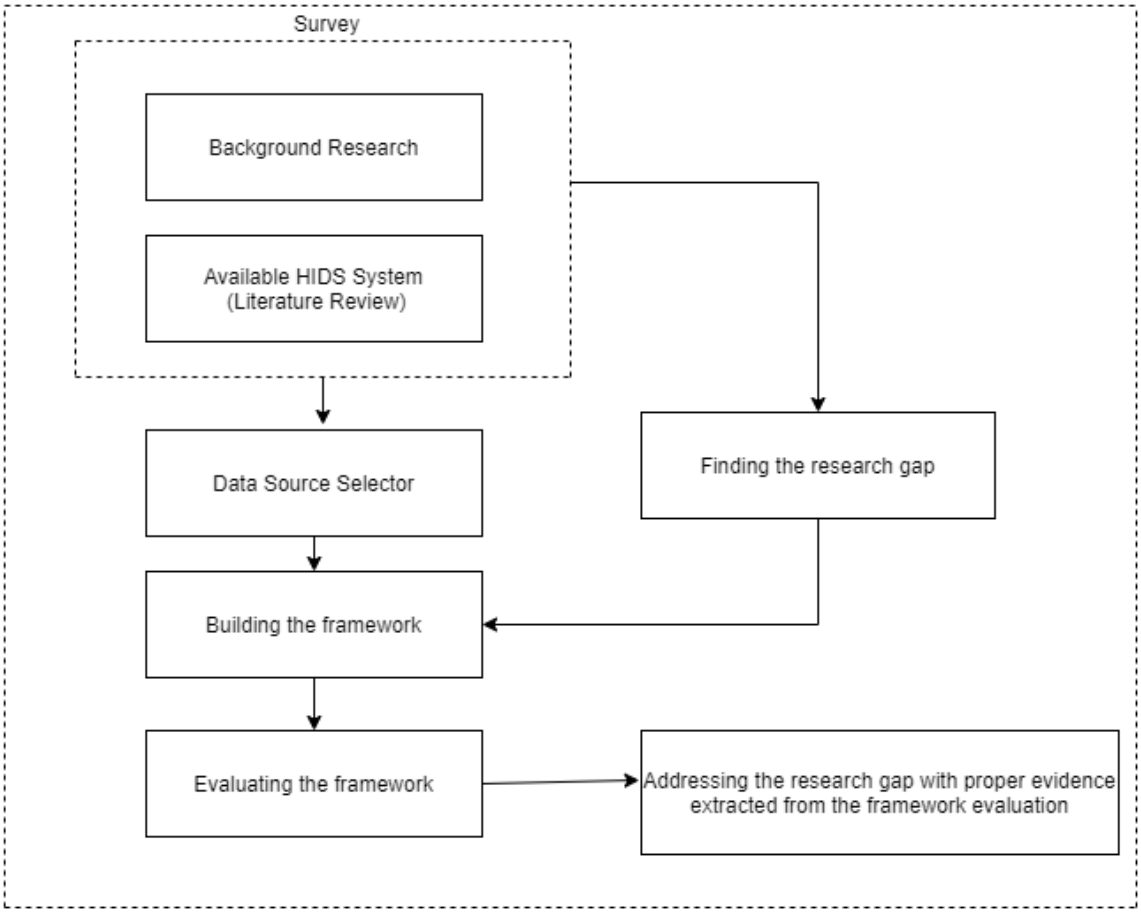

Figure 1.1: Overview of the Research Work

2) level of representation of the state of the art host-based attacks, and 3) level of separability between malicious and benign traces. Finally, ADFA-LD12 was chosen to evaluate the proposed framework.

I further conducted an in-depth literature survey to capture the development in this field over the last two decades. The literature survey was categorized into five parts as per the detection approach used by other researchers: 1) Enumerating Sequence-based approaches , 2) Hidden Markov Model-based approaches, 3) Machine learning and clustering-based approaches, 4) Neural Network-based approaches, and 5) Rule and Filter-based approaches. Detailed knowledge mining was executed to understand and analyze different metrics to evaluate the detection mechanisms. I decided to build the HIDS that focuses on reducing the number of false positives, increase the detection rate, and reduce the training time.

It has been realized from the literature review that feature retrieval techniques play a major role in differentiating malicious traces from non-malicious traces. Existing feature retrieval techniques mainly focus on three different approaches: 1) Sequence-based approach (short and long) [3], 2) Frequency-based approach [18], and 3) Hidden feature retrieval approach using Integer Data Zero Watermark (IDZW) [7].

I propose a feature retrieval technique that combines frequency-based system call modeling and hidden representation of system calls based on IDZW method. This hybrid feature retrieval approach retains the features extracted from the frequency of an individual system call in a trace. It also carries the hidden abstraction of a trace. Several normalization and scaling techniques were implemented to conduct further data pre-processing on the retrieved features.

I used three dimensionality reduction techniques to extract representative information from retrieved features: 1) Neural Network-based dimensionality reduction approach; Auto Encoder, 2) Statistical feature extraction approach; Principle Component Analysis(PCA), and 3) Random Forest-based Recursive Feature Elimination. Dimensionally reduced data was used to train learning classifiers using both supervised and semi-supervised approach. Classifiers were trained with only benign traces to build a normal profile in order to implement a semi-unsupervised approach.

Malicious traces were then compared against the normal profile to determine outliers. If a sample produces an output that is higher than the previously specified threshold, then it is considered as a malicious trace. One of the major reasoning behind choosing the semi-supervised approach was to tackle highly imbalanced data efficiently. Highly imbalanced training is a true representation of a real-time scenario, which makes the semi-supervised approach rational to implement.

The dataset was divided into training data, testing data, and validation data, before being used to train the classifier with a supervised approach. All subdivisions contain both malicious and non-malicious system call traces. The classifiers were trained using both normal and malicious system calls and their respective levels. The validation training set was used to understand "how well the model is trained" and tune the parameters of the classifier. Finally, the test set was used to evaluate the classifier.

To evaluate the proposed framework, and analyze the effectiveness of the hybrid

feature retrieval techniques, classifiers were trained with a different configuration of data normalization/scaling and dimensionality reduction techniques. For each of the learning models, I have reported accuracy, precision, recall, f-score, false alarm rate,and detection rate,. In addition to that, the time required for the training process was also recorded for each of the classifiers. The performance of classifiers and the calculated metrics were visualized using a Receiver Operating Characteristic curve for a simplified and efficient interpretation.

The performance of classifiers was compared, and the best one was chosen according to their overall performance, considering the trade-off between false alarm rate and detection rate. Finally, the result of the selected model was compared with the results achieved by other researchers. The reasoning behind the effectiveness of the results is discussed in the Conclusion (Chapter 8).

1.2 Motivation

The sensitive information is compromised very often through several cyber intrusions. In recent times, Internet hackers are coming up with different innovative approaches to breach the security of digital platforms to access and exploit sensitive information. Scientists are coming up with solutions to fight against modern approaches, but that is not enough because the amount of data in the digital platform is increasing exponentially, as is the number of threats. More resources need to be deployed to counter cyber-attacks, and cybersecurity researchers have long way to go before we catch up with the modern threats.

It has been observed (2016) that 95 percent of breached records mainly come from three industries; Government, Retail, and Technology [1]. These industries are targeted because they store large amounts of personal identification details. As per a study conducted at the University of Maryland [19], a cyber attack occurs every 39 seconds. As per data published by Juniper Research [20], the average cost of a data breach will exceed 5 trillion dollars by 2024, and 75 percent of the healthcare department has already been affected by malware over the last years. "According to the Q2 2018 Threat Report, Nexusguard's quarterly report, the average number of distributed denial-of-service (DDoS) attacks grew more than 26Gbps, increasing in size by 500% [21]." It is also expected that 6 trillion dollars will be spent globally on

cybersecurity by 2021 [21].

I was a victim of "WannaCry," a trojan worm, in May 2017, which exploited the vulnerabilities of the Windows Operating System. It was a two-step attack, produced a self-propagating worm that locked up files on a host computer, and a module that introduced ransom extortion in systems [1]. I began exploring articles on this topic to learn about prevention from similar attacks in the future. I read essays and papers [3, 22–24] about the classical techniques of the antivirus software to prevent and detect cyber threats. I also gained knowledge regarding firewalls and understood the need for developing an independent Intrusion Detection System (IDS). The learning process helped me realize that the performance of Machine Learning and Deep Learning-based IDS has significantly improved in the course of time [2, 23].

As the number and types of cyber threats increase exponentially, we need to come up with more dynamic and innovative methods and approaches to tackle them efficiently. During this learning process, I encountered multiple roadblocks for implementing a standalone IDS. High false positive rates left the system incapable of detecting zero-day attacks in real-time. All these reasons and learning have intrigued me to deep dive into the domain of the research of building an Intrusion Detection system using Machine Learning.

1.3 Roadmap of the Thesis

This section provides a roadmap that describes the content of each chapter in the thesis.

- Chapter 2 contains the background and literature review. In this chapter, different types of cyber threats and intrusion detection systems are introduced. Then, the research work accomplished by the other researchers in the domain has been presented in a detailed manner.

- Chapter 3 narrows down the research problem I have been working on and represents the thesis contribution.

- Chapter 4 focuses on the comparative study of the different datasets that are publicly available to evaluate HIDS. Chapter 5 includes the algorithms that were explored and experimented with to build the HIDS framework.

- Chapter 6 describes the methodology and the course of action taken to address the research problem.
- Chapter 7 focuses on describing the experimentation and results, and analysis of the achieved result.
- Chapter 8 summarizes the present work with concluding remarks, limitation, and includes an overview of the research work that can be expanded based on the current course of research.

Chapter 2

Background and Literature Review

2.1 Cyber Intrusion/attack

A cyber attack can be defined as a process that makes a computer system vulnerable by breaking the security of the system that causes it to enter into an insecure state. The major target of a cyber attack is computer information systems, infrastructures, computer networks, and personal computer devices [25].

2.1.1 Definitions

Since late 1980, cyber attacks have evolved to manipulate the vulnerabilities of the intelligent software systems. In recent times, the scale and robustness of the cyber attacks have increased exponentially. As per the 2018 report by World Economic Forum, cyberattacks can be defined as: "Offensive cyber capabilities are developing more rapidly than our ability to deal with hostile incidents [26]."

The Internet Engineering Taskforce defined the attack in RFC2828 as: "An assault on system security that derives from an intelligent threat, i.e., an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system [27]."

As per CNSS instruction No. 409, a cyber attack is defined as: "An attack, via cyberspace, targeting an enterprise's use of cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment/infrastructure; or destroying the integrity of the data or stealing controlled information [28]."

Committee of National Security System by the United States of America defines cyber attacks as: "Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself [28]."

2.1.2 Type of attacks

This sub-section discusses different attacks and intrusion attempts made at large-scale infrastructures and government networks. I classify and present an overview of different types of malicious attacks. The resulting classification has helped me to gain a broad knowledge about different types of cyber-attacks, which paves the path for the research on the mitigation techniques. Cyber attacks can be categorized as below:

1. Attacks classified by purpose
2. Classification by attacker type
3. Classification by target

Attacks classified by Purpose

In this section, I categorize of security attacks based on the intended effect. They can be further divided into 1) Confidentiality 2) Availability, and 3) Integrity. Each of the categories of this section is attached to one specific goal, but some attacks broadly fall under multiple categories.

1. **Confidentiality :** Attacks under this category attempt to gain access to private information. The scope of this type of attack ranges from information gathering (an initial step for future attacks) to password cracking. This specific subcategory can be further divided into **Reconnaissance, Access attacks, and attacks on passwords and secret keys**.

   In Reconnaissance [29], an attacker attempts to gather in-depth knowledge of a system before launching an attack. They help the attackers to gain a complete understanding of the network that they are attempting to intrude, accessing data to realize evasion mechanisms, and design attack strategies. Probing (nmap ipsweep,satan) [30–33] and Portscan [34] attacks are the example of Reconnaissance. The access related attacks attempt to gain access to the network, server, or system in question, usually with elevated privileges. In the event where an attacker is external to the network, they will attempt to exploit vulnerabilities to gain access to the network. Some examples of the access

attacks are User to Root [35] , Remote to Local, Infiltration, Privilege escalation, Backdoor, spoofing, and Webshell attacks [36–39]. A few examples of attacks on password and secret keys are Statistical Inference Attack, Active Dictionary attack, Power analysis attacks, Keylogger attack, and SSH brute force attack [40–42, 42, 43].

2. **Availability :** The information on a network and services provided by channels and nodes should be up and available at all times. Critical systems pay heavily and result in deeper external impact when they are made unavailable and inaccessible to legitimate requests. Some of the common attacks in this subcategory are different types of DDos attacks,Scheduling attacks, Stealthy Collision attack,and Stealthy decoy attacks [44–48].

3. **Integrity :** Attacks that fall under this category involve tampering and manipulation of data on the network. Malware based attacks such as Worm, Ransomware, Spyware, Trojan horse, and web-based attacks such as Heartbleed, Web-based SQL Injection attacks fall under this category [49–53].

Attacks classified by Attacker type

In this section, cyber attacks are classified by attacker type. This classification determines whether an attack has resulted in compromise of performance or data, also serves as a precursor to an incoming attack. These categories of attacks can be further categorized into 1) Passive and 2) Active.

Passive: Passive attacks do not result in disruption of service or network performance. They are instead primarily used for reconnaissance.

1. **Eavesdropping:** Passive sniffing can be used to gather information regarding IP addresses, domain names, active ports and machines, operating systems, and network topology [54].

2. **Traffic Analysis:** Interception of traffic to realize traffic and information flow does not necessarily need to enable plain-text message interception. Rate of traffic flow, channels with higher routing bandwidth, and information flow metrics through slow and distributed scans provide information for reconnaissance that can help with information inference [55].

Active:

1. **Masquerade:** The masquerade attack is a type of attack, in which a user of a system illegitimately poses as another legitimate user. One of the best example of masquerade attack is Identity theft in financial transaction system. Masquerade attacks can be very dangerous in the case of an attacker who has the capability to cause damage to an organization [56].

2. **Man-in-the-middle:** In Man-in-the-middle attacks, perpetrators positions themselves in a conversation between a user and an application. The attackers eavesdrop or impersonate as one of the legitimate parties in the communication in order to make it appear as if a normal communication is underway. One of the major goals of this attack is stealing personal information of the users such as log in credentials, credit card information [56].

3. **Session Hijacking:** Session Hijacking occurs when an attacker steals the information of a legitimate user for a specific website and uses it to bypass the authentication to that website. Session Hijacking is one most popular cross-site attack since every website that uses session identifier are vulnerable to this attack [57].

Attacks classified by the target of the attacker

This section classifies attacks by the intended target of the attack. Bringing the specified target may fall under one of the more categories of purpose.

Server:

These attacks primarily attempt to bring down web-servers. They disrupt the availability of service, inject malicious code into the server, or access information that is otherwise unavailable to client nodes.

1. **Cross-site request forgery:** Illegitimate requests can be sent to the server through legitimate, authenticated users. The end-user is unaware of the request and cannot observe the response. Using POST requests, attackers can exploit an end-user's trust in their browser to access information from the server that should otherwise only be available to legitimate users alone [58].

2. **SQL Injection:** An SQL injection attack gives an attacker unrestricted access to the database that the server refers to and protects. They allow the attacker to use SQL queries to access sensitive information and to manipulate not just the data, but the structure of the database and its tables [59].

3. **Command Injection:** A code that handles user input to generate queries for the user is treated as a lexical entity and does not take into consideration the structure of the programming language. When user inputs are not sanitized, they can be used to alter the output, making the server access data that they otherwise should not be accessed [60].

4. **DoS Attacks:** These attacks affect the availability of the server and halt its services.

5. **Directory Traversal:** An attacker can alter HTTP requests to execute path traversals on the server. The intention is to navigate the restricted directories that require elevated privileges to access [61].

Network:

1. **Man-in-the-middle:** In a man-in-the-middle attack, the attacker acquires access to data in transit between a sender and a receiver.

2. **Session Hijacking:** Session Hijacking allows a user to bypass authentication to gain access.

3. **Attacks on Network Performance:** These attacks attempt collisions on certain channels, decrease channel availability, such as DoS attacks.

4. **Rerouting:** An attacker may use stealthy collisions or use IDS flow rerouting responses to DoS attacks to their advantage. Rerouting traffic to compromised networks or low bandwidth channels include the objectives of a rerouting attack.

Client:

1. **Social Engineering:** Social Engineering [62] is a process of getting users to compromise the information system. Instead of attacking a system technically,

social engineers target humans to access confidential information. By following this process, social engineers carry out malicious attacks by influence and persuasion. There are several social engineering techniques, which include baiting, phishing, pretesting (pretending to be someone else to get access to privileged data).

2. **Phishing:** Phishing is an approach that broadly comes under Social Engineering where the cyber attackers manipulate the users into revealing personal details such as password, credit card, social security number. Phishing can be carried out by sending the users fake emails or redirecting to the untrusted website. Phishing messages usually originate from legitimate organizations that request account information. On providing the account information, users are redirected to an untrusted website and tricked into entering confidential information, which can result in an identity theft [63].

3. **Client Application:** The application-level attacks target the layer of the internet that usually faces to the end-users. These types of attacks refer to the vulnerabilities present in the code of the web application. Application layer attacks are designed to attack the application itself, resulting in the application not being able to deliver the content to the users as expected [64].

4. **Client Machine:** Attacks under this type cover attempts at compromising the host machine of an end-user on a network.

    - **Rootkits:** A rootkit [65] is a program that is designed to provide attackers administrative access to a system without letting the users know about it. It can be used to use a system and manipulate it remotely.A rootkit hides its presence within the lower layers of the operating system.
    - **Code Mutation:** Malicious dynamic links [66] to libraries in client code can be altered to execute adversarial code on the execution of regular programs on an end-user machine.
    - **Anti-emulation:** Malicious code [66] is often run as executable in virtual sandboxes that help them evade detection. These sandboxes sequester them to environments where detection systems have limited access.

2.2 Intrusion Detection Systems (IDSs)

In order to protect a network or host from the attackers, many traditional techniques such as password protection, firewall, rule-based network detection have been used extensively. But these techniques are not enough to completely protect the network or a host from the intrusions. A dedicated IDS is required to detect, classify, and generate alerts for any malicious activity. IDS can be categorized into two ways, such as types of analyzed data and types of methodology it follows [3].

2.2.1 Categories based on the types of analyzed data

This category concentrates on the type of data an IDS analyzes. It can be further sub-categorized into Network-based Intrusion Detection System(NIDS), Host-base Intrusion Detection System(HIDS), and Collaborative Intrusion Detection System(CIDS) [67].

Network-based Intrusion Detection System (NIDS)

NIDS monitors and analyzes network communication to detect any intrusion. In the past few years, research communities from the cybersecurity field have majorly focused on developing intelligent network intrusion detection systems, and they were able to achieve notable success in this field. To satisfy the rapid development of research, the researchers have also developed efficient datasets to evaluate a NIDS such as NSL-KDD, CICDDoS2019, CSE-CIC-IDS2018, UNSW-NB15, ISCXIDS2012 [68]. Researchers from various fields used several methods such as rule-based learning, pattern mining, netflow based sequence analysis, machine learning, and deep learning-based techniques to enhance the performance of the NIDS. As NIDS is deployed as hardware devices between the internet and intranet, NIDs can perform real-time intrusion detection. As mentioned by Liu et al. [3] , traditional NIDS cannot process the encrypted packets, which reduces the detection speed of the system and downgrade the speed of the network flow. One of the most popular open-source rule-based NIDS is Snort [69], which can detect intrusion based on the stored attack signatures but not able to detect if the traffic deflects significantly from the stored signatures.

Host-based Intrusion Detection System (HIDS)

HIDS monitors the activity of individual hosts to detect any unauthorized activity. HIDS is mainly deployed to detect internal and local attacks. It monitors the system calls, shell logs, and application logs to find patterns and extract information to detect illegal behavior. System calls are exchanged between an operating system's kernel service and running application process. Every resource adaptive process running in a host generates a huge number of system calls. The raw system calls are unstructured and need a significant level of fine-tuning before extracting any meaningful information from it. By capturing the complete system call traces, a generalized working approach of the processes can be understood. As system calls are operating system level highly sensitive information, there is only few research group who came up with system call based datasets. System call based HIDS gained significant attention in the last two decades because of the increasing number of attacks in Linux servers. System call based HIDS has not only been developed for individual hosts but also developed for virtual hosts and embedded platforms [3]. As the real-time system call generation increases exponentially with the number of an active process, HIDS needs to handle those large amounts of system calls very efficiently for the training and evaluation process. Unlike NIDS, HIDS needs to be installed in all the individual host to analyze the system calls and needs to be connected to a central server to achieve generalization. For these reasons, research groups from this field aim to accomplish a HIDS with lesser training time with improved accuracy and lower computation. According to [3], the traditional system call based HIDS does not reach robust performance as per the expectation. So, various research work in this area is under progress and has ample scope of further development. OSSEC [70] is one of the open-source commercial tools that is used for detecting host base attacks.

HIDS can be classified into two classes, such as operating-system-level intrusion detection systems and application-level intrusion detection systems, based on the audit data it analyzes [71]. Operating system-level IDS uses the information generated by low-level system operations such as system calls, file system modifications, and user log-on. As this information represents the low-level event stream, it is challenging to tamper with, unless the system is compromised at the kernel level. Application-level IDS are capable of analyzing the raw information produced by a running application

such as application logs.

Collaborative Intrusion Detection System (CIDS)

Collaborative IDS are an amalgamation of NIDS and HIDS. CIDS focuses on detecting intrusion on both the host and the network. As per the survey conducted by Vasilomanolakis et al. [72], CIDS can be classified in three ways according to their communication architecture, such as centralized, decentralized, and distributed. Researchers have mentioned about few mandatory requirements such as accuracy, minimized overhead, scalability, resilience, privacy, self-operation, and interoperability to build the CIDS that can protect large IT systems. A Traditional CIDS lacks state summarizing and comprehensive analysis of network traffic and host logs.

2.2.2 Categories based on the types of methodology

This category of systems majorly concentrate on the type of methodology an IDS analyzes. It can be further sub-categorized into Misuse Detection system, Anomaly Detection system, and Stateful Protocol analysis [3, 73]. Figure 2.1 depicts the categorization.

Signature-based (knowledge-based)	**Anomaly-based (behavior-based)**	**Stateful protocol analysis (specification-based)**
Pros • Simplest and effective method to detect known attacks. • Detail contextual analysis.	• Effective to detect new and unforeseen vulnerabilities. • Less dependent on OS. • Facilitate detections of privilege abuse.	• Know and trace the protocol states. • Distinguish unexpected sequences of commands.
Cons • Ineffective to detect unknown attacks, evasion attacks, and variants of known attacks. • Little understanding to states and protocols. • Hard to keep signatures/patterns up to date. • Time consuming to maintain the knowledge	• Weak profiles accuracy due to observed events being constantly changed. • Unavailable during rebuilding of behavior profiles. • Difficult to trigger alerts in right time.	• Resource consuming to protocol state tracing and examination. • Unable to inspect attacks looking like benign protocol behaviors. • Might incompatible to dedicated OSs or APs.

Figure 2.1: Pros and cons of the IDSs categorized based on the followed methodology [24]

Misuse Detection System (MDS)

In the misuse detection system, libraries with attack signatures are created and stored. The Incoming traffic is compared with the stored attack signatures. If the traffic

matches the attack signature, it is considered as a malicious traffics. The systems try to capture all the abnormal traffic behaviors effectively to create attack signatures in order to detect the known malicious behaviours of the network. As signatures the previously defined by the system, every traffic flow deviates from the stored signatures are considered as nonmalicious [3]. So, these types of systems are not capable of detecting unforeseen novel attacks. As a result, misuse detection systems have a high amount of false negative rate. These systems are not at all effective options for detecting zero day and low footprint attacks. If an intruder gets access to the signature libraries, it is an easy job for them to bypass that. But, one of the advantages of this rule-based approach is that the detection rate is impressively high.

Anomaly Detection System (ADS)

Anomaly detection systems are not required to compare the incoming traffic with any of the existing signatures. These types of systems can detect zero day novel attacks. Anomaly detection systems can be further classified into Network Anomaly Detection System(NADS) and Host-based Anomaly Detection System(HADS). Chandola et al. [74] and Liao et al. [24] have provided a detailed and comprehensive idea about different types of IDS in their survey research paper. According to Liao et al., the detection approach for an anomaly can be classified into five categories, such as statistical-based, pattern-based, rule-based, state-based, and heuristic-based.

As per a few of the recent researches, system call based HADS are created by building a normal profile of system calls by training a model only with nonmalicious calls [74]. This training can be executed using data mining and artificial intelligence techniques, and a generalized platform for in-depth normal behaviors of a system is created. HADS are usually deployed in the hosts where the normal nature of the system call traces does not change rapidly. The malicious system calls are compared to the normal profile to understand if it is deviating significantly and considered as malicious if it deviates more than a specific threshold configured by the system. These types of detections are less dependable on operating systems and very efficient for detecting privilege escalations. Another popular approach is to use both malicious and nonmalicious system calls to train the AI-based models and use it for attack detection and classification.

It is a challenging task to create a generalized normal profile for an Anomaly detection system based on system calls due to the rapidly evolved computer systems' constant change of observed behaviour. Two of the major advantages of Anomaly detection systems are the unavailability of the system during profile generation and real-time detection.

Stateful Protocol Analysis (SPA)

In this approach, the IDS analyzes the trace and protocol by pairing port requests with replies [24]. Unlike anomaly-based detection, SPA depends on the vendor-developed generic profiles of the specific protocols that are based on protocol standards from an international standard organization such as IETF. SPA is a specification based detection that can detect the unexpected sequence of commands. This detection process is resource-intensive as it needs to analyze the protocol state information. Unlike system call based HADS, it is unable to detect the attacks by only getting trained with benign protocol behaviour.

2.3 Literature Review

In recent years, researchers innovated techniques to detect host specific attacks. They have used different data sources, detection strategies, and methodologies to accomplish their intended research gaps. In this section, I will present the development in this field in the last twenty years. As the attack detection approaches have evolved over the time with respect to modern attack generation, the literature review can be categorized into five categories based on their detection approaches [3] as follows; 1) Enumerating Sequence approach, 2) Hidden Markov Model based approaches, 3) Machine Learning based approaches(Non-Neural Networks), 4) Neural Network based approaches, and 5) Rule-based and Filter based approaches.

2.3.1 Enumerating Sequence

Forrest et al. [9, 11, 75] primarily introduced the method of detecting anomaly using the system call sequences. The researchers have introduced a method for detecting intrusion at the level of privileged processes. As per their claim in their previous

research works, short sequences of system calls were used to discriminate between normal and abnormal characteristics of the Unix processes. A normal behavior profile was created by collecting data synthetically from a live user environment by tracing the actual execution of a program. This approach is based on an enumeration sequence-based method known as STIDE (sequence time delay embedding) [9] which is simple and efficient while detecting anomalies in real-time. In this technique, the sliding window algorithm is used to generate short sequences of system call traces and then a database with the signature of the normal behavior is created with the short sequences. For efficiency purposes, the short sequences are stored as trees in which each tree rooted as a specific system call. The size of the database is usually described by the number of unique short sequences. As the short sequences are stores as trees, the storage management of this process is significantly efficient. During the anomaly detection, the short sequences are extracted from the testing sequences and compared against the normal profile database. Intrusion is characterized by analyzing the number of mismatches between the testing system calls and normal profile behavior. If the normal database includes at the possible variation of a normal behavior, then a test sequence that has mismatch count more than a specific threshold considered as malicious activity. In the further course of the research, Hofmeyr et al. [10] have also applied Hamming distance to calculate the amount of mismatch between the test sequence and the normal sequences. If the Hamming distance is larger than the user-defined threshold value, it denotes that the test sequence is deviating significantly from the normal profile to be considered malicious.

Challenges of Enumerating Sequence based approach:

It is a challenging task to capture all possible normal behaviour to build a generalized normal profile. As every individual program creates a normal database for itself, updating and managing a normal database is a complex and computationally expensive task. Efficient methods need to implement to execute the job of building normal profiles [3]. In the process of finding an anomaly, each test sequence needs to be compared with all the entries in the normal profile to achieve the optimum hamming distance to compare with the pre-defined threshold. If the normal profile is long enough or the test sequence is many in numbers, the process of detection gets slower and computational complexity increases significantly.

2.3.2 Hidden Markov Model (HMM) based approaches

In this subsection, the research development regarding the application of the Hidden Markov Model in detecting anomalies is discussed thoroughly. Hidden Markov model is the simplest form of a dynamic Bayesian network that contains finite numbers of the hidden state. In this model, each state is not visible, but the output that is dependent on the state is visible. HMMs can be trained with several states according to the number of system call types. For each system that can be transmitted through the model, the probability of state movement and the system call generation is stored. If the probability of the incoming system call is under a specific threshold, it is considered as an anomaly. Warrender et al. [75] have initiated the application of HMM for building a HIDS using system calls. The researchers have trained the HMM with several state records which are defined by the number of types of system calls in a program. The states are internally connected to each other which are reachable when required. As the state movement probabilities and system call generation are stored, this approach is heavily expensive in the field of computation and storage. This proposed approach by Warrender and their colleagues need to examine each system calls separately and calculate the probability of the state movement. If the probability of a system call does not satisfy the pre-defined threshold, it can be considered as an anomaly. As mentioned by the authors, this HMM training required a huge amount of time and resources.

Qiao et al. [76] propose a HMM based approach where it processes the system call traces as an input and transforms it into a sequence of hidden states. They applied the enumeration method to build the normal database based on the sequence of hidden states. A part of pre-processed system calls is considered to train HMM filter using the Baum-Welch algorithm. All traces are passed to the HMM filter to develop the state transition sequence using the Viterbi algorithm. One of the major drawbacks of this research work was the creation of incomplete and inconsistent databases because a complete database of hidden sequences cannot be created from an incomplete input of system call sequences due to the one to one relationship.

Hoang et al. [77] propose an approach that addresses the incomplete database issues by creating a multi-layer model. This proposed model is an amalgamation of the enumeration method, frequency based approach , and HMM. This model is a

combination of two phases: the training phase and testing phase. In the training phase, normal database and an HMM are created with system calls generated by normal program execution [78]. In the testing phase, test sequence is compared against the normal database. If any mismatch is found, sequences are sent through the HM model for further verification. As per the researchers, this two-way verification process reduced the false positive rate significantly but still requires high computation if test sequences are considerably large. This approach is effective for false positive reduction but inefficient for online detection.

To address the issue with the training time and resource utilization by HMM, Hoang et al. [79] have extended the research further. The authors trained the HMM using a modified version of multiple observation sequence algorithm(HMMMOSA). In this method, long system call sequences were divided into sub-sequences and each sub-sequences are used to train sub-models. The sub-models are merged to build the final model. This incremental approach can be used to further reduction of resources and training time.

Haider et al. [12] randomly sampled Host-based IDS data at different time intervals to analyze the frequency of system call identifiers, sequences of which can be fed into Hidden Markov Models (HMM's). They found little difference between anomalous (attack) sequences and normal sequences. They propose the use of a stochastic model to detect anomalous behaviour on Linux-based Intrusion Detection Systems. They collected and encoded sequences of system calls (SS) from simulated attack vectors. These sequences are used to extract Feature State (FS) information of these SS sequences. Feature states represent characteristics of the sequence, for example, the frequency of system call identifiers in the sequence. Transition-less FS chains are used to generate a summary of system states (SSS). Considering each summary SSS as a single state, the external HMM, includes underlying states, to compute Complete States (CS). Both FS and CS chains are used to develop a Nested-Arc Hidden Semi-Markov Model (NAHSMM). Their approach includes state summary (FS chains) that adapt to varying lengths of state duration. They apply the FS chain, using SSS, to cope with large data (as captured on cloud servers) in a condensed form, to reduce computational and storage costs. These Feature States capture the highest and least appearing system call identifiers in a given sequence for their count and concerning

their encoded integer value. Once the state summary has been constructed and established, they are then used to train and test the detection engine. The proposed algorithm trains the engine to learn normal System State behaviour given a tolerance threshold. The resulting model produces a lower false alarm rate, a higher detection rate, at half-the processing time of a regular Hidden Markov Model. Accuracy and training costs were measured to evaluate the model. For further research, the scalability and resilience of the model were evaluated too. Though this approach is very efficient for detecting an anomaly in the cloud servers, generalizing a normal profile is still a challenge. The methodology developed by Murtaza et al. [80] focuses on the trace abstraction technique that helps to reduce the training time without affecting the detection accuracy. They represented the system call traces as kernel module interaction and use the representation as an input for the anomaly detection techniques such as Sequence Time Delay Embedding(STIDE) and Hidden Markov Model. The researchers have experimented on the three publicly available datasets namely UNM, Firefox-DS, and ADFA-LD. Using Kernel module trace has significantly reduced the training time of the model but the accuracy and detection rate was not compromised. The STIDE technique works better than HMM while detecting the false positives, but both the model performs similarly if the accuracy is the main method of measurement.

Challenges of HMM based approach: HMM based models are criticized for high time complexity and computation complexity. As most of the proposed approaches mentioned about training HMM with short sequences, the system call traces with longer length takes considerable time for getting detected as an anomaly which is unacceptable for online detection. As the system call database for creating a normal profile increases exponentially, performance tends to degrade for the HMM models. In some of the research works, it has also been mentioned that deep system call patterns cannot be mined by single layer HMMs. [3, 12].

2.3.3 Machine Learning and Clustering based approaches(Non-Neural Networks)

In this section, research development regarding the application of machine learning techniques in detecting anomalies is discussed. Haider et al. [8] have developed a

Host-based Anomaly detection system by utilizing the ADFA-LD [14] dataset as a data source. This dataset has low footprint attacks that make both the normal and abnormal host data homogeneous and difficult to separate. This specific of ADFA-LD represents the modern cyber threat environment. The research team has observed the drawbacks of the existing approaches while detecting anomalies in the ADFA-LD dataset is the inability to extract features that represent the system calls. The researchers have proposed a character data zero watermark inspired global statistical features acquisition strategy for integer data for extracting the reliable and hidden features from the systems calls [7]. Three machine learning algorithms, namely, KNN, SVM with linear or RBF kernel were used to train and test the host-based ADFA-LD dataset. As the low footprint attacks data are nonseparable from the normal data, SVM with RBF kernels was used in order to perform an effective non-linear separation. The major purpose of the kernel was to transform a data point into a new feature space where it is possible to separate normal and low footprint attack data using SVM. The researchers have used the nonparametric and computationally inexpensive KNN model to train a profile with normal behaviours of system calls and then evaluated by calculating the deviation of a test system call from that normal profile. One of the main drawbacks of this model is it becomes significantly slow when the size of the data increases rapidly. Several experiments were performed with the values of k and euclidean distance in order to find the similarity of the data point with its neighbor.

Yuxin et al. [81] proposed a behavior-based detection approach using semantic analysis. The researchers have presented an executable as assemble code and generated a control flow graph from it. From a control flow graph, the system call execution path is extracted and merged to build a system call stream from an executable. The authors have used the decision tree algorithm to classify the system call sequences. Decision trees are used for inductive learning and it is a method for approximating discrete-valued functions. The internal nodes of a decision tree represent an attribute and the leaf nodes correspond to class labels, and the path from the root to leaf defines the classification rule. Decision trees classify the examples by sorting the tree from the root to a leaf node. It builds such a tree that checks the best set attribute for each instance and split the training sample into proper classes. This proposed

approach was compared with the dynamic detection approach and reported a higher accuracy with a lower false positive rate.

Aghaei et al. [82] proposed a method for building a host based intrusion detection system which is a combination of semi-supervised one class learning algorithm and Principal Component Analysis(PCA) based feature extraction technique. The researchers have defined the one class classification as combining the target class probability function with artificial class density function to estimate the target class density by using Bayes theory. PCA [83] based Eigentraces method was used to extract the representative features. They have majorly used two learning networks such as Radial Basis Function neural network and Random forest to train and evaluate the proposed methodology. Benchmark dataset ADFA LD [14] was used to simulate and evaluate the experiments. Th have reported high performance for detection malicious system calls in terms of accuracy, precision, recall.

Xie et al. [84] have proposed a frequency based feature retrieval technique from the system call traces to detect the anomaly. They have addressed the research problem of larger training time required for the short sequence based intrusion detection systems. The researchers have used the ADFA-LD12 dataset to evaluate their proposed framework. The ADFA-LD dataset is generated in a Linux machine that had kernel version 2.6.38 which provides 325 different system calls. Every system call in a trace is indexed with system call numbers which vary from 1 to 325. The authors developed a technique to calculate the number of frequency of an individual system call in a trace. The frequency based process converts the system call traces of different length into equal sized frequency vectors. Because of the sparseness of frequency vectors, PCA was used to reduce the dimension that reduces the computation cost of the model too. The researchers used 2 frequency-based machine learning algorithms such as k-Nearest Neighbour(kNN) [84–86] and k-Means Clustering(kMC) [87] to detect the anomaly. Xie and their colleagues have compared the performance of the learning models for detecting each type of attack. The kMC algorithm outperformed the kNN in terms of accuracy and computation time. In [88], Xie et al. extended the research by applying one class SVM on ADFA-LD dataset based on the short sequences of system calls. In this approach, the repeating short sequences were dropped to differentiate the anomalies from the normal profile efficiently. The researchers were able

to reach an acceptable performance by keeping the computation cost low. They have also suggested some improvement in their existing methodology by applying some weight to the short sequences which maximize the separability between normal and abnormal activity.

Challenges of Machine Learning based approach: Using Machine Learning models in a distributed system is very challenging. In a distributed network, HIDS components are installed in each individual host and their training is performed in the centralized server. Every individual host creates different types of normal profiles as per the system calls it generates. Training a machine learning model with differently structured data in a parallel manner is complex and computationally expensive. Most system call based datasets contain a large number of normal traces and very few numbers of anomalous data. This type of data imbalance is actually a generalized representation of a real-world scenario. When the machine learning models are trained with heavily imbalanced data, it tends to overfit towards the most occurring class which is inefficient to detecting true negatives. Therefore, different strategies such as parameter tuning, semi-supervised methods were used to tackle these problems.

2.3.4 Neural Network based approaches

Kim et al. [89] proposed a hybrid methodology which is an amalgamation of Long Short-Term Memory(LSTM)-Based System Call Language Modeling and robust ensemble method to design a HIDS. As per the argument presented by the researchers, the traditional methods are not capable of capturing the call level and phrase-level features. Kim et al. propose to create a language model(In natural language processing (NLP), a language model represents a probability distribution over sequences of words) of system calls that learn the semantic values of individual system calls and the relationship between multiple system calls which represents a different meaning altogether. Deep Recurrent Neural Networks(RNN) have shown significant performance in other fields(machine translation, text summarization) where sequence-based and temporal modeling is required [90]. In the proposed approach, the individual system calls are considered as words and sequences of system call sequences as a sentence. The system calls correspond to the language of communication between the user and the system. In this approach, a neural language modeling technique, LSTM was

used to develop and train the language model for enhanced long-range dependence learning. While training the language model, backpropagation through time algorithm was used to minimize the cross-entropy loss which maximized the likelihood of a system call sequence. In order to ensure efficient false positives reduction, an ensemble method based integrated classifier build the team.An integrated classifier is a group of threshold classifiers that is individually trained with system calls that have a high probability of being normal. The threshold classifiers are merged into a robust classifier to reduce the number of false positives significantly. Leaky Rectifier Linear based function on the ensemble method was used to maintain the linearity of the method. Creech et al. [91] have proposed a detection mechanism using Extreme Learning Machine (ELM). ELM was used to train the word and phrase dictionaries formed with various lengths of system call sequences. The key characteristic of ELM is, it can complete its training in one pass/iteration using the Moore-Penrose pseudo-inverse to solve a least-squares equation, hence avoiding traditional training problems associated with neural networks. As per the authors, the higher training speed consumes high resources but that does not impact the training as HIDS should be installed in every host individually. While training the ELM, the weights between the hidden layer and output were adjusted efficiently, whether the weights between the input layer and the hidden layer along with bias values can be randomly and statistically assigned. The training process of ELM is significantly faster than the traditional deep learning methods which also avoids local minima concerns.

Chawla et al. [92] has proposed a framework which is a combination of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), to detect malicious system calls. The group of the researchers proposed to use CNN to extract the local features of the system call sequences and use this as an input to the RNN layer. The output of the RNN layer was processed through a fully connected softmax layer that outputs a probability distribution over the system calls processed through the network. While processing the input sequence with CNN, filters slide over the system call sequences to find the local pattern in the system calls. This technique creates a representation of fixed-sized contexts and effective context size is increased by stacking several layers of CNN. The model was trained on normal system calls predicts the probability for next system call. The trained model was used to

predict the probability of a testing sequence to detect a malicious trace according to the pre-defined threshold. The researchers have claimed to reduce the training time of the RNN based Long Short Term Memory model but could not achieve a state of the art result achieved by Kim et al. [89]. Chen et al. [93] proposed an anomaly recognition and detection technique which performs probabilistic inference. The techniques revolve around building an incremental network that analyzes the feature dependency and develops a self-structuring method by learning a confabulation network. This network continuously refines its knowledge with the help of streaming unlabelled data. The experimentation was executed on the graphic processing unit and Xeon Phi processor which performs better than the general-purpose microprocessor. The framework offers real-time detection on a concurrent data stream that has different contextual knowledge. Chen and their colleagues have achieved high computing performance with memory efficiency.

Challenges of Neural Network based approach: Deep neural network models are data-hungry and computationally expensive. In a real-time environment, the system call produced by a computer system increases exponentially with the usage of the application. Though the quality of the training increases with this exponentially growing huge amount of data, that increases the computation complexity too. One solution to this problem is using GPU in each host dedicated to the HIDS that ensures parallel processing. Efficient strategies need to be followed to maintain a large number of system calls and their training using the best hyperparameters [3].

2.3.5 Rule based and Filter based approaches

In [3], a survey regarding the development of HIDS based on Rule and Filter based approaches has been presented. Lee et al. [94, 95] proposed a framework for developing, designing, and evaluating an IDS. In a traditional rule based framework, a sliding window was used to extract unique system calls from the normal system call traces. The short sequences are then extracted from malicious traces and matched against the normal short sequences. If the exact match occurs, it is considered as normal, else malicious. This process was inefficient as a malicious system call trace also contains a large number of normal system calls. The researchers have applied RIPPER rule learning to the training data. The RIPPER rule outputs if-then rules on the training

data. The same rule is applied to testing system call traces. If the "if-then" rule generated from the testing sequence deviates from the pre-defined rule, the trace is considered as anomalous. Tandon [96] designed an approach based on LEARD rule and created rules based system calls sequences and arguments. Ye et al. [96] proposed an approach based on rough set theory to differentiate the malicious system calls from the normal traces. Minimized set of rules were extracted in order to define the normal profile. Ying et al. [97] proposed a log file analysis technique as a part of Host based Intrusion Detection System. The autors have collected system level log files and application level log files and decoded them to extract useful information such as user name, source IP address.After pre-decoding and decoding of rules, the authors have made constructed rules based 400 existing OSSEC rules. The combined the rule based method with Back propagation network based anomaly detection system in order to increase the efficiency and accuracy of the proposed system. Zimmerman et al. [13,98] proposed a novel approach to build a policy based HIDS based on information flow control at the operating system level. The proposed technique enforces restriction on the operation that restricts the activity to exploit the unprecedented behaviour such as buffer overflow, race condition, and confused deputies. The goal of the experiment was to develop a policy based HIDS that ensures high reliability and high detection rate. In this research, two set of result were presented, one experiments was done on a controlled environment and other was executed on operational server. The result from the experiment fulfilled the researchers expectations.

Wang et al. [99] proposed a anomaly detector, Anagram that models a mixture of higher order ngrams. High order ngrams were used to generate extract a robust signature to increase the generalization while creating normal profile. The Anagram content models were implemented using Bloom Filter which reduces the space complexity and ensures the privacy preserving cross-site correlation. The proposed system were used to detect mimicry attacks which with high accuracy and significantly lower false positives. The authors have implemented a feedback loop based on gained knowledge by the system that helps to improve the detection efficiency over the time. The framework was not only used in the stand alone sensors, but also a perfect fit for fault tolerant host-based environment.

Challenges of Rule and Filter-based approach: Most of the experiments

using the rule based approach executed on small scale datasets. Though extracting rules from non malicious system calls and creating a normal profile is time efficient and involves less computation but false positive and false negative detection rate cannot match to the state of the art techniques. Rule based approaches fail to mine pattern and rules from the deep and long traces. One of the major drawback of the bloom filter based approach is it allows the false positives. Even if this approach is fast and computationally inexpensive, a major improvement needs to done in order to use it for detecting anomalous system calls.

Table 2.1 depicts an comparative representation of different detection approaches for system call analysis.

Table 2.1: Summary of Literature Survey

Detection Approach	Papers	Advantages	Challenges
Enumerating Sequence based approach	Forrest et al. [9, 11, 75], Hofmeyr et al. [10]	• Simple and efficient for creating normal database. • Acceptable approach for fast real time detection. • Short sequence-based approaches are highly efficient when traces are small.	• Almost impossible to capture all possible normal short sequences to build a generalized profile. • Updating and managing the normal database is complex and computationally expensive. • Detection process gets slower if the traces are significantly long. • Does not consider the effect of long trace or sequence-based features of a trace.
Hidden Markov Model based approach	Warrender et al. [75], Qiao et al. [76], Hoang et al. [12, 77, 78]	• Moderate detection rate. • Mostly used for theoretical researches.	• High computational complexity and time complexity. • Higher false positive rate. • Traces with longer length take significantly more time to be detected which is not ideal for online detection.
Machine Learning and Clustering based approach	Haider et al. [8] Haider et al. [7], Yuxin et al. [81], Aghaei et al. [82]	• Moderate accuracy and detection rate.	• Machine Learning models tend to overfit while training on highly imbalance data. Needs to find the optimal hyper parameter which is computationally expensive

Detection Approach	Papers	Advantages	Challenges
		• Efficient for real-time detection. • Adaptive towards the novel attacks.	• Training machine learning model in a distributed manner is challenging and a complex process.
Neural Network based approach	Kim et al. [89], Creech et al. [91], Kim et al. [89], Chen et al. [93], Chawla et al. [92]	• Moderate accuracy and detection rate. • Efficient for real-time detection. • Adaptive towards the novel attacks.	• Training can be computationally expensive depends on the configuration of the network. • Neural Network based models are data hungry. The quality of the training increase with the amount of the data but becomes computationally expensive eventually.
Rule based approach	Lee at al. [94, 95], Ye et al. [96],Ying et al. [97], Zimmerman et al. [13, 98]	• Extracting rules from the short trace is fast and the rule-based detection is time efficient.	• Not able to mine pattern from the long traces. • Not able to detect the novel attacks which does not match with the existing rules. • Some filter such as bloom filter allows false positives.

On the basis of the conducted literature survey, I analyzed and refined research gaps further and formulate the research question and hypothesis in the next chapter.

Chapter 3

Research Problem

After conducting the literature survey, I realized that the techniques for retrieving the features from the system call traces are one of the major factors that control the nature of the traces for the evaluation of decision engine. The hidden features can be extracted by the feature retrieval process that differentiates the malicious system call traces from the normal system call traces. It has been observed from the literature survey that the researchers have experimented with sequence based feature retrieval or frequency based feature retrieval or hidden watermark based feature retrieval. Just using one feature retrieval technique to extract the feature leads to a significant information loss. As an example, if the features are retrieved using only frequency based technique, then we ignore the sequence and hidden watermark related attribute of the system call traces. Retrieving the efficient features directly affects the classification capability of the classifiers. The recent development in this domain has indicated that the false alarm rate and detection rate of system call based Intrusion Detection Systems have improved by the course of the time but my proposed solution will focus to improve and enhance those metrics even more. One of the major concerns that showed HMM, neural network, and machine learning-based approaches are the high computational cost. Dimensionality reduction techniques need to be applied efficiently to extract the necessary, unbiased, uncorrelated features from the pool of retrieved features. Representing the feature in lower-dimensional space reduces the computational and time complexity of the learning learning models. So, the proposed framework will be focused on two major gaps identified during the literature survey.

- As low footprint traces are inseparable from the malicious traces, feature retrieval techniques play a major role to extract the representative hidden information that differentiates the malicious traces from the normal traces. The proposed framework focuses on an efficient hybrid features retrieval technique to improve the detection rate and false positive rate. This research work will

also address the trade-off between false positive and false negatives to achieve an optimal solution.

- Feature vector constructed using frequency-based modeling approach is large in length as every individual system call elements in the original dictionary are responsible to generate one element in the feature vector. The proposed framework will also focus on dimensionality reduction by representing the original feature vector into a lower-dimensional feature space that improves the time complexity of the learning models without deteriorating the performance of it. The training time of the proposed models will be measured in order to justify the improvement of time complexity. Reducing the training time is a very importannt measure for realtime detection. As the volume of system calls increase exponentially with the number of active processes, the training process becomes more complex. In a distributive achitecture, the HIDS becomes unavailable while training. The longer training time of an HIDS increases its chance of being attacked due to its unavailability.

Chapter 4

Data Sources and Datasets

In this chapter, I will introduce different information sources used to evaluate host based intrusion detection systems. This chapter will mainly focus on discussing different system call-based datasets but also provides a brief outline regarding other information sources such as system calls, application logs, and system logs.

Some of the host-based information sources in Linux/Ubuntu OS are Accounting, Syslog, and Linux Audit. Accounting maintains the record regarding the resource usage, such as memory, disk, CPU, network usage, and the application or processes invoked by the users. Syslog is an audit service made available by the OS to the application program to store the logs generated by them. Audit enables users to perform various tasks such as mapping a user to a process, generating audit reports using 'aureport', and filtering events of interest at different levels. Other publicly available tools such as strace, Itrace, LTTng, BSM can also be used in order to trace and capture the system calls [3]. The information to evaluate an HIDS can broadly be categorized into three classes : 1) System calls and sequences , 2) Application and process logs , and 3) System logs.

System Calls and Sequences: Linux provides options to monitor system calls from the command line [100]. These commands can be used to detect file access, program executions, and network port access.
Microsoft Windows also provides auditing systems that can be leveraged to perform host-based intrusion detection. There are three types of logs produced by the auditing facility: system logs, security logs, and application logs.
The system log (SYSEVENT.EVT) contains events that are related to windows services and drivers. It can track events during system startup, as well as hardware and controller failures.
The security log (SECEVENT.EVT) tracks security-related events such as log-ons, log-offs, changes to access rights, and system startup and shutdown, as well as events

related to resource usage.

The application log (APPEVENT.EVT) contains events generated by applications.

Application and Process Logs: Logs generated by the applications can also be considered as one of the most important sources of audit data. " Application audit data is rich, reliable, and focused. Therefore, it is easy to determine which program is responsible for a particular event. On the downside, application data is also very specific and different applications have to be dealt with on an individual basis by the HIDS [71]."

System Logs: System-level log files can be used to evaluate the host-based intrusion detection system. Log files can be pre-decoded to extract three parameters; Time/date, Hostname, and Program name. Then critical information can be extracted by using regular expressions. For example, depending on the type of log, the following data can be extracted: Source IP address, Username, and Log. A rule tree can then be constructed with the help of the OSSEC system [70] and can be used to determine if any suspecious or malicious event is observed.

4.1 Datasets

In this section, I will discuss different system call-based datasets that were considered to evaluate our HIDS. I conducted a comprehensive search in order to select the most relevant dataset that represents attacks on contemporary operating system environments such as Linux and Windows. In the following sub-sections, we explore popular, publicly available system call-based datasets and their features.

4.1.1 Firefox-DS

Murtaza et al. [17] developed this dataset on Linux using contemporary test suites and hacking techniques. This dataset contains both the normal and anomalous system calls. It was created using modern penetration testing techniques such as Metasploit. Normal system call traces are collected by executing seven different testing frameworks on Firefox 3.5. Each test framework was responsible to execute a different set of operations which covers most of the functionalities of Firefox in order to ensure the completeness of normal behavior. The researchers have achieved 60% source code average, 5931 passing test case files, 1.3 TB of traces, and average 19,000 - 4000,000

system calls per test case files from seven different test suites. This dataset includes five state-of-the-art attacks including memory corruption exploits, integer overflow, dangling pointer exploits, DOM exploits, and null pointer exploits. [3].

4.1.2 ADFA-LD12

Creech et al. [14] proposed the ADFA-LD12 [8, 101–103] dataset and Linux version 11.4 was selected as a host operating system for executing all the operations required to generate the dataset. In order to instantiate the web attack vectors, Apache version 2.2.17, running PHP version 5.3.5 was installed. In the patched operating system, FTP, SSH, and MySQL version 14.4 were run as services and a web collaborative tool TikiWiki version 8.1 [104] was installed.

The host system was configured in such as way that it represents a generalization of the modern Linux based processes such as file sharing, database service, remote access, and web server functionality. The researchers considered the intelligent methodologies followed by the new generation attackers while generating the dataset to increase realism and generalization. The target server for the ADFA-LD12 dataset was fully patched and TikiWiki was used to to introduce flaws.

ADFA-LD12 includes six different attack payload such as password brute force attacks, privilege escalation, Java-based Meterpreter, Linux Meterpreter, and C100 Webshell. The password brute force attacks are mostly the last resort of an attacker because of their prominent and large footprint.

In ADFA-LD, password bruteforce attacks were simulated on the open services such as FTP and SSH which represent a realistic threat on the service and are exposed to the external sources.

Privilege escalation was implemented by a client side attack were a payload is included into a Linux executable using Metasploit [6] and uploaded to the server to simulate social engineering attacks.

The Meterpreter-based attacks were initiated using open source Metasploit framework. In the process of generating this attack, Tikiwiki was used to upload the Java vulnerability payload which resulted in a reverse TCP connection to the root system. Once the shell was established, attacks like privilege escalation and installation of backdoor tool was carried out.

In a similar manner, Linux executable Meterpreter was uploaded to the host using social engineering or similar convenient platforms. Lastly, a PHP based remote file exclusion attack was generated to upload C100 Webshell to manipulate host system and escalate privileges. C100 shell provides an illegal GUI to the attacker in order to manipulate the operating systems.

As ADFA-LD12 was developed for anomaly detection, it has three different groups in the dataset; training, validation, and attack. In the training and validation dataset, only the normal raw system calls are included which are recorded while running normal activities in the host such as web browsing and file preparation. Traces were generated using *auditd* and were finally filtered by size in order to avoid unnecessary processing. The normal group contains 833 traces and validation group contains 4373 traces. In the pool of attack data 10 attacks were included for each attack vector. Table 4.1 provides an overview of ADFA-LD12 dataset and the distribution of the trace vectors.

Table 4.1: Distribution of the different types of traces in ADFA-LD12

Type of trace	Label	No of traces	Payload/Effect	Process of the vector generation	Filtering size
Training	Normal	833	NA	Captured the normal operation in Linux local server including file sharing, database service, remote access.	[300 Bytes,6kB]
Validation	Normal	4373	NA	Captured the normal operation in Linux local server including file sharing, database service, remote access.	[300 Bytes, 10 kB]
Adduser	Attack	162	Privilege escalation. Add new superuser with admin privilege.	Client side poisoned executable	N.A.
Hydra-FTP	Attack	148	Password brute force	FTP by Hydra	N.A.
Hydra-SSH	Attack	91	Password brute force	SSH by Hydra	N.A.
Java-Meterpreter	Attack	125	Java based Meterpreter	Tiki wiki vulnerable executable	N.A.
Meterpreter	Attack	75	Linux Meterpreter payload	Client-side poisoned executable	N.A.
Webshell	Attack	118	C100 Webshell	PHP Remote File Inclusion vulnerability	N.A.

4.1.3 NGIDS-DS

Haider et al. [105] developed the NGIDS-DS dataset with the help of next-generation infrastructure at Australian Centre for Cyber Security (ACCS) in the University of New South Wales and Australian Defence Force Academy (ADFA), Canberra. This dataset is a collection of abnormal and normal host and network activities which were introduced during the emulation.

The researchers designed the dataset according to the guidelines provided by Davis et al. [106]. The IXIA Perfect Strom tool was used in order to generate both representative traffic and host-based connectivity. The tool provides major advantages such as producing a mixture of normal and abnormal cyber traffic, generating the highest number of attacks with different dynamic behaviors, creating profile of the cyber traffics of multiple enterprises, along with the automatic generation of ground truth. The test-bed architecture for developing NGIDS-DS can be divided into two parts, namely, Part A and Part B.

In the Simulation Phase, Part A was designed to generate a mixture of normal and anomalous traffic as well as operation timing that is related to five different enterprises such as e-commerce, military, academia, social media, and banks. Part B of the test-bed was used as a group of victim network.

Part A of the network replicates the real-world scenario that the attack vectors required to destroy an enterprise network are complex in nature and it might come from different directions. Part B represents a collective synthetic presentation of an enterprise network that can be a victim of an attack.

The abnormal/malicious traffic includes seven major attack labels; Exploit, DOS, Worms, Generic, Reconnaissance, Shellcode, and Backdoors. There were two different machines deployed in the network. In the first machine Ubuntu 14.04 was installed and an auditing mechanism was installed in order to act as a critical enterprise machine that runs several services such as storage, FTP, and email. The second machine has Ubuntu 14.04 and tcpdump installed on it in order to collect the traffic that was transferred from testbed Part A to Part B.

Table 4.2: Distribution of malicious and benign host logs in NGIDS-DS

File type	Feature	Quantity
Ground-truth.csv	Total Records	313926
	Attributes	7
99.csv files of host logs	Total records	90054160
	Attack records	1262426
	Normal records	88791734
	Attributes	9
NGIDS.pcap	Total Capture Packets	1094231
	Unique IPs	18

This dataset specifically concentrates on generating the traffic that resembles with the real-world network vectors and attacks. The researchers have followed certain techniques and guidelines in order to generate more realistic and reliable traffic data. NGIDS-DS was specifically designed for the Australian Defence Force Academy (ADFA) within a controlled environment.

Altogether, this dataset provides strong external validity which is a key factor for the research problem that is addressed in this document.

NGIDS-DS dataset consists of five different files: (1) groundtruth.csv; (2) 99 csv files of host logs; (3) NGIDS.pcap of the network packets; (4) feature-descr.csv; and (5) readme.txt.

The number of captured network packets are less than that of host records because there are 82 corresponding host logs for one network activity (packets). In the host logs, the ratio of the normal traffic and attack traffic is 90:1 that demonstrates the furtive behavior of the network activities in NGIDS-DS.

Table 4.2 provides a summarized quantitative description of NGIDS-DS dataset. Table 4.3 provides description of different features in NGIDS-DS dataset.

Table 4.3: Description of different features in NGIDS-DS dataset

Files	Feature Name	Type	Description
99 host logs csv files	date	date	Contains the date on which the particular activity of the NGIDS-DS is performed
	time	time	Contains the time on which the activity of the NGIDS-DS is executed
	pro_id	number	Represents the process of unique identification executed in the host during the simulation
	path	nominal	It contains the execution path of any process executed in the host
	sys_call	number	Contains the system calls identifiers executed by the processes
	event_id	number	Represents the event's unique identification which happened in the host

Files	Feature Name	Type	Description
	attack_cat	nominal	It contains the names of the main categories of the attacks such as Exploits, DoS, Worms, Backdoors, Shellcode, Reconnaissance and Generic. Also contains "null" where label=0 which means normal activity record
	attack_subcat	nominal	It contains the subtype of attack of main attack category such as the subcategory "Browser" belongs to the main category "Exploits". Also contains "null" where label=0 which means normal activity record
	label	binary	Contain 0 or 1 to show the main category of the activity whether normal or abnormal

4.1.4 UNM and DARPA

University of New Mexico (UNM) [15] dataset and Defense Advanced Research Projects Agency (DARPA) [16] dataset are two of the most commonly used datasets which were very popular for evaluation of HIDSs [107]. DARPA was generated in 1999 and UNM dataset was released in 2004. System call traces in these datasets are very simple which does not represent a real world cyber attack scenario. There are a huge number of researchers built intrusion detection systems using the these datasets because of the unavailability of datasets which represent the current state-of-the-art attack methods. DARPA dataset was created by the Defense Advanced Research Projects Agency and generated in a Solaris-based system. The DARPA dataset consists of both host and network auditing data but UNM dataset consists only of system call identifiers.

Chapter 5

Algorithms

In this chapter, I will unpack and provide refreshers on the algorithms that have been explored while building the proposed Host-based Intrusion Detection System (HIDS). It has been understood from the literature that an efficient HIDS mainly relies on three major components; 1) Efficient feature retrieval component that differentiates malicious traces from non-malicious trace, 2) Dimensionality reduction component to extract true representative, uncorrelated features and reduce the computational cost , and 3) Decision engines component to classify system call traces. This chapter is categorized according to these components and focuses discussing on algorithms which were explored to develop each of the module individually.

5.1 Feature Retrieval

As system calls are collected by different tracing tools, they are heavily unstructured and unorganized. Feature engineering and data pre-processing techniques need to be applied in order to extract representative features from system calls. In the area of system call analysis for detecting intrusions, two techniques are majorly used: Short Sequence-based System Calls Analysis and Frequency-based System Call Analysis.

The short sequence-based technique mines patterns from the short sub-sequences of the system calls but it does not consider the whole trace at a time. In this technique, a model is built with non-malicious patterns in order to build a normal profile and malicious system calls are compared against the normal profile. The procedure for building a normal profile is complex and time consuming.

In the frequency-based method, systems calls are transformed into equal-sized vectors on the principle of frequency of each system call in a trace. This procedure might experience some significant loss of information while building the normal profile. As the short sequence-based method does not consider a full trace, this approach will not be used to develop the system. Feature retrieval algorithms can be categorized

into 1) Frequency-based algorithms [18] and 2) Integer Data Zero Watermark-based algorithms [7].

5.1.1 Frequency-based algorithms

In sequence-based system call analysis, the traces of different lengths are converted into equal-sized frequency vectors. In the ADFA-LD12 dataset, each system call in a trace is denoted by an integer which is the index of that individual system call. The index of a system call is determined by the version of the operating system in which the system call traces were generated. As an example, Linux kernel version 2.6.38 provides a total number of 325 different system calls. In this frequency-based method proposed by Xie et al. [18] requires recording the occurrence of each system call in a trace and dividing the number of occurrences with the length of the system call trace in order to build the frequency vector. If the system call index has a range from 1 to p (maximum index of a system call in a trace) and f_i denotes number of occurrence of an individual system call indexed by i, where $i = 1, 2, 3....p$. Each element in the frequency vector can be represented as

$$\bar{f}_i = \frac{f_i}{|s|}, \tag{5.1}$$

where $|s|$ denotes the length of system call trace s. The frequency-based transformation converts system call traces into frequency vectors of same size. If m number of system call traces are provided and the highest index value of those system calls is p, the shape of the frequency transformed data will be $m * p$. The transformed data can be further used for feature selection and training a classifier.

Haider et al. [22] proposed a host-based anomaly detection system where they have applied Sine transformation and Fourier transformation to transform raw system call identifiers time-domain signal into frequency-domain signal. The researchers have used a sliding window on 1 second to log unit data and apply transformation strategies. In this 2-step process, input time-domain signal of system call identifiers in 1 second is transformed by sine transformation which is represented in the equation

$$f(x) = \sin(x)_t, \tag{5.2}$$

where x is considered as the system call identifier and $t = 1$ to T and T can be 1 second. Fourier transformation was applied to the sine transformed signal.The Fourier transformed components on the input signal are considered as sequence vectors which are used to train the learning models. Frequency domain vectors generated from non-malicious short sequence system calls are used to train and build a normal profile. Malicious systems calls are validated against the normal profile. Fourier transformation can be represented by the equation

$$f(\epsilon) = \int_{-\infty}^{\infty} f(x)e^{-}2\pi\epsilon^{\epsilon}, \tag{5.3}$$

where for any real number ϵ, x represents time in SI unit of seconds and ϵ represents frequency in Hertz.

5.1.2 Integer Zero Data Watermark (IDZW)

The principle of Zero Watermarking revolves around utilizing the actual context for a stream of information and generating a watermark based on it. The zero-watermarking algorithm does not change the characteristics of the the original data whose watermark needs to be generated. This approach is very broad and general in nature.

Most researchers have used this approach in different domains including detecting malicious modification in database relations [108], detecting tampering of text documents [109], and copyright authentication [110]. Jalil et al. [109] proposed a novel watermarking technique for checking text integrity and authentication. In this proposed strategy the text information that needs to be sent is not changed to embed the watermark which is one of the key principle for exchanging text documents between multiple parties. The context of the text was used to generate the watermark pattern and later the pattern is matched with a pattern matching technique in order to identify any tampering.

Haider et al. [7] proposed a reliable Host-based Anomaly Detection System (HADS) where a novel data zero watermarking strategy was developed to extract representative hidden features from system calls. This group of researchers have experienced two major challenges while constructing the HADS to develop strategies for detecting new generation attacks (NGA's). Firstly, NGA causes low footprint activities which

are very difficult to differentiate from the normal system calls in terms of accuracy and processing time. Secondly, it is very challenging to extract natural difference between normal and malicious system calls. Thirdly, considering system call values and handling strategies, deviation between the values may differ respective to the time and space complexity. This might introduce bias in the learning model which affects its outcome. Six different hidden features were extracted using the Zero-Watermarking technique. The process of feature retrieval described in Figure 5.1. In Figure 5.2, all the notations that were used in order to retrieve the abstract hidden features from the system traces are described in detail.

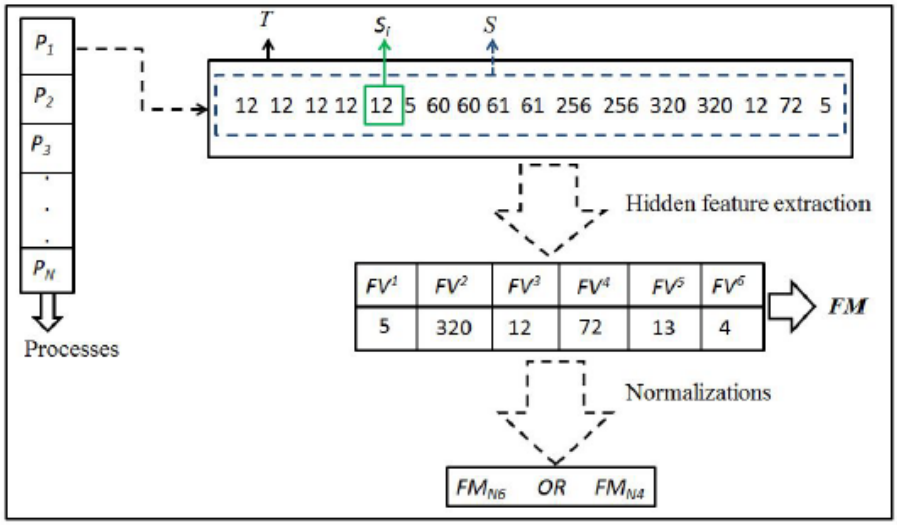

Figure 5.1: Zero-Watermark feature retrieval [7]

$$FV^1 = argmin_{T1:N \epsilon R} \forall_T \tag{5.4}$$

$$FV^2 = argmax_{T1:N \epsilon R} \forall_T \tag{5.5}$$

$$FV^3 = argmin_{T1:N \epsilon R} \sum_{i=1}^{N} \forall sm_{C_{T_{1:N}}} \tag{5.6}$$

$$FV^4 = argmax_{T1:N \epsilon R} \sum_{i=1}^{N} \forall sm_{C_{T_{1:N}}} \tag{5.7}$$

$$FV^5 = \forall_{Ti:N} \sum_{i=1}^{N} T_{i:N} \% 2 = 0 \tag{5.8}$$

$$FV^6 = \forall_{Ti:N} \sum_{i=1}^{N} T_{i:N} \%2! = 0 \tag{5.9}$$

Equation 5.4 extracts the system call with lowest index whether equation 5.5 extracts the system call with highest index value. Equation 5.6 and 5.7 analyzes the count of the same system calls in a single trace. $Argmax$ denotes the most repeated system call S_i in Trace T and $argmin$ denotes the least repeated system call in a trace. Equation 5.8 and equation 5.9 initiates the process to count even and odd system calls in a trace and assigned the results to FV^5 and FV^6. According to the research conducted in [7], normal and malicious system calls generate dissimilar feature vectors using the above equations which can efficiently represents the "natural" difference between normal and malicious system calls.

1) P =single process, P_N = N processes
2) S =system calls $\Rightarrow \{R^1, \ldots R^N\}$, R denotes real numbers
3) T =trace of Nth P, $T \epsilon$ S
4) S_i =i^{th}System call in T
5) FV^k =integer data zero watermark assisted hidden feature vector of T, where *k=1,2,3,4,5 and 6 or k =1,2,3 and 4 respectively*
6) $FV^k = FV^{kof6} \cup FV^{kof4}$
7) $FV^{kof6} = FV^1 \cap FV^2 \cap FV^3 \cap FV^4 \cap FV^5 \cap FV^6$
8) $FV^{kof4} = FV^1 \cap FV^2 \cap FV^3 \cap FV^4$
9) FM_6=Feature matrix of Nth trace with order $T \times FV^{kof6}$
10) FM_{N6}=Normalized $FM_6 \epsilon FM_{NZ6} \cup FM_{NP6} \cup FM_{NMD6}$
11) FM_4= Feature matrix of Nth trace with order $T \times FV^{kof4}$
12) FM_{N4}=Normalized $FM_4 \epsilon FM_{NZ4} \cup FM_{NP4} \cup FM_{NMD4}$
13) $FM = FM_6 \cup FM_4$

Figure 5.2: Notations for Integer Data Zero Watermark retrieval [7]

5.2 Dimensionality Reduction

In order to extract/select representative features from the retrieved features, I explored three different existing algorithms: 1) Principle Component Analysis [111], 2) Auto-Encoder, and 3) Random Forest-Recursive Feature Elimination [112].

5.2.1 Principal Component Analysis (PCA)

Principal Component Analysis is a multivariate feature extraction technique that captures the most valuable information from the input variables and drops the least important one [113, 114].

In PCA, the effect of highly correlated independent features is minimized and eliminated from the final calculated component. Each new variable after applying PCA is independent of the others. It is a linear dimension reduction technique that exploits the orthogonal transformation. In this process, the orthogonal linear combination of the variable with the largest variance is calculated and used for reducing dimension data.

Abolhasanzadeh et al. has described the process of PCA mathematically in [83]. For a given sample $\{x_i\}_{i=1}^{N}$ which has mean $x_{mean} = 1/n \sum_{i=1}^{n} x_i$, the covariance matrix can be written as $\sum = E\{(x - x_{mean})(x - x_{mean})^T\}$. If the spectral decomposition can be written in equation 5.10, which signifies the principal component transformation process generates a system that has mean zero and a diagonal covariance matrix $\wedge$, which has the eigenvalues of Σ. Finally, PCA generates a set of uncorrelated features that has a maximum variance. The hyperplane obtained from the first L principle components is considered as the regression that minimizes the orthogonal distance to the data.

$$\Sigma = \cup \wedge \cup^T \tag{5.10}$$

5.2.2 Recursive Feature Elimination with Random Forest (RF-RFE)

Random Forest ranks the importance of each predictor by constructing a multitude of decision trees [115, 116]. Each node of the tree has a different subset of randomly selected attributes, and the best attribute is selected based on various measures. Recursive feature elimination is a process that ranks the features according to the importance in the specific context. During each iteration, the significance of the features is calculated, and the feature with the least value is dropped. Another computationally efficient approach can be implemented by removing a group of features in every iteration.

In the case of evaluating highly correlated features, the importance of the same

feature can significantly vary while estimated over a different subset of features. The iterative approach is therefore practical for this feature elimination process. During training, a subset of the training set is selected in a Random Forest, and these are used to provide unbiased measures of prediction error. Figure 5.3 provides the pseudo-code of the RFE algorithm.

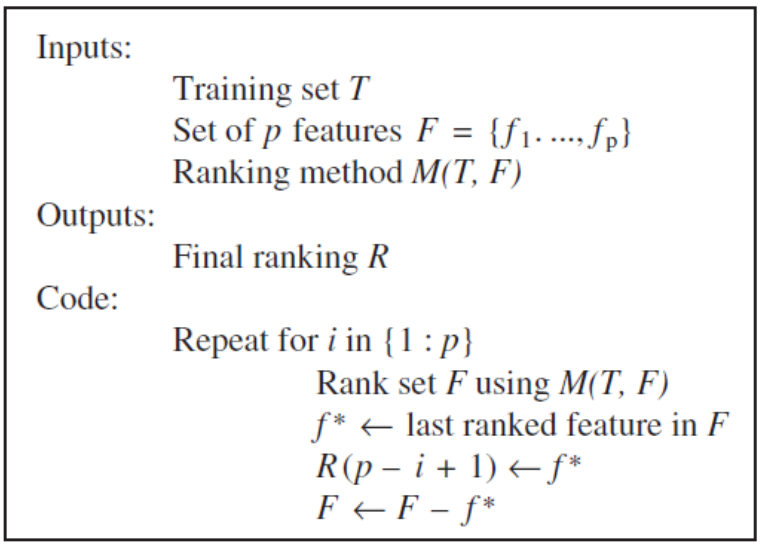

Figure 5.3: Pseudo code of RFE algorithm [115]

5.2.3 Autoencoder

Autoencoder is an unsupervised neural network which learns input vectors and reproduces them as an output [117]. Every autoencoder based network has two parts i.e., an encoder and a decoder. The number of neurons in the first layer of an autoencoder is the same as the number of features in the input data. Encoder module reduces the dimensions of input data as specified in the network configuration, and the decoder reconstructs lower-dimensional data to produce the output with the same proportion as the input.

Autoencoder can be used for dimensionality reduction by separating the encoder and decoder. The hidden layer of the autoencoder efficiently learns from highly dimensional data to generate output with premium quality that aligns with the original input. Autoencoder helps to compress data, visualize data in the lower dimension,

and extract hidden relation between the features that were not identifiable in a higher dimension. A hidden layer of autoencoder does not simply sub-select the elements from the previous layers, but it combines multiple features and represents original data in a lower-dimensional space.

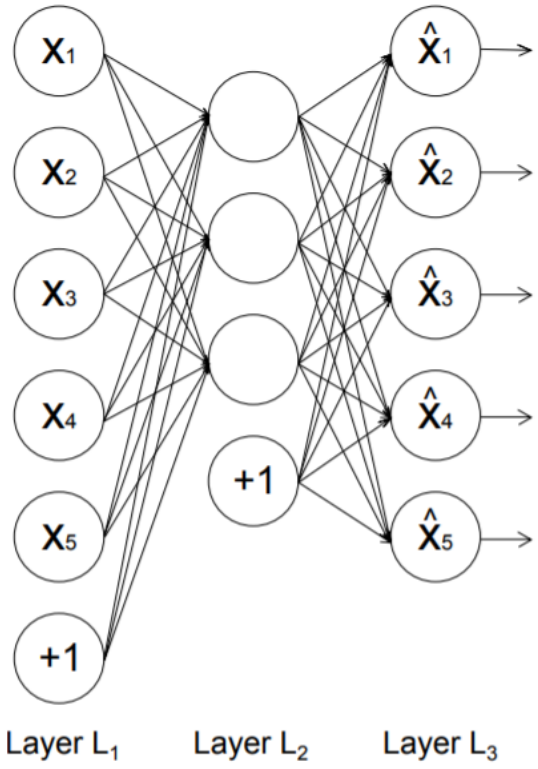

Figure 5.4: Autoencoder Architecture [117]

$$a_i^l = f\left(\sum_{j=1}^{n} W_{ij}^{(l-1)} a_j^{(l-1)} + b_i^l\right) \tag{5.11}$$

Figure 5.4 shows the basic architecture of an autoencoder. The input vector of that autoencoder is $x(1), x(2), ..., x(5)$ and output is $\hat{x(1)}, \hat{x(2)}, ..., \hat{x(5)}$. In the hidden layer L_2, the original dimension of feature has been reduced to a smaller number of neurons. The activation function of unit i and Layer l can be represented by Eq. 5.11, W and b are denote weight and bias respectively. The weight and bias can be controlled by the requirement of the network. For the activation function f in the hidden layer, Rectified Linear Unit (ReLU) function can be used.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{n}\left(\frac{1}{2} \parallel x(i) - \hat{x}(i) \parallel^2\right) \tag{5.12}$$

During training the reconstructed output is compared with the original data during every iteration and the error is calculated. During the process of training, the error needs to be minimized. The objective function that calculates the error is presented in by Eq. 5.12.

5.3 Decision Engine (DE)

The Decision Engine, a term coined by Haider et al. [7], is a component that decides if a system call trace is malicious. I focused on machine learning-based approaches to build the decision engine. This machine learning-based approach includes both the neural network based classifiers and non-neural network-based classifiers including Decision Tree and Random Forest.

According to the working principles of machine learning algorithms, I categorized the algorithms used for building a decision engine into three categories: 1) Unsupervised Algorithms, 2) Supervised Algorithms, and 3) Semi-supervise algorithm. To address the research problems discussed in Chapter **??**, Decision Engines were implemented using only supervised and semi-supervised machine learning approaches. In this section, both approaches and their feasibility will be discussed.

5.3.1 Supervised Algorithms

Using the supervised approach, the classifiers are trained with the samples and their respective labels. These models learn with the existing data and learn until a optimal performance is achieved on the training data. The classifiers are further evaluated on the unforeseen data. The following subsection will discuss different supervised machine learning and neural network-based approaches.

Decision Tree (DT)

A decision tree is used for classification and regression problems. It breaks down a set into smaller subsets to construct a tree comprising decision nodes and leaf nodes. Decision nodes branch out to possible decision paths. Leaf nodes represent a final

classification or decision. The decision tree-based algorithms build the trees based on the principle of information entropy.

While building the tree, the normalized information gain is calculated for each node. The feature with the highest value of information gain is chosen for making decisions. The decision tree works recursively in the same manner until all the examples of the dataset are classified.

One of the significant advantages of the decision tree is the calculation of the biased information gain values for the categorical features. Extracting rules from wider and deeper decision trees are complicated but provide excellent accuracy. These deep decision trees lead to over-fitting, which has a lack of generalization capabilities. Pruning can be performed on the deep decision tree to get a small tree that has comparatively better generalization capabilities.

Support Vector Machine (SVM)

The SVM classifier is based on finding a hyperplane that differentiates two classes in such a way that the distance between the hyperplane and the closest point of a class is maximized [118].

The approach of SVM is based on the principle of minimized classification risk [119]. SVM is well known for its generalization ability and it works efficiently when the number of features is significantly on the higher side but the number of rows is low. While classifying the non-separable data by SVM, slack variables are added and cost parameters assigned to every data point to achieve the optimal hyperplane by performing a quadratic optimization. Several types of classification surfaces can be achieved by controlling the application of SVM kernels such as linear, polynomial, Gaussian radial basis function, and hyperbolic tangent.

SVM is majorly used for performing binary classification, but it can also be used for multi-class classification by finding the optimal hyperplane between each pair of classes. In the case of anomaly detection, One-class SVM was used to build a “normal” profile on non-malicious data and test the malicious data against the normal profile. If the data deviates more than a specific threshold from the model built on a normal profile, it can be considered malicious.

Random Forest (RF)

A common criticism of decision trees is their tendency to overfit. Random forest is a combination of ensemble and decision trees that constructs multiple decision trees and outputs the mode of their classes as the result of the trained model. The trees in the random forest select random data features while building each tree [118].

The process of forest generation is based on the principle of collecting the trees with controlled variance. The prediction of the random forest model can be executed by the majority or weighted voting by the individual trees. There are a couple of advantages of Random Forest, including a low number of model parameters and its resistance to overfitting. While the number of trees increases in a random forest, the variance of the model decreases without affecting the bias. The random forest has a couple of disadvantages, such as low interpretability and its dependence on a random generator.

Extreme Gradient Boosting (XG-Boost)

Extreme Gradient Boosting [120, 121] is a relatively new ensemble approach to gradient boosting for classification. Multiple models (Decision Trees) are constructed sequentially. The output of one model is used to determine gradient increments and introduce weak classifiers before training the next model in the sequence. This process is carried out until there are no more improvements to be made.

Dense Neural Network (DNN) and Convolutional Neural Network (CNN)

Dense Neural Networks are consisting of dense layers on an architecture that is similar to neural network. Each neuron of a layer in a dense neural network is connected to every neuron of next layer. The number of neurons of the input layer is equal to number of input feature in the dataset and number of the neuron in the output layer equals to the number of distinct features that dataset has. The main strength of the neural network is their ability to learn from their own mistake. In the network, the weights and biases are defined in such a way that they can predict a value based on provided input. Then a cost function is deployed to calculate the deviation between

the expected output and their predicted output. The main goal of the network is to find a set of weight and bias values that minimizes the cost function. The principle of gradient descent helps to reduce the cost function. The gradient descent works by computing the gradient repeatedly and then update the weight and bias is a way that minimizes the cost function. The parameter which is used for choosing the direction of the gradient descent towards the global minimum is the learning rate. If the learning rate is very small, the network takes more time to reach the global minimum [122]. Whereas a larger learning rate might speed up the learning process, but the network might not be able to reach the global minimum at all. Another way to speed up the gradient descent process is to use stochastic gradient descent. This works by calculating the gradient of a small set of randomly chosen samples and average them to get a true estimate. The backpropagation algorithm is used to calculate to compute the gradient of the cost function. It computes the partial derivative of the cost function with respect to weight and bias and propagates this back to update the weight and bias. The backpropagation algorithm calculates the gradient of the cost function of one training example. However, it needs to be combined with some learning algorithms, such as stochastic gradient descent, to compute the gradient of all the samples. This is how the neural network efficiently learns by itself [123].

CNN is variant of neural network which is majorly used for computer vision [124, 125]. LeCun et al. [126] proposed a CNN-based architecture LeNet-5 to classify handwritten digits. The hidden layers of the Convolutional Neural Network mainly consists of convolution layer, pooling layer, fully connected layer, and batch normalization layer. Figure 5.5 provides a visual representation of a CNN architecture.

In the pooling layer, the convolution operation which is basically a dot product between the input data and a filter defined by the network results an output modified output preferably of lower dimension than the input. Figure 5.6 is a representation of convolution operation.

The pooling operation is applied on the output of the convolutional layer. Pooling is a sample-based discretization process. The primary goal of pooling is reducing the dimensionality of the data without losing significant information from the sub-region of the original data. There are few types of pooling strategies such as max pooling and average pooling. As per the naming convention, the max pooling selects the maximum

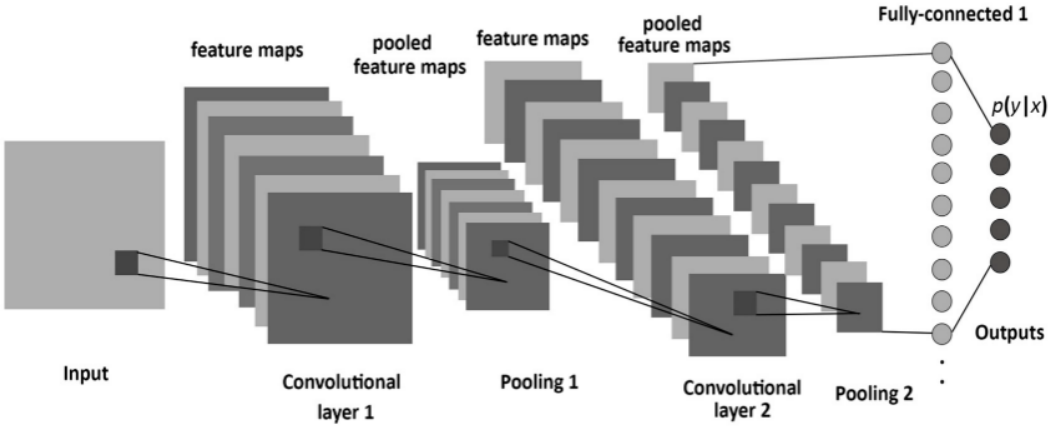

Figure 5.5: CNN Architecture [124]

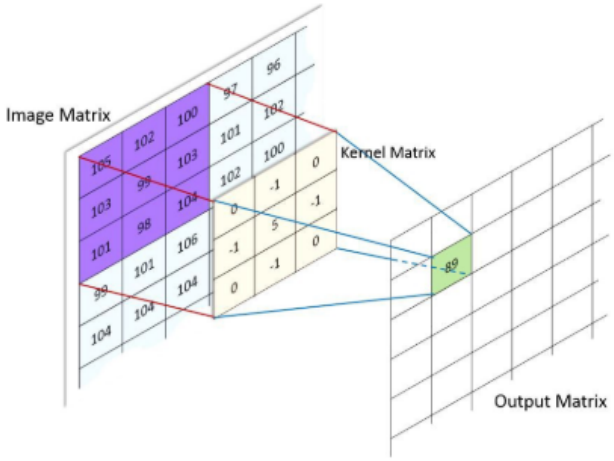

Figure 5.6: Convolution Operation [124]

value from a selected region and average pooling averages the value of the selected region which is defined by the pooling size. Pooling operation in explained in Figure 5.7. Several convolution and pooling layer can be stacked as per the requirement and complexity of the problem. The final output from the pooling is flattened in fed to a fully connected neural network in order to perform the classification.

5.3.2 Semi-supervised Algorithms

For experimental purposes, decision engines are built using a semi-supervised approach. The classifier is trained with only non-malicious traces in order build a

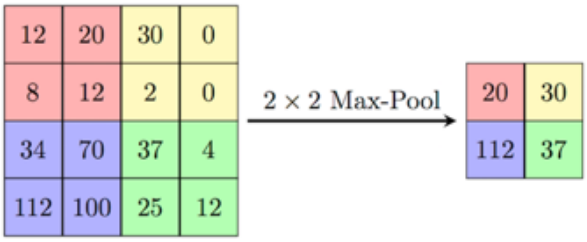

Figure 5.7: Pooling Operation [124]

normal profile. This normal profile is an efficient and generalized representation of all the normal system call traces. The test system calls are then evaluated against the normal profile. In the semi-supervised approach, we have used One-Class Support Vector Machine. While conducting the literature survey, it was noticed that the researchers have used custering techniques such as k-means clustering and k-nearest neighbour to build the anomaly detection system but they did not obtain optimal result in terms of detection rate and false alarm rate. As a result, I decided not to conduct any experiments using clustering techniques.

One-Class Support Vector Machine (OCSVM)

The One-Class SVM algorithm, proposed by Scholkopf al. [127] tries to separate the whole dataset from the origin. This algorithm operates to find a hyperplane which separates the data from the origin with the maximal margin. OCSVM can be trained with only one class and it eventually learns the boundary of these points. If the test samples are compared against this model, the algorithm checks if the sample lies outside the boundary [88].

Chapter 6

Methodology

The proposed framework for Host-based Intrusion Detection System contains seven different components: 1) Data Source, 2) Feature Retrieval, 3) Prepossessing and Normalization, 4) Dimensionality Reduction, 5) Data Splitting 6) Training Decision Engine, and 7) Evaluation of the Decision Engines. Each of the components will be discussed in detail in the sections.

Figure 6.1 is a simplified representation of the components of the proposed framework. At the end of this chapter, a detailed representation of the proposed framework is provided.

6.1 Data Source

I have conducted an in-depth literature review discussing different datasets to evaluate Host-based Intrusion Detection Systems in Chapter 4. As the Linux based computers perform efficiently with lowest resource footprint, I have specifically focused on the datasets which are generated based on Linux system. Finally, I chose ADFA-LD12 to conduct further experiments. In this section, I will discuss supporting reasons behind choosing the dataset for conducting further experiments.

Advantages of ADFA-LD12 dataset

- Ubuntu Linux version with Linux kernel 2.6.38 was employed in the host computer during the generation of ADFA-LD12 dataset [14]. The server was configured to allow different operations such as file sharing service,database service, remote access, web server functionalities which is a reasonable representation of modern Linux server-based computer systems. In order to capture the data with a representation of realism, FTP, SSH, and MySQL were enabled in the default ports. Apache and PHP were installed to enable capture web service activities. Altogether, this is a fairly valid representation of a modern Linux-based

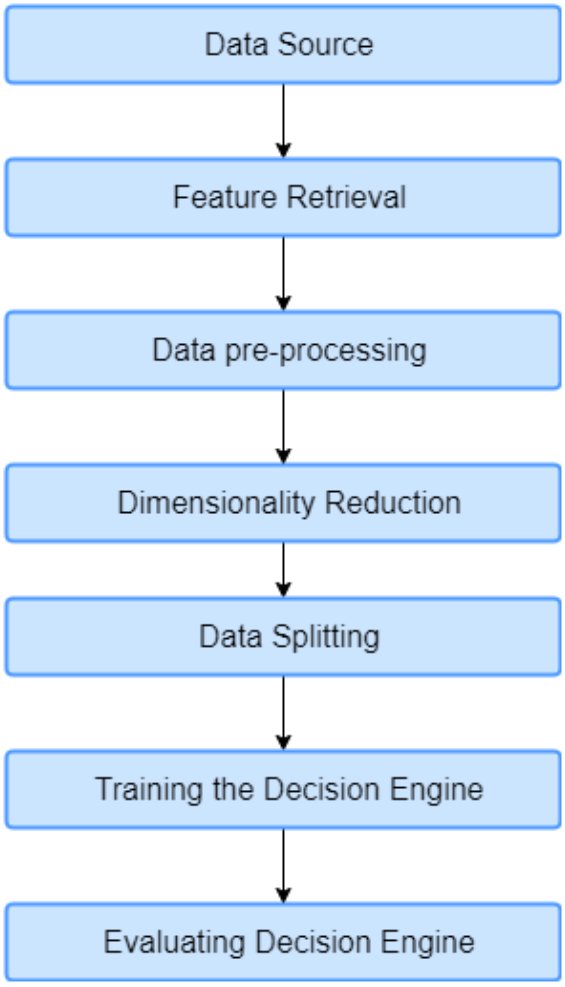

Figure 6.1: Simplified representation of individual component of the proposed framework

computer system which is capable of generating complex system call-based information.

- State of the art methodologies were adopted while generating the attacks.In order to introduce realism , the host server was fully patched but a minor remote code execution vulnerability was introduced. This represents a model secured server with a small vulnerability. The datasets, UNM and KDDCUP, used highly porous, unsecured servers that could be easily exploited. The attacks were chosen such that they cover the major vulnerabilities experienced by modern Linux based systems [14].

- In order to evaluate he complexity of the ADFA-LD dataset, Creech et al. [14]

performed a cluster analysis technique based on "Bag of System Calls" (BSC) technique. The BSC technique is based on the popular technique “bag of words”, which is majorly used in the field of natural language processing [128]. The BSC technique was applied to KDD98 and ADFA-LD and the result is presented in Figure 6.2. It can be observed from the graphical representation that the attack data and the normal data are easily separable in KDD98 dataset but not in ADFA-LD12. In the KDD98 representation, the red attack data and blue normal data are separable even with manual observation. The degree of separability can be measured by the minima and maxima generated by the key system calls which are affecting the nature of the trace. In the representation generated by ADFA-LD12, the traces are not separable easily, as they are homogeneous in nature.The minima and maxima alone are insufficient to separate those two types of traces [129]. This is a representation of realistic representation of low footprint attacks mixed with normal system call traces.

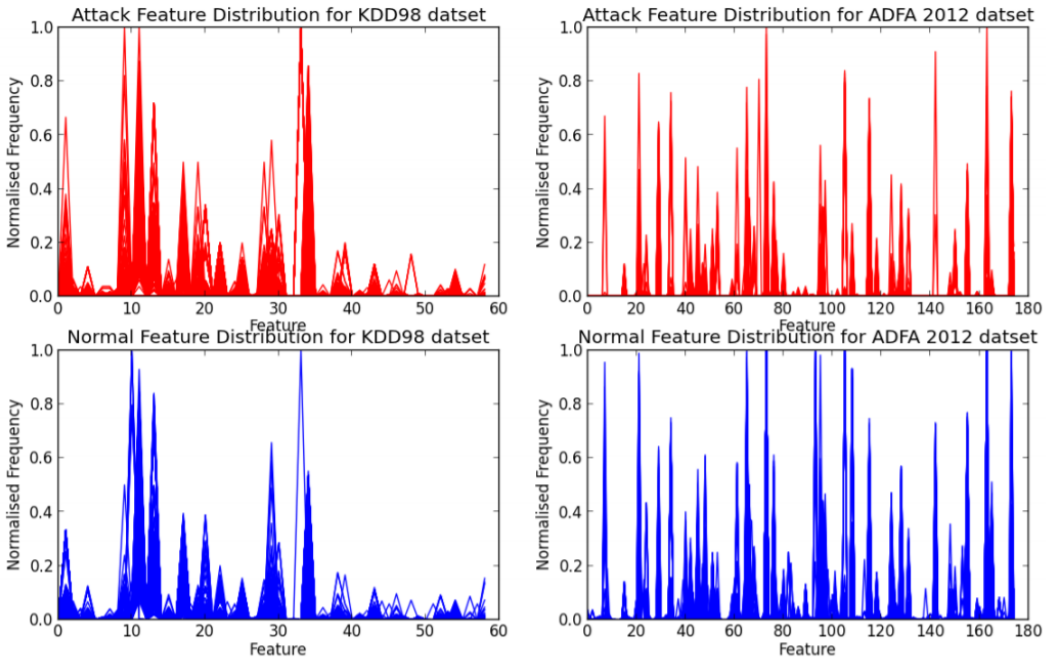

Figure 6.2: Complexity of ADFA-LD 12 ns KDD98 dataset [14]

These criteria were strictly maintained in order to ensure the maximum effectiveness to achieve the highest quality of realism while simulating the data.

6.2 Feature Retrieval

The feature retrieval process was applied on the chosen dataset. As per the development in this domain, the feature retrieval techniques from system calls can be broadly categorized into three parts: 1) Frequency-based techniques , 2) Sequence-based techniques, and 3) Hidden abstract representation techniques. This component of the framework proposes a hybrid feature retrieval technique based on the combination of Hidden abstract representation technique [7] and frequency modeling of system calls [18] discussed in section 5.1 in Chapter 5.

The reasoning behind choosing Hidden abstract representation based method "Integer Data Zero Watermark" (IDZW) [7] combined with frequency modeling method [18] are discussed below [108, 110].

- The ADFA-LD12 dataset contains low footprint attacks that are highly similar to non-malicious data. In order to classify malicious data from non-malicious data efficiently, the natural difference between the malicious and non malicious system calls need to be captured. The IDZW method is capable of capturing the natural hidden difference of malicious traces from benign instances that helps the decision engine to classify it.

- For the unsupervised approach, the decision engine needs to be trained with all possible patterns of normal system calls in order build a generalized normal profile. As the nature and the pattern of the systems calls tend to change depending on the operating system's kernel version, the short sequence-based methods and standalone use of frequency- based methods become ineffective in extracting the hidden representation of the system calls. But IDZW method can cop up with dynamic nature of the system call generation.

- As per the development in this domain, short sequence based and semantic extraction based strategies fail to capture hidden representative features. But the IDZW method and the frequency-based method shows promising result when applied separately. I therefore propose a feature extraction strategy that poses the quality of both, the method, IDZW, and frequency modeling.

- The time complexities for both the semantic-based feature extraction and language modeling are significantly higher while training and predicting a new sequence. Language models and feature-based extractions are inefficient for real-time intrusion systems. Both the IDZW strategy and frequency-based methods are a lot quicker individually and it takes a few seconds to extract the features and to train a decision engine.

- The hybrid feature extraction technique is designed in such a way that it extracts the hidden representation from the system call and it records the effectiveness of the frequency of each system call in a long trace.

- The hybrid approach takes into consideration the whole trace rather than specific system calls or short sequences extracted from the long traces. In low-footprint attacks , the short sequences of system calls do not represent the nature of the whole system call trace. There might be some possibilities where the trace starts with non-malicious nature but at the end of the trace, some malicious activity is posed by the trace. The sequence-based method does not capture the importance of each system call in a trace.

Each trace present in the ADFA-LD dataset is of a different length. Using the feature retrieval technique, traces of different sizes are converted into equal-sized vectors which are further processed and finally fed into a learning classifier for evaluation. In order to retrieve 6 features using the IDZW method, Eq. 5.4, Eq. 5.5, Eq. 5.6, Eq. 5.7, Eq. 5.8, and Eq. 5.8 were used (refer section 5.1). 325 frequency-based features were retrieved using Eq. 5.1 (refer section 5.1).

Table 6.1: Feature Retrieval from a normal system call trace using IDZW method

Type of system calls	Unprocessed version (before feature retrieval)	After feature retrieval with IDZW
Training(Normal)	168 54 102 6 102 102 102 102 168 54 102 6 102 102 102 102 91 102 168 54 102 6 102 102 102 102 168 54 102 6 102 102 102 78 240 240 240 168 54 102 6 265 78 78 78 265 240 240 78 240 265 102 102 102 13 102 102 102 102 6 195 5 197 192 3 3 6 91 240 195 102 102 78 168 102 168 54 102 6 195 5 197 192 3 3 6 91 195 102 102 78 168 102 168 54 102 6 265 78 78 265 102 102 102 13 102 102 102 102 6 195 5 197 192 3 3 6 91 195 102 102 78 168 102 168 54 102 6 265 78 78 265 102 102 102 13 102 102 102 102 6 195 5 197 192 3 3 6 91 195 102 102 78 168 102 168 54 102 6 265 78 78 265 102 102 102 13 102 102 102 102 6 195 5 197 192 3 3 6 91 195 102 102.....	265,3,102,219,91,256

Table 6.2 provides an example of the retrieved features using frequency modelling.

Table 6.2: Retrieved features using Frequency modeling

After Feature Retrieval with Frequency Modelling
0.0,0.0,0.0130475302889096,0.07921714818266543,0.004659
832246039142,0.001863932898415657,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.014911463187325256,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.011183597390493943,0.0,
0.0,0.0,0.0,0.0,0.0,0.0065237651444548,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.06337371854613234,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
.....(more features are present)

6.3 Data Preprossessing and Normalization

Data normalization and scaling techniques have been applied to the features that were retrieved using both IDZW method and frequency modelling. The goal of applying scaling is to change the numerical values of the features to common scale in such a manner that the ranges of the values are not distorted. Scaling is very effective when distance-based algorithms such as k-Nearest Neighbour and SVM are used as classifiers.

Normalization techniques mainly change the shape of the distribution of the data so that any other type of distribution can be described as normal distribution. In neural networks, a model learns by mapping an input variable to the output variable. The input variables that have different scales and distribution make it difficult for a model to learn efficiently. Large input value leads the model to learn large weights which affects the learning process of models which in turn result in major generalization errors.

I have experimented with three data scaling/normalization techniques: 1) Standard Scaler or Z-score, 2) Min-Max Scaler, and 3) Robust Scaler. In this section , I will further discuss these three techniques and their corresponding results through visualization. At the end of this section, I will provide a detailed reasoning for choosing the best option for that fits perfectly with this proposed methodology.

6.3.1 Min-Max Scaling

In Min-Max scaling each datapoint subtracts the minimum value in its feature and divides the quantity with its range. As Min-Max scaler does not distort the shape of

the original distribution, it does not change the valuable information embedded inside the original data. Min-Max scaler does not reduce the importance of an outlier. The operations of the Min-Max scaling can be formulated by Equation

$$z_{min-max} = (x^i - x_{min})/(x_{max} - x_{min}), \tag{6.1}$$

where $z_{min-max}$ denotes the Min-Max scaler transformed value, x_{min} denotes minimum value of feature x, x_{max} denotes the maximum value of feature x, and x^i denotes individual data point in feature x.

Figure 6.3 represents the Kernel Density Estimate(KDE) plot of the original distribution of the features retrieved from IDZW method. Figure 6.4 describes the distribution of the transformed features after applying the Min-Max scaling.

It can be observed from Figure 6.4 that the shape of the original distribution is preserved and the relative difference between the features of original distribution and the Min-max scaler transformed distribution is maintained.

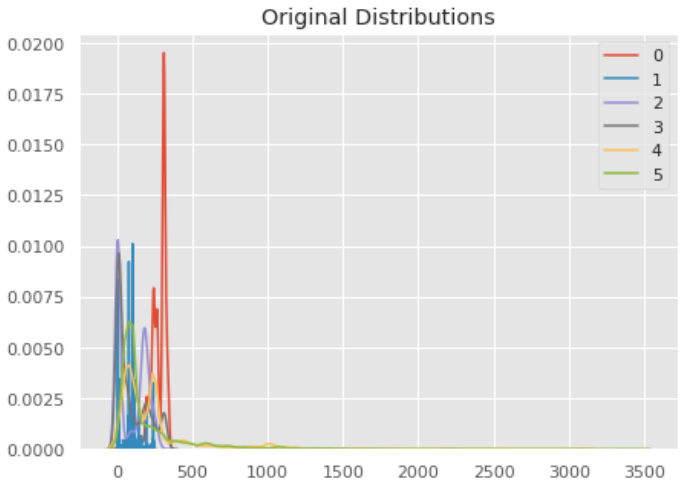

Figure 6.3: Original distribution of the features retrieved from IDZW method

6.3.2 Standard Scaling (Z-score Normalization)

It rescales the distribution of the values of the dataset. The mean of the observed value becomes zero and standard deviation becomes 1 [7]. This approach assumes that the observed data fits a Gaussian distribution with a controlled mean and standard deviation. The mathematical operation for standard scaling is described in Eq. 6.2.

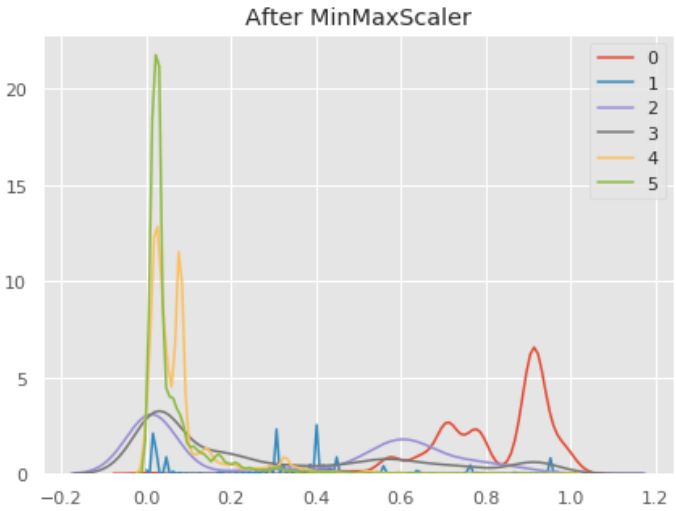

Figure 6.4: Min-Max scaler transformed data

$$z_{standard} = (x^i - \mu)/\sigma \tag{6.2}$$

where $z_{standard}$ represent transformed values of individual feature vectors, μ and σ denote the mean and standard deviation of the feature matrix after the feature retrieval.

Figure 6.5 describes the distribution of the transformed features after applying the standard scaling [130].

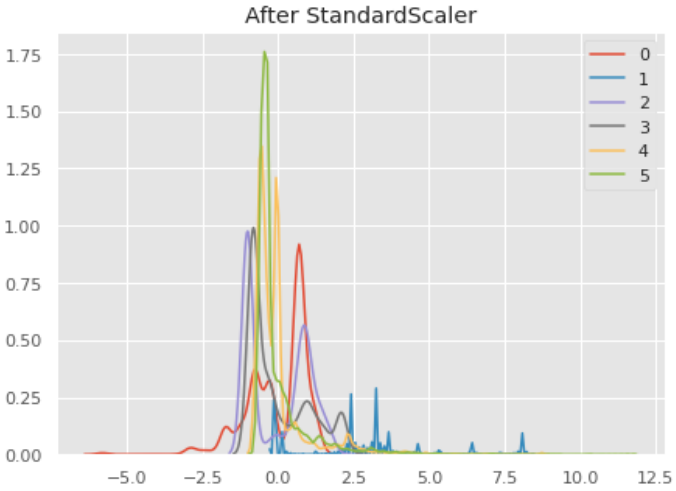

Figure 6.5: Standard scaler(z-score normalization) transformed data

6.3.3 Robust Scaling

The Robust Scaling technique uses a similar method as Min-Max scaling but uses inter-quartile range rather than min-max range. As a result, this scaling technique is robust towards the outliers [131]. Robust scaling technique can be efficiently applied to each of the individual feature vectors, as mentioned in Eq. 6.3

$$z_{robust} = (x^i - Q1(x))/(Q3(x) - Q1(x)), \tag{6.3}$$

where z_{robust} denotes the transformed datapoint after robust scaling is applied, x^i denotes the individual datapoint of a feature vector, $Q1(x)$ denotes the middle value of the 1st half of the ranked order datapoints, and $Q3(x)$ denotes the middle value of the second half of the ranked order datapoints.

Figure 6.6 represent the distribution of the data after Robust scaling is applied on the original distribution

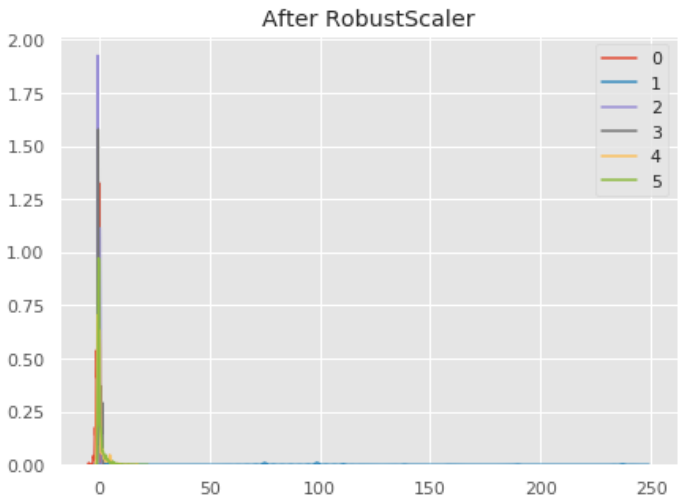

Figure 6.6: Robust scaler transformed data

6.3.4 Choosing a Scaling Technique

Several experiments were conducted using these three scaling/normalization techniques discussed above. The reasonings behind selecting these specific scaling/normalization techniques are provided below [131, 132].

- Distance-based algorithms such SVM is sensitive towards the magnitude of the datapoints. Therefore, it is necessary to scale the features to weigh-in equally.
- Scaling is very critical while performing dimnensionality reduction using Principle Component Analysis [83]. PCA tends to select the feature with higher variance. As the features with high magnitude have high variance, it skews the PCA towards the high variance scaling incorrectly.
- Scaling can significantly accelerate the gradient descent process while training a neural network. If the data is scaled, it quickly reaches the global minimal rather than oscillating inefficiently away from the convergence point.

6.4 Dimensionality Reduction

In this section, I will discuss several dimensionality reduction techniques which are used to build this component. The feature matrix produced by the frequency-based feature retrieval method is high dimensional in nature. As the frequency of each system call is taken into consideration, there are many system calls which does not appear in a specific trace at all. This method produces a significant numbers of zeroes as feature vectors. Dimensionality reduction algorithms represent the original features in a lower dimensional space which reduces storage requirements and computational complexity of the overall system. These methods help to identify the valuable representative features of a dataset that actually control the nature of the data.

The are two major approaches that comes under the umbrella of Dimensionality Reduction i.e. Feature selection and Feature extraction. It has been noticed in previous rearch work that both of these terms have been used interchangeably. But in the proposed work, both the terminologies have different inteepretation and explained in the following. Feature selection methods select a subset of the original features. As a result, some information loss occurs in this process as some of the features are directly discarded. On the other hand, feature transformation and extraction methods introduce new features that are constructed from the original features but in a lower dimensional space. In order to maintain the diversity, detailing, and completeness, I have experimented with three different dimensionality reduction techniques on frequency modelled features: 1) Principal Component Analysis , 2) Recursive Feature

Elimination, and 3) Simple Autoencoder.

6.4.1 Principal Component Analysis (PCA)

PCA was applied on the high dimensional data retrieved by frequency-based feature retrieval method. As discussed in 5.2.1, covariance matrices were created from standardized data to represent correlation and dependencies between different features of the dataset. Then Eigen vectors and Eigen values were calculated from the covariance matrix. The Eigen vector with the highest Eigen value was considered as the most significant principal component.

Principal components with lesser significant values were dropped to reduce the dimensionality of the data. Finally, original data axis was replaced with newly formed principal components by multiplying the transpose of original dataset with transpose of the obtained vector. Figure 6.7 denotes that nearly 75 principal components required to retain 95% variance of the original data. We, finally use 114 features extracted by PCA.

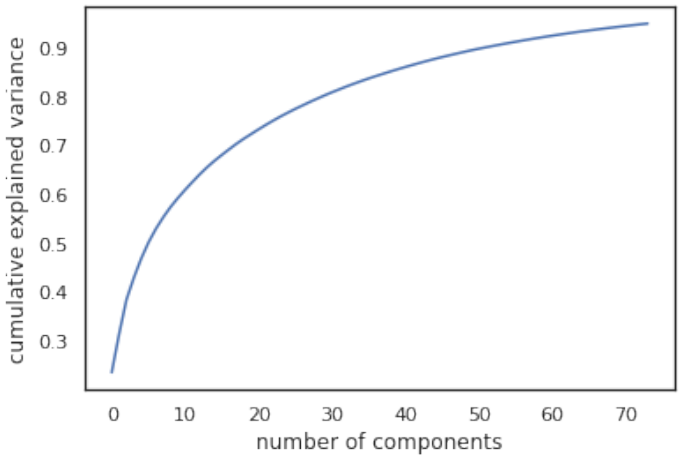

Figure 6.7: The number of Principal Components required vs.Retained Variance

6.4.2 Autoencoder

The autoencoder network which is built for dimensionality reduction has two parts: 1) an Encoder, and 2) a Decoder. The Encoder compresses the original number of features from 326 to 80. The Decoder reconstructs those compressed features to

map with the original features. The autoencoder is trained with 50 epochs and the difference was calculated between reconstructed features and actual features after every epoch. After completion of all the epochs, only the encoder module of the network was used to generate the compressed features form the scaled frequency modelled data. The compressed features were merged with the features retrieved by the IDZW method and used for training and evaluation of the classifiers. Figure 6.8 and Figure 6.9 represents the configuration of the encoder and decoder module respectively.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 160)               52320
_________________________________________________________________
dense_2 (Dense)              (None, 80)                12880
=================================================================
Total params: 65,200
Trainable params: 65,200
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.8: Configuration of the Encoder module in Autoencoder

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 160)               12960
_________________________________________________________________
dense_4 (Dense)              (None, 326)               52486
=================================================================
Total params: 65,446
Trainable params: 65,446
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.9: Configuration of the Decoder module in Autoencoder

Figure 6.10 plots the training loss against the validation loss of AutoEncoder network.

6.4.3 Random Forest - Recursive Feature Elimination (RF-RFE)

RF-RFE method was used in order to identify the most important features. The above methods such as Autoencoder and PCA extracts new features from the original features in lower dimensional space but feature selection methods like RF-RFE provides the optimal subset of the existing feature that represents the most of the

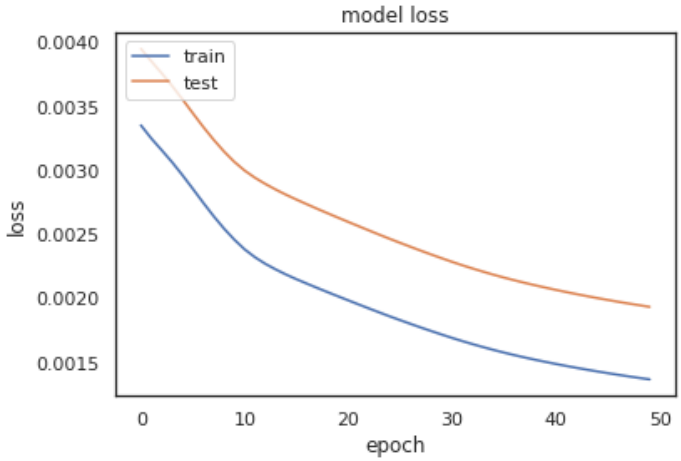

Figure 6.10: Training Loss v/s Validation Loss (Autoencoder)

information in a dataset. As the data is highly imbalanced, I use the RF-RFE method with stratified cross-validation in order to get rid of over fitting. In this technique, the model is built with the entire set of predictors and importance score is calculate for every individual predictors. The predictor with the lowest score is dropped and again the model is trained with the rest of the predictors.By this recursive approach, the optimum number of features are extracted by calculating the importance score. The model is finally trained with the optimal features. As this approach selects the subset of the original features, it simply discards the importance of those features those are dropped. I plotted a graph for Recursive Feature Elimination with Cross Validation (RFECV) in order to visualize the classification accuracy with respect to the number of selected features. It can be extracted from the Figure 6.11 that, the features between the range of 25-320 shows almost same classification accuracy. As the major goal of this approach is to reduce the dimensionality of the feature, I have used first 86 features selected by RFECV for the further training of the learning models [133].

I experimented with all the dimensionality reduction algorithms in order to find the optimal set of features and trained the learning models with the obtained features. A thorough detailing of the results and their respective interpretation is provided in the 7 Chapter.

PCA and RF-RFE was implemented using Python Scikit-learn libraries [134] [135]. Autoencoder was built using Keras [136] which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.

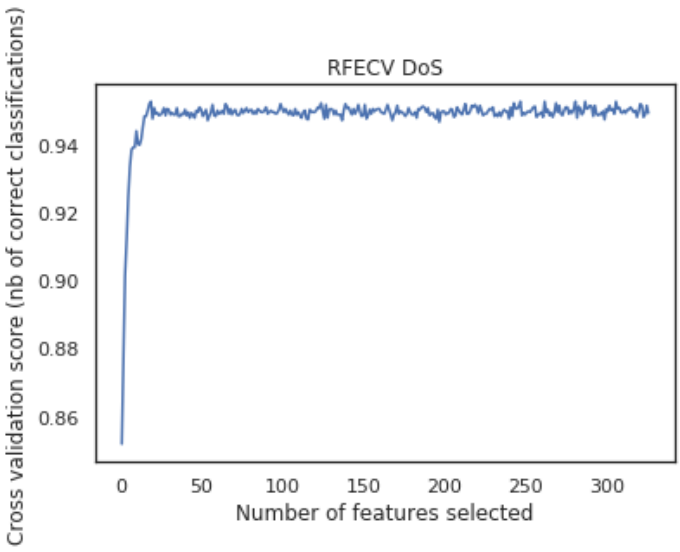

Figure 6.11: Classification accuracy v/s Number of features selected in RFECV

6.5 Data Splitting

The dimensionally reduced data were distributed into three sections: 1) Training, 2) Validation, and 3) Testing. As the decision engine was built using both the supervised and semi-supervised approach, the distributions of the samples were also managed in different manner.

6.5.1 Supervised Approach

In the supervised approach, classifiers were trained using both the malicious and non-malicious traces. The training data was used to train the learning models. Validation data was used to provide an unbiased evaluation of a learning model on the training dataset while tuning the hyperparameters of the model. The samples from the testing dataset were used to provide an unbiased and generalized evaluation of the trained learning model. 10-fold stratified cross-validation was used to evaluate the learning models. The distribution of the trace are represented in table 6.3.

Table 6.3: Distribution of the traces for training supervised algorithms

Normal Training traces	Attack Training traces	Normal Validation traces	Attack Validation traces	Normal Test trace	Attack Test trace
3346	196	962	225	962	225

6.5.2 Semi-supervised Approach

In the semi-supervised approach, I trained the classifiers using only the non-malicious traces to build a normal profile. Then validation data which contains only normal traces was used to evaluate the correctness of the trained normal profile. The test set was used to provide an unbiased and generalized evaluation of the trained learning model. The distribution of the trace for semi-supervised approach are represented in table 6.4.

Table 6.4: Distribution of the traces for training semi-supervised algorithm

Normal Training traces	Normal Validation traces	Test traces	Attack
4205	1000	962	

6.6 Decision Engines (DE)

The dimensions of the retrieved features were reduced significantly and prepared for training the decision engines. As mentioned in the Section 5.3, Chapter 5 , I trained the decision engine using both, supervised and semi-supervised approaches.

6.6.1 Supervised approach

I trained five traditional machine learning based classifiers(non-neural network) to train the classifier such as Decision Tree, Random Forest, Logistic Regression, SVM with radial basis function kernel, and XGBoost. In addition to that, we also trained a 5-Layer dense neural network In the supervised approach the classifiers were trained using both the normal and malicious data.

Several experiments were conducted for searching a hyper parameter space for a set of values that optimizes the performance of the learning models.

To build the decision tree classifier, I concentrated on few hyperparameters such as criterion, min_samples_leaf, min_samples_split, and splitter.

Information gain-based entropy measure was used to split the nodes in decision tree, "Best" splitter was used to identify the most important feature to split the nodes based on impurity measure. The min_samples_split value is chosen as 2 and min_samples_leaf is chosen as 1 [137]. I have also configured Random Forest using hyperparameters including the number of trees in the forest, the maximum number of features considered for splitting a node, the maximum number of levels in each decision tree, and the minimum number of data points allowed in a leaf node. The values of the hyperparameters were assigned as per the knowledge gained about the classifiers functionality and the dataset distribution. The similar approach is followed when Logistic Regression and XG-Boost Classifiers were tuned with optimal hyperparameters.

While configuring SVM , Radial Basis Function (RBF) kernel was used because of non-linear nature of the dataset. RBF kernel combines and uplift the samples to higher dimensional feature space separates the classes using a linear decision boundary.

I also built a 5-layer neural network. In the first layer 25 units were used, along with random_uniform as kernel initializer, and ReLU as the activation function. In a similar way other layers are configured. After every two layers, one dropout layer was added to prevent the model from over fitting. In the final layer, sigmoid function is used as it is a binary classification problem. In the Chapter 7, I discuss all the results using these learning models.

6.6.2 Semi-Supervised approach

A varied set of experiments were carried out using One Class SVM (OCSVM) where the classifier was trained using only non-malicious system calls in order build a normal profile. Then the malicious system calls are tested on the trained model and verify if the test traces are significantly deviating from the trained model to be considered as malicious. If the test sequence deviates more than a previously configured threshold, it is considered as malicious, else benign.

6.7 Evaluation

In this section, different metrics to evaluate the performance of the classifiers are discussed. As the proposed methods performs a binary classification task, I am proposing to use several popular classification metrics that help to justify the efficiency of the trained models.

One of the most efficient tabular visualization model to represent a classifiers performance in a confusion matrix or an error matrix. The confusion matrix contains the predictions of the classifier and the ground truth labels. Each row of a confusion matrix contains the instances of an original instance and each row denotes the predicted instances. The non-malicious instances were labelled as '0' and malicious instances were labelled as '1'.

Learning models were trained using the training data and their respective labels. The confusion matrix was generated by mapping the prediction of the trained models on the test data against the labels of the original test data. Table 6.5 represents a confusion matrix for anomaly detection problem. In the confusion matrix, the instances marked as true positives are predicted as an attack, and also originally an attack. The false positive instances are predicted as an attack by the classifier but not actually an attack. True negatives are the instances which are detected as benign by the classifier and also actually a benign instance, whereas, false negatives are detected as non-malicious instances but actually an attack. The diagonal elements of the elements represents the correct predictions of different classes and other elements denotes the miss-classified instances. Metrics such as classification accuracy, precision, recalls, f-score, sensitivity, and specificity can be extracted from the confusion matrix.

Table 6.5: Confusion Matrix for Anomaly Detection

		Predicted label	
		Not attack(0)	Attack(1)
True label	Not attack(0)	True Negative(TN)	False Positive(FP)
	Attack(1)	False Negative(FN)	True Positive(TP)

Accuracy

Accuracy can be defined as number of correct prediction divided by number total number of predictions.Accuracy is the simplest and intuitive approach to measure the performance of the classifier. But accuracy does not consider the records which are classified as false positives and false negatives. So, it does not provide a whole picture of how the false positive and false negative affect the result. Accuracy can be formulated as:

$$Accuracy = \frac{\Big(TP + TN\Big)}{\Big(TP + FP + TN + FN\Big)} \tag{6.4}$$

Precision

Precision can be defined as the number of true positives divided by the summation of true positive and false positives. As ADFA-LD12 dataset is highly imbalanced and biased towards to non-malicious system call traces, precision is a very useful metric as it measures the correctly classified instances against all the original instance of that specific class.Precision can be formulated as:

$$Precision = \frac{\Big(TP\Big)}{\Big(TP + FP\Big)} \tag{6.5}$$

Recall

Recall can be defined as number of true positives divided by summation of true positive and false negatives. As ADFA-LD12 dataset is highly imbalanced and biased towards to non-malicious system call traces, Recall is a very useful metric as it measures the correctly classified instances against all the predicted system call traces of that specific class .Recall can be formulated as:

$$Recall = \frac{\Big(TP\Big)}{\Big(TP + FN\Big)} \tag{6.6}$$

F-score

F-scores is a combination of precision and recall. As my research question addresses the issues with false positives and false negatives, F-score is one of the most appropriate metrics to evaluate the correctness of the model. F-score is the harmonic mean of precision and recall, can be defined as:

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.7}$$

Detection Rate (DR)

Detection rate indicates the measure that how the trained model is able to classify. Detection rate can be defined as the ratio of True Negatives to the all actual negative instances.

$$DetectionRate = \frac{\Big(TP\Big)}{\Big(TP + FN\Big)} \tag{6.8}$$

Receiver Operating Characteristic (ROC) curve

The ROC curve is plotted by keeping True Positive Rate(TPR)(Eqn. 6.10) on the Y axis, and False Positive Rate(FPR)(6.9) on X axis. Most of the classification algorithms that I use as a decision engine predicts a probability of an instance [138]. If the probability is smaller/larger than a specific threshold , it is considered as a part of specific class as per the pre-defined configuration. ROC curve plots the TPR against FPR of all possible threshold values. ROC curve helps to find the optimal probability threshold where TPR is high and FPR is low. ROC curve is a trade-off between TPR and FPR. As our research problem revolves around reducing the number of false positives and false negatives, ROC curve helps efficiently to find threshold for classifiers.In ROC curve, the 45 degree diagonal of ROC space represents the baseline classifier. Another important metric that is closely related to ROC curve is the are under ROC curve which is known as AUC. AUC provides overall measure of performance of a model throughout all the possible thresholds [139, 140].AUC is scale-invariant that measures the quality of the ranked prediction.AUC measures the degree of separability that represents how the model is able to create a separation between the classes.

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{6.9}$$

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} \tag{6.10}$$

False Alarm Rate (FAR) False alarm rate can be defined as an average of false positive rate and false negative rate. FAR can be represented by the Eq. 6.9.

$$FalseAlarmRate(FAR) = \frac{FalsePositiveRate + FalseNegativeRate}{2} \tag{6.11}$$

The proposed framework is built by efficiently merging all the individual module. Fig.6.12 represents the proposed framework.

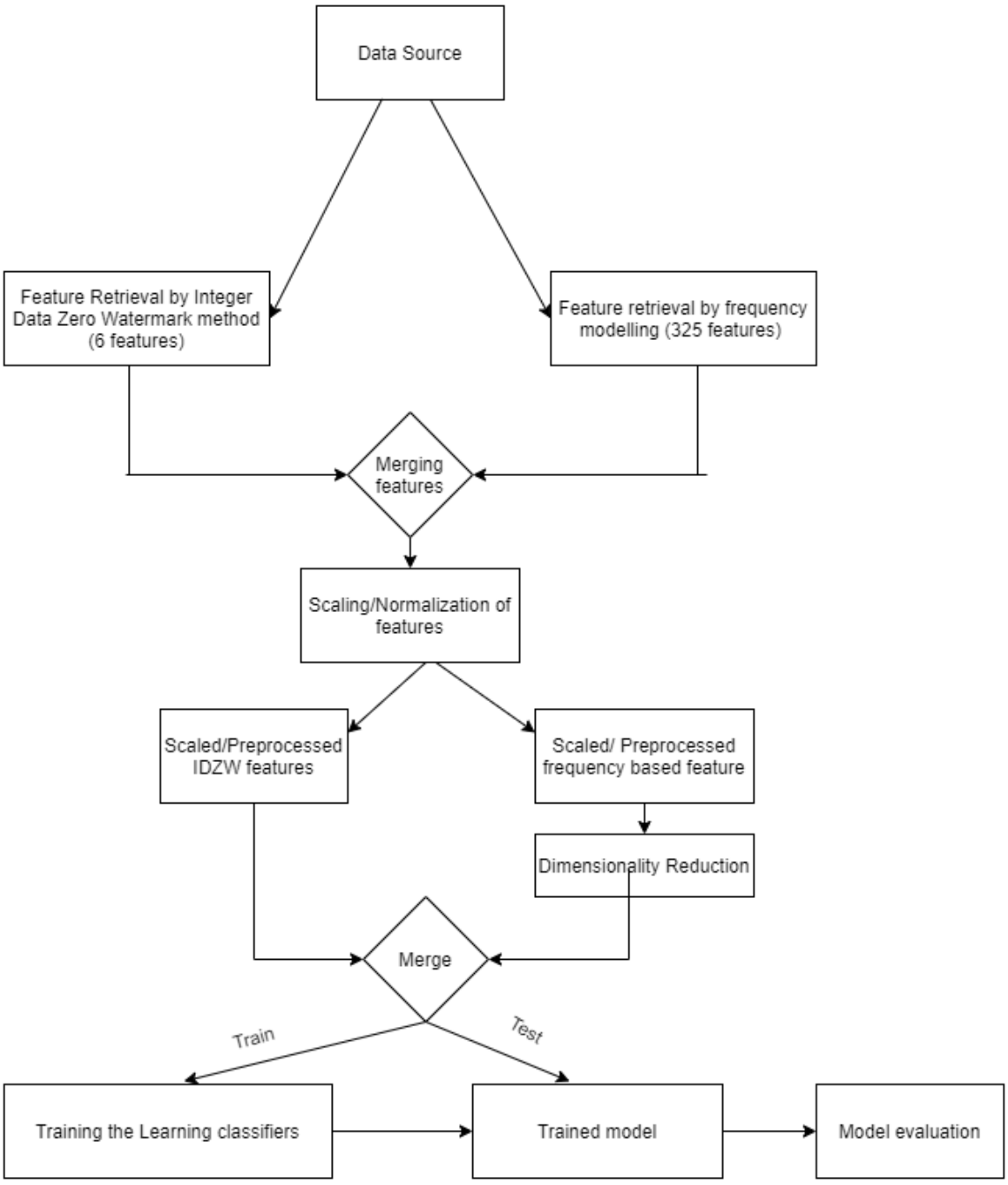

Figure 6.12: Representation of the proposed framework

Chapter 7

Experiments, Results, and Discussion

In this Section, I will present and analyze the results that were generated from the experimentation. As feature retrieval process play a major role in building the framework, I evaluated the performance of each classifier trained with retrieved features clubbed with different combination of dimensionality reduction, and data scaling techniques. To address the research problems described in 3, I have followed the below set of tasks to complete the experiments.

- Firstly, two separate sets of features were retrieved using the IDZW method (6 features) and frequency-based feature retrieval (325 features) method.
- Secondly, Data preprocessing was performed on both the features by applying different scaling and normalization techniques.
- Since the number of features retrieved by the frequency-based approach is large, dimensionality reduction techniques were applied to represent the feature to a lower-dimensional space.
- The low-dimension frequency-based features was merged with preprocessed features retrieved using the IDZW method.
- The merged data was split into training, validation, and testing. The training data was used to train the learning models. Validation data was used to provide an unbiased evaluation of a learning model on the training dataset while tuning the hyperparameters of the model. The samples from the testing dataset were used to provide an unbiased and generalized evaluation of the trained learning model. 10-fold stratified cross-validation was used to evaluate the learning models.
- I evaluated the effectiveness of the hybrid feature retrieval technique and dimensionality reduction techniques by focusing fewer metrics including false positive

rate and detection rate with addition to Accuracy, precision, recall, and f-score. The detection rate was given importance because it effectively indicates how well the model can detect an attack. The false positive rate indicates the efficiency of classifiers to evaluate the miss-classification of the traces that are considered benign.

7.1 Hardware Requirements and Configuration

Most of the experiments were executed using Google Colab which is a platform that allows us to write and execute Python scripts in the browser and it does not require the installation of dependencies separately.

To calculate and compare the training time of the classifiers, I used a personal computer that has the following configuration. In the personal computer all the experiments were done using Python installed with an Anaconda Distribution.

- Operating System: Microsoft Windows 10 Home 64- bit Operating system, x-64 based processor
- Version: 10.0.18363 Build 18363
- Processor: Inter(R) Core(TM) i5-8265U CPU @1.60 GHz 1.80GHz
- Installed RAM: 12.0 GB
- BIOS version: LENOVOAPCN31WW

7.2 Performance Evaluation

In order to represent the result in an organized approach, this section has been categorized into three parts as per different dimensionality reduction techniques used [1].

Each of the sub-categories will describe the performance of the classifiers trained with features extracted/selected by a specific dimensionality reduction techniques.

[1] I have also conducted several experiments with CNN Where I represented the long trace of system call as an image and classified it with CNN. Though the approach is novel, but this approach was not able to provide satisfactory result while comparing with existing approaches. So the results related to CNN is not provided here.

7.2.1 Performance Based on the Features Extracted by PCA

Table 7.1 describes the performance of different machine learning classifiers when PCA was used as the dimensionality reduction technique with min-max normalization. It can be observed from Table 7.1 that the Random Forest classifier has outperformed other machine learning classifiers while evaluated using this configuration. Even if the required time to train the random forest model is higher than other classifiers, the model performs considerably better while reducing the number of false positives and false negatives. The same configuration was used to train the 5-layer neural network, which resulted in a high detection rate of 96.8% with a false alarm rate of 2.1%. Table 7.2 provides a detailed result of the neural network-based classifier trained with the features extracted by PCA.

Table 7.1: Performance of machine learning based classifiers trained with the features extracted by PCA and scaled with Min-Max normalization

PCA with variance 0.95 and Min-Max scaler

	Decision Tree	Random Forest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boost	One Class-SVM
Accuracy	94.76%	96.76%	92.17%	95.66%	95.56%	73.36%
Precision	0.88	0.95	0.87	0.91	0.94	0.72
Recall	0.88	0.90	0.73	0.89	0.85	0.71
F-Score	0.88	0.91	0.77	0.90	0.89	0.72
False Alarm Rate (FAR)	1.4%	0.76%	2.1%	1.6%	0.76	8.6%
Detection Rate	88.2%	91.3%	55%	88.12%	86.25	72.26%
Training Time	0.3 seconds	6.189 seconds	0.0005 seconds	0.54 seconds	2.69 seconds	0.06 seconds
AUC score	0.93	0.99	0.97	0.99	0.99	0.83

Table 7.2: Performance of neural network based classifier trained with the features extracted by PCA.

	PCA(with 95% variance) with Min-Max scaler [80 features]	PCA(with 95% variance) with Standard scaler [120 features]	PCA (with 95% variance) with Robust scaler [120 features]
Accuracy	94.78%	93.48%	91.37%
False Alarm Rate (FAR)	2.1%	3.4%	4.4%
Detection Rate	96.8%	94.8%	91.6%
Training Time	3.25 seconds	5.09 seconds	5.09 seconds

	PCA(with 95% variance) with Min-Max scaler [80 features]	PCA(with 95% variance) with Standard scaler [120 features]	PCA (with 95% variance) with Standard scaler [120 features]
AUC score	0.98	0.97	0.96

Table 7.3 summarizes the performance of machine learning classifiers when PCA was used as the dimensionality reduction with z-score normalization. While considering PCA with z-score normalization, Random Forest and Support Vector Machine with rbf kernel shows the similar false alarm rates which is as low as 1.6%. But, Random Forest outperforms the SVM model while detecting malicious traces.

Table 7.3: Performance of the classifiers trained with the features extracted by PCA and scaled with z-normalization

PCA with variance 0.95 and Standard scaler

	Decision Tree	Random Forest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boost	One Class-SVM
Accuracy	94.76%	97.1%	94.88%	95.21%	96.05%	78.84%
Precision	0.88	0.96	0.89	0.88	0.94	0.72
Recall	0.88	0.90	0.87	0.91	0.87	0.78
F-Score	0.87	0.92	0.87	0.89	0.90	0.75
False Alarm Rate (FAR)	2.4%	1.6%	2.4%	1.6%	2.8%	8.46%
Detection Rate	90.2%	92.3%	85%	88.12%	88.25	71.36%
Training Time	0.4 seconds	6.9 seconds	0.0005 seconds	0.83 seconds	3.86 seconds	0.08 seconds
AUC score	0.94	0.99	0.97	0.99	0.99	0.80

The same configuration was used to evaluate the neural network-based classifier, which showed a promising detection rate but a higher false-positive rate. When PCA was clubbed with robust-scaling, SVM showed the lowest false alarm rate, and the neural network-based classifier indicated the highest detection rate.

Finally, it can be deduced from all the experimentation related to PCA that the Neural Network and Min-Max scaling together result in the highest detection rate of 96.8%, and Random Forest with Min-Max scaling result the lowest false alarm rate of 0.76%.

7.2.2 Performance Based on the Features Extracted by Autoencoder

Table 7.4 describes the performance of different machine learning classifiers when min-max normalization was applied to the features extracted by the Autoencoder. It can be observed from Table 7.4 that the Random Forest classifier has outperformed other machine learning classifiers while evaluated using this configuration. Even if the required time to train the random forest model is higher than the other classifiers, the model performs considerably better while reducing the number of false positives and false negatives. It can be observed that all the classifiers have performed significantly well while reducing the false positives. The same configuration was used to train a neural network, which resulted in a higher detection rate of 93.8%, but the classifier was not efficient enough to reduce the false alarms. Table 7.5 provides a performance summarization of the neural network-based classifiers trained with the features extracted by Auto Encoder.

Table 7.4: Performance of machine learning classifiers trained with the features extracted by Auto Encoder and scaled with Min-Max normalization

AutoEncoder with Min-Max scaler

	Decision Tree	Random Forest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boos t	One Class-SVM
Accuracy	94.36%	96.70%	91.11%	94.31%	95.6%	82.3%
Precision	0.87	0.95	0.83	0.88	0.94	0.82
Recall	0.86	0.89	0.69	0.83	0.84	0.78
F-Score	0.86	0.92	0.74	0.86	0.88	0.80
False Alarm Rate (FAR)	2%	0.7%	1.5%	1.5%	0.79%	7.89%
Detection Rate	87.2%	91%	46%	77.77%	77.11%	76.62%
Training Time	0.28 seconds	6.12 seconds	0.0004 seconds	0.54 seconds	2.83 seconds	0.08 seconds
AUC score	0.92	0.99	0.97	0.99	0.99	0.86

Table 7.5: Performance of neural network based classifier trained with the features extracted by Auto Encoder.

	AutoEncoder with Min-Max scaler (86 features)	AutoEncoder with Standard scaler(86 features)	AutoEncoder with Robust scaler (86 features)
Accuracy	94.59%	95.59%	91.2%
False Alarm Rate (FAR)	4.4%	1.7%	8.2%
Detection Rate	94.8%	90.5%	91.2%
Training Time	3.55 seconds	3.48 seconds	3.81 seconds
AUC score	0.97	0.97	0.94

Table 7.7 and 7.6 summarizes the performance of machine learning classifiers when Auto Encoder was used as the dimensionality reduction technique with z-score normalization and robust scaling. While considering Auto Encoder as a dimensionality reduction technique with z-score normalization, Random Forest, and Decision Tree shows a similar detection rate which is around 88%. But, Random Forest outperforms the Decision Tree-based model while reducing the False Alarm rate. As expected, another tree-based model XG-Boost also displays promising results with a false alarm rate of 0.7%.

Table 7.6: Performance of machine learning classifiers trained with the features extracted by Auto Encoder and scaled with Robust Scaling

	AutoEncoder with Robust scaler					
	Decision Tree	RandomForest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boost	One Class-SVM
Accuracy	94.42%	96.33%	92.75%	92.05%	95.10%	76.9%
Precision	0.87	0.93	0.84	0.84	0.93	0.75
Recall	0.85	0.88	0.80	0.73	0.82	0.69
F-Score	0.86	0.91	0.81	0.77	0.86	0.72
False Alarm Rate (FAR)	2.4%	0.8%	1.5%	1.5%	1.4%	8.2%
Detection Rate	85.8%	84%	46%	53.08%	72%	72.2%
Training Time	0.31 seconds	6.23 seconds	0.0004 seconds	0.72 seconds	2.76 seconds	0.005 seconds
AUC score	0.91	0.99	0.97	0.99	0.98	0.82

The same configuration was used to evaluate the neural network based classifier, which showed a promising detection rate of 94.8% but a higher false alarm rate of

4.4%. When AutoEncoder was clubbed with robust-scaling, Random Forest again outperforms the other classifiers straightaway with a high detection rate of 91.65% and a low false alarm rate of 0.8%.

Finally, it can be deduced from all the experimentation under this specific configuration that the Random Forest-based classifier Z-score normalized data shows the highest detection rate of 91.93%, as well as the lowest false alarm rate of 0.5%.

Table 7.7: Performance of machine learning classifiers trained with the features extracted by Auto Encoder and scaled with z-score normalization

AutoEncoder with z-score normalization						
	Decision Tree	RandomForest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boost	One Class-SVM
Accuracy	94.70%	96.78%	93.81%	91.4%	95.10%	81.24%
Precision	0.87	0.95	0.87	0.89	0.93	0.79
Recall	0.88	0.91	0.83	0.66	0.83	0.78
F-Score	0.88	0.92	0.85	0.71	0.87	0.78
False Alarm Rate (FAR)	3.1%	0.5%	2.4%	1.7%	0.7%	7.12%
Detection Rate	87.8%	86.7%	78.6%	80.5%	75.3%	76.6%
Training Time	0.31 seconds	6.23 seconds	0.0005 seconds	0.65 seconds	2.76 seconds	0.056 seconds
AUC score	0.92	0.99	0.97	0.98	0.98	0.82

7.2.3 Performance based on the features selected by RF-RFE

Table 7.8 describes the performance of different machine learning classifiers when Random Forest-based Recursive Feature elimination is used as the dimensionality reduction technique with min-max normalization scaling. It can be observed from that Random Forest classifier has outperformed other machine learning classifiers while evaluated using this configuration. Even if the required time to train the random forest model is higher than the other classifiers, the model performs considerably better while reducing the number of false positives and false negatives. It can be observed from the table 7.8 that all the classifier has performed significantly well while reducing the false positives. The same configuration was used to train a neural network, which resulted in a higher detection rate of 95.7% and false alarm rate of

0.8%. Table 7.9 provides a summarization of the neural network-based classifiers trained with the features extracted by RF-RFE.

Table 7.8: Performance of machine learning classifiers trained with the features selected by RF-RFE and scaled with Min-Max normalization

RF-RFE and Min-Max scaler						
	Decision Tree	Random Forest	Logistic Regression	Support Vector Machine(with rbf kernel)	XG-Boost	One Class-SVM
Accuracy	96.28%	97.05%	92.05%	94.66%	95.88%	76.62%
Precision	0.92	0.95	0.87	0.90	0.95	0.76
Recall	0.90	0.91	0.72	0.83	0.85	0.71
F-Score	0.91	0.93	0.76	0.86	0.89	0.73
False Alarm Rate (FAR)	2.1%	0.7%	1.7%	1.9%	0.8%	7.96%
Detection Rate	89.5%	91%	55%	84.15%	72.9%	71.16%
Training Time	0.07 seconds	1.78 seconds	0.0002 seconds	0.61 seconds	1.9 seconds	0.006 seconds
AUC score	0.93	1.00	0.96	0.98	0.99	0.82

Table 7.9: Performance of neural network based classifier trained with the features selected by RF-RFE

	RFE with Min-Max scaler (92 features)	RFE with Standard scaler(92 features)	RFE with Robust scaler (92 features)
Accuracy	97.8	98.25	96.9
False Alarm Rate (FAR)	0.72%	3.1%	4.3%
Detection Rate	96.3%	96.2%	94.7%
Training Time	3.52 seconds	3.22 seconds	3.64 seconds
AUC score	0.982	0.987	0.98

I have conducted experiments using other data scaling techniques with RF-RFE but results were almost similar. The neural network classifier trained on the RF-RFE based features showed a promising detection rate of 96.2% but a higher false alarm rate of 3.1% when trained and evaluated normalized data using z-score. When RFE was clubbed with Min-Max-scaling, it provides a higher detection rate of 96.3% as well as lower false alarm rate of 0.72%.

7.3 The Trade-off between False Alarm Rate and Detection Rate

The effectiveness of classifiers is visually presented by plotting false alarm rate vs detection rate. The framework was built in such as a manner that maximizes the detection and minimizes the false alarm rate. It can be depicted from Figure 7.1 that, the neural network based classifier results the highest detection rate and random forest shows the lowest false alarm rate while using PCA as feature reduction technique.

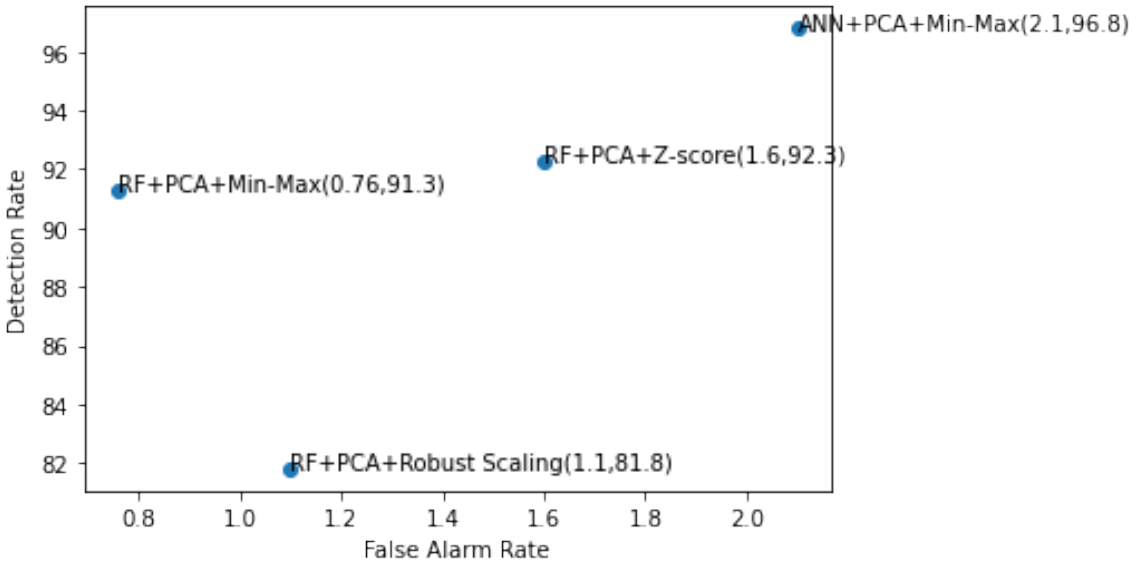

Figure 7.1: False Alarm Rate vs Detection Rate(Classifiers trained with the features retrieved by PCA)

Figure 7.2 visualizes the performance of the classifier when Auto Encoder is used as dimensionality reduction technique. It can be observed that all the learning models were able reduce the false alarm rate very efficiently. The performance of Auto Encoder feature extraction based models has a trade-off between the False alarm rate and detection rate. As an example from Figure 7.3 , random forest indicates the lowest false alarm rate rate but lowest detection rate too which is not acceptable. By analyzing the trade-off for false alarm rate and detection rate, it has been realized that a the final model needs to be chosen in such a way that shows a higher detection rate as well as very low false alarm rate. While comapring different proposed methods, I mainly focused on false alarm rate and detection rate because it represents the optimal efficiency of the model by interpreting classification capability as well as misclassification performance of the model. The false alarm rate not only identifies

false positives, but also recognizes false negatives too.

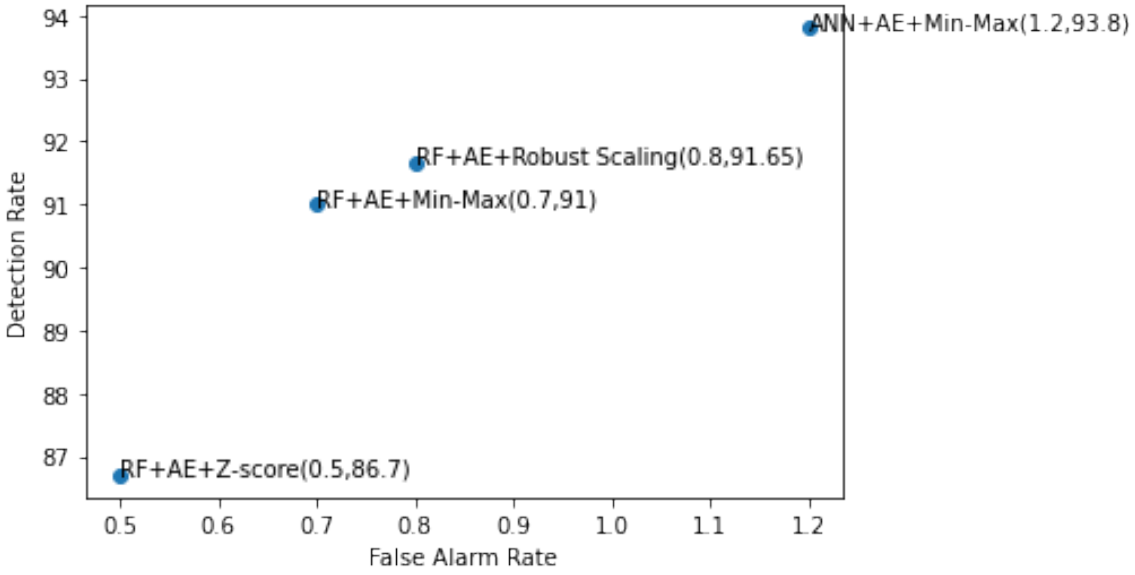

Figure 7.2: False Alarm Rate vs Detection Rate(Classifiers trained with the features retrieved by Auto Encoder)

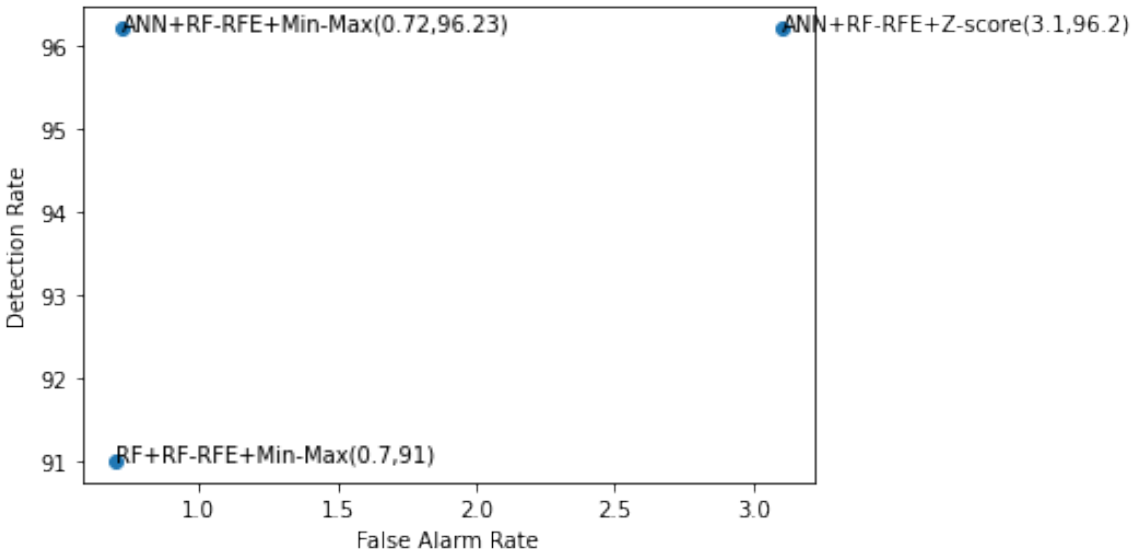

Figure 7.3: False Alarm Rate vs Detection Rate(Classifiers trained with the features selected by RF-RFE)

7.4 Comparison with the existing approaches

Table 7.10 presents a detailed comparison of performance of the proposed approach with the existing state of the art approaches. While performing the experiment and comparing the proposed approaches with the existing ones, I put a significant effort to build a similar environment (system configuration, hardware, dependencies) followed by the other researchers to justify the comparisons [2].

Procedures such as semantic feature extraction, frequency-domain feature selection show a promising detection rate but fail to reduce false-positive rate eventually. Similarly, the LSTM and CNN based language models show an impressive detection rate of 100% but fail to generalize while reducing false alarms. In order to address the trade-off between false alarm rate and detection rate, the designed approach can provide a better result. From all the experimented configurations, I finally chose six models for comparing with the existing methods. From the pool of developed models,**PCA+Min-Max normalization + ANN** model performs best while detecting malicious traces. **Auto Encoder+Min-Max normalization + Random Forest** model performed better then other models while reducing the false alarm rate. But I always aimed for a solution that indicates a perfect balance between false alarm rate and detection rate. I could notice that **RF-RFE+Min-Max normalization + ANN** model showed a high detection rate of 96.8% and a low False alarm rate of 0.72%. This Neural Network-based model exhibits the lowest training time of all the chosen models. I can confidently conclude from these observations that the proposed framework was able to perform signifcantly better than the existing approaches in terms of reducing False alarm rate and increasing detection rate. Additionally, the proposed approach was also ably reduced the training time of the model, which is better than most of the existing approach, as indicated in table 7.10. Though the method proposed by Haider et al. [7] takes lesser time to train the model. However, our proposed approach exhibits a better performance in terms of false alarm rate and detection rate. Even if the raining time is more but still acceptable because of the efficient handling of trade off between false alarm rate and detection rate.

[2]The proposed models are based on the features retrieved by IDZW method [7] and frequency based method [84]. As the creators of these techniques already published a set of results using these methods independently, I compared the result of my proposed hybrid approach with those existing approaches individually and presented my interpretation.

These set of the experiments and evaluation of the models confirm that the proposed framework successfully addresses the research problem discussed in chapter 3.

Table 7.10: Comparison of the performance of the proposed framework with existing systems

	False Alarm Rate	Detection Rate	AUC Score	Processing time
Semantic features and ELM [91]	15%	90%	Not Reported	1 week
System calls short sequence and one class SVM [88]	20%	70%	Not Reported	Few Hours
System calls short sequence and kNN, k-mean clustering [18]	20%	60%	Not Reported	Few Seconds(as reported)
Frequency domain Feature selection approach [22]	16%	95%	Not Reported	Not reported
Using Hidden Markov Model	42%	90%	Not Reported	Few hours
LSTM-Based language modeling [89]	50%	100%	0.84	More than 10 seconds
Combined CNN/RNN based model [92]	60%	100%	0.81	Not reported
Integer Zero Watermark Feature retrieval (using z-score normalized feature and ELM) [7]	1%	91%	Not Reported	1 millisecond
Proposed Approach(PCA+Min-Max normalization + Random Forest)	0.76%	91.33%	0.99	6.18 seconds
Proposed Approach(PCA+Min-Max normalization + ANN)	2.1%	96.8%	0.98	3.25 seconds
Proposed Approach(Auto Encoder+Min-Max normalization + ANN)	4.4%	93.8%	0.97	3.55 seconds
Proposed Approach(Auto Encoder+Min-Max normalization + Random Forest)	0.7%	91.65%	0.99	6.12 seconds
Proposed Approach(R-RFE+Min-Max normalization + ANN)	0.72%	96.3%	0.98	3.25 seconds

Chapter 8

Conclusion

The thesis has broadly focused on building a machine learning framework for host-based intrusion detection using system call identifiers. As stated in the research problem in Section 3, I primarily focused on building a hybrid feature retrieval technique that retains both the frequency features of the system calls, as well as the hidden representative information. It has been realized that feature retrieval plays a crucial role in extracting valuable information from the system call that helps to differentiate the malicious system calls from the nonmalicious calls. In addition to the feature retrieval approach, I further concentrated on the efficient application of dimensionality reduction techniques and development decision engines. Dimensionality reduction techniques represent the features in the lower dimensional space that reduces the processing cost of the decision engine. Decision engines help to classify if a system call is malicious or not. An efficient amalgamation of all these above approaches pave the path for building the HIDS framework that increases the detection rate, reduces the false alarm rate, and minimizes training time than most of the existing methods.

It has been realized during the framework development that feature retrieval plays a major role that affects the evaluation of the proposed framework significantly. In ADFA-LD, some long traces contains more than a thousand system call but only a small part of it contains anomaly. Short sequences extracted from long malicious traces can be similar to short sequences extracted from benign traces. Training with these ambiguous short sequences might lead to major confusion for the decision engines. I have noticed during the literature survey that researchers have focused on the feature retrieval to improve the performance of the classifiers and I reached to the same conclusion while evaluating the proposed framework.

The feasability of the application of the feature retireval approach is very generic in nature and it does not depend on any external dependencies. Even if the kernel

version is changed, the proposed feature retireval technique can work similarly to extract both the natural difference and sequence based features for each system call in a trace.

Another important observation that has been made during the experimental phase is a balance between false alarm rate and detection rate. It has been noticed that,if I tune the classifier to maximize the detection rate, it reduces the false alarm rate eventually and vice versa. It is very important to find the "perfect" spot that balances between these metrics.

I have used both supervised and semi-supervised approaches in order to train the classifiers. Both the approach focuses the same set of features but in the supervised approach, the classifiers are trained with both the benign and malicious traces but in the semi-supervised approach the classifiers are trained with only benign system call traces in order create a generalized benign profile. I have two major contributions in this thesis: 1) The hybrid feature retrieval technique that was able to discriminate efficiently between benign and malicious traces. I could analyze and justify from the previous work that the proposed model has performed better the state of the art approaches based on IDZW method, frequency modeling method, and language modeling methods, and HMM based methods. 2) Using different dimensionality reduction techniques, the training time of the classifiers have reduced drastically without affecting the evaluation metrics negatively.

8.1 Limitations

The framework is designed and validated to detect low footprint attacks. As the nature of low footprint and high footprint attacks are diferent and the degree of separability between these two are significanlty low, it is not sure if the proposed frmaework will work efficienly for detecting low footprint attacks.

The results of the learning models were compared in term of detection rate, and false alarm rate. No statistical tests such as comparison of p value have not been performed to compare the perfomace of learning techniques.

8.2 Future Work

As a future scope of the current work, I want to extend my research in the field of building a Cognitive Module(CM) which can be integrated with the decision engine module of the existing framework. The cognitive module works on the principle of rule-based detection. The cognitive module can be placed in such a way that all the test system call traces can be passed through the Cognitive module at first. Then the traces coming out of the CM can be fed into the machine learning based decision engine. The main reasoning behind this approach is to accelerate the attack detection rate. As the rule based detection is fast and efficient, the traces which are detectable by the rule-based system, are not required to be passed through the classification model. This approach can reduce the computation cost of the system dramatically. This is an amalgamation of the misuse detection approach and the anomaly detection approach. In addition to that, the observation from the decision engine can be used to create and update rules and policies for the rule-based detector. This approach makes will make the framework adaptive for detecting novel attacks.

In addition to the above scope, an Intrusion Response System (IRS) and Alert Generation module can be integrated with the proposed HIDS framework. An IDS with an Intrusion Response Module enables a system to mitigate the damage and recover from any detected attacks. There are a few important criteria that classify and define the working principles of the Intrusion Response System such as Level of automation, response cost, response time, and adjustment availability. To implement this module with effective generalization, I can build several sub-modules such as the Alert Integration module, Alert correlation module, Adaptive Risk Assessment module, and Recovery and Eradicate module.

Another future scope of present work can be unfolded by implementing the proposed HIDS based on the principle of Federated Learning. Federate Learning, proposed by McMahan et al. [141] is a collaborative learning technique that jointly trains global machine learning models without sharing their privacy-sensitive data on a server. Federated learning can be considered a perfect fit for distributed multitask computing. In a typical horizontal federated learning structure, there is one server connected to multiple clients. Each client trains its own data and sends the trained

model (updated parameters) to the server in order to execute aggregation. The training data in each of the individual client is never uploaded to the server. Only, the aggregated and updated model in the server is sent to the individual host. This training process gets repeated iteratively until the global model in the server converges or the number of the maximum round during the training procedure is reached. In the federated learning structure, the federated averaging algorithm is used to calculate the average of the local model weights or the gradient updates in each of the individual clients. This privacy-preserving approach will ensure the security of the training process as the individual hosts are not required to send any sensitive training data to the centralized server. As the server and hosts involved in parameter exchanging, the training cost can be significantly reduced using this approach. Another major advantage of this approach is the fast and efficient development of the generalized profile.

This is how the existing research can be further developed in different directions.

Bibliography

[1] SYSTEMS and TECHNOLOGY, "Detecting cyberattacks through the ever-evolving cybersecurity industry," 2019. [Online]. Available: https://www.getsmarter.com/blog/career-advice/how-to-detect-a-cyberattack/

[2] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227 – 2235, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036641100209X

[3] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-Based Intrusion Detection System with System Calls: Review and Future Trends," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, November 2018. [Online]. Available: http://doi.acm.org/10.1145/3214304

[4] Y. Wang, S. Wen, Y. Xiang, and W. Zhou, "Modeling the Propagation of Worms in Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 942–960, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6644336/

[5] N. Moustafa and J. Slay, "The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems," in *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. Kyoto, Japan: IEEE, November 2015, pp. 25–31. [Online]. Available: http://ieeexplore.ieee.org/document/7809531/

[6] Metasploit, "Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit." [Online]. Available: https://www.metasploit.com/

[7] W. Haider, J. Hu, X. Yu, and Y. Xie, "Integer Data Zero-Watermark Assisted System Calls Abstraction and Normalization for Host Based Anomaly Detection Systems," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, November 2015, pp. 349–355.

[8] W. Haider, J. Hu, and M. Xie, "Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, June 2015, pp. 513–517.

[9] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, "A sense of self for Unix processes," in *Proceedings 1996 IEEE Symposium on Security and Privacy*, May 1996, pp. 120–128, iSSN: 1081-6011.

[10] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, July 1998. [Online]. Available: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JCS-980109

[11] S. Forrest, S. Hofmeyr, and A. Somayaji, "The Evolution of System-Call Monitoring," in *2008 Annual Computer Security Applications Conference (ACSAC)*. Anaheim, CA, USA: IEEE, December 2008, pp. 418–430. [Online]. Available: http://ieeexplore.ieee.org/document/4721577/

[12] W. Haider, J. Hu, Y. Xie, X. Yu, and Q. Wu, "Detecting Anomalous Behavior in Cloud Servers by Nested-Arc Hidden SEMI-Markov Model with State Summarization," *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 305–316, September 2019.

[13] L. Mé, S. Profile, J. Zimmermann, L. Mé, and C. Bidan, "Experimenting with a policy-based hids based on an information flow control model," in *In Proceedings of the Annual Computer Security Applications Conference (ACSAC*, 2003.

[14] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2013, pp. 4487–4492.

[15] U. o. N. M. Computer Science Department, Farris Engineering Center, "Computer Immune Systems - Data Sets and Software." [Online]. Available: https://www.cs.unm.edu/~immsec/systemcalls.htm

[16] M. V. Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," in *Recent Advances in Intrusion Detection*, G. Goos, J. Hartmanis, J. van Leeuwen, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2820, pp. 220–237. [Online]. Available: http://link.springer.com/10.1007/978-3-540-45248-5_13

[17] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture, "A host-based anomaly detection approach by representing system calls as states of kernel modules," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, November 2013, iSSN: 1071-9458, 2332-6549.

[18] M. Xie, J. Hu, X. Yu, and E. Chang, "Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD," in *Network and System Security*, M. H. Au, B. Carminati, and C.-C. J. Kuo, Eds. Cham: Springer International Publishing, 2014, vol. 8792, pp. 542–549. [Online]. Available: http://link.springer.com/10.1007/978-3-319-11698-3_44

[19] "Study: Hackers Attack Every 39 Seconds," library Catalog: eng.umd.edu. [Online]. Available: https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds

[20] "Business Losses to Cybercrime Data Breaches to Exceed $5 trillion by 2024," library Catalog: www.juniperresearch.com. [Online]. Available: https://www.juniperresearch.com/press/press-releases/business-losses-cybercrime-data-breaches

[21] D. Milkovich, "13 alarming cyber security facts and stats," 2018. [Online]. Available: https://www.cybintsolutions.com/cyber-security-facts-stats/

[22] W. Haider, J. Hu, and N. Moustafa, "Designing Anomaly Detection System for Cloud Servers by Frequency Domain Features of System Call Identifiers and Machine Learning," in *Mobile Networks and Management*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2018, pp. 137–149.

[23] W.Lin, H.Lin, P.Wang, B.Wu, and J.Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *2018 IEEE International Conference on Applied System Invention (ICASI)*. Chiba, Japan: IEEE, 2018, pp. 1107–1110.

[24] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, January 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804512001944

[25] "Cyberattack," February 2020, page Version ID: 940598565. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cyberattack&oldid=940598565

[26] Weltwirtschaftsforum and Z. I. Group, *Global risks 2019: insight report*. World Economic Forum, Switzerland, 2019. [Online]. Available: http://www3.weforum.org/docs/WEF_Global_Risks_Report_2019.pdf

[27] R. Shirey, "Internet Security Glossary," RFC Editor, Tech. Rep. RFC2828, May 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2828

[28] he National Counterintelligence and S. C. (NCSC), "Ncsc home." [Online]. Available: https://www.dni.gov/index.php/ncsc-home

[29] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, December 2015.

[30] G. R. Zargar and P. Kabiri, "Identification of effective network features for probing attack detection," in *2009 First International Conference on Networked Digital Technologies*, July 2009, pp. 392–397.

[31] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* USA: Insecure, 2009.

[32] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/586110.586144

[33] P. Ram and D. K. Rand, "Satan: double-edged sword," *Computer*, vol. 28, no. 6, pp. 82–83, June 1995.

[34] J. Gadge and A. A. Patil, "Port scan detection," in *2008 16th IEEE International Conference on Networks*, December 2008, pp. 1–6.

[35] A. Javaid, Q. Niyaz, W. Sun, and M. 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)*, ser. BICT'15. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26. [Online]. Available: http://dx.doi.org/10.4108/eai.3-12-2015.2262516

[36] R.Masood, U. e Ghazia, and Z. Anwar, "Swam: Stuxnet worm analysis in metasploit," in *2011 Frontiers of Information Technology.* Islamabad, Pakistan: IEEE, December 2011, pp. 142–147.

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

[38] J. A. Mahal and T. Charles Clancy, "Analysis of pilot-spoofing attack in miso-ofdm system over correlated fading channel," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, October 2018, pp. 341–346.

[39] T. Dinh Tu, C. Guang, G. Xiaojun, and P. Wubin, "Webshell detection techniques in web applications," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 2014, pp. 1–7.

[40] R. Zhu, T. Shu, and H. Fu, "Empirical statistical inference attack against PHY-layer key extraction in real environments," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, October 2017, pp. 46–51.

[41] O. Nakhila, A. Attiah, Y. Jin, and C. Zou, "Parallel active dictionary attack on wpa2-psk wi-fi networks," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, October 2015, pp. 665–670.

[42] E. Erdin, H. Aksu, S. Uluagac, M. Vai, and K. Akkaya, "OS Independent and Hardware-Assisted Insider Threat Detection and Prevention Framework," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, October 2018, pp. 926–932.

[43] E. Ficke, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Characterizing the Effectiveness of Network-Based Intrusion Detection Systems," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, October 2018, pp. 76–81.

[44] Y. Dai, M. Xie, K. Poh, and G. Liu, "A study of service reliability and availability for distributed systems," *Reliability Engineering & System Safety*, vol. 79, no. 1, pp. 103 – 112, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832002002004

[45] A. Muallem, S. Shetty, L. Hong, and J. W. Pan, "Tddeht: Threat detection using distributed ensembles of hoeffding trees on streaming cyber datasets," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, October 2018, pp. 1–6.

[46] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, June 2009, pp. 268–273.

[47] I. Almomani and B. Al-Kasasbeh, "Performance analysis of leach protocol under denial of service attacks," in *2015 6th International Conference on Information and Communication Systems (ICICS)*, April 2015, pp. 292–297.

[48] A. H. Anwar, J. Kelly, G. Atia, and M. Guirguis, "Stealthy edge decoy attacks against dynamic channel assignment in wireless networks," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, October 2015, pp. 671–676.

[49] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 475–488. [Online]. Available: http://doi.acm.org/10.1145/2663716.2663755

[50] Avast, "What is Spyware | Free Spyware Scanner & Removal Tool | Avast." [Online]. Available: https://www.avast.com/c-spyware?hsSkipCache=true&hs_ungate__cos_renderer_combine_all_css_disable=true

[51] Norton, "What is a computer worm and how does it work?" [Online]. Available: https://us.norton.com/internetsecurity-malware-what-is-a-computer-worm.html

[52] Avast, "What is ransomware & how to remove it avast." [Online]. Available: https://www.avast.com/c-ransomware?hsSkipCache=true&hs_ungate_cos_renderer_combine_all_css_disable=true

[53] S. W. Boyd and A. D. Keromytis, "SQLrand: Preventing SQL Injection Attacks," in *Applied Cryptography and Network Security*, M. Jakobsson, M. Yung, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 292–302.

[54] D. Kapetanovic, G. Zheng, and F. Rusek, "Physical layer security for massive mimo: An overview on passive eavesdropping and active attacks," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 21–27, 2015.

[55] G. Conti and K. Abdullah, "Passive visual fingerprinting of network attack tools," in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, ser. VizSEC/DMSEC '04. New York, NY, USA: ACM, 2004, pp. 45–54. [Online]. Available: http://doi.acm.org/10.1145/1029208.1029216

[56] M. B. Salem and S. J. Stolfo, "Masquerade attack detection using a search-behavior modeling approach," *Columbia University, Computer Science Department, Technical Report CUCS-027-09*, 2009.

[57] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen, "Sessionshield: Lightweight protection against session hijacking," in *Engineering Secure Software and Systems*, Ú. Erlingsson, R. Wieringa, and N. Zannone, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 87–100.

[58] N. Jovanovic, E. Kirda, and C. Kruegel, "Preventing cross site request forgery attacks," in *2006 Securecomm and Workshops*, August 2006, pp. 1–10.

[59] D. Hartley, "What Is SQL Injection?" in *SQL Injection Attacks and Defense*. Elsevier, 2012, pp. 1–25. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9781597499637000013

[60] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," in *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '06. New York, NY, USA: ACM, 2006, pp. 372–382. [Online]. Available: http://doi.acm.org/10.1145/1111037.1111070

[61] T. Threepak and A. Watcharapupong, "Web attack detection using entropy-based analysis," in *The International Conference on Information Networking 2014 (ICOIN2014)*, February 2014, pp. 244–247.

[62] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Journal of Information Security and applications*, vol. 22, pp. 113–122, 2015.

[63] Avast, "What is Phishing? | Avoid Phishing Emails, Scams & Attacks | Avast." [Online]. Available: https://www.avast.com/c-phishing?hsSkipCache=true&hs_ungate___cos_renderer_combine_all_css_disable=true

[64] D. Scott and R. Sharp, "Abstracting application-level web security," in *Proceedings of the 11th International Conference on World Wide Web*, ser. WWW '02. New York, NY, USA: ACM, 2002, pp. 396–407. [Online]. Available: http://doi.acm.org/10.1145/511446.511498

[65] Avast, "Rootkit Definition | Free Rootkit Scanner & Remover | Anti-rootkit." [Online]. Available: https://www.avast.com/c-rootkit?hsSkipCache=true&hs_ungate___cos_renderer_combine_all_css_disable=true

[66] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1145–1172, 2017.

[67] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–41, September 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2808687.2808691

[68] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets," *arXiv:1806.03517 [cs]*, June 2018, arXiv: 1806.03517. [Online]. Available: http://arxiv.org/abs/1806.03517

[69] M. Roesch, "Snort - lightweight intrusion detection for networks," *LISA '99: Proceedings of the 13th USENIX conference on System administrationNovember 1999*, p. 11, 1999.

[70] OSSEC, "OSSEC - World's Most Widely Used Host Intrusion Detection System - HIDS." [Online]. Available: https://www.ossec.net/

[71] G. Vigna and C. Kruegel, "Host-based Intrusion Detection," in *Handbook of Information Security*, 2006, p. 35.

[72] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and Survey of Collaborative Intrusion Detection," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1–33, May 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2775083.2716260

[73] Z. A. Haddad, M. Hanoune, and A. Mamouni, "A collaborative framework for intrusion detection (C-NIDS) in Cloud computing," in *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, May 2016, pp. 261–265.

[74] V. Chandola, E. Eilertson, L. Ertoz, G. Simon, and V. Kumar, "Minds: Architecture & Design," in *Data Warehousing and Data Mining Techniques for Cyber Security*. Boston, MA: Springer US, 2007, vol. 31, pp. 83–107. [Online]. Available: http://link.springer.com/10.1007/978-0-387-47653-7_6

[75] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*, May 1999, pp. 133–145, iSSN: 1081-6011.

[76] Y. Qiao, X. Xin, Y. Bin, and S. Ge, "Anomaly intrusion detection method based on HMM," *Electronics Letters*, vol. 38, no. 13, pp. 663–664, June 2002.

[77] D. Hoang, J. Hu, and P. Bertok, "A multi-layer model for anomaly intrusion detection using program sequences of system calls," in *The 11th IEEE International Conference on Networks, 2003. ICON2003.*, 01 2003, pp. 531– 536.

[78] R. Davis, B. Lovell, and T. Caelli, "Improved estimation of hidden Markov model parameters from multiple observation sequences," in *Object recognition supported by user interaction for service robots*, vol. 2. Quebec City, Que., Canada: IEEE Comput. Soc, 2002, pp. 168–171. [Online]. Available: http://ieeexplore.ieee.org/document/1048264/

[79] X. Hoang and J. Hu, "An efficient hidden Markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955)*, vol. 2, November 2004, pp. 470–474, iSSN: 1531-2216.

[80] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and S. Gagnon, "A trace abstraction approach for host-based anomaly detection," in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. Verona, NY, USA: IEEE, May 2015, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7208644/

[81] D. Yuxin, Y. Xuebing, Z. Di, D. Li, and A. Zhanchao, "Feature representation and selection in malicious code detection methods based on static system calls," *Computers & Security*, vol. 30, no. 6-7, pp. 514–524, September 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016740481100068X

[82] E. Aghaei and G. Serpen, "Host-based anomaly detection using Eigentraces feature extraction and one-class classification on system call trace data," *ArXiv*, p. 11, 2019.

[83] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *2015 7th Conference on Information and Knowledge Technology (IKT)*, May 2015, pp. 1–5.

[84] M. Xie, J. Hu, X. Yu, and E. Chang, "Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD," in *Network and System Security*, M. H. Au, B. Carminati, and C.-C. J. Kuo, Eds. Cham: Springer International Publishing, 2014, vol. 8792, pp. 542–549. [Online]. Available: http://link.springer.com/10.1007/978-3-319-11698-3_44

[85] M. Xie, S. Han, and B. Tian, "Highly Efficient Distance-Based Anomaly Detection through Univariate with PCA in Wireless Sensor Networks," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, November 2011, pp. 564–571, iSSN: 2324-9013.

[86] M. Xie, J. Hu, S. Han, and H.-H. Chen, "Scalable Hypergrid k-NN-Based Online Anomaly Detection in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1661–1670, August 2013, conference Name: IEEE Transactions on Parallel and Distributed Systems.

[87] A. N. Mahmood, J. Hu, Z. Tari, and C. Leckie, "Critical infrastructure protection: Resource efficient sampling to improve detection of less frequent patterns in network traffic," *J. Netw. Comput. Appl.*, vol. 33, no. 4, p. 491–502, July 2010. [Online]. Available: https://doi.org/10.1016/j.jnca.2010.01.003

[88] M. Xie, J. Hu, and J. Slay, "Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD," in *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, august 2014, pp. 978–982.

[89] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems," *arXiv:1611.01726 [cs]*, November 2016. [Online]. Available: http://arxiv.org/abs/1611.01726

[90] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

[91] G. Creech and J. Hu, "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 807–819, April 2014.

[92] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host Based Intrusion Detection System with Combined CNN/RNN Model," in *ECML PKDD 2018 Workshops*, C. Alzate, A. Monreale, H. Assem, A. Bifet, T. S. Buda, B. Caglayan, B. Drury, E. García-Martín, R. Gavaldà, I. Koprinska, S. Kramer, N. Lavesson, M. Madden, I. Molloy, M.-I. Nicolae, and M. Sinn, Eds. Cham: Springer International Publishing, 2019, vol. 11329, pp. 149–158.

[93] Q. Chen, R. Luley, Q. Wu, M. Bishop, R. W. Linderman, and Q. Qiu, "AnRAD: A Neuromorphic Anomaly Detection Framework for Massive Concurrent Data Streams," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 1622–1636, May 2018, conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[94] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," Defense Technical Information Center, Tech. Rep., October 2000. [Online]. Available: http://www.dtic.mil/docs/citations/ADA401496

[95] W. Lee, S. Stolfo, and P. Chan, "Learning patterns from unix process execution traces for intrusion detection," *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, May 1997.

[96] G. Tandon and P. Chan, "Learning rules from system call arguments and sequences for anomaly detection," *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, p. 10, 2005.

[97] Y. Lin, Y. Zhang, and Y.-j. Ou, "The Design and Implementation of Host-Based Intrusion Detection System," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, April 2010, pp. 595–598.

[98] J. Zimmermann, L. Mé, and C. Bidan, "Introducing reference flow control for detecting intrusion symptoms at the os level," in *RAID'02: Proceedings of the 5th international conference on Recent advances in intrusion detection October 2002*, vol. 2516, 10 2002, pp. 292–306.

[99] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: A Content Anomaly Detector Resistant to Mimicry Attack," in *Recent Advances in Intrusion Detection*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, D. Zamboni, and C. Kruegel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4219, pp. 226–248, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11856214_12

[100] T. Y. Win, H. Tianfield, and Q. Mair, "Detection of Malware and Kernel-Level Rootkits in Cloud Computing Environments," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. New York, NY, USA: IEEE, November 2015, pp. 295–300. [Online]. Available: http://ieeexplore.ieee.org/document/7371497/

[101] E. . I. T. U. C. U. Creech, Gideon, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," 2014.

[102] Q. A. Tran, F. Jiang, and J. Hu, "A Real-Time NetFlow-based Intrusion Detection System with Improved BBNN and High-Frequency Field Programmable Gate Arrays," June 2012, pp. 201–208, iSSN: 2324-9013.

[103] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *Journal of Network and Computer Applications*, vol. 87, pp. 185–192, June 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517301273

[104] T. Community, "HomePage." [Online]. Available: https://tiki.org/HomePage

[105] W. Haider, J. Hu, and N. Moustafa, "Designing Anomaly Detection System for Cloud Servers by Frequency Domain Features of System Call Identifiers and Machine Learning," in *Mobile Networks and Management*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J. Hu, I. Khalil, Z. Tari, and S. Wen, Eds. Springer International Publishing, 2018, pp. 137–149.

[106] J. Davis and S. Magrath, "A Survey of Cyber Ranges and Testbeds." Edinburgh, Australia: Cyber Electronic Welfare Division, 2013, p. 38.

[107] A. I. Abubakar, H. Chiroma, S. A. Muaz, and L. B. Ila, "A Review of the Advances in Cyber Security Benchmark Datasets for Evaluating Data-Driven Based Intrusion Detection Systems," *Procedia Computer Science*, vol. 62, pp. 221–227, 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050915025788

[108] A. Khan and S. A. Husain, "A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations," in *The Scientific World Journal*, vol. 2013, 2013, pp. 1–16. [Online]. Available: http://www.hindawi.com/journals/tswj/2013/796726/

[109] Z. Jalil, A. M. Mirza, and H. Jabeen, "Word length based zero-watermarking algorithm for tamper detection in text documents," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 6, April 2010, pp. 378–382.

[110] A. Li, B. Lin, and G. Lü, "Authentication of gis vector data based on zero-watermarking," in *The international archives of the photogrammetry, remote sensing and spatial information sciences.* Beijing, China: ISPRS, 2008.

[111] K. Pearson, "LIII. *On lines and planes of closest fit to systems of points in space*," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, November 1901. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/14786440109462720

[112] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002. [Online]. Available: https://doi.org/10.1023/A:1012487302797

[113] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," in *2010 10th International Conference on Intelligent Systems Design and Applications*, November 2010, pp. 363–367.

[114] H. Abdi and L. J. Williams, "Principal component analysis: Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, July 2010. [Online]. Available: http://doi.wiley.com/10.1002/wics.101

[115] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi, "Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 2, pp. 83 – 90, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743906000232

[116] B. F. Darst, K. C. Malecki, and C. D. Engelman, "Using recursive feature elimination in random forest to account for correlated variables in high dimensional data," *BMC Genetics*, vol. 19, no. S1, p. 65, September 2018. [Online]. Available: https://bmcgenet.biomedcentral.com/articles/10.1186/s12863-018-0633-8

[117] M. Sakurada and T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis - MLSDA'14*. Gold Coast, Australia QLD, Australia: ACM, 2014, pp. 4–11. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2689746.2689747

[118] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7307098/

[119] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Science & Business Media, June 2013.

[120] J. Brownlee, "A gentle introduction to xgboost for applied machine learning," August 2016. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[121] XGBoost, "Xgboost." [Online]. Available: https://xgboost.ai/

[122] E. Inzaugarat, "Understanding Neural Networks: What, How and Why?" May 2019, library Catalog: towardsdatascience.com. [Online]. Available: https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31

[123] X. Yao, "Evolving artificial neural networks," vol. 87, no. 9. IEEE, September 1999, pp. 1423–1447.

[124] V. Nigam, "Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning," February 2020. [Online]. Available: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90

[125] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[126] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," in *Shape, Contour and Grouping in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. [Online]. Available: https://doi.org/10.1007/3-540-46805-6_19

[127] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, July 2001. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/089976601750264965

[128] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, December 2010. [Online]. Available: http://link.springer.com/10.1007/s13042-010-0001-0

[129] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: A preliminary analysis of ADFA-LD," in *2013 6th International Congress on Image and Signal Processing (CISP)*. Hangzhou, China: IEEE, December 2013, pp. 1711–1716. [Online]. Available: http://ieeexplore.ieee.org/document/6743952/

[130] X. H. Cao, I. Stojkovic, and Z. Obradovic, "A robust data scaling algorithm to improve classification accuracies in biomedical data," *BMC Bioinformatics*, vol. 17, no. 1, p. 359, December 2016. [Online]. Available: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016-1236-x

[131] S. Asaithambi, "Why, How and When to Scale your Features," December 2017, library Catalog: medium.com. [Online]. Available: https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e

[132] B. A. Keen, "Feature scaling with scikit-learn- ben alex keen." [Online]. Available: http://benalexkeen.com/feature-scaling-with-scikit-learn/

[133] X. Huang, L. Zhang, B. Wang, F. Li, and Z. Zhang, "Feature clustering based support vector machine recursive feature elimination for gene selection," *Applied Intelligence*, vol. 48, no. 3, pp. 594–607, March 2018. [Online]. Available: http://link.springer.com/10.1007/s10489-017-0992-2

[134] s.-l. d. B. L. 2007 2019, "sklearn.decomposition.pca scikitlearn 0.22.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[135] s. d. B. L. 2007-2019, "sklearn.feature selection.rfe scikitlearn 0.22.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[136] "Home - Keras Documentation." [Online]. Available: https://keras.io/

[137] R. G. Mantovani, T. Horváth, R. Cerri, S. B. Junior, J. Vanschoren, and A. C. P. d. L. F. de Carvalho, "An empirical study on hyperparameter tuning of decision trees," *arXiv:1812.02207 [cs, stat]*, February 2019, arXiv: 1812.02207. [Online]. Available: http://arxiv.org/abs/1812.02207

[138] R. Toshniwal, "Demystifying ROC Curves," January 2020, library Catalog: towardsdatascience.com. [Online]. Available: https://towardsdatascience.com/demystifying-roc-curves-df809474529a

[139] N. R. Cook, "Use and Misuse of the Receiver Operating Characteristic Curve in Risk Prediction," *Circulation*, vol. 115, no. 7, pp. 928–935, February 2007. [Online]. Available: https://www.ahajournals.org/doi/10.1161/CIRCULATIONAHA.106.672402

[140] R. A. Maxion and R. R. Roberts, "Proper use of roc curves in intrusion/anomaly detection," *University of Newcastle upon Tyne, Computing Science Tyne, UK*, p. 33, 2004.

[141] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv:1602.05629 [cs]*, February 2017, arXiv: 1602.05629. [Online]. Available: http://arxiv.org/abs/1602.05629

[142] E. . I. T. U. C. U. Creech, Gideon, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," 2014.

Appendix A

Extracting Short Sequence from Long Traces

The method proposed by Forrest et al. [9] used STIDE(Sequency Time Delay Embedding) algorithm to extract short sequence from system call traces. The STIDE algorithm is designed to model individual program and it needs to trained for every individual process. The process has two parts such as creating the database with normal traces, and comparison of new sequences against the database. While creating the database, the allowable short sequences are extracted using a sliding window of specific size to extract an understanding reagrding which system calls typically follow the other system calls. The length of the system calls is arbitary and defined by the user. Figure A.1 describes an example where short sequences are extracted using a window of size 5 [142].

Figure A.1: Short Sequence Extraction using Sliding Window [142]

This sliding window with same window size is applied to the test system calls. Then, the created databased is searched with the short sequences retireved from the test system calls. If a short sequence is not found in the database, that is considered as a mismatch. Forrest et al. defined maximum number of possible mimatch by considering length of the trace and windown size. Any new trace that has surpasses the maximum number of mismatch considered as a malicious trace.