A METHODOLOGY TO DESIGN INTRUSION DETECTION SYSTEMS (IDS) FOR IoT/NETWORKING PROTOCOLS

By

Pratik Satam

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2019

THE UNIVERSITY OF ARIZONA

GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Pratik Satam titled "A Methodology to design Intrusion Detection Systems (IDS) for IoT/Networking protocols" and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

[illegible] Date: March 28, 2019

Dr. Salim Hariri

[illegible] Date: March 28, 2019

Dr. Ali Akoglu

[illegible] Date: March 28, 2019

Dr. Gregory Ditzler

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

[illegible] Date: March 28, 2019

Dissertation Director: Dr. Salim Hariri

Acknowledgements

Foremost, I would like to express my sincere appreciation to my advisor, Prof. Salim Hariri, whose invaluable advice and support have made this work possible. I am most grateful for his encouragement, broad knowledge and deep insights which have made the research experience fascinating and exciting throughout my whole graduate study.

I would like to express my love and appreciation for my parents, family and friends, for encouraging me in my academic pursuits and their support over these years. Without them I would have never gotten here.

I sincerely thank my doctoral committee members, Dr. Ali Akoglu, Dr. Gregory Ditzler, and Dr. Beichuan Zhang for serving on my dissertation and comprehensive exams committees, providing helpful advices and suggestions.

I would like to really appreciate my colleagues from ACL lab Shalaka Satam, Jesus Pacheco, Cihan Tunc, and all faculty and staff in the department of Electrical and Computer Engineering which made a great atmosphere for me to enjoy my research. A special thanks Nancy Emptage, Anna Seiple, Tami Whelan and Christine Eisenfeld for the administrative help provided during my tenure as a Ph.D. student in the ACL lab.

.

INDEX

List of Figures

List of Tables

ABSTRACT

Over the last few decades, the Internet has grown from a network that connected two research Universities to a juggernaut that encompasses the whole world with over 4.1 billion users as of July 2018, with a growth rate of 1052% from 2000-2018 [47]. Modern internet supports a wide variety of features and services like cloud computing and storage, social networking, content services, blogs and social interactions, Online banking and shopping, etc. Alongside the development of the internet, technologies relating to sensors [1,2,3,4], wireless communications [5,6,7], and mobile computing have seen an unprecedented growth [8,9,10], which has contributed to the development of a new paradigm: Internet of Things (IoT). The core concept of IoT involves forming connected devices that can be accessed ubiquitously from anywhere. These IoT devices have sensors and some processing and programming capabilities to support smart or Intelligent operations. Smart devices include wearables like smart watches, shoes, glasses, smart phones, smart refrigerators, smart cars, etc. This fast-paced growth of the internet and IoT infrastructures and services has introduce a challenging security problem due to the exponential growth in vulnerabilities and potential exploitations by cyber attackers. There is a dire need for an effective means to secure the cyber space against any type of threats that are known or unknown. This research presents a methodology to design Anomaly Behavior based Intrusion Detection Systems (AB-IDS) to secure networking and IoT protocols.

An AB-IDS has a complete understanding of the semantics of the normal behavior of its target system, consequently allowing it to detect any malicious attacks on the system that forces it to operate abnormally. This approach of monitoring and accurately characterizing the normal behavior instead of looking for specific attack signatures (as done by signature-based IDS' [15]) allows the AB-IDS to detect new and modified attacks. Since each protocol has its own specification, it is hard to develop one AB-IDS that is able to secure all the protocols. Instead, we adopt a more granular approach that involves developing multiple micro AB-IDS' where each one is specialized in detecting anomalous behavior in its protocol, and the results from each of these micro AB-IDS' are aggregated to present a wholistic picture of the current operational state of the complete system. Designing of these micro intrusion detection systems is a time-consuming task that requires an in depth understanding of the protocols. To aid this research approach, in this *dissertation* we develop a methodology to design the micro AB-IDS' using machine learning models.

The approach methodology involves following steps: 1. Threat modelling analysis; 2. Feature selection and protocol foot printing to characterize the behavior of the protocol; and 3. Use the protocol foot printing data structures to develop machine learning models that characterize accurately the normal behavior of the protocol to be protected by the micro AB-IDS. The threat modelling provides a formal approach to model the behavior of the protocol, identify potential attack vectors that target the protocols and develop mechanisms to protect protocol operations against these attack vectors. The feature selection step involves selection of correct features that helps characterize the behavior of the protocol.

This step also involves designing and using different innovative data structures that help capture/represent the behavior of the protocol. In our research we concluded that Observation flows (OF) and n-grams are powerful data structures that can be used to characterize the behavior of the protocols. The last step involves developing machine learning models using the features obtained in Step 2 to differentiate the normal behavior of the machine learning model from the abnormal. We have evaluated our approach by designing micro intrusion detect systems to detect attacks on the Wi-Fi protocol, the DNS protocol and the HTML protocol. The experimental results show that the IDS' designed using this approach have a very high accuracy with very low false positives and false negatives for new and modified attacks.

CHAPTER 1: INTRODUCTION

1.1 MOTIVATION

Over the last few decades, the Internet has grown from a network that connected two research Universities to a juggernaut that encompasses the whole world with over 4.1 billion users as of July 2018 with a growth rate of 1052% from 2000-2018 [1]. Modern internet supports a wide variety of features and services like, cloud computing and storage, social networking, content delivery services, blogs and social interactions, online banking and shopping, just to name a few. Deployment of these technologies has made the internet a household commodity. Alongside the development of the internet, technologies relating to sensors [2,3,4,5], wireless communications [6,7], and mobile computing have seen an unprecedented growth [8], which has contributed to the introduction of the new paradigm: Internet of Things (IoT).

Internet of Things is a paradigm that will enable all devices and objects to have the capability of being connected, sense surroundings, communicate over multiple communication networks, and connect with other Internet enabled devices and services [2,3,4,5,8]. The core concept of IoT involves interfacing all devices with sensor data and consequently making the device ‘smart’ by providing a wide range of intelligent services that cover all aspects of our life and economy. Smart devices are proliferating our daily lives rapidly, wherein wearable technology (see Figure 1.1) like smart watches, shoes, glasses, devices like smart phones, smart refrigerators, smart cars, etc. are becoming an integral part of our lives.

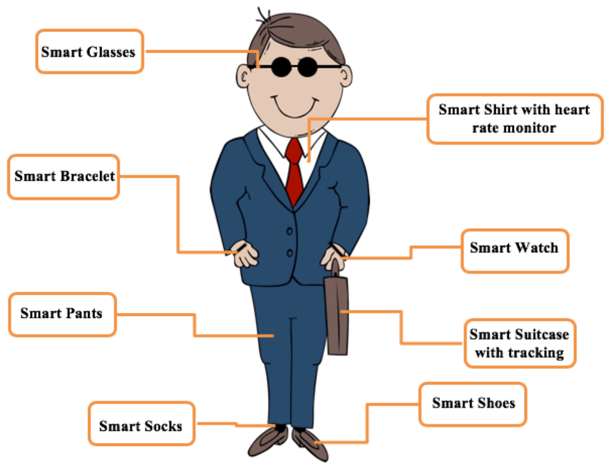


Figure 1.1: Rise of wearables and future wearable technology [16]

The core technologies of IoT are not new. Sensing technologies have been used in the past on factory manufacturing floors, for monitoring and tracking livestock, etc. The idea of machine to machine communication is also not new as that is the core concept of the Internet. IoT is an evolution in the use of these technologies in terms of number of devices, the type of devices and the services provided via these devices [3]. IoT promises, it will lead to the development of a new generation of devices that will provide personalized services; services that are tailored and modified to each user’s needs and demands.

For instance, let’s consider smart speakers like Google Home [9] or Amazon Alexa [10] which are traditional speakers, with a processor (attached to a microphone) connected to the internet (Wired or wireless). These smart speakers' interface with the user via voice commands and provide services like playing music of the internet that is tailored to the music favored by the user. Smart devices like the Apple watch and fitness trackers will soon find their way into the healthcare market, allowing doctors to monitor the health of their patients, perform predictive data analytics and thus help save lives and significantly improve the quality of healthcare services.

The number of IoT smart objects is expected to reach 212 Billion entities by the end of the year 2020 (see Figure 1.2) [11]. This growth will constitute an unprecedented increase in internet traffic and wireless network traffic. It is predicted that IoT will impact the economy in the range of $3.9 trillion to $11.1 trillion by 2025 [12]. All these statistical observations point to a fast-paced growth of the IoT industry till 2025, which will result in ubiquitous presence of smart devices that will be monitoring their users in different capacities, sharing data with each other and with services on the internet. The downside to this fast-paced growth in the IoT industry is the fact that the vulnerability of IoT devices has grown exponentially and made their security a major research challenge.

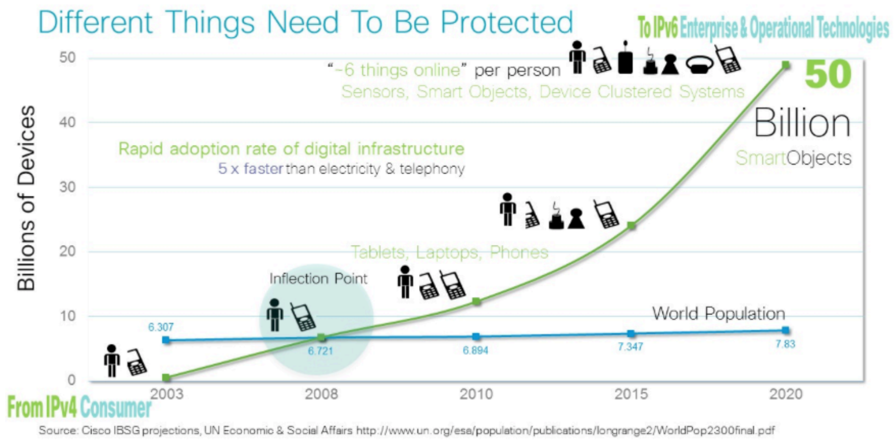


Figure 1.2: Growth of IoT by 2020

Cyber attackers can compromise IoT devices to execute attacks to hinder the operations of large computer networks, factories, nuclear powerplants, destroy global financial systems, power grids, water distribution systems etc. To make the situation worse, we are observing an alarming trend of increasing attack sophistication while reduction in the knowledge required to launch sophisticated attacks as shown in Figure 1.3.

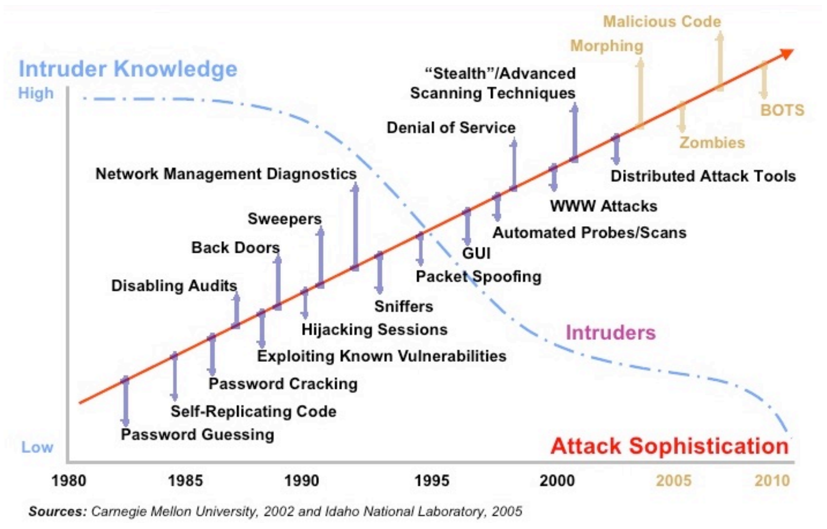


Figure 1.3: Attack sophistication versus intruder technical knowledge

Furthermore, the attack propagation time used to range in weeks in the 1980s now ranges to a fraction of a second to reach a sizable number of Internet enabled devices as shown in Figure 1.4. For instance, in May 2017 WannaCry attack was able to attack over 200,000 computers in 150 countries in 4 days where it was propagating at the speed of 3600 computers per hour [13]. This attack would have had impacted more systems at even a faster rate if aimed at the deployed IoT devices and sensors.

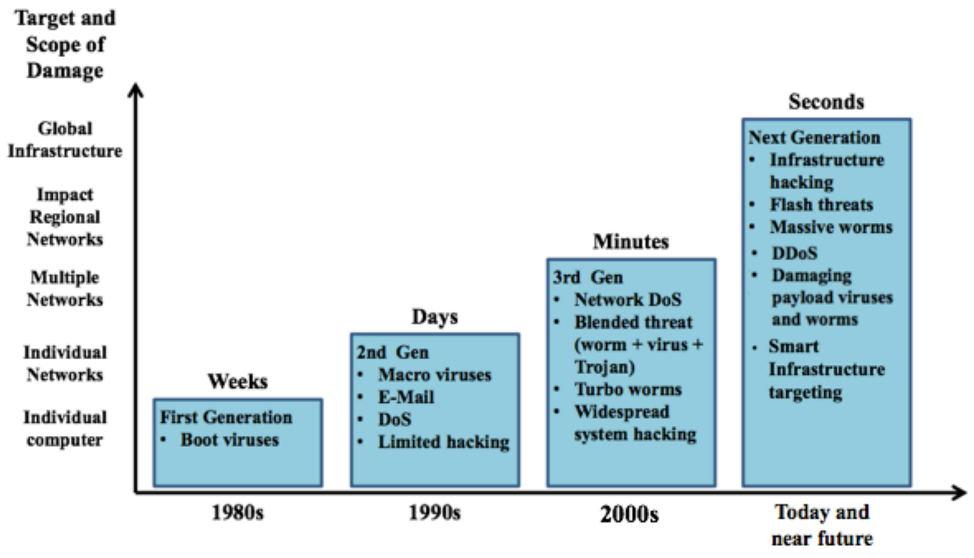


Figure 1.4- The timeline of cyber-threads scope of damage and impact time [12]

To address the security challenges associated with IoT infrastructure and services, we need a paradigm shift in how we monitor, analyze and protect their operations and services.

1.2 Problem Statement

With the growth of internet, cloud computing and IoT, smart devices have proliferated into our daily lives. These smart devices support a wide variety services and applications, which has resulted in a plethora of networking protocols.

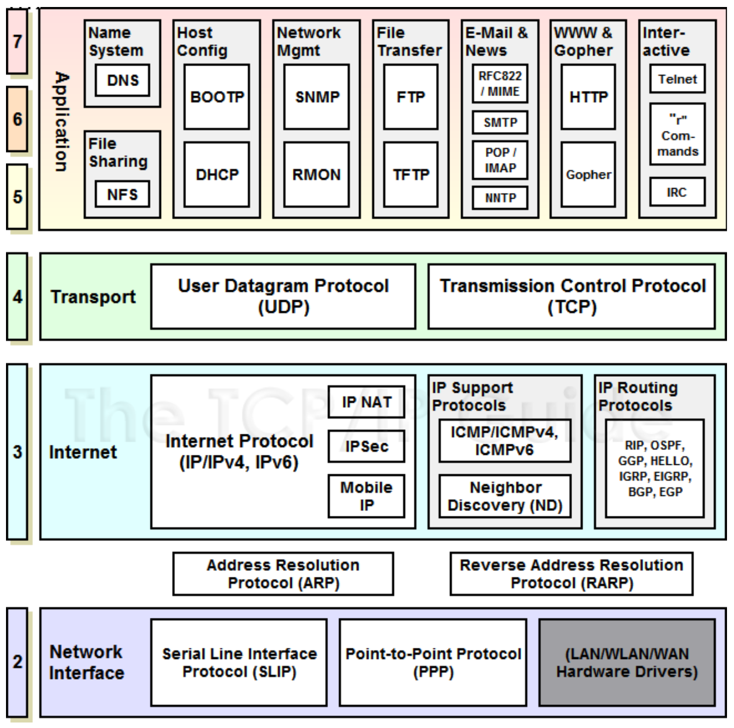


Figure 1.5- Network Layers and Protocols

Figure 1.5 highlights a subset of protocols deployed on the internet. Each of these protocols is vulnerable to attacks and needs to be secured. It is difficult to design one IDS that is able to detect attacks on all the protocols. Yousef et al. [14], presented an approach to design a multi-level intrusion detection system (ML-IDS) as shown in figure 1.6 to secure the computer networks. Yousef et al. proposed building multiple intrusion detection systems called micro intrusion detection systems to secure individual protocols and performing decision fusion to detect threats on the whole network stack.

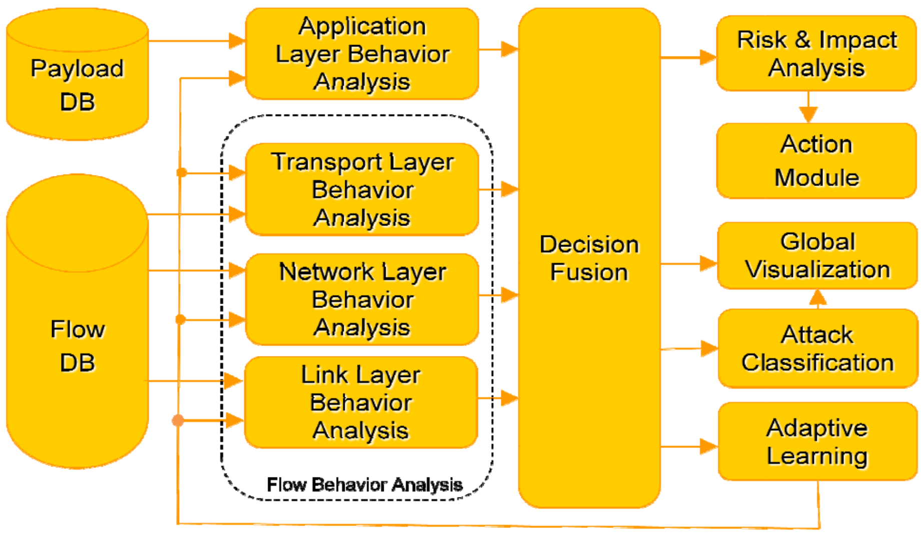


Figure 1.6- Multi-Level Intrusion Detection System (ML-IDS)

This research extends that work and presents a methodology that aids in developing micro intrusion detection systems to secure each protocol, and the results from each of these micro AB-IDS' are aggregated by the ML-IDS. The proposed methodology has 3 steps: 1. Threat modelling analysis; 2. Feature selection to characterize the behavior of the protocol; 3. Use the selected features to develop machine learning models that characterize accurately the normal behavior. Considering the projected growth of the internet and the growth of IoT systems, it is critically important to formalize this methodology to design AB-IDS' that can secure each protocol.

1.3 Research Objectives

Our main goal for this research is to design a general methodology that can streamline the development process of micro IDS for different protocols. The specific goals of this research are highlighted as follows:

- Develop a methodology to design specialized AB-IDS' that can detect attacks on their protocols. This will help the user achieve proactive intrusion detection capabilities that can detect malicious activities that trigger anomalous behavior including zero-day attacks.
- Develop innovative data structures like ObservationFlow (OF) for each protocol type, that will be used to accurately and efficiently characterize the normal behaviors of each protocol. These footprint data structures will lead to significant reduction of the amount of data that need to be monitored and stored in real-time. These footprints will model normal and abnormal operations of different networking protocols.
- Evaluate the performance of the developed micro AB-IDS on the Wi-Fi, DNS and the HTML protocols. In the evaluation, different performance statistics like accuracy, false positives and false negatives will be used.

1.4 Dissertation Organization

The remaining chapters of the dissertation are as follows:

Chapter 2 describes the background and the related work on intrusion detection systems and data mining techniques. We discuss the work done by Axelsson et al. in [15], and follow their taxonomy for Intrusion Detection Systems, discussing signature-based intrusion detection systems, anomaly-based intrusion detection systems, and compound intrusion detection system. We then discuss different characteristics of intrusion detection systems like the time of detection of the intrusion detection system, granularity of intrusion detection system, etc. We then give a brief introduction to machine learning, classification learning, associative learning and clustering, and highlight some machine learning algorithms like Isolation forest, and C4.5 that were used in this research.

In Chapter 3, we present our AB-IDS development methodology. We begin by presenting our architecture for IoT devices. We then present our threat modelling methodology designed to model IoT devices. We apply this methodology to detect threats and design intrusion detection systems for IoT/networking protocols. We discuss data structures like the observation flow, and n-grams that we use to model the normal behavior of the protocol.

In Chapter 4, we apply our methodology to design an IDS to secure IEEE 802.11 (Wi-Fi protocol). We describe the Wi-Fi protocol and its operational state diagram. We then present related work of Intrusion detection systems for the Wi-Fi protocol. We then apply our intrusion detection system to Wi-Fi protocol by performing threat modelling on the Wi-Fi protocol, using the threat modelling to identify vulnerabilities. We then model the normal behavior of the Wi-Fi protocol using n-grams and observation flows and train machine learning models that we use to accurately differentiate normal behavior of the Wi-

Fi protocol from the abnormal behavior. We then present the results of evaluating of our approach on different datasets.

In Chapter 5, we apply our methodology to design an IDS to secure the DNS protocol. We introduce the DNS protocol and describe its normal operations. We discuss the related work addressing the security of the DNS protocol. Then we apply our IDS design methodology to develop an AB-IDS for the DNS protocol. We use n-grams and observation flows to characterize the normal behavior of the DNS protocol. The n-grams and extracted characteristics feature set from the traffic is used to train machine learning models to differentiate the normal behavior of the DNS protocol from the abnormal. We then present the experimental results of our approach.

In Chapter 6, we present an AB-IDS developed to detect attacks on the HTML protocol. We introduce the HTML protocol and describe its operation. We discuss related work that focuses on HTML protocol security. We introduce a static analysis-based approach to detect malicious HTML files. Following the introduction of the static analysis-based approach, we introduce a dynamic analysis-based approach to detect malicious HTML files, wherein suspicious HTML files are opened in a Sandbox and the runtime flow of the HTML file is analyzed to conclude on the normality of the file. We then present the experimental results for our approach.

In Chapter 7, we present the conclusion and future work.

CHAPTER 2: BACKGROUND AND RELATED WORK

2.1 INTRODUCTION

In this dissertation we present a methodology to develop AB-based Intrusion Detection Systems (IDS) for different Networking protocols. In this chapter we present the background for this research approach. We begin by discussing the IDS taxonomy presented by Axelsson [15]. We discuss the types of intrusion detection systems, characteristics of intrusion detection system and provide a brief overview of the machine learning algorithms.

2.2 OVERVIEW OF THE INTRUSION DETECTION SYSTEM TAXONOMIES:

Axelsson, el at. [15] presented a complete taxonomy of intrusion detection systems. According to Axelsson, Intrusion Detection Systems are like 'Burglar Alarms', which are built with the aim of protecting a system against attacks by sounding off warnings on detection of an attack.

According to Axelsson intrusion detection systems are of three major types: Anomaly based intrusion detection systems, signature-based intrusion detection systems, and compound intrusion detection systems.

2.2.1 ANOMALY BASED INTRUSION DETECTION SYSTEMS:

According to Axelsson and [17][18][19][20], an anomaly-based intrusion detection system is designed to model the normal behavior of the system and makes an inherent assumption that any attack will lead to an anomalous behavior. Designing and implementation of anomaly-based intrusion detection systems begins with collection of information on what constitutes normal behavior for the system that is called the "target system", or the "target". Anomaly based intrusion detection systems use this understanding of the normal behavior of the target to build models that allow classification of harmful anomalies from the normal behavior.

Anomaly based intrusion detection systems can be classified into two types based on the method used to build models that identify the normal behavior of the system: Self learning and Programmed (supervised).

2.2.1.1 Self Learning

Self-learning systems as the name suggests, perform the task of learning the normal behavior on their own. They observe the target at runtime and have the capabilities to judge and extract features that are characteristics of the target's normal behavior. They then build models that incorporate these characteristics. The systems that fall in this category may use different approaches to model the normal behavior of the system. Some of the approaches involve the use of stochastic modeling, which may involve formulation of rules that are able to mark the conditions of normality of the system or use of distance vectors to measure the difference of certain measured features at runtime and classify certain events as abnormal. Artificial neural networks and clustering algorithms have been used to build such systems.

2.2.1.2 Programmed/Supervised learning

In case of a programmed intrusion detection system [46] [47] [48] [49] [50], a third party other than the original intrusion detection system itself, teaches the system by feeding it with information to detect abnormal events. This is generally done by feeding the system with different parameters that have statistical values that help deciding if the system is operating normally or not.

2.2.2 SIGNATURE BASED INTRUSION DETECTION SYSTEMS

The signature-based intrusion detection system [21][22][23][24][25][26][27][28][29] operate on the knowledge obtained from analyzing the behavior of known intrusions on the target. In a signature-based intrusion detection system, the IDS check's for signs of signatures of previously known attacks or intrusions, by comparing current behavior to a set of known signatures stored in a database. Whenever there is a match, the intrusion detection system gives an alert. In a signature-based intrusion detection system, the system does not model the normal behavior of the target. The Intrusion detection systems of this type are programmed intrusion detection systems, where in the intrusions are programmed as either state-based models, or audit event (string matching models).

2.2.3 Compound Intrusion Detection Systems

Compound intrusion detection systems are a composite of a signature-based intrusion detection systems and anomaly-based intrusion detection systems. These systems generally use signature-based detection on normal traffic.

2.3 System Characteristics of Intrusion detection Systems

System characteristics of intrusion detection systems are independent of the type of detection the intrusion detection system performs.

2.3.1 Time of Detection

Intrusion Detection Systems can be either Real Time (Near Real Time) Intrusion Detection Systems [43][44][45] or Non-Real Time Intrusion Detection Systems. The Real Time detection Systems are systems that check for Intrusions at runtime and respond to attacks in a timely manner. However, these systems must run their algorithms with low overhead to be deployed. Non-Real Time Intrusion Detection Systems analyze network traffic offline and can run very sophisticated models to improve detection and accuracy.

2.3.2 Granularity of Detection

Granularity of Detection is the smallest unit of Data that is processed by the Intrusion Detection System. The Intrusion Detection System can process data continuously or in small groups or batches.

2.3.3 Source of Audit Data

Source of Audit Data is the data input source for the Intrusion Detection system. Source of Audit Data is either network packets tapped directly from the network interface or system logs like Kernel logs that are maintained by the operating system.

2.3.4 Response to Detected Intrusions

Based on the Response to detected intrusions, Intrusion Detection Systems can be of two types: Passive and Active systems, which are discussed below.

2.3.4.1 Passive Response Systems

Passive response systems are Intrusion Detection Systems that respond to detection of an intrusion on the system by sending an alarm. They warn the user of the attack, but they do not take any preventive actions or countermeasures against the detected attacks.

2.3.4.2 Active Response Systems

Active response systems are intrusion detection systems that respond to detection of an intrusion on the network by sounding an alarm and then taking counter measures against the detected attack. The counter measures range from closing of the network connections, to even attacking the resources used by the attacker.

2.3.5 Locus of Data Processing

The data processed in the intrusion detection system can be either performed at a central location or at distributed locations.

2.3.6 Security

This is to measure the security of the intrusion detection system itself from attacks, wherein this measure determines how secure an intrusion detection system is from attacks.

2.3.7 Degree of Inter-Operability:

This is a measure of the intrusion detection systems ability to operate with other intrusion detection systems.

2.4 Machine Learning and Data mining

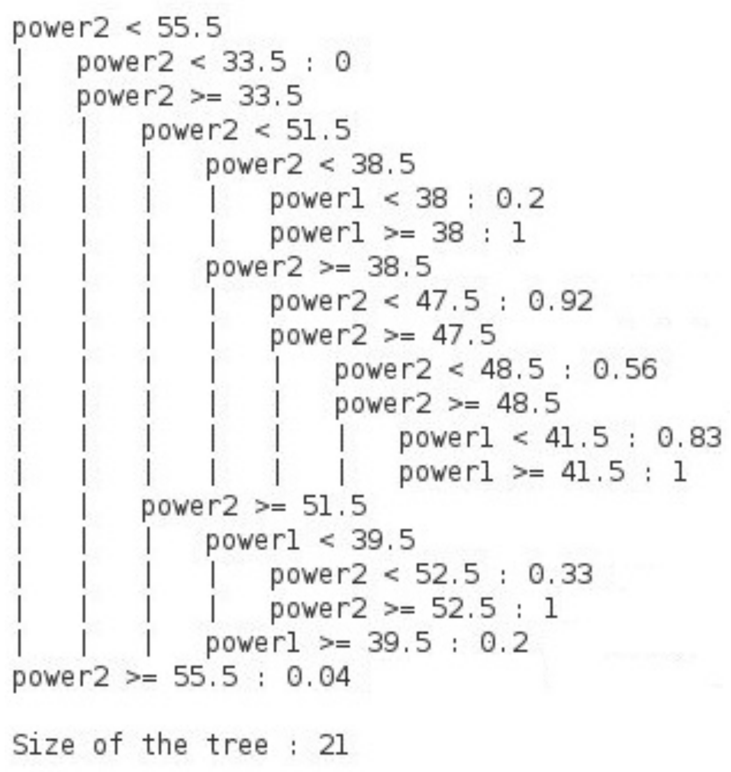
The amount of data that is collected in databases today is growing exponentially. This increase in the amount of collected data can be attributed to many reasons, such as the increasing computing power and number of connected devices, the increase in the channel capacity of computer networks, faster memory devices, to just mention a few. It has been observed that as the size of data increases, the ability of a human to make sense out of the data decreases rapidly. This has brought about the need for means to process the data and obtain knowledge from mining and analyzing the collected data.

Machine learning is the study of algorithms and statistical models to obtain knowledge on how networks and protocols operate. Machine learning is generally of two types: Supervised machine learning where the data is marked into classes for training by a human, Unsupervised machine learning where the knowledge is gained without human input. Data Mining is the process of finding patterns in data sets by the means of use of various data mining algorithms. The data mining process can be automatic or semi-automatic involving human interference. These algorithms are used to learn patterns from the data and thus learn conditions or patterns of behavior that help in prediction of a behavior of another data set having similar patterns.

The event space in data mining algorithms [32][33][34][35] represents the space or the set that holds all the events or the data points that are to be analyzed. Data mining could be viewed as a search to look for conditional statements that are able to group all the elements of the set with correct descriptions.

Machine learning algorithms are given inputs in the form of files that holds the data to be analyzed. The data rows describe specific instances, while the data columns describe different attributes. The attributes that are selected to represent the data set to be used by the data mining algorithm are typically defined by a process called Attribute Selection [36]. On completion of training on the data in the databases using different machine learning algorithms, the performance of the trained models are analyzed using methods like k-fold cross validation. There are a number of means by which the trained on a particular data set can be represented. The methods in which these results can be represented are discussed below.

Decision tables are the easiest means to represent the output of the learning algorithms, where in it is a table with conditions in the first column and the result of the machine learning algorithm in the next. Decision Trees are tree like structures that represent the results of the machine learning algorithms. The nodes represent the conditions to be taken while the leaves represent the class of the result. Figure 2.1 below shows an example of a Decision Tree.



```
power2 < 55.5
|   power2 < 33.5 : 0
|   power2 >= 33.5
|   |   power2 < 51.5
|   |   |   power2 < 38.5
|   |   |   |   power1 < 38 : 0.2
|   |   |   |   power1 >= 38 : 1
|   |   |   power2 >= 38.5
|   |   |   |   power2 < 47.5 : 0.92
|   |   |   |   power2 >= 47.5
|   |   |   |   |   power2 < 48.5 : 0.56
|   |   |   |   |   power2 >= 48.5
|   |   |   |   |   |   power1 < 41.5 : 0.83
|   |   |   |   |   |   power1 >= 41.5 : 1
|   |   power2 >= 51.5
|   |   |   power1 < 39.5
|   |   |   |   power2 < 52.5 : 0.33
|   |   |   |   power2 >= 52.5 : 1
|   |   |   power1 >= 39.5 : 0.2
power2 >= 55.5 : 0.04

Size of the tree : 21
```

Figure 2.1: Decision Tree

Classification Rules are the simplest means to represent the results of machine learning algorithms. They provide the conditions in form of simple condition statements that can be used as test to classify the results. The conditional statements can also be ANDED together with other conditional statements to obtain longer rules. Example rules are shown in the Table 2.1 below

Table 2.1: Classification Rules

RULES 1:	IF **FS1<97** AND **FS2> 279732** AND **FS3>1020** THEN TRAFFIC IS NORMAL;
RULES 2:	IF **FS<1881** AND **FS2> 591472** AND **FS4>505** THEN TRAFFIC IS NORMAL

Clusters are obtained as a result of machine learning when a clustering algorithm is used instead of a classification algorithm. The Clusters that are obtained help representing the distance of the instance from the center of the cluster. Clusters can be mapped in two-dimensional space which is the simplest form to represent the cluster. The clusters could also be represented in multi-dimensional space. Figure 2.2 below shows an example of a cluster representation.

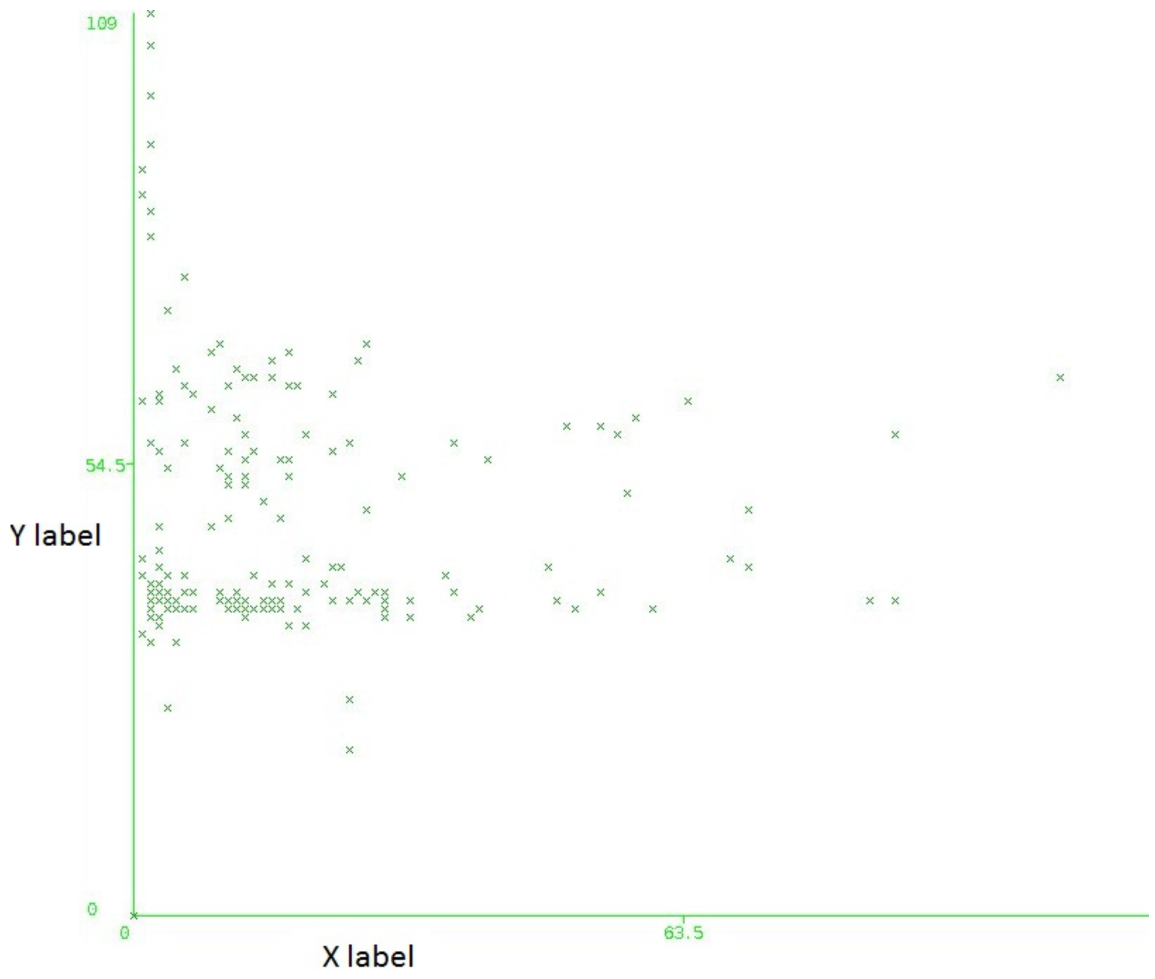


Figure 2.2: Clustering Example

- **Learning Types:**

According to Witten and Frank el in [7] three different types machine learning approaches. The choice of the approach used depends on the type of data that is being processed and the application of the resulting conditions or rules from machine learning. These machine learning approaches are discussed below.

i. Classification Learning:

Classification learning involves using algorithms to understand the conditions which allow classification of unseen examples into predefined classes. Thus, classification learning allows default classification of the unseen data into different classes.

ii. Association Learning:

Association Learning allows the building of associations between different unseen examples by predicting attributes as well as classes. Association learning helps in learning strong rules of association between attributes and not just their classes. Association learning involves use of different algorithms like Apriori Algorithm [37], Eclat Algorithm [38], FP-growth Algorithm [39].

iii. Clustering

Clustering is the process of grouping together objects that are similar to each other compared to other objects. Clustering is generally distance based, where in the clusters are judged depending on the center of the clusters. Clustering is generally done using Centroid based clustering, Distribution- based clustering or Density-based clustering. Some of the clustering methods include k-means clustering [40][41].

- **Machine learning algorithms**

In this section we list and describe the machine learning algorithms we used in this research.

i. Isolation forest

Isolation forest [30] isolates observations by randomly selecting a feature to split the data with a random value between the maximum and the minimum values of that feature. As anomalies are smaller in number and have values different from the normal entries, they are easier to isolate. The results of the isolation forest are represented in term of iTrees, where the anomalies are isolated closer to the root of the tree.

ii. C4.5

C4.5 [31][32], is a tree-based decision tree classifier that splits its set of classified samples into subsets to enrich one class for each of the attributes, where the attribute with the highest normalize information gain is chosen.

iii. Random Forest

Random forest [33] is an ensemble learning algorithm that constructs multiple decision trees during the training time and outputs the mean prediction of the different classes at the time of classification.

iv. AdaBoost

AdaBoost [34] is a collection of weak classifiers combined by weighted sum wherein the results of classifiers are tweaked to in favor of instances wrongly classified by the previous classifier.

v. Ripper

Ripper [35] is based in incremental reduced error pruning, where rules are grown one at a time such that the training set is split into 2; 2/3 being for growing the rules and 1/3 for pruning the rules. Rules are built for smaller classes first.

vi. Hoeffding tree

Hoeffding tree [37] is an incremental decision tree that trains on massive data streams, provided that the characteristics of the data streams do not change over time.

vii. Random tree

Random tree constructs a tree using K randomly chosen attributes at each node allowing estimation of class probabilities based on the hold out set.

viii. Support Vector Machine (SVM)

Support vector machine [38][39] constructs a hyperplane that separates the data into classes and thus can be used for classification.

ix. Bagging algorithm

Bagging algorithm [40] is an ensemble learning algorithm, that generates new sub training sets from the given training sets and builds models on these subsets, combining the results by averaging or voting.

x. Logistic Regression

Logistic regression [41] uses a logistic function to model a binary dependent variable, such that it can classify the labeled data into correct classes.

CHAPTER 3: DESIGN METHODOLOGY FOR INTRUSION DETECTION SYSTEMS

This chapter presents a methodology for designing the AB-IDS'. In this chapter we introduce the IoT Architecture and use it as an example to describe our methodology. We present IoT threat modelling methodology, anomaly behavior analysis, intrusion detection system design, and the data structures used to model the normal behavior of the analyzed protocols.

3.1 IOT ARCHITECTURE

IoT and cyber physical systems have become an integral part of modern computer networks. Thus, securing computer networks and networking protocols has to start with understanding the interactions between IoT devices/cyber physical devices and computer networks. In this section we present our IoT architecture to model the behavior of any cyber physical system/IoT device. We then introduce IoT threat modeling methodology to perform systematic threat modelling analysis of the cyber physical systems followed by some examples of the threat modelling analysis. The same threat modelling analysis is then applied to secure IoT protocols and networking protocols.

IoT architecture [51] helps model the IoT/ cyber physical devices and helps understand the behavior of the IoT device and identify their vulnerabilities. As shown in Figure 3.1. the IoT modelling architecture consists of four layers: End devices, Communications, Services, and Applications. Cyberattacks can be launched against the functions and services at each layer.

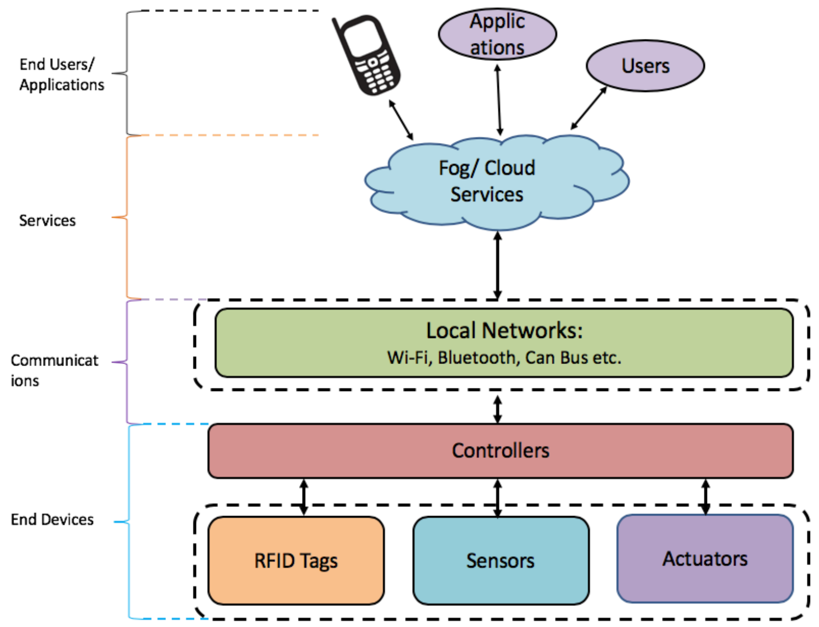


Figure 3.1- IoT architecture

The first layer (bottom layer) is composed of end devices and controllers that control these end devices. Devices like RFID tags, sensors like temperature sensors and actuators interact with the IoT users, collect data and pass it to the processing services or present the results of the data processed [52][53]. The second layer (communications layer) is responsible for providing the required connectivity and communications among all the sensors and actuators associated with IoT services or applications [53][54]. The communication technologies that can be used include Internet, satellite, mobile cellular networks, wireless sensor networks, and internal car network infrastructures (e.g. CAN, Bluetooth, I2C, MOST). Various functionalities of the IoT devices are provided through the services layer i.e. the third layer. The services layer provides common middleware and functions to build sophisticated IoT services in the application layer. The applications layer provides end user applications which are used by the users to access the IoT devices.

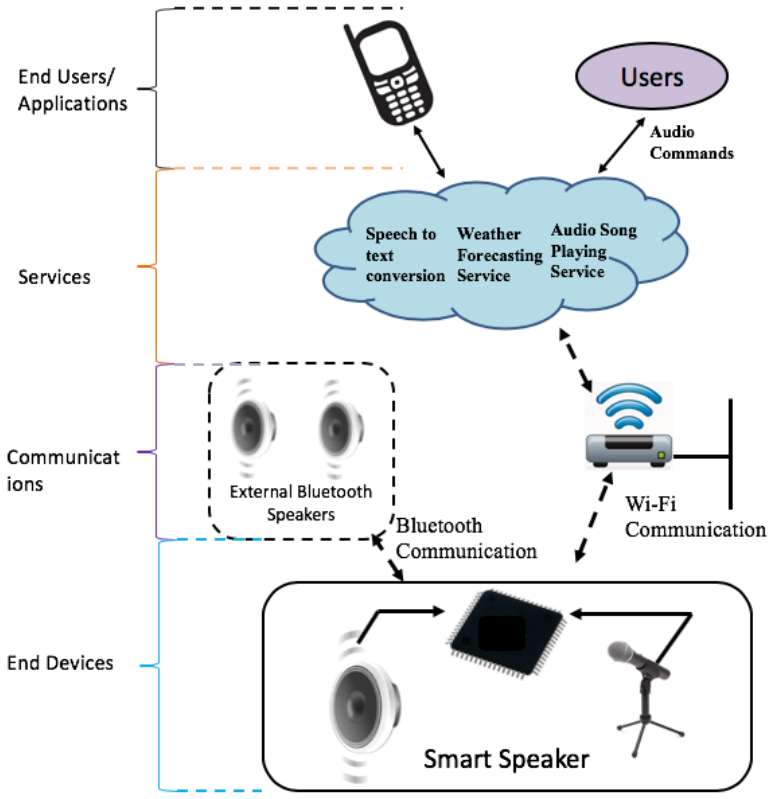


Figure 3.2- Application of IoT architecture to a Smart speaker

In Figure 3.2, we apply our IoT architecture to a smart speaker system. The speakers, onboard controllers and microphones constitute the End devices layer of the IoT methodology. These speakers use Wi-Fi, Bluetooth and Ethernet to communicate with other devices and webservices; constituting the communications layer. These speakers use cloud-based speech to text conversion services (STT) [55][56][57], weather forecasting services, song playing platforms like YouTube or Pandora or Spotify etc. which constitute the services layer of the methodology. All commercially available smart speakers have applications that allows the users to control these devices through their smart phones, and other smart devices which constitute the services layer.

3.2 IoT Threat Modeling Methodology

Figure 3.3 shows our IoT threat modelling methodology, which helps analyze the vulnerabilities in cyber physical systems like IoT devices and help develop methodologies to remove or mitigate the impact of these vulnerabilities if they were exploited by attackers in the cyber physical systems. As shown in Figure 3.3, the IoT threat modelling consists of four layers and each layer has the following 5 tasks: Modelling the target layer's function, investigating the attack surfaces for the model, investigating the targets of the attack surface, investigating the impact of the attack if executed, investigation of the attack mitigation strategies. For each layer, we first develop a model that captures the functions to be provided by that layer. The layer model will then be analyzed to identify the Attack Surface (AS) that characterizes the entry points that can be exploited by attackers to inject malicious events or behaviors into that layer functions. For each vulnerability or AS entry point, we then identify the potential target(s) that can be exploited by attackers, and the risk and impact if the attack was successful. The final step in each row of the IoT threat modelling methodology is to develop methods to mitigate or eliminate these vulnerabilities in order to achieve the required secure operations of the functions provided by each layer.

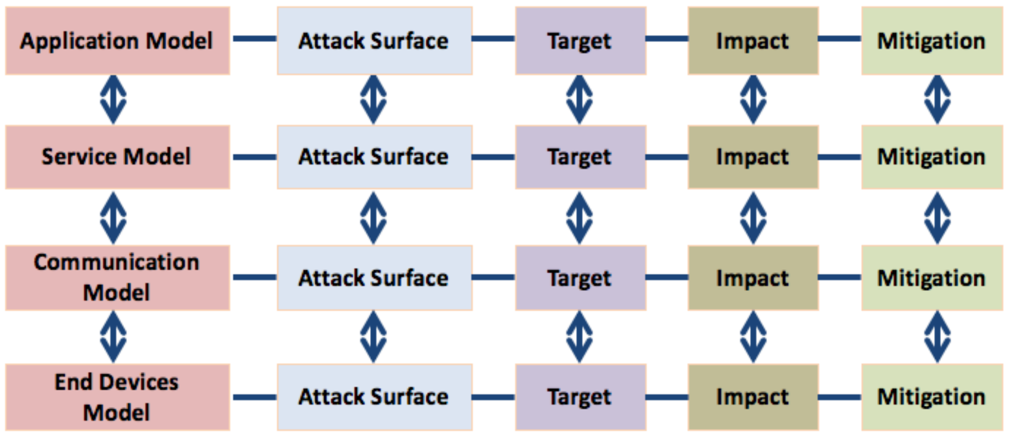


Figure 3.3. IoT threat modelling methodology

3.2.1 Attack Surface (AS) Identification, Impact Analysis and Mitigation Strategies

After modeling the operations of a given layer, we identify the attack surfaces for each layer model, perform risk and impact analysis, then develop mitigation methods to protect against each detected vulnerability. This analysis helps identify the imminent threats to the IoT system, and helps in securing the system in a systematic fashion. In the subsequent subsections, we perform AS identification, impact analysis and mitigation methods for the IoT system shown in Figure 3.2.

- **End-Devices Layer**

Table 3.1 shows the attack surface, impact, mitigation, and mitigation mechanisms associated with the end-devices layer.

Table 3.1: End Nodes Layer

Attack Surface	Target	Impact	Mitigation mechanism
Controllers	Control, information	Lost control, human life safety, failures	Anomaly behavior analysis (ABA) to detect abnormal control information, encryption
Sensors	Information, access to the system	Lost control, human life safety, failures	Lightweight encryption, ABA, secure sensor identification and authentication
Actuators	Control	Lost control, human life safety, failures	Lightweight encryption, ABA, anti-jamming
Entertainment	Access to the system	Long time, waste energy, cyber crime	Encryption, moving target defense, ABA

- **Communication Layer**

Table 3.2 shows the attack surface, target, impact and mitigation mechanisms associated with the communication layer.

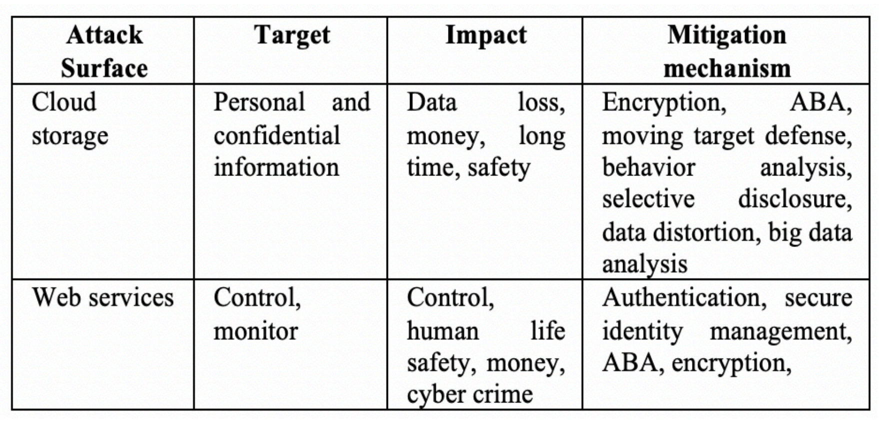
Table 3.2: Communications Plane

Attack surface	Target	Impact	Mitigation mechanism
Protocols	Access control, information,	Lost control, human life safety, failures, money, longer time	Authentication, ABA, moving target defense, anti-jamming
Firewalls	Access control to the system	Lost control, human life safety, long time, energy waste	IDS, behavior analysis, authentication
Routers	Access, information	Control, human life safety, time	ABA, anti-jamming, encryption
Communication Bus	Information, control,	Privacy, money, human life safety, long time, failures	Encryption, IDS, moving target defense, ABA

- **Services Layer**

Table 3.3 shows the attack surface, target, impact and mitigation mechanisms associated with the services layer.

Table 3.3: Services Layer

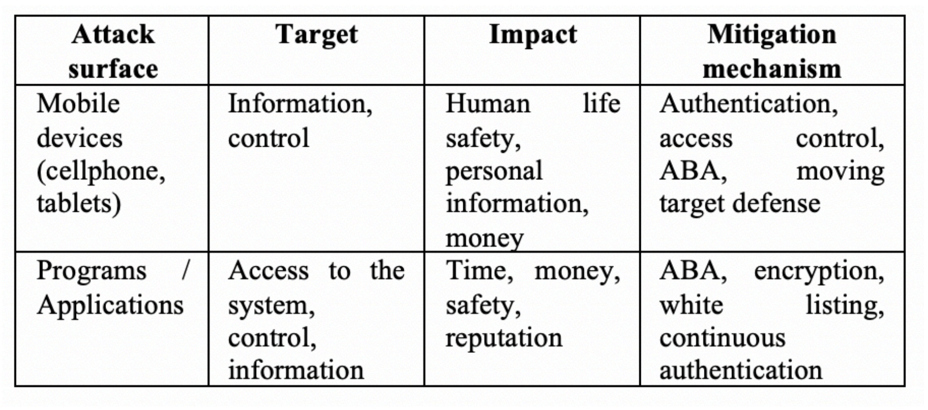


Attack Surface	Target	Impact	Mitigation mechanism
Cloud storage	Personal and confidential information	Data loss, money, long time, safety	Encryption, ABA, moving target defense, behavior analysis, selective disclosure, data distortion, big data analysis
Web services	Control, monitor	Control, human life safety, money, cyber crime	Authentication, secure identity management, ABA, encryption,

- **End Users/ Applications Layer**

Table 3.4 shows the attack surface, target, impact and mitigation mechanisms associated with the End Users/Applications layer.

Table 3.4: Applications Layer



Attack surface	Target	Impact	Mitigation mechanism
Mobile devices (cellphone, tablets)	Information, control	Human life safety, personal information, money	Authentication, access control, ABA, moving target defense
Programs / Applications	Access to the system, control, information	Time, money, safety, reputation	ABA, encryption, white listing, continuous authentication

3.3 Anomaly Behavior Analysis (AB-Analysis)

The AB-Analysis approach aims at the analyzing the normal behavior of the protocol. The normal operations of a system can be represented by an n-dimensional data structure as shown in Figure 3.4. As long as the system is performing normally, its operation point is confined to the normal operation regions. When an anomalous event occurs, the operation point of the system moves outside the normal region and consequently, it can be detected by the AB module. Countermeasures are then taken to bring the operational point back inside the cube. We have applied the concept of AB analysis to detect intrusions on computer networks or attacks on different protocols [46][47][49][50].

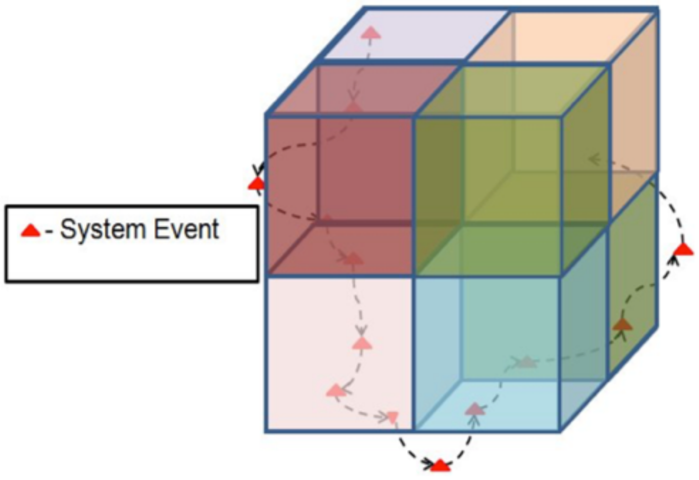


Figure 3.4: Anomaly Behavior Analysis

3.4 Intrusion Detection System Design Methodology

The designing of an AB-IDS using the AB-analysis approach presented in Section 3.3 for any protocol involves 3 steps;

1. Threat modelling analysis of the protocol
2. Feature selection and protocol foot printing to characterize the behavior of the protocol;
3. Use the selected set of features to develop machine learning models that characterize the normal behavior.

In the first step, we apply the threat modelling methodology discussed in Section 3.2 to the protocol. We begin with obtaining a visual model (like state machines, architectures etc.) that describes the normal behavior for the protocol. The protocol state transition machine serves as the ideal candidate for the protocol's visual model. Then attack surfaces are identified and impact analysis is performed on these attack surfaces. Mitigation strategies are devised to mitigate the threats from these attack vectors.

In the second step, the attack surfaces are used to select features that characterize the behavior of the protocol. A subset of the features is passed through data structures that allow better data analytics and modelling of normal behavior of the protocol.

In the third step, the protocol footprint data structure is used to create machine learning models that can separate normal behavior of the protocol from the abnormal during runtime analysis. Figure 3.5 shows the AB-IDS design methodology.

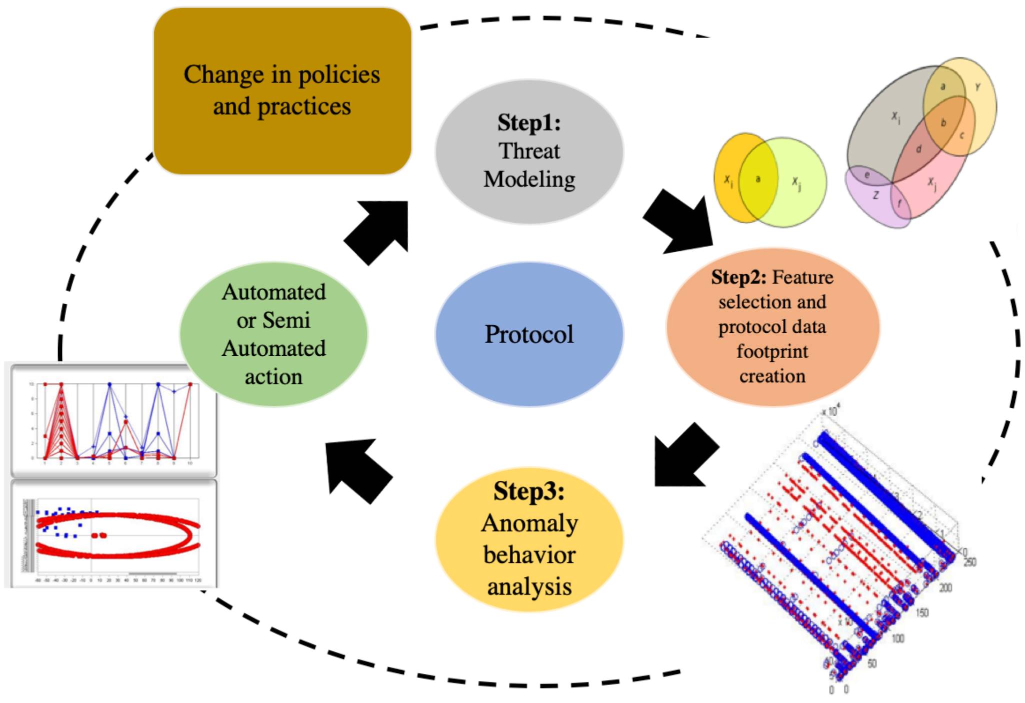

Change in policies and practices
Step1: Threat Modeling
Step2: Feature selection and protocol data footprint creation
Protocol
Automated or Semi Automated action
Step3: Anomaly behavior analysis

Figure 3.5: AB-IDS design methodology

3.4 Protocol Fingerprinting Data structures

A footprint data structure is needed to capture the state machine transitions made by the protocol. We use N-grams and Observation-flow data structures to effectively fingerprint the normal behavior of different protocols [46][47][49][50].

- **Observation-flow:**

It is a continuous flow of frames or packets between a source-destination pair sampled at specific intervals of time ‘t’. The observation-flow characterizes the state transitions made by the protocol. Figure 3.6. shows a simplified example of an Observation-flow for the Wi-

Fi protocol. The flow begins with receiving Authentication frames (Auth), followed by Association request frames (Asso Req), followed by Data frames (Data) and ends with De-Authentication frames (De-Auth).

- **N-gram:**

An N-gram is a sliding window of a pre-defined size, sampled from the observation-flow. The n-grams are used to model the temporal behavior of the target protocol, in this case the Wi-Fi protocol as shown in the Figure 3.6. {Auth, Auth, Asso Req, Data} forms an n-gram of size 4. An n-gram can be of any size and n-gram size analysis experiment has to be performed to determine the size of the n-gram to be adopted for use in an IDS for a protocol.

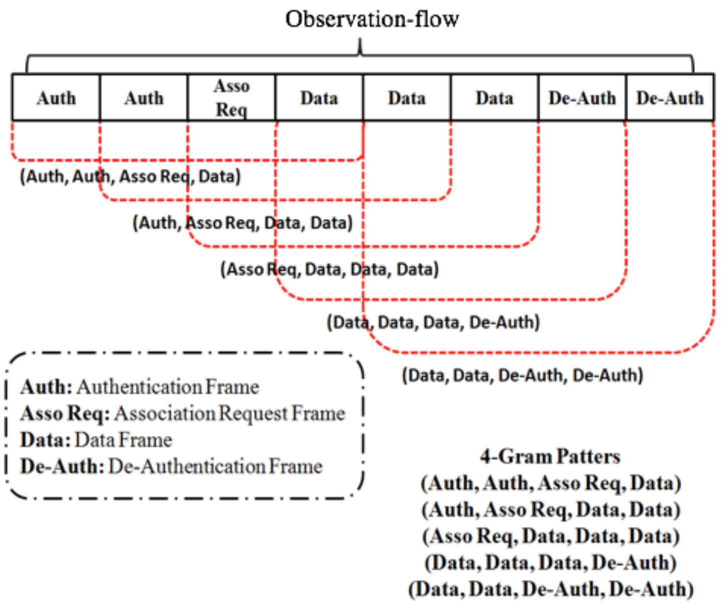

Observation-flow
Auth
Auth
Asso Req
Data
Data
Data
De-Auth
De-Auth
(Auth, Auth, Asso Req, Data)
(Auth, Asso Req, Data, Data)
(Asso Req, Data, Data, Data)
(Data, Data, Data, De-Auth)
(Data, Data, De-Auth, De-Auth)
Auth: Authentication Frame
Asso Req: Association Request Frame
Data: Data Frame
De-Auth: De-Authentication Frame
4-Gram Patters
(Auth, Auth, Asso Req, Data)
(Auth, Asso Req, Data, Data)
(Asso Req, Data, Data, Data)
(Data, Data, Data, De-Auth)
(Data, Data, De-Auth, De-Auth)

Figure 3.6: Observation-flow and n-grams

CHAPTER 4: DESIGNING AB-IDS FOR THE WI-FI PROTOCOL

Wi-Fi protocol also known as the IEEE802.11[58] [59][60][61][62][63] is a wireless local area network protocol. It is the Ethernet's equivalent for wireless networks. The protocol was first formalized in the year 1997. This protocol over the years of its existence has been upgraded and has incurred many changes. But most of these upgrades have been to enhance the data rate and the link quality of the network and little has been done to improve the security of the network.

4.1 THE IEEE 802.11 STANDARD

The Wi-Fi protocol [58] operates in the 2.4 GHz UHF and the 5 GHz SHF bands both of which fall under the category of ISM bands and hence have been sanctioned for unlicensed use. The original Wi-Fi protocol that was declared in the year 1997 specified bit rates of 1 or 2 Mbits/s while specifying 3 alternate physical layer configurations which were Diffuse infrared at 1Mbps, Frequency Hopping Spread Spectrum (FHSS) at 1 Mbps or 2 Mbps, Direct Sequence Spread Spectrum (DSSS) at 1 Mbps or 2 Mbps.

4.1.1 IEEE 802.11B

This specification of the protocol [60] supports a maximum data rate of 11 Mbps using the same physical specifications as the original Wi-Fi protocol. This protocol supported the use of the DSSS in the 2.4 GHz UHF ISM band. Carrier Sense Multiple Access/ Collision Avoidance (CSMA/CA) is the protocol that is used to manage the media access.

4.1.2 IEEE 802.11A

This standard was added to the original 802.11 protocol in the year 1999 [59]. This standard specified the operations of the Wi-Fi in the 5 GHz SHF supporting 52 subcarrier Orthogonal Frequency Division Multiplexing supporting data rates up to 54Mbps.

4.1.3 IEEE 802.11G

This standard was ratified to the original 802.11 protocol in the year 2003 [61]. This standard increased the data rate in the 2.4GHz UHF ISM band to 54Mbps. This was done by the adoption of the use of Orthogonal Frequency Division Multiplexing (OFDM).

4.1.4 IEEE 802.11N

This enhancement to the 802.11 protocol enhanced the data rates up to 600Mbps [62]. This increase in data rate is achieved by the use of multiple data streams with channels widths of 40MHz. It can operate in the 2.4GHz UHF ISM band and the 5 GHz SHF band. Moreover, one of the enhancements that have been added includes the use of multiple antennas that allows seamless maintenance of simultaneous data streams.

4.1.5IEEE 802.11AC

This standard was approved in the year 2014 [63]. This protocol allows each single link to have a throughput of 500 Mbps while the overall throughput for a multi-station WLAN would be at least 1 Gbps. This increase in throughput is achieved by the use of wider channels, and up to 8 MIMO spatial streams and use of 256 QAM.

Table 4.1: IEEE 802.11 Physical Layer Standards

Release Date	**Standard**	**Frequency Band**	**Bandwidth**	**Modulation**	**Data Rate**
1997	802.11	2.4GHz	20 MHz	DSSS, FHSS	2 Mbps
1999	802.11b	2.4GHz	20 MHz	DSSS	11 Mbps
1999	802.11a	5GHz	20 MHz	OFDM	54 Mbps
2003	802.11g	2.4GHz	20MHz	DSSS, OFDM	542 Mbps
2009	802.11n	2.4GHz, 5Ghz	20MHz, 40MHz	OFDM	600Mbps
2013	802.11ac	5Ghz	40MHz, 80MHz,160MHz	OFDM	6.93Gbps

4.2 Data Link Layer and Frame Structure

The 802.11 protocol breaks the continuous stream of data into smaller data units and sends it over the network encoded as payload data in the frames. The protocol also defines other types of frames that are responsible for the maintenance and management of the link. The general structure of the frame is as shown in the Figure 4.1.

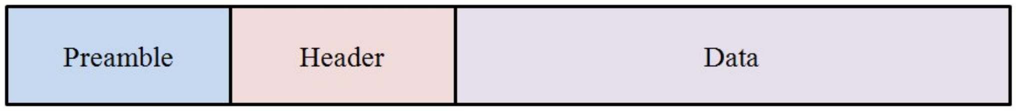


Figure 4.1: Wi-Fi Frame structure

- **Preamble**

The preamble is the first set of bit sequence that follow the actual header. The preamble marks the start of the frame. The preamble allows the receiver frequency synchronization and receiver time synchronization. This helps the receiver extract the header and the data from the modulated signals that are sent over the channel.

- **Header**

The Wi-Fi Header is made up of a 2 Byte frame control field, a 2 Byte Duration ID, 48 bit long address field that includes the source and the destination address. The structure of the Wi-Fi frame header is as shown in the Figure 4.2.

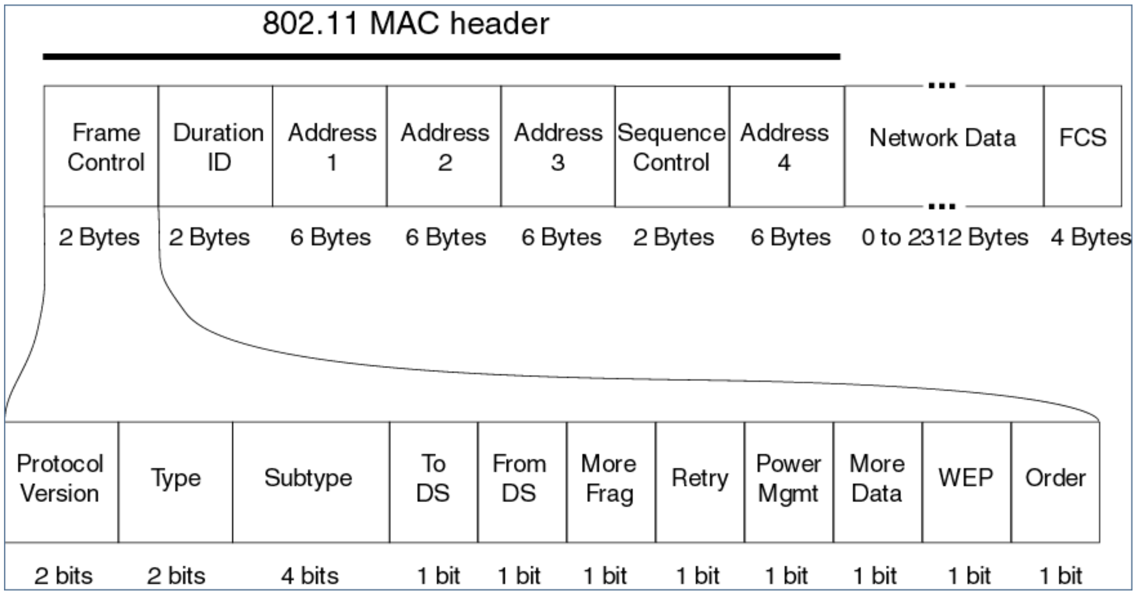


Figure 4.2: Wi-Fi Header

- **Frame Control**

The frame control field in the Wi-Fi header is a 2 Byte long field that acts as a control for the frame. This field is further sub divided into sub fields as shown in the Figure 4.2. The sub fields include Protocol Version, Type of the frame, Subtype of the Frame, the direction of the frame movement, power management specifics and many more. A Wi-Fi frame can be of three type namely Management frame, Control Frame and Data Frame.

- **Management Frames**

Management frames more specifically are the link management frames that help in the control of the link. These frames setup the link and tear it down once the communication is complete.

Management Frames are described below:

i. Authentication Frame

An exchange of authentication frames takes place with an access point when the link setup between the access point and the user device takes place. It helps in establishment of the identity of the device connecting to the network.

ii. Association Request Frame

This Frame informs the access point that the device is ready to send data on the network and hence the access point allocates resources for the device.

iii. Association Response Frame

This frame is sent by the access point in response to the Association Request Frame. The response frame may be a positive response or a negative response to the device.

iv. Beacon Frame

This is the frame that is broadcast by the access point after a fixed interval of time. This frame informs the devices that are trying to connect to the access point of the various characteristics of the access point, like the name, the operating frequency, the transfer rates, Type of encryption scheme used and more.

v. De-Authentication Frame

De-Authentication Frame is a complement of the Authentication frame. It is the frame that is sent over the network by the user device to the access point when the user device wants to disconnect from the network.

vi. Disassociation Frame

Disassociation Frame is a complement of the Association Frame. It informs the access point that it can de-allocate the resources that it had allocated for the device as the device no longer plans to use the network.

vii. Probe Request Frame

This frame is sent from a station to another station to get information about that station.

viii. Probe Response Frame:

Probe response frame is the response sent by a station for the probe request.

ix. Reassociation Request Frame:

Reassociation Request Frame is a frame that is sent when a device moves out of the range of one access point and moves into the range of another. The device sends a Reassociation request to another access point with signal strength more than the current access point.

x. Reassociation Response Frame:

This is the response frame that is sent in response to the Reassociation Request. The response may be a positive response or a negative response.

xi. Control Frame:

The control frames are sent over the network and control the contention issues of the network.

xii. Acknowledgement (ACK) Frame:

On the reception of a data frames the device sends an acknowledgement frame to the source.

xiii. Request to Send (RTS) Frame:

It is the request to send that acts as an optional contention control over the network.

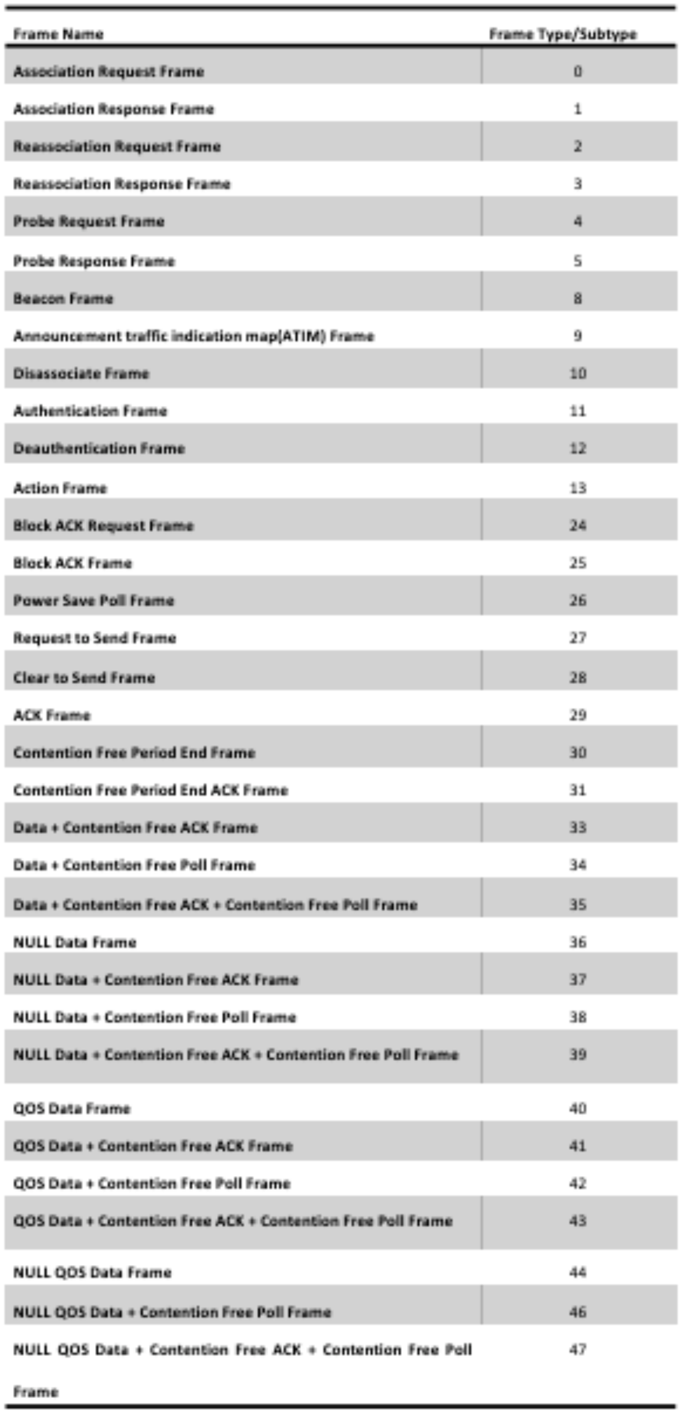
xiv. Clear to Send (CTS) Frame:

It is the optional Clear to Send Frame that is sent in response to the Request to Send Frame.

xv. Data Frame:

The Data frames are the frames that are used to move the data from the source to the destination. They generally carry higher level protocols in their data sections.

Table 4.2: Wi-Fi Frame Types



Frame Name	Frame Type/Subtype
Association Request Frame	0
Association Response Frame	1
Reassociation Request Frame	2
Reassociation Response Frame	3
Probe Request Frame	4
Probe Response Frame	5
Beacon Frame	8
Announcement traffic indication map(ATIM) Frame	9
Disassociate Frame	10
Authentication Frame	11
Deauthentication Frame	12
Action Frame	13
Block ACK Request Frame	24
Block ACK Frame	25
Power Save Poll Frame	26
Request to Send Frame	27
Clear to Send Frame	28
ACK Frame	29
Contention Free Period End Frame	30
Contention Free Period End ACK Frame	31
Data + Contention Free ACK Frame	33
Data + Contention Free Poll Frame	34
Data + Contention Free ACK + Contention Free Poll Frame	35
NULL Data Frame	36
NULL Data + Contention Free ACK Frame	37
NULL Data + Contention Free Poll Frame	38
NULL Data + Contention Free ACK + Contention Free Poll Frame	39
QOS Data Frame	40
QOS Data + Contention Free ACK Frame	41
QOS Data + Contention Free Poll Frame	42
QOS Data + Contention Free ACK + Contention Free Poll Frame	43
NULL QOS Data Frame	44
NULL QOS Data + Contention Free Poll Frame	46
NULL QOS Data + Contention Free ACK + Contention Free Poll Frame	47

4.3 Related Wi-Fi Work:

Depending on the layer on which the Intrusion Detection System (IDS) focuses on, we can classify them into three types: 1) Physical layer based IDS[64][65][66][67][68], 2) MAC layer based IDS [69][70][71][72], and 3) Physical layer and data link layer based IDS [73]. Most of the approaches to detect the physical layer attacks on the Wi-Fi network involve the use of signal strength or multiple antennas to detect the angle of the attack on the Wi-Fi network. This method can efficiently detect attacks on the physical layer of the Wi-Fi network such as network jamming or detect attacks on the data link layer as in MAC address spoofing. This approach is complex because it takes into account the effects of signal fading, noise, changes in the medium, and effects due to the movement of the target and may negatively affect the performance of the intrusion detection system. The intrusion detection systems that operate on the data link layer use the data obtained from the Wi-Fi frame to detect attacks. Open source intrusion detection systems like Snort and most of the commercial Intrusion detection systems available like AirMagnet [74] and some detection engines in Air Defense [75] use the misuse detection approach to detect Wi-Fi attacks. But as this approach involves the use of attack signatures to detect attacks, modified attacks or zero-day attacks cannot be detected by these methods.

In [76] the authors focus on the use of the round-trip time (RTT) of the signal to track down the location of the user from the access point. In [65] the authors use the signal strength of the received signal to track spoofing attacks. In their work, the authors profile the received signal using Gaussian Mixture models. Then the profiles that are generated by the use of these models for each transmitter used to detect spoofing attacks on the networks. In this paper we present a similar approach to track the location of the attacking device. The system uses the signal strength to track the location of the attacker once the attack on the network has been detected. In our method, we use machine learning algorithms to generate profiles for different access points.

Kolias et.al in [77] present an extensive study of the Wi-Fi protocol and the attacks possible on the Wi-Fi protocol. As a part of this study, that authors collected a family of datasets that not only have normal Wi-Fi traffic, but also collection of attack traces. This family of datasets is used to measure the performance of our approach. The authors also use classification-based algorithms to classify the attacks in the family of dataset they presented. They trained on the datasets marked in the family as training datasets and tested the performance of the models on the datasets marked as the testing datasets. This approach is not as effective as our approach as they train on training datasets in its raw form, while we use data structures like the n-gram and flows to extract the normal behavior of the protocol.

Allahdadi et. al in [78], model the Wi-Fi protocol's behavior using hidden Markov models (HMM) to model the state transitions of the Wi-Fi protocol. They use this approach to detect anomalies like access point shutdown, access point over load, noise, and flash crowd.

Usha et.al in [79], present an approach to perform anomaly-based intrusion detection on the Wi-Fi protocol. Usha et.al use normalize gain (NG), to select the optimal features from the AWID dataset to train semi-supervised clustering(SSC) algorithms to detect normal and abnormal behavior in Wi-Fi protocol. They were able to get different degree of accuracy on the different datasets, with upto 98% accuracy.

Hamid et.al in [46] presented an approach, that used n-grams combined with bloom filters to detect attacks on Wi-Fi protocol. Hamid et.al presented the same approach as a part of his Ph.D dissertation. This work is an extension of the work done in that dissertation. The authors of this dissertation also noticed that the runtime IDS designed by Hamid et.al did not detect the minimal deauthentication attack presented in this research. The approach presented by Hamid used a threshold parameter that classified normal events from the abnormal. This threshold parameter needed manual tuning. This work uses machine learning models to detect attacks, thus eliminating the need to manually tune the threshold parameter.

Satam et.al in [80] presented an approach, that used n-grams with machine learning to detect attacks on Wi-Fi networks. In that work, the authors applied their approach to detect attacks on single access point Wi-Fi networks and distributed Wi-Fi networks. They also presented an approach that could detect the location of the attacker once the attack has been detected. This work is an extension of the work done by Satam et. al. This work expands on the work by including a more diverse feature set, more extensive testing, with testing on larger and more diverse datasets, including an external dataset not collected at the University of Arizona. This work tests the approach against all attacks possible on the Wi-Fi protocol using five different machine learning algorithms instead of just a conjunctive rule-based algorithm as done in [80].

4.4 Using The Proposed Methodology To Design An AB-IDS For The Wi-Fi Protocol

- **Wi-Fi State Machine:**

The Wi-Fi protocols is a stateful protocol that is, each transition in the protocol follows state machine. The Wi-Fi state machine is as shown below.

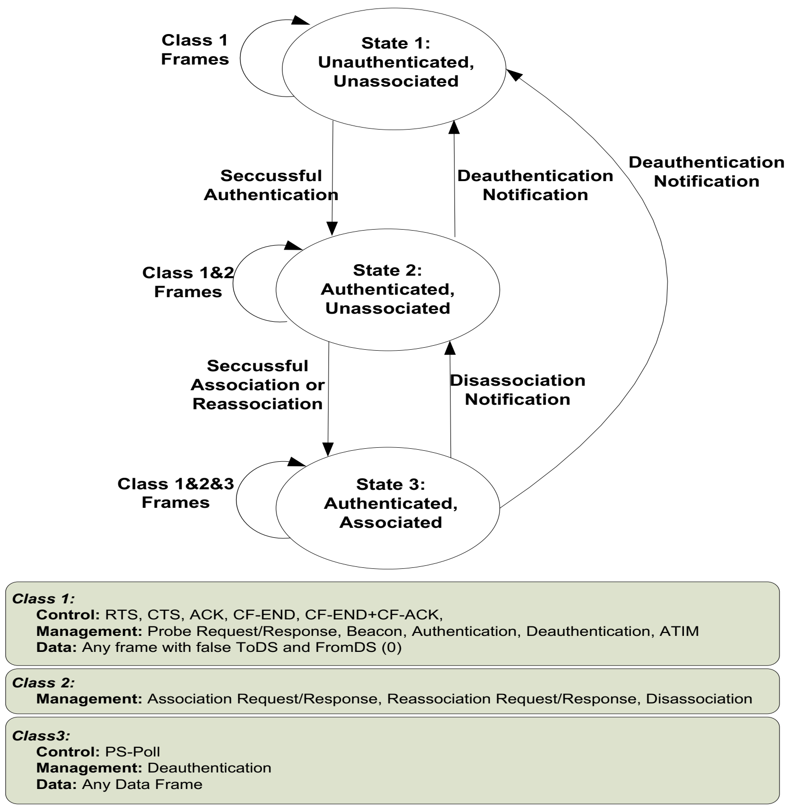

Class 1 Frames
State 1: Unauthenticated, Unassociated
Seccussful Authentication
Deauthentication Notification
Deauthentication Notification
Class 1&2 Frames
State 2: Authenticated, Unassociated
Seccussful Association or Reassociation
Disassociation Notification
Class 1&2&3 Frames
State 3: Authenticated, Associated
Class 1:
Control: RTS, CTS, ACK, CF-END, CF-END+CF-ACK,
Management: Probe Request/Response, Beacon, Authentication, Deauthentication, ATIM
Data: Any frame with false ToDS and FromDS (0)
Class 2:
Management: Association Request/Response, Reassociation Request/Response, Disassociation
Class3:
Control: PS-Poll
Management: Deauthentication
Data: Any Data Frame

Figure 4.3: Wi-Fi State machine.

As shown in Figure 4.3 the Wi-Fi state machine has three different states. State 1 is the unauthenticated and unassociated state. In this state, Wi-Fi device is not connected to the network. State 2 is the Authenticated and Unassociated state. In this state, the device is connected to the network but the access point has no resources allocated to for that device. In the State 3 of operation, the device is Authenticated and is also Associated to the network. This allows the device to send the traffic over the network.

- **Identification of Attack surfaces, Impact and mitigation analysis for the Wi-Fi protocol**

Figure 4.3 highlights the state machine used by the Wi-Fi protocol. A careful analysis of this state machine also highlights the different attack surfaces that can be used to target the Wi-Fi protocol. Wi-Fi protocol is vulnerable to jamming attacks. Such jamming attacks pose a threat to all wireless protocols and can be easily detect systems monitoring signal strengths in frequencies. Attacks on the datalink layer of the Wi-Fi protocol performed through frame spoofing and pose a bigger threat to the Wi-Fi protocol. These attacks could be used to perform De-authentication attacks, Man in the middle attacks, Denial of Service Attacks (DoS), Fake authentication attacks etc. The above-mentioned attack surfaces exist on any device using Wi-Fi networks including IoT devices.

Notwithstanding the above-mentioned attack vectors, a particular IoT device might have other possible attack vectors as a result of bad system design. For instance Sivaraman et.al [81] in their work identified IoT devices like the Withings Baby Monitor that allows the user to use an IP camera to monitor his/her baby, exchanges image data over Wi-Fi in plain text, allowing the researchers to execute a man in the middle attack on the baby monitor; in Withings Smart Body Analyzer, a weighing scale to measure fat, heartrate and BMI the researchers observed that the users personal information was exchanged on Wi-Fi channels unencrypted. These attack vectors are a result of bad design or low computation power of the IoT device and hence make it easier for the attacker to execute attacks like man in the middle attacks.

- **Feature Extraction**

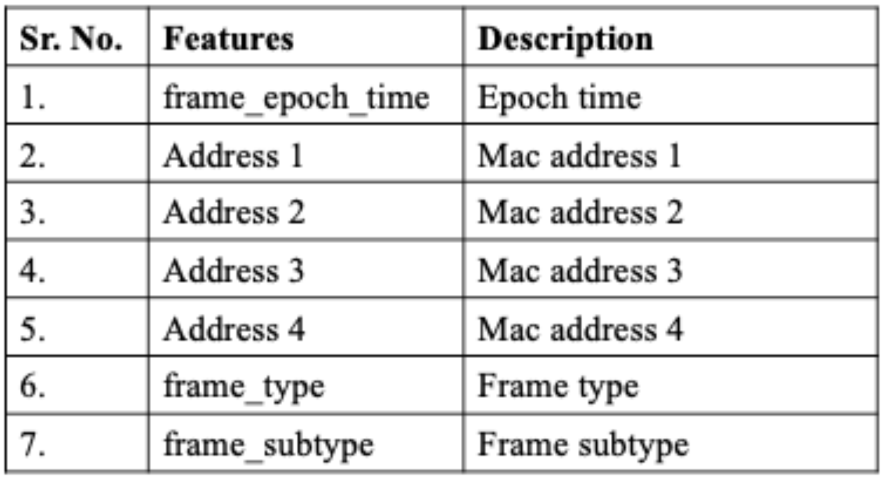
While designing and implementing an AB analysis system, it is important to identify the operational point of the system at any instant of time. The process of identification of the operation point of the Wi-Fi network begins with collecting Wi-Fi frames in the network. Literature review and study of the Wi-Fi protocol characterized by Figure 4.3 allowed us to extract an initial feature set to characterize the operational point of the Wi-Fi protocol. This feature set is listed in Figure 4.4.

i. Wi-Fi Flow/session extraction

From a raw Wi-Fi frame observed on the Wi-Fi network, the frame_epoch_time, that is the time that frame was observed; Address 1-4 are the four different addresses from the Wi-Fi frame; frame_type, and frame_subtime is extracted from the raw frame. Wi-Fi traffic can be split into flows or sessions of time interval 't seconds' based on frame source address and frame destination address pair. The frame source and destination addresses are extracted from the addresses 1-4 based on the type of Wi-Fi frame and the working of the Wi-Fi protocol.

ii. N-Gram extraction

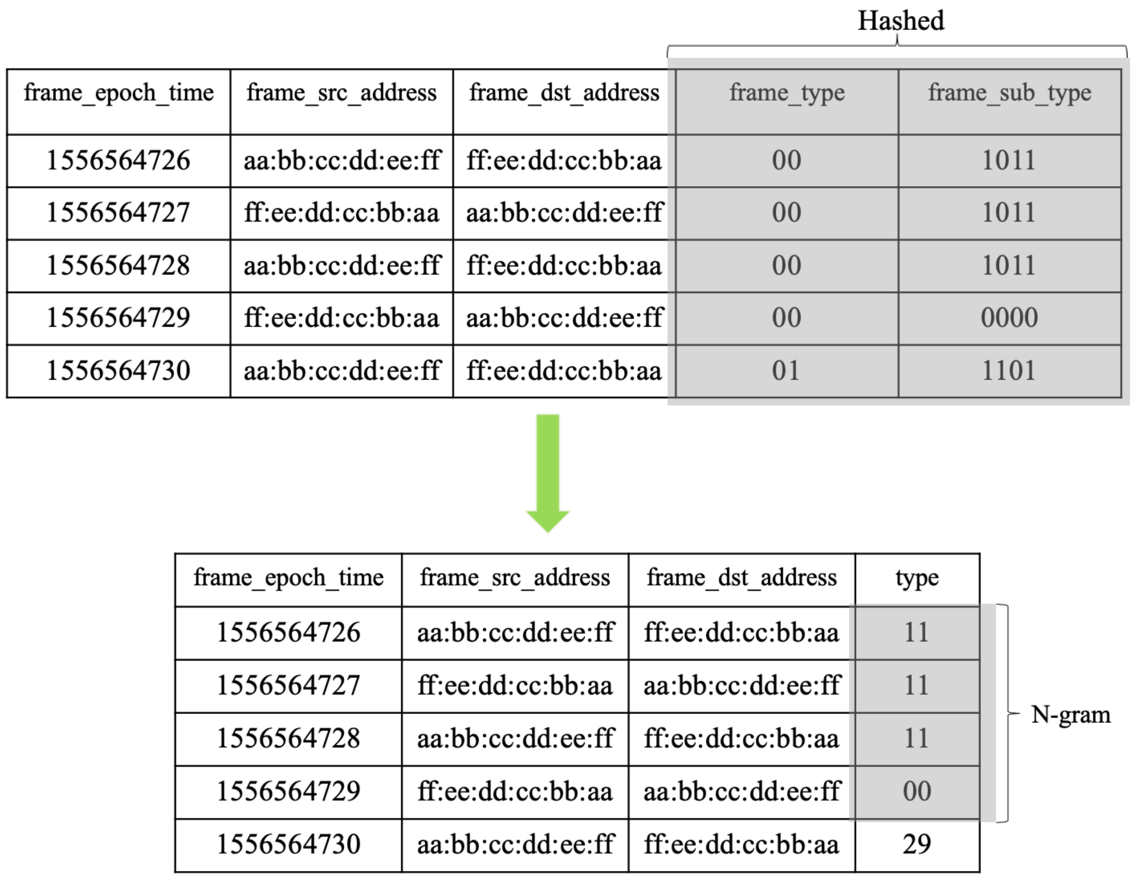
Figure 4.5 shows how a Wi-Fi session is converted into a flow of n-grams. To



Sr. No.	Features	Description
1.	frame_epoch_time	Epoch time
2.	Address 1	Mac address 1
3.	Address 2	Mac address 2
4.	Address 3	Mac address 3
5.	Address 4	Mac address 4
6.	frame_type	Frame type
7.	frame_subtype	Frame subtype

Figure 4.4: Initial Feature Set

transform the flow into n-grams, the frame_type and frame_subtype is hashed together to form a combined field called type. The n-grams are formed using a sliding window over the field type, where the size of the sliding window is equal to the size of the n-grams.



frame_epoch_time	frame_src_address	frame_dst_address	frame_type	frame_sub_type
1556564726	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	00	1011
1556564727	ff:ee:dd:cc:bb:aa	aa:bb:cc:dd:ee:ff	00	1011
1556564728	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	00	1011
1556564729	ff:ee:dd:cc:bb:aa	aa:bb:cc:dd:ee:ff	00	0000
1556564730	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	01	1101

frame_epoch_time	frame_src_address	frame_dst_address	type
1556564726	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	11
1556564727	ff:ee:dd:cc:bb:aa	aa:bb:cc:dd:ee:ff	11
1556564728	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	11
1556564729	ff:ee:dd:cc:bb:aa	aa:bb:cc:dd:ee:ff	00
1556564730	aa:bb:cc:dd:ee:ff	ff:ee:dd:cc:bb:aa	29

Figure 4.5: Feature set to fingerprinting data-structure mapping

iii. N-gram counting

During the initial stages of the training, we observe all possible unique n-grams for a n-gram size. While collecting this set of unique n-grams, we calculate the frequency of observation of each unique n-gram.

iv. Features to build the machine learning model

Figure 4.6 shows the set of features used to build the machine learning models. Frequency of each n-gram observed in a flow is used to get the probability of the flow.

According to Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.1}$$

where,

A and B are events

P(A|B) is the probability of event A occurring given event B is true.

Hence, for n gram of size n we have:

$$P(W_n|W_1^{n-1}) = \frac{P(W_1^{n-1}|W_n)P(W_n)}{P(W_{n-1}|W_1^{n-2})} \tag{4.2}$$

where,

$P(W_n|W_1^{n-1})$ is the probability of n-gram of size n given probability of n gram of size 1 to (n-1)

$P(W_{n-1}|W_1^{n-2})$ is the probability of n-gram of size (n-1) given probability of n gram of size 1 to (n-2)

P(W_n) is the probability of n-gram of size 1

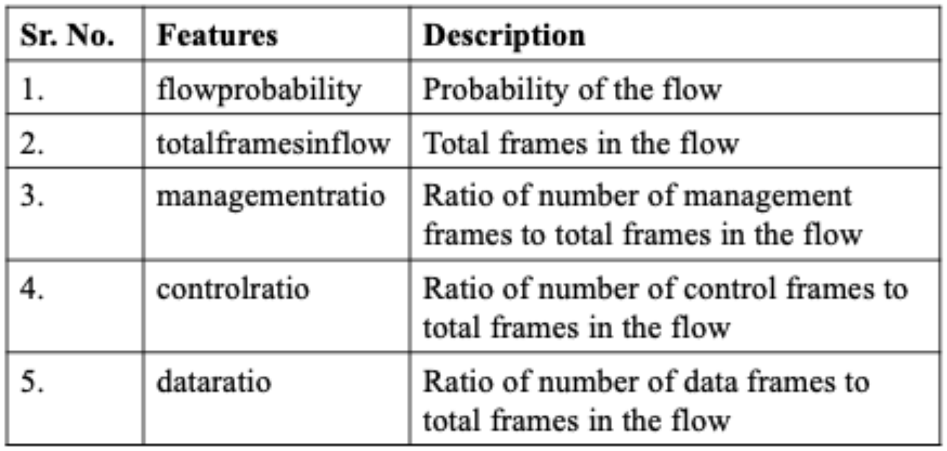
Therefore, to obtain $P(W_1^{n-1}|W_n)$:

We use the smoothing model to calculate the probability of Flow.

By using the Jelinek-Mercer smoothing model [121], we calculate the value of constant λ and calculate the probability of the flow:

$$P(W_1^{n-1}|W_n) = \lambda P(W_1^{n-1}|W_n) + (1-\lambda)P(W_1^{n-2}|W_n) \tag{4.3}$$

After obtaining the probability of the flow, we calculate the total number of n-grams in the flow, the ratio of new/unseen n-grams to the total number of frames in the flow, the ratio of number of management frames to the total number of frames in the flow, the ratio of number of control frames to the total number of frames in the flow, and the ratio of number of data frames to the total number of frames in the flow.



Sr. No.	Features	Description
1.	flowprobability	Probability of the flow
2.	totalframesinflow	Total frames in the flow
3.	managementratio	Ratio of number of management frames to total frames in the flow
4.	controlratio	Ratio of number of control frames to total frames in the flow
5.	dataratio	Ratio of number of data frames to total frames in the flow

Figure 4.6 Features used to build machine learning models

- **Building behavior models using machine learning**

Features presented in Figure 4.6 are used to build machine learning models that can identify the normal behavior from the abnormal behavior. Figure 4.7 shows the architecture of the Wi-Fi IDS, that was used to perform runtime analysis of the approach. As shown in Figure 4.7 the IDS have 2 modules; Sniffer module and Behavior Authentication Module.

i. Sniffer Module

The Sniffer module is a libpcap [82] based implementation that collects the Wi-Fi traffic using the configured network card. The collected Wi-Fi traffic is stored in the memory using a linked list.

ii. Behavior Authentication Module(BAM)

The BAM reads raw Wi-Fi packets from the shared database and performs analysis by converting the traffic into flows, extracting the n-grams, and uses machine learning models to classify the flows as either normal or abnormal.

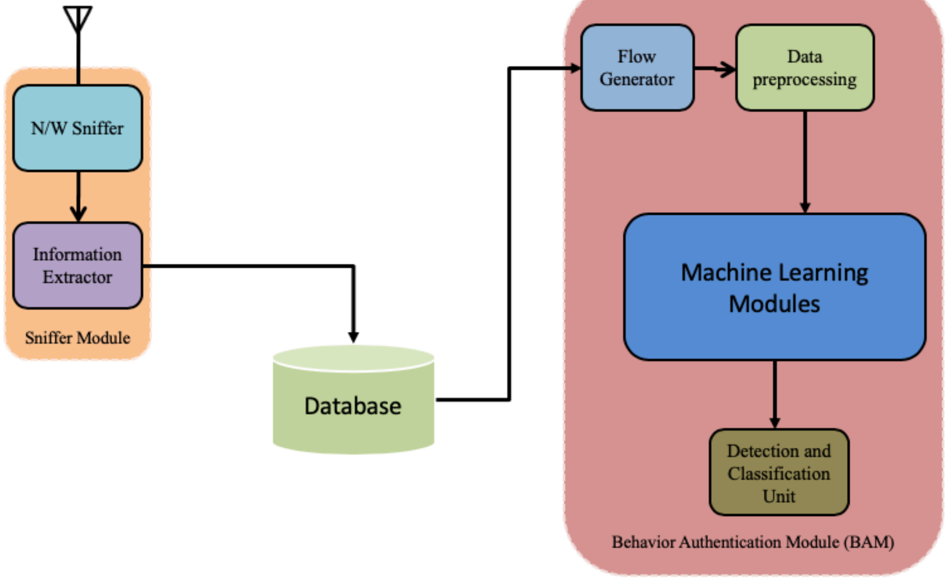


Figure 4.7: Wi-Fi IDS architecture

4.5 Attacks on Wi-Fi networks

In this section we describe the attacks on Wi-Fi networks. The attacks listed here target the Wi-Fi encryptions systems (WEP, WPA, WPA2) and the availability of the Wi-Fi networks. Attacks on the physical layer of the Wi-Fi network are outside the scope of this work.

- **Attacks targeting the availability of the Wi-Fi protocol**

All the attacks listed here take advantage of the fact that Wi-Fi management frames are transmitted unprotected and can be spoofed easily.

i. Deauthentication attack

Deauthentication attack is a Denial of Service (DoS) attack on the Wi-Fi network. As shown in the Figure 4.8, the attacker spoofs deauthentication frames being the access-point and the victim, resulting in the victim getting disconnected from the Wi-Fi network.

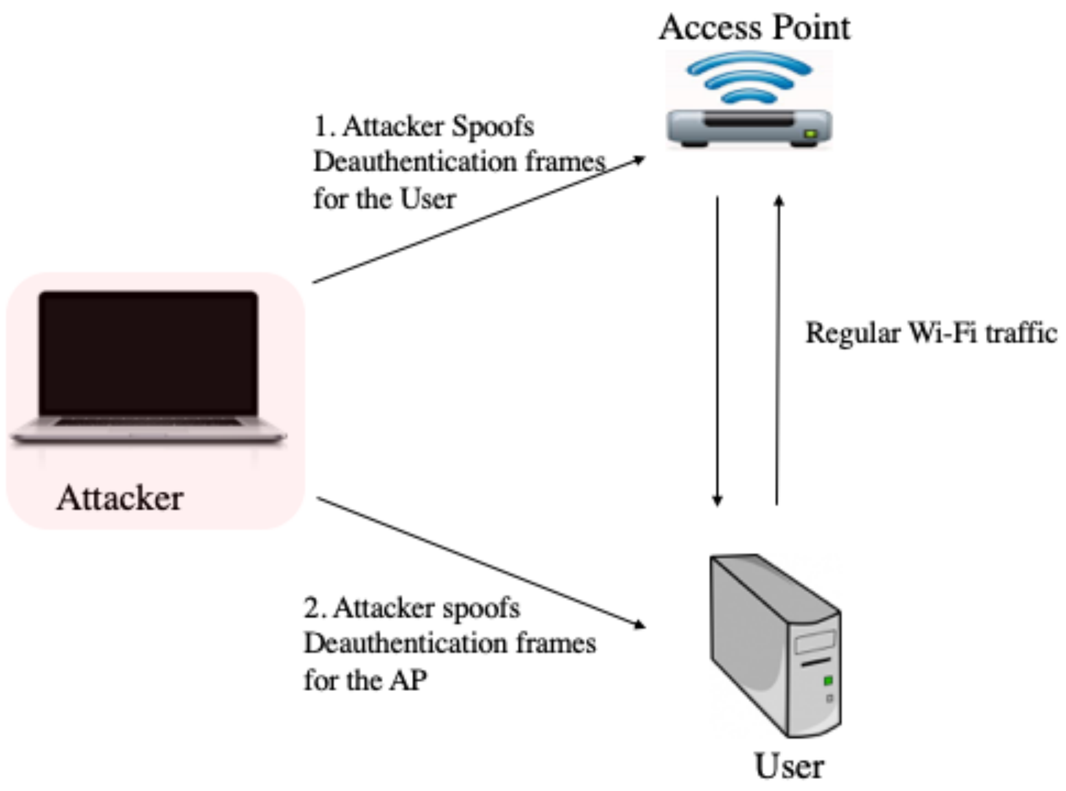

Access Point
1. Attacker Spoofs Deauthentication frames for the User
Regular Wi-Fi traffic
Attacker
2. Attacker spoofs Deauthentication frames for the AP
User

Figure 4.8 Deauthentication attack

ii. Disassociation attack

As shown in the Figure 4.9, the attacker spoofs disassociation frames being the access-point and the victim, resulting in the victim getting disassociated from the Wi-Fi access point.

In a single access point Wi-Fi network, this results into a weak DoS attack (compared to deauthentication attack) as the transition from being disassociated to communicating on the Wi-Fi network is shorter than the transition from deauthenticated to communicating on the Wi-Fi network.

In a distributed access point Wi-Fi network, this attack may result in a DoS attack on the whole distributed network by causing congestion on the network. When a disassociation attack is executed on a victim in a distributed Wi-Fi network, the victim will associate himself with another access point in the vicinity (on the same Distribute Wi-Fi network). Thus, the victim will not face any service disruptions. But when this attack is performed on a large number of users, this attack will severely degrade the capacity of the network to support the large number of users that it normally supports. This can be done by the attacker preventing users to use a few specific access points in the network, forcing the users to use other access points in the network that might already be overloaded with a lot of users. If this attack is done on a large enough number of users, it will cause the network to crash.

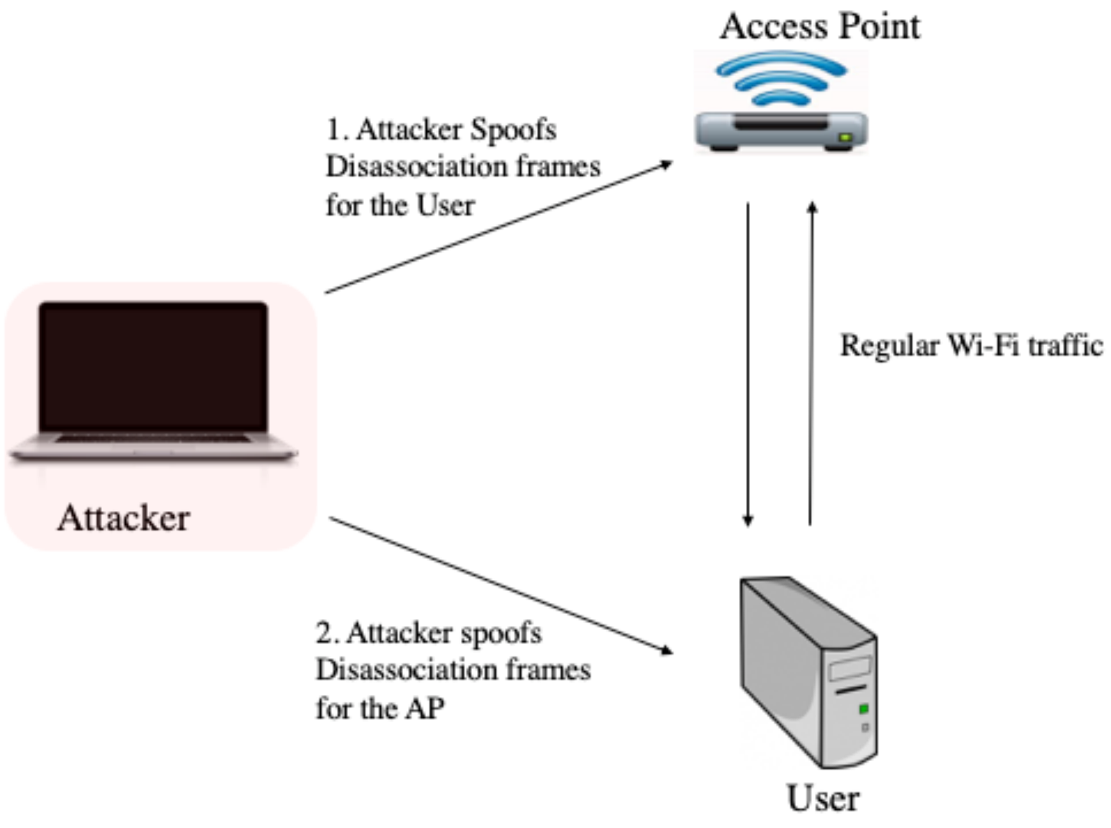

Access Point
1. Attacker Spoofs Disassociation frames for the User
Regular Wi-Fi traffic
Attacker
2. Attacker spoofs Disassociation frames for the AP
User

Figure 4.9 Disassociation attack

iii. Fake Authentication attack/Authentication request flooding

In this attack the attack tries to exhaust the client association table by sending a flood of authentication requests to the access point. This attack is shown in Figure 4.10. This attack exploits the fact that an access point can at any given point serves only a fixed number of users which is fixed by the physical memory limit on the access point or the value setup while configuration and an entry is inserted in the association table when an AP receives an authentication request (even if authentication is not completed).

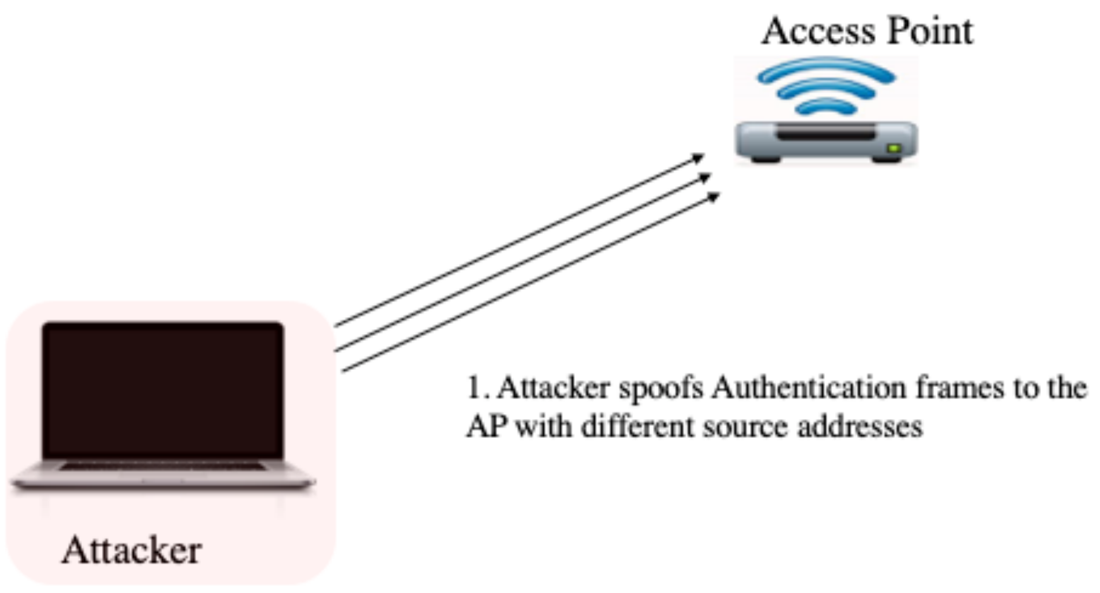

Access Point
1. Attacker spoofs Authentication frames to the AP with different source addresses
Attacker

Figure 4.10 Authentication Flood attack

iv. Deauthentication broadcast attack

In this attack the attacker spoofs deauthentication frames for an access point with broadcast address set as the client address. This attack is similar to the deauthentication attack described before, except it causes all the users to deauthenticate from the network. This attack might cause increase in network load as most users will try and connect to the same network at the same time. Figure 4.11 shows the attack.

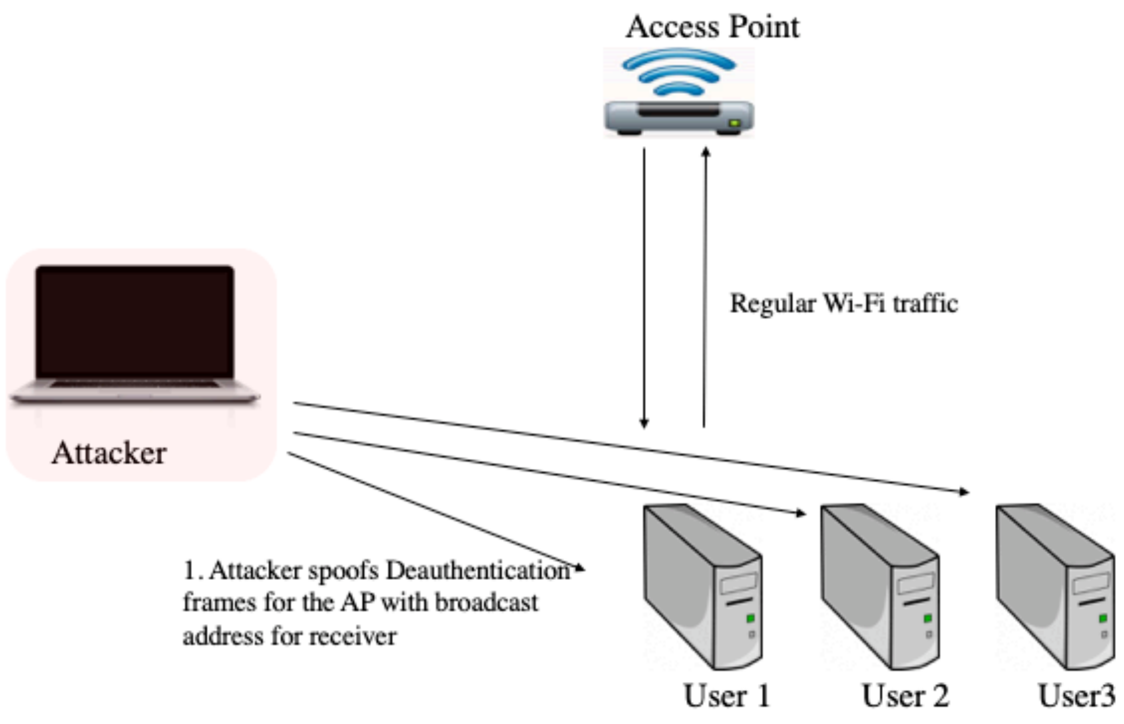

Access Point
Regular Wi-Fi traffic
Attacker
1. Attacker spoofs Deauthentication frames for the AP with broadcast address for receiver
User 1
User 2
User3

Figure 4.11 Deauthentication broadcast attack

v. Disassociation broadcast attack

In this attack the attacker spoofs disassociation frames for an access point with broadcast address set as the client address. This attack is similar to attack ii. except it will cause all the users to disassociate from the access point. This attack might cause an increase in network load as most users will try and connect to the same network at the same time. Figure 4.12 shows the attack. This attack will cause network load problems in distributed Wi-Fi networks.

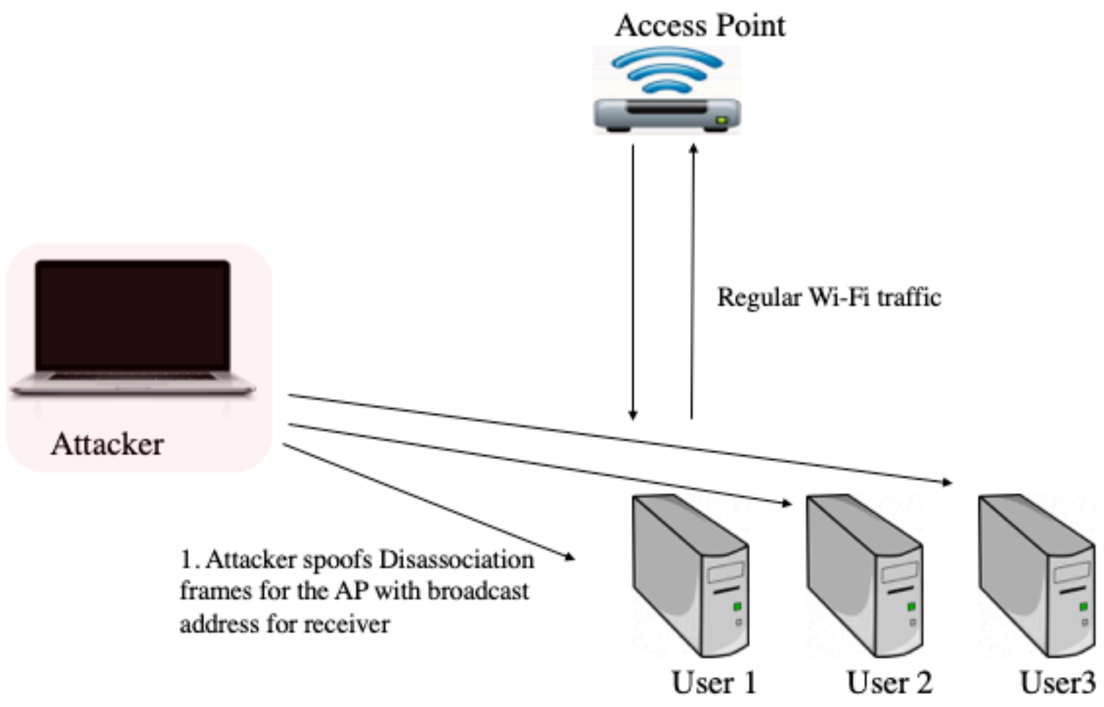

Access Point
Regular Wi-Fi traffic
Attacker
1. Attacker spoofs Disassociation frames for the AP with broadcast address for receiver
User 1
User 2
User3

Figure 4.12 Disassociation broadcast attack

vi. Fake power saving attack

In this attack, the attacker sends a frame with data set to null and the power saver bit set to 1. The Access Point (AP) will accept the frame thinking the user's device wants to go into power saver mode and will start buffering all the traffic for the user. The TIM field in the next beacon will have the address of the user but will be ignored by the user as the user is not in power saver mode. Eventually, the AP will drop the buffered traffic based on its setting and memory load. Figure 4.13 shows the attack.

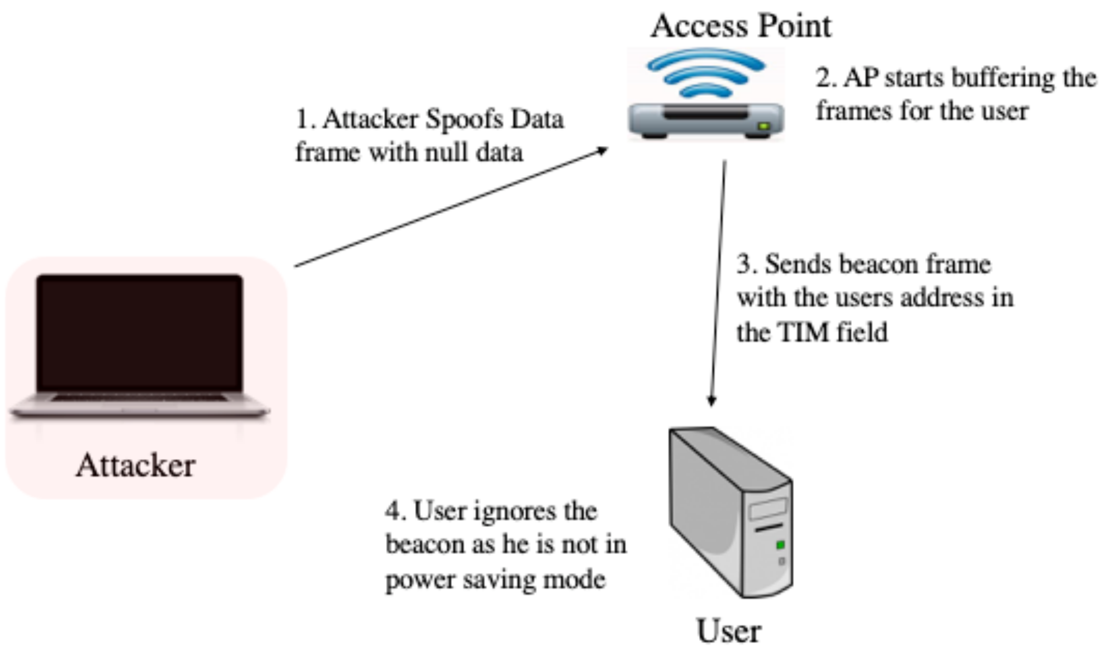

Access Point
2. AP starts buffering the frames for the user
1. Attacker Spoofs Data frame with null data
3. Sends beacon frame with the users address in the TIM field
Attacker
4. User ignores the beacon as he is not in power saving mode
User

Figure 4.13 Fake power saving attack

viii. CTS Flooding Attack

In Wi-Fi Request to Send (RTS), Clear to Send (CTS) are optional frames that support a mechanism that can be used to control the congestion and access to the physical medium. When enabled, a legitimate user sends CTS frames before transmitting, thus telling other users to stay of the medium for a fixed amount of time. An attacker may flood the Wi-Fi network with CTS frames making other users wait and stay of air for a long time. Figure 4.14 shows the attack

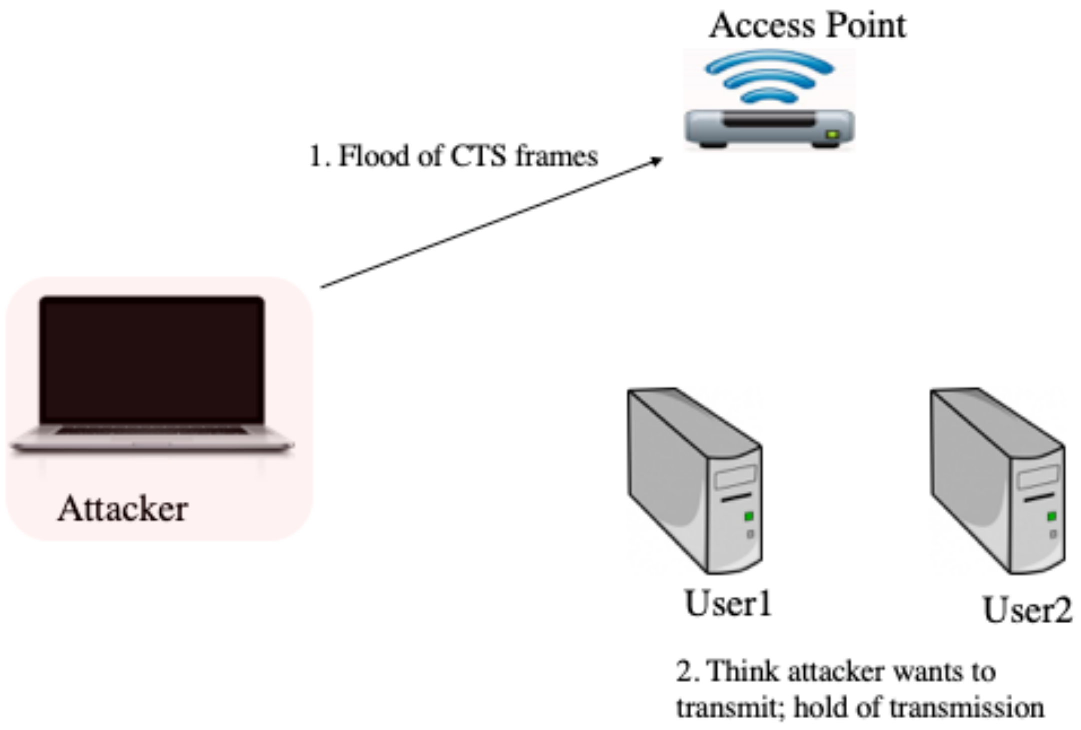


Figure 4.14 CTS Flooding attack

ix. RTS Flooding attack

RTS flooding works similar to the CTS flood attack, where the attacker spoofs and floods the network with RTS frames, asking for large transmission windows, hoping to keep other legitimate users off air for those periods. Figure 4.15 shows the attack

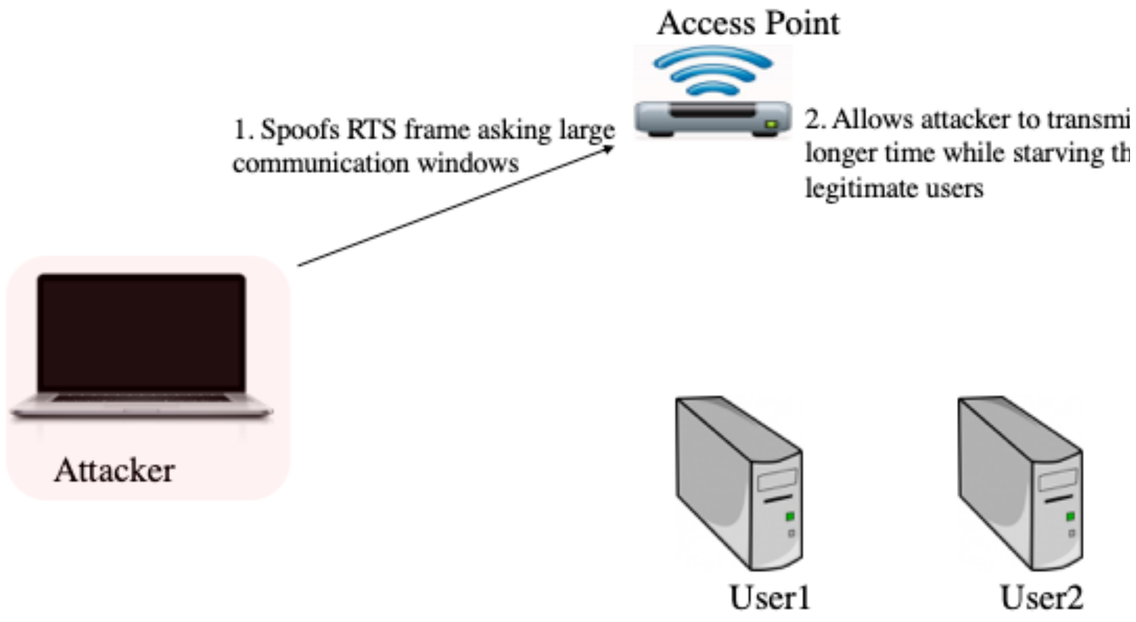

Access Point
1. Spoofs RTS frame asking large communication windows
2. Allows attacker to transmi longer time while starving th legitimate users
Attacker
User1
User2

Figure 4.15 RTS flooding attack

x. Probe request flooding attack

Wi-Fi protocol requires an AP to respond to all the probe request it receives. In this attack, the attack will flood the AP with probe requests, forcing it to respond to a huge amount of probe requests, thus bogging it down and making it incapable of responding to other users and legitimate probes, preventing users from connecting to the network. Figure 4.17 shows the attack.

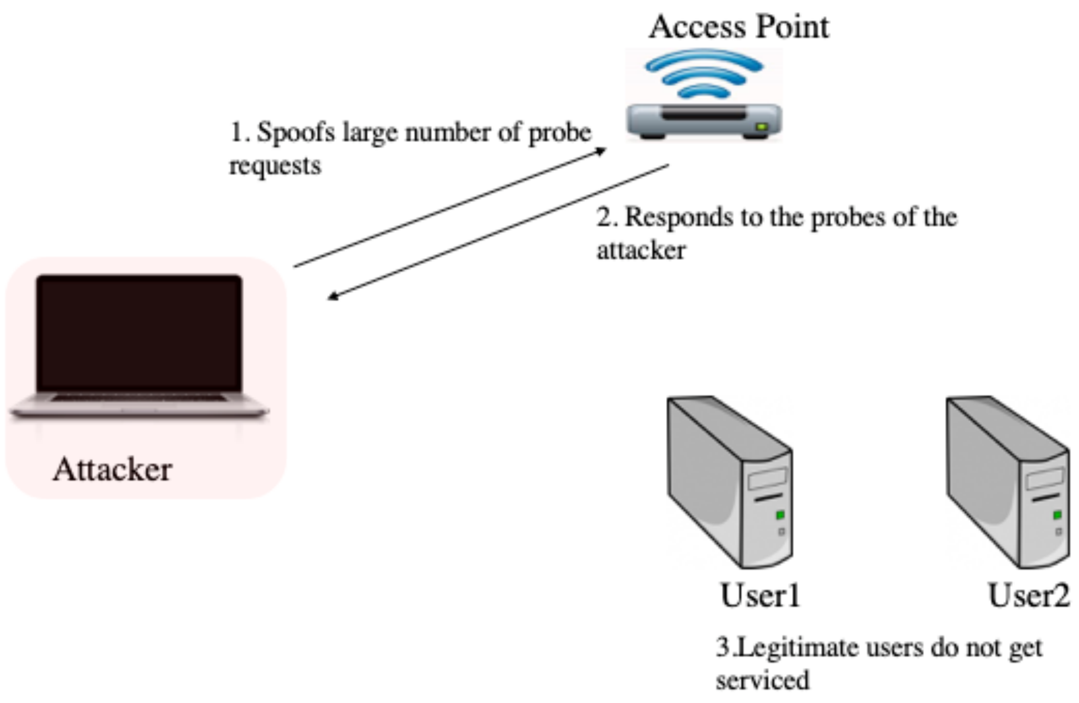

Access Point
1. Spoofs large number of probe requests
2. Responds to the probes of the attacker
Attacker
User1
User2
3.Legitimate users do not get serviced

Figure 4.17 Probe request flooding attack

xi. Probe response flooding

In the probe response flooding attack, the attack monitors the network and waits for the victim to send out a probe request to an AP. On seeing the victim send the probe request, the attacker floods the network with fake probe responses, filled with wrong information. Because of the volume of the received probes, the chances of the victim accepting the fake response frames is high. If the victim accepts the fake probe responses, it will not be able to connect to the AP as its setting will not match those of the AP. Figure 4.18 shows the attack.

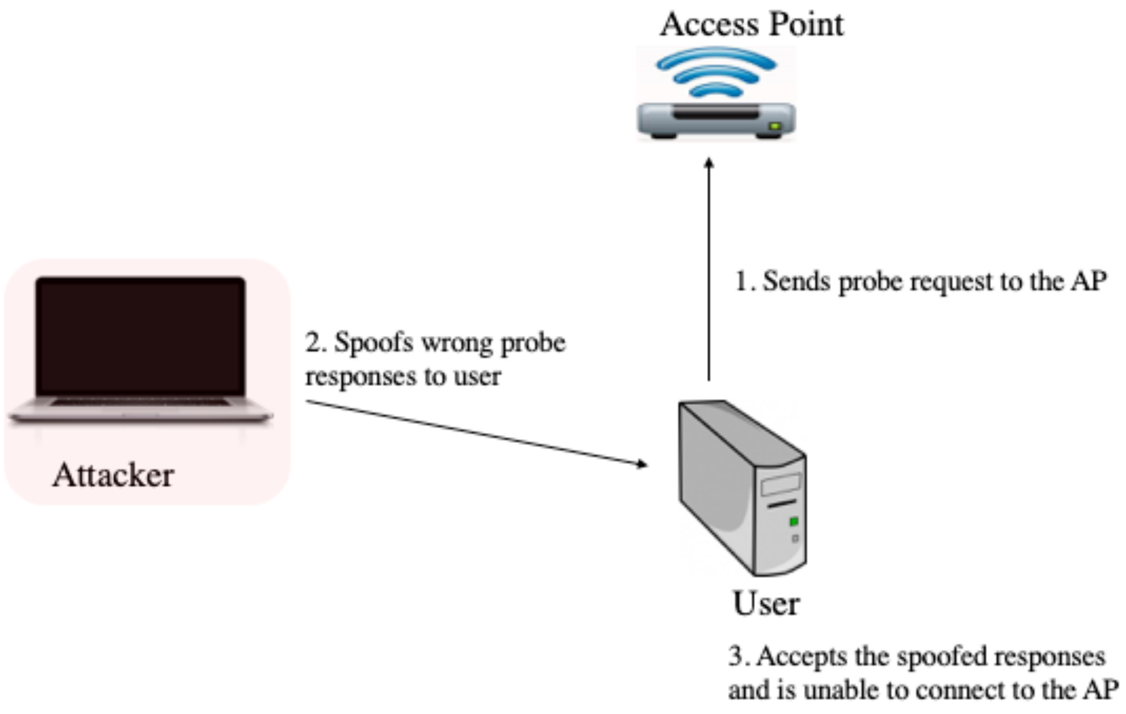

Access Point
1. Sends probe request to the AP
2. Spoofs wrong probe responses to user
Attacker
User
3. Accepts the spoofed responses and is unable to connect to the AP

Figure 4.18 Probe response flooding attack

xii. Man in the middle attack using rogue access point or evil twin

In this attack, the attack sets up an AP with the same name as the legitimate access point. When new and naïve users try joining the network, there is a high chance that they will join the network setup by the attacker.

The attacker can force the users to join his fake network by performing deauthentication attack on the legitimate network. As most Wi-Fi device are set to automatically reconnect to another access point, they will give the rogue access point or the evil twin preference over others (as it has the same SSID). Figure 4.19 shows the attack.

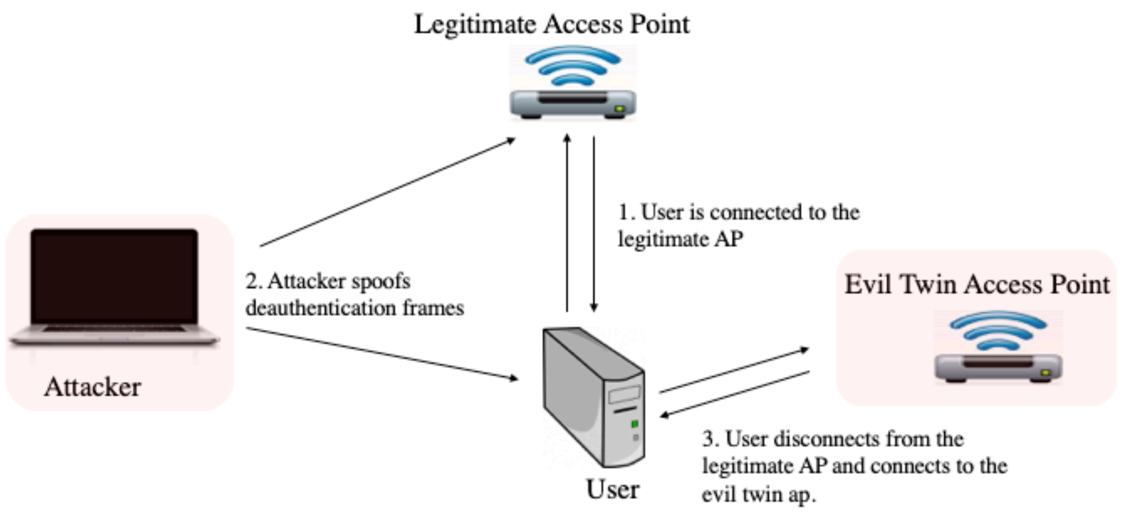

Legitimate Access Point
1. User is connected to the legitimate AP
2. Attacker spoofs deauthentication frames
Evil Twin Access Point
Attacker
User
3. User disconnects from the legitimate AP and connects to the evil twin ap.

Figure 4.19 Man in the middle attack using rogue access point or evil twin

xiii. Beacon flooding attack

In this attack, the attacker will spoof fake beacon frames for non-existent APs. This will cause the users to see a long list of APs available, making it harder for him to search the correct AP. Most GUIs limit the number of Aps that they list (in ascending order). This attack could result in the users, access point not being listed on the list of Aps. Figure 4.20 shows the attack.

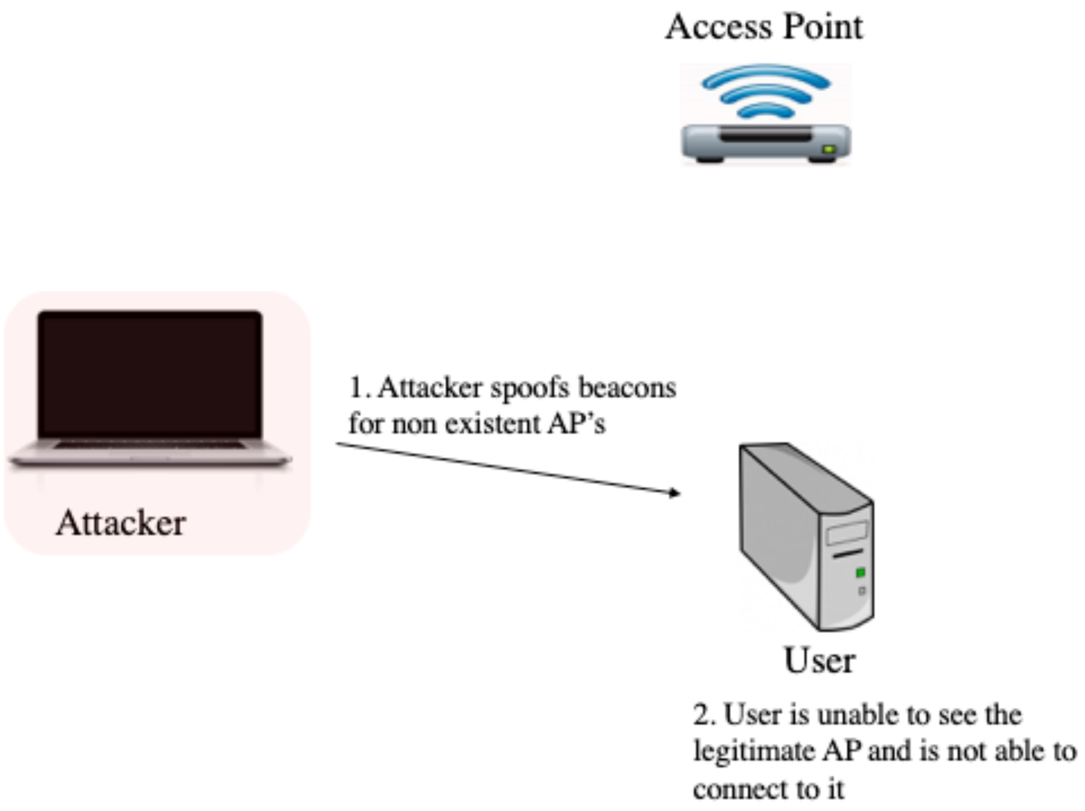

Access Point
Attacker
1. Attacker spoofs beacons for non existent AP's
User
2. User is unable to see the legitimate AP and is not able to connect to it

Figure 4.20 Beacon flooding attack

xiii. Modified Deauthentication attack

This attack is a modification of Deauthentication attack that we created. In this attack, unlike the other deauthentication attack, where the attacker floods the network with deauthentication frames, the attacker spoofs deauthentication frames on the network in a timely manner. The attacker monitors the network traffic. In our experimentation we observed that when one deauthentication frame is spoofed to the network and the user (while both the devices are communicating), they both respond to the deauthentication frames while trying to keep the communication alive. If an attacker spoofs another frame during this response, it harder for the two devices being attacked to recover. We were able to successfully deauthenticate Wi-Fi devices with as low as 8 frames in a 10 second window, that is a very small attack footprint when compared to a few hundred frames spoofed in 3-4 seconds by the other deauthentication attacks. Figure 4.21 shows the attack.

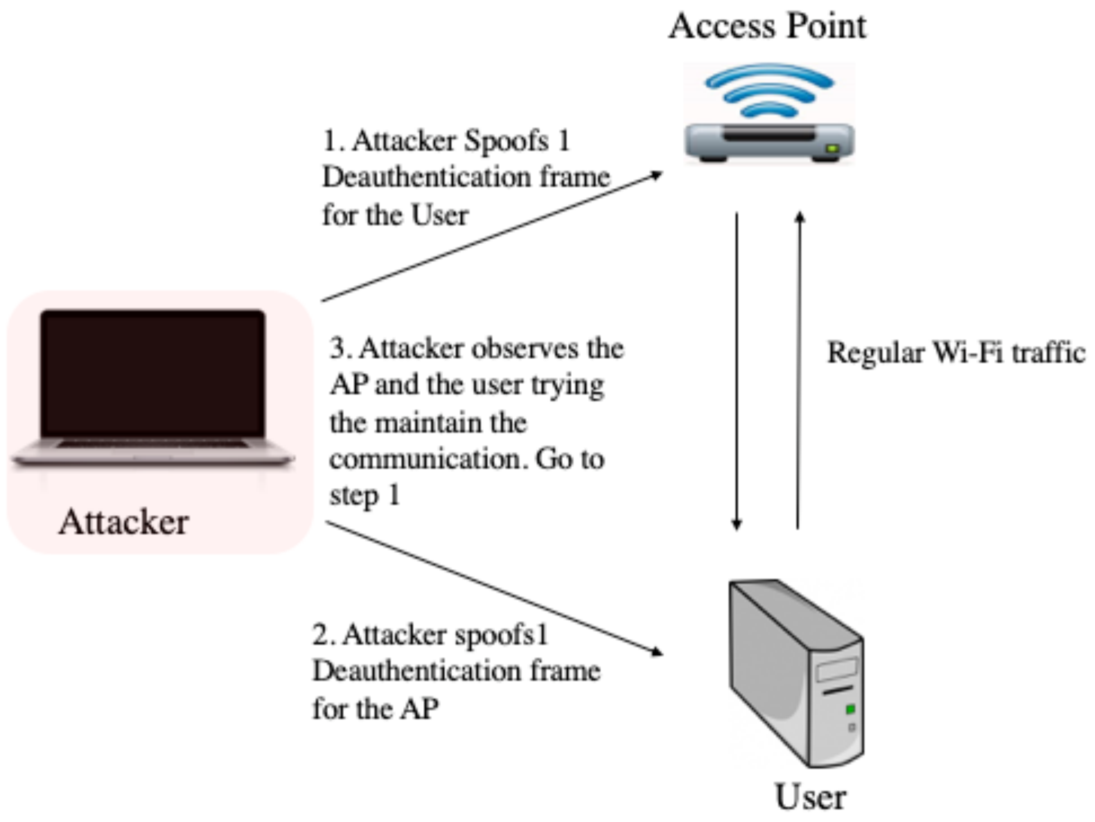

Access Point
1. Attacker Spoofs 1 Deauthentication frame for the User
3. Attacker observes the AP and the user trying the maintain the communication. Go to step 1
Regular Wi-Fi traffic
Attacker
2. Attacker spoofs1 Deauthentication frame for the AP
User

Figure 4.21 Modified deauthentication attack

- **Attacks targeting the encryption protocol**

Following are the attacks that target the encryption used to secure data over Wi-Fi network.

i. ChopChop Attack

This attack targets the WEP protocol, as in WEP CRC-32 was wrongly utilized for message integrity and WEP allows message reply. While performing this attack, the attacker chops the last byte of the encrypted data that he/she sniffed and XORs it with a chosen value, with the hope that it will lead to a sequence valid for a specific ICV. Then, the attacker injects this modified frame into the network, AP itself tells the attacker if his guess was correct or wrong w(obligated by the protocol). The attacker repeats the process until the access point tells him he was correct.

ii. Fragmentation Attack

The Wi-Fi protocol allows Wi-Fi frames to be fragmented into smaller fragments, if the packet being sent over the network is larger than the maximum length. In the fragmentation attack the attacker tries to achieve the same result as the chopchop attack, but by sending lesser frames on the Wi-Fi network. The attacker guesses the first 8 bytes of the keystream with high probability and takes advantage of the fragmentation mechanism to construct frames of size 8 bytes and send them to the network with broadcast address. The AP accepts the frames and broadcasts the frames on the network. The attacker sniffs new encrypted data, but knows the message, allowing the attacker to get keystream by simple xoring.

iii. Caffe Latte Attack

Caffe latte allows the attacker to get the IV when the user is away from the target Wi-Fi network. Most Wi-Fi devices are actively looking for known Ap's (by probing) and try and automatically connect to an AP with similar name. The attacker observes such probes from the user's device and creates an evil twin with the same SSID. The user's device will automatically try to connect with the AP. As WEP does not verify the AP, it will get authenticated and send out DHCP requests. On failure to get an IP via DHCP, the user's device will allocate itself a local IP and send ARP request (encrypted) to the AP. The attacker floods the user with ARP responses, while modifying the encrypted ARP packet to guess the IP address. If the attacker is able to guess the users default IP address, then the user's device will respond with an ARP response. Now that the attacker knows the users default IP address, he can send his own ARP requests to the user to get more encrypted ARP responses to help figure out the IV.

iv. Hirte Attack

Hirte attack is a combination of fragmentation and Caffe latte attack, where the attacker responds to the gratuitous ARP packet with fragmented ARP packets. Thus, the user's device responds with more fragmented packets allowing the attacker to figure out the IV.

v. FMS Attack

This attack aims at deriving the WEP shared key. When a weak IV is used to encrypt the frames, the attacker can guess byte n+1, when he has knowledge of the first byte of the keystream and the first n bytes of the key.

vi. KoreK Family of Attacks

This family of attacks is similar to FMS except statistical analysis is performed to vote amongst the probability of different keys.

vii. PTW Attack

PTW attack work in the same way as FMS, but cracks WEP encryption in a more efficient manner by the use Arp injections to speed up the process;

viii. ARP Injection Attack

ARP injection itself is not an attack on the Wi-Fi protocol. ARP injections allow the attackers to speed up the process of WEP encryption cracking like it was done in PTW attack.

ix. Dictionary Attack

This attack is used to target WPA/WPA2 encryptions using the PSK configuration. The sniffs the 4-way handshake, by either observing a new user join the network or deauthenticating the current user and forcing him to rejoin. Once the handshake has been captured, the attacker computes the PMK, PTK, and MIC while parsing through the dictionary. If the passphrase from the dictionary use to calculate the PMK is correct, then the MIC will match the MIC sniffed from the legitimate handshake.

4.6 Experimental Results and Evaluation

- **Experimental setup for collecting the Normal data**

Figure 4.21 shows the testbed used to collect the normal data. The testbed consists a monitoring node that operates in monitor mode and collects the data from the Wi-Fi channels. The data was collected on the second floor of the Electrical and Computer Engineering building at the University of Arizona, that at any given time has around 6-7 access points with 15-200 users using the network depending on the time of the day and the month of data collection. For example, during the semester hours, the network has around 200 users at a given time, while during the break hours the number of users decreases drastically. We collected 3 normal training datasets over the period of the development of this IDS. Table 4.3 shows the characteristics of these 3 datasets. The first dataset was collected in June 2016 (called Dataset1 henceforth) over a period of 9 days, the second dataset was collected over a period of 14 days in August 2017 (called Dataset2 henceforth), the third dataset was collected in November 2018 (called Dataset3 henceforth) over a period of 38 days.

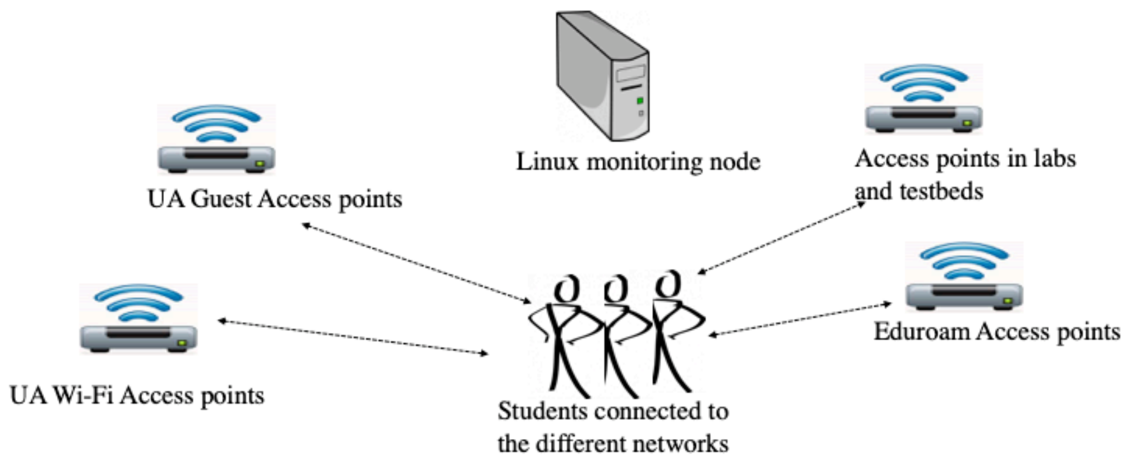


Figure 4.21: Experimental setup for collecting the Normal data

Table 4.3 Characteristics of the normal datasets

Dataset Name	Time of Collection	Total number of frames collected	% of Beacon frames	% of Authentication frames	Percentage of Deauthentication frames
Dataset 1	June 2016	16271211	85.55%	0.006%	0.001%
Dataset 2	August 2017	25810597	89.54%	0.004%	0.002%
Dataset 3	November 2018	64170469	87.52%	0.004%	0.001%

- **Experimental setup for collecting the Attack data**

Figure 4.22 shows the experimental setup used to collect the attack data. The Wi-Fi access point is a Linux machine acting as an access point with using hostapd [83], with Atheros chipset. Table 4.4 shows the list of attacks used to collect the attack traffic. All the attacks that were tested were targeting the availability of the attack network and none of them focused on targeting the encryption protocol. Most of the attacks were executed using Aircrack-ng attack library. The flooding-based DoS attacks were executed using hping3[84] and bash shell commands.

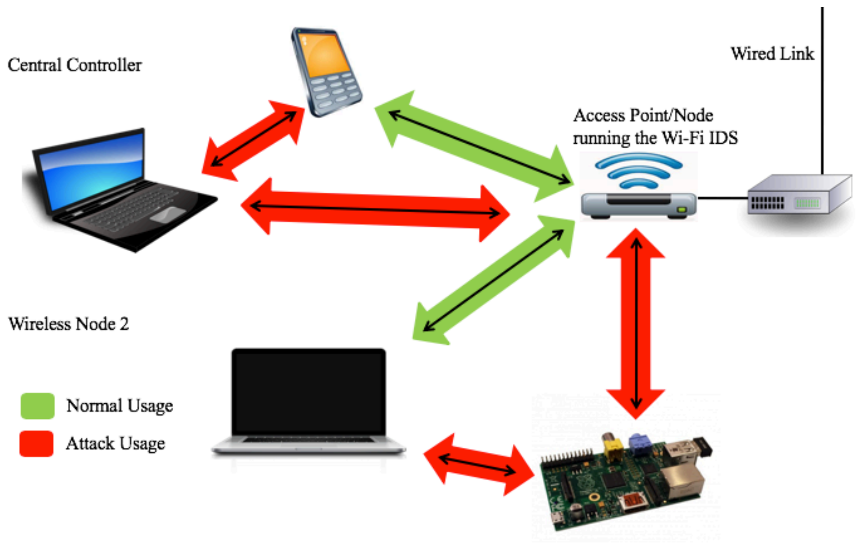


Figure 4.22: Experimental setup for collecting attack data

Table 4.4 List of attacks in the attack dataset

Sr. No.	Attack Name
1.	Deauthentication attack
2.	Fake Authentication
3.	Syn flood
4.	Udp flood

- **Experimental setup for testing the runtime performance of the IDS**

Figure 4.23 shows the experimental setup while testing the runtime operations of the Wi-Fi IDS. The test-bed consists of an access point with multiple devices acting as user devices, and 2-3 different devices acting as attacking devices.

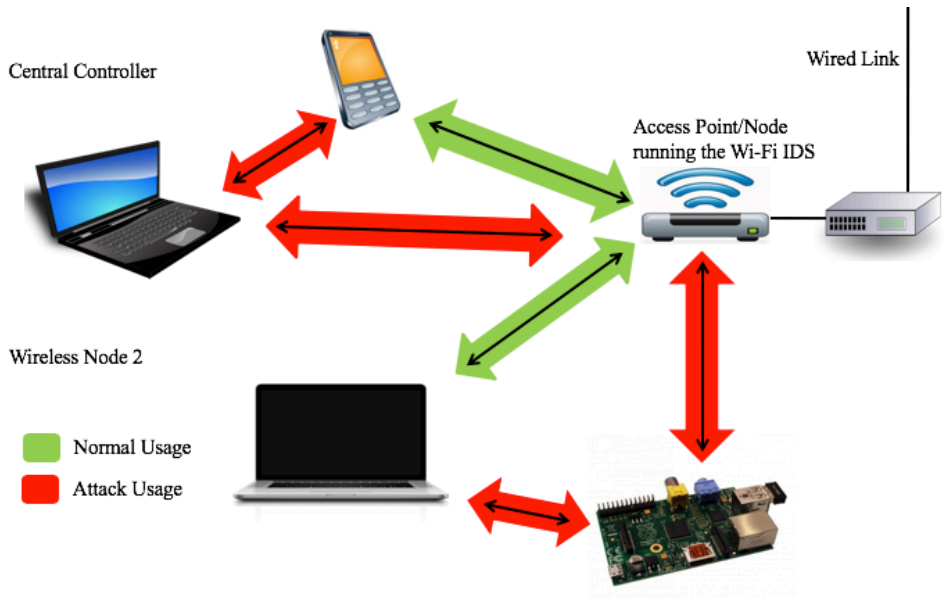


Figure 4.23: Experimental setup for performing runtime tests

- **External dataset to evaluate the approach**

The approach was tested on the AWID [77] family of datasets. We performed detection analysis on AWID-ATK-R, that is an attack database collected on the testbed shown in Figure 4.24, and has attack traffic of 14 different attacks, listed in Table 4.5. Table 4.6 shows the characteristics of the AWID dataset.

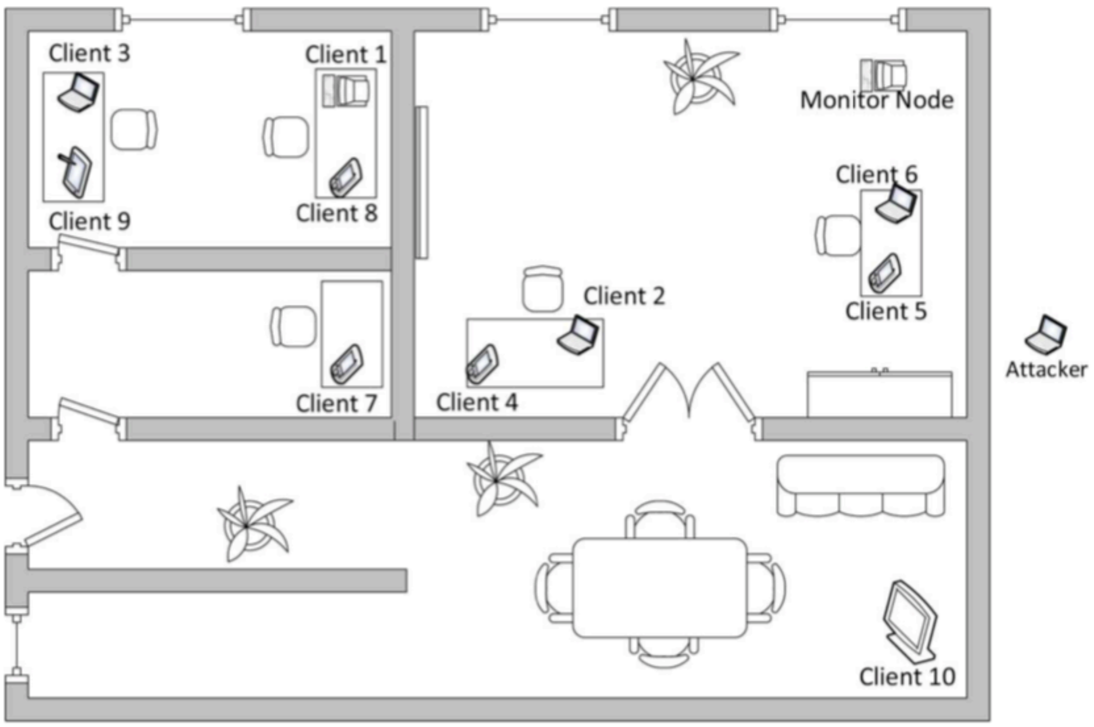


Figure 4.24: AWID dataset data collection environment

Table 4.5 AWID-ATK attacks list

Sr. No.	Attack Name
1.	Amok attack
2.	Arp flood
3.	Beacon flood
4.	Café latte attack
5.	Chopchop attack
6.	Cts attack
7.	Deauthentication attack
8.	Disassociation attack
9.	Evil Twin attack
10.	Fragmentation attack
11.	Hirte attack
12.	Power saving attack
13.	Probe request attack
14.	Rts attack

Table 4.6 Characteristics of AWID dataset

Dataset Name	Time of Collection	Total number of frames collected	% of Beacon frames	% of Authentication frames	Percentage of Deauthentication frames
Awid_atk_r_tst	March 2014	575643	41.83%	0.001%	1.4%

- **Experimental Analysis**

In the remainder of this section, we present the experiments performed on the Wi-Fi IDS.

i. Experiment 1- N-gram size analysis:

This experiment helps answer the design choice made while building the IDS. It answers the question related to the n-gram size, and most importantly it answers the question on how much data is required to obtain a complete modeling of the normal behavior of the Wi-Fi protocol. This experiment was also performed on the Dataset 1,2,3 as shown in Figures 4.25, 4.26, 4.27, 4.28, 4.29, 4.30.

Figures 4.25,4.26, 4.27 show the total number of unique n-grams observed over time for flow sizes of 10 seconds, 60 seconds, and 3600 seconds for n-gram sizes of 1-10. In these three figures we see a similar trend, where there is a steep rise in the number of unique n-grams initially. Then the graph flattens out as a limiting value is reached.

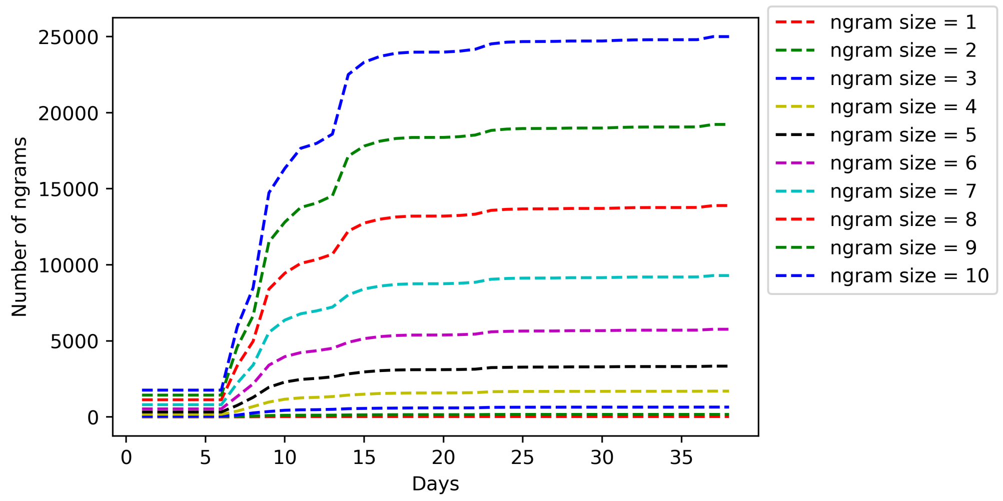


Figure 4.25: Total number of unique n-grams observed over the time for a flow size of 10 seconds

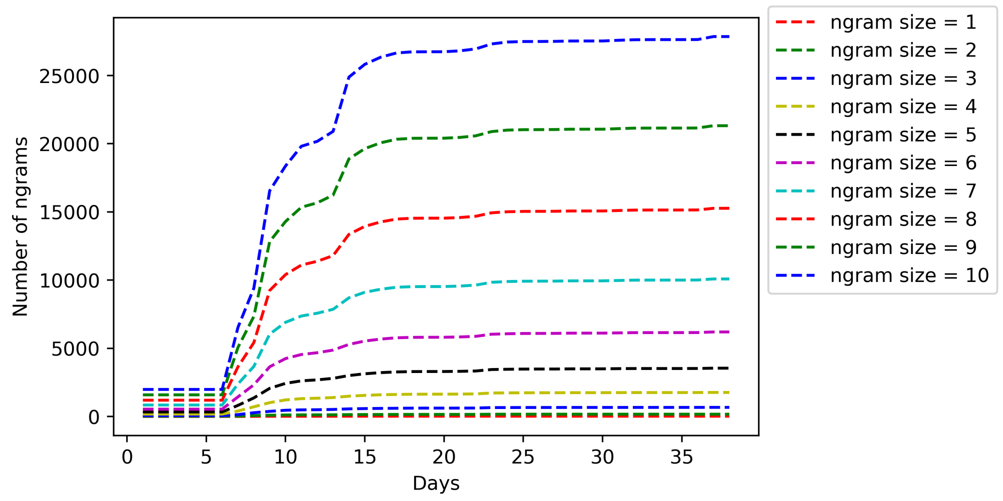


Figure 4.26: Total number of unique n-grams observed over the time interval for a flow size of 60 seconds

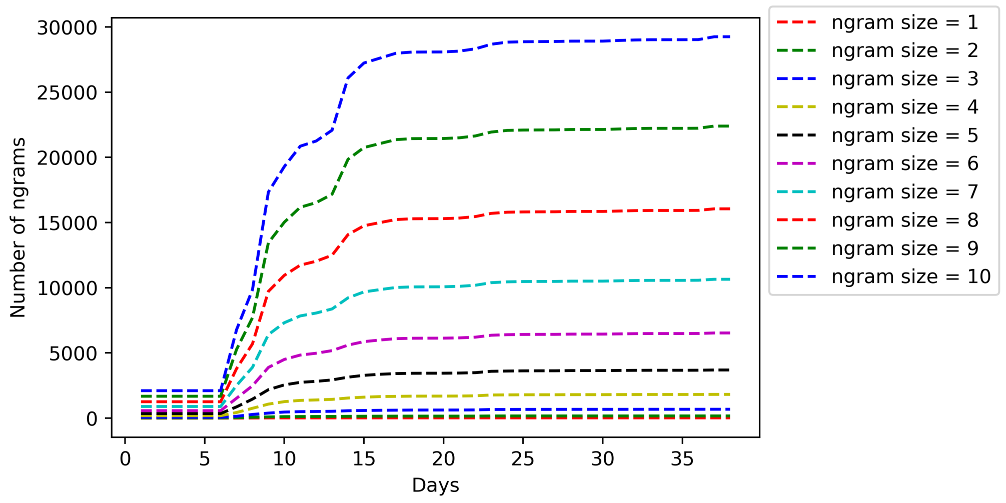


Figure 4.27: Total number of unique n-grams observed over the time interval for a flow size of 3600 seconds

Figures 4.28,4.29, 4.30 show the unique n-grams observed per day over time. We can see that most of the unique new n-grams were collected over a period of 10 days and we conclude that a data collection time of 12-15 days is enough to gain a complete understanding of the normal behavior of the Wi-Fi network.

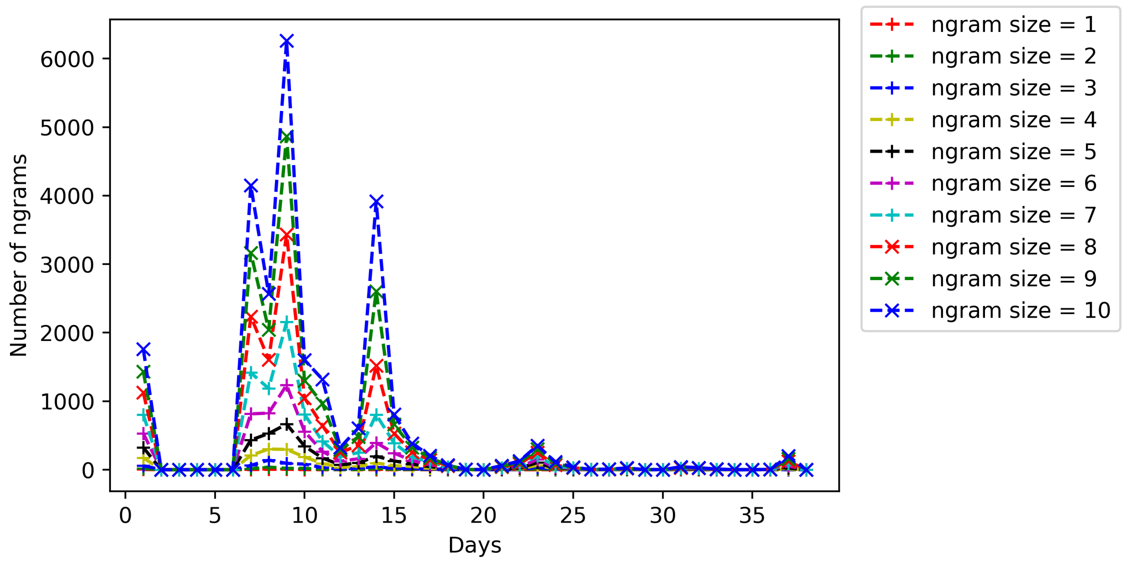


Figure 4.28: Unique n-grams observed per day for a flow size of 10 seconds

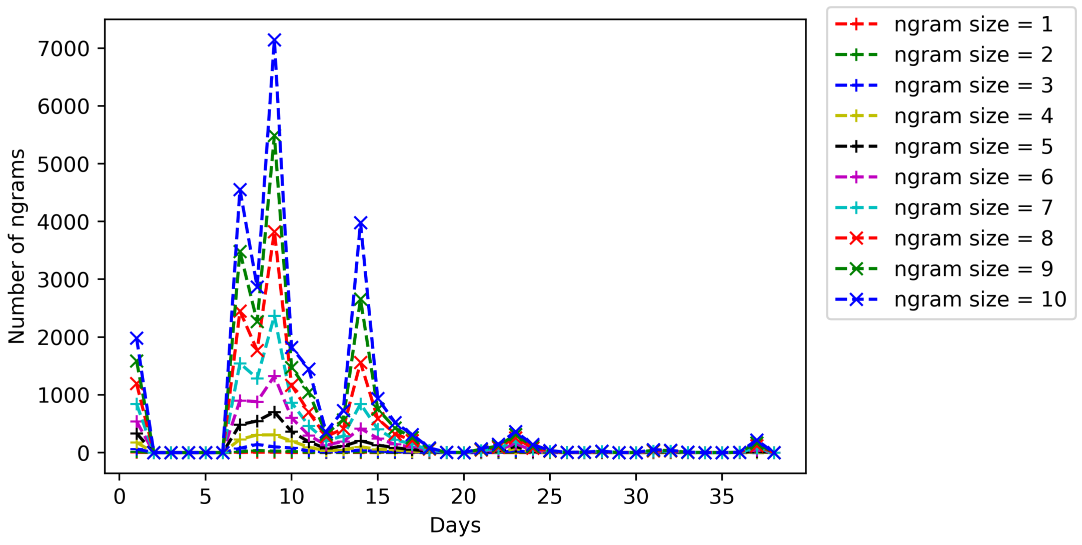


Figure 4.29: Unique n-grams observed per day for a flow size of 60 seconds

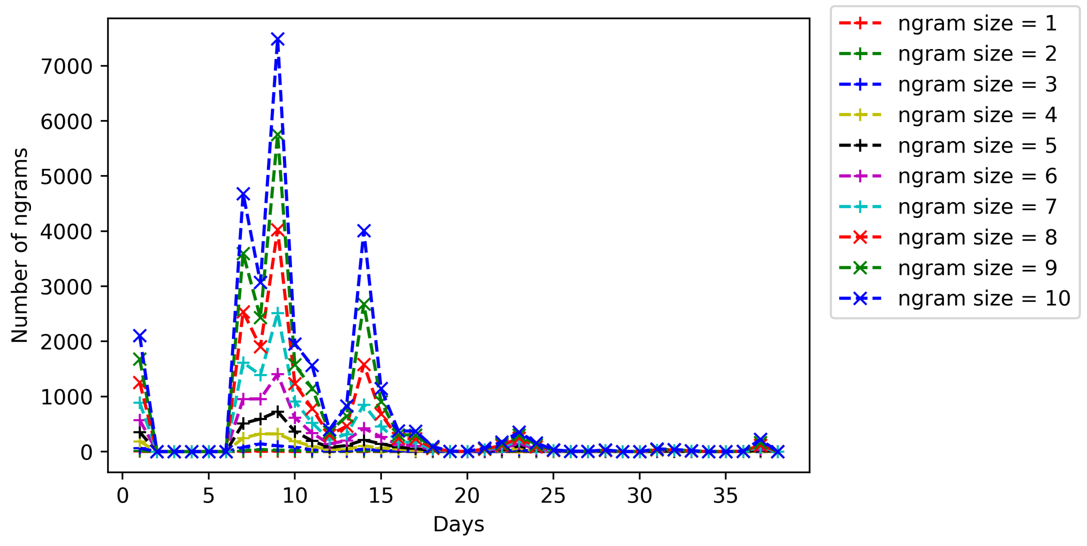


Figure 4.30: Unique n-grams observed per day for a flow size of 3600 seconds

For experimental purposes, we chose an n-gram size of 4, and from here on all the experiments for the Wi-Fi IDS will be performed with an n-gram size of 4.

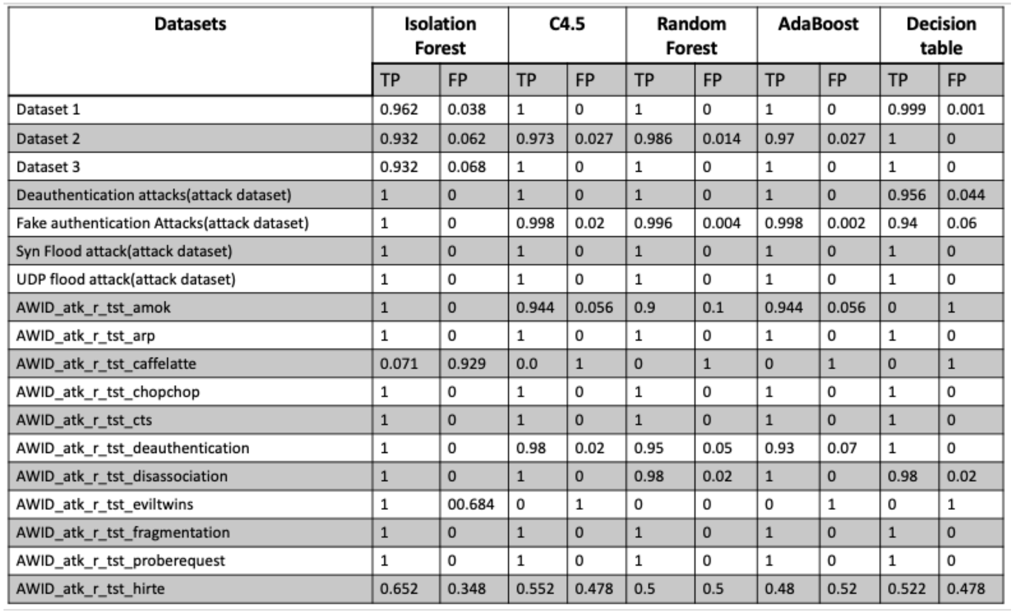
ii. Experiment 2: Performance of the machine learning algorithms

In this experiment, we observed the performance of different machine learning models. A training set was sampled from the dataset3 of 100,000 entries to train the Isolation Forest and 30,000 entries for the other classification algorithms. The classification algorithms need a few malicious entries to work properly and hence the training file with 30,000 entries was appended with 15 abnormal entries from the attack dataset for the deauthentication attack. Table 4.7 shows the performance of these different models. The fake beacon flood attack from AWID dataset was excluded from this test as it does not violate the Wi-Fi protocol, and is an attack on the user of the Wi-Fi networks. From the results, we can observe that the algorithms perform really well when detecting attacks targeting the Wi-Fi protocols operations. The performance for the different algorithms drops when trying to detect caffe latte attack, hirte attack, and the evil twins attack. Caffe Latte attack and the hirte attack target the WEP encryption protocol. As described in section 4.5 Caffe latte and hirte attack, that is a type of Caffe latte attack, are executed in the absence of a legitimate access point. The attacker uses the probes sent out by the user's device to figure out the name (SSID) of the vulnerable AP and sets up a new AP with that name. The unconnected Wi-Fi device used by the user thinks there is a known AP in the vicinity and connects to the attacker's AP. The attacker then proceeds to extract the WEP key using the vulnerabilities in the WEP protocol. The method in which these two attacks are executed, they do not break the functioning of the Wi-Fi protocol till the attacker starts spoofing packets to break the WEP key, justifying the bad performance of the models. To execute the evil twins attack, the attacker sets up an AP with the same name as the legitimate AP and hopes that the users will connect to his AP instead of the legitimate AP. This makes it impossible for the models to detect this attack as the attackers AP is behaving like any other legitimate AP, justifying the performance of the models in table 4.7.

The results for the model obtained from Isolation forest shows that the model is able to detect attacks it has not seen, while giving low false positives when looking at the normal data.

Overall, the high accuracy and the performance of all the models on seen traffic and unseen traffic is attributed to the normal behavior modelling done during the feature selection and extraction process. In our approach the normal behavior modelling happens as a part of the system design, feature choice and the feature extraction process, which can be quantified by the performance of these models.

Table 4.7: Performance of the machine learning models



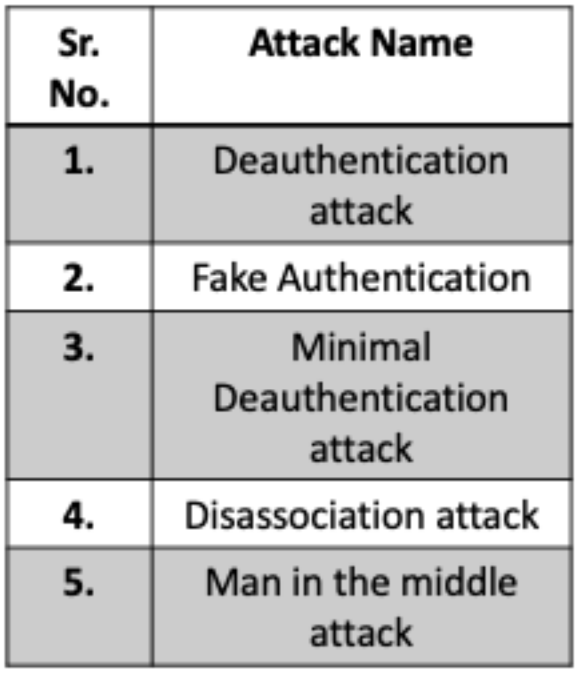
Datasets	Isolation Forest		C4.5		Random Forest		AdaBoost		Decision table	
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
Dataset 1	0.962	0.038	1	0	1	0	1	0	0.999	0.001
Dataset 2	0.932	0.062	0.973	0.027	0.986	0.014	0.97	0.027	1	0
Dataset 3	0.932	0.068	1	0	1	0	1	0	1	0
Deauthentication attacks(attack dataset)	1	0	1	0	1	0	1	0	0.956	0.044
Fake authentication Attacks(attack dataset)	1	0	0.998	0.02	0.996	0.004	0.998	0.002	0.94	0.06
Syn Flood attack(attack dataset)	1	0	1	0	1	0	1	0	1	0
UDP flood attack(attack dataset)	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_amok	1	0	0.944	0.056	0.9	0.1	0.944	0.056	0	1
AWID_atk_r_tst_arp	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_caffelatte	0.071	0.929	0.0	1	0	1	0	1	0	1
AWID_atk_r_tst_chopchop	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_cts	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_deauthentication	1	0	0.98	0.02	0.95	0.05	0.93	0.07	1	0
AWID_atk_r_tst_disassociation	1	0	1	0	0.98	0.02	1	0	0.98	0.02
AWID_atk_r_tst_eviltwins	1	00.684	0	1	0	0	0	1	0	1
AWID_atk_r_tst_fragmentation	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_proberequest	1	0	1	0	1	0	1	0	1	0
AWID_atk_r_tst_hirte	0.652	0.348	0.552	0.478	0.5	0.5	0.48	0.52	0.522	0.478

iii. Experiment 3: Runtime analysis of the IDS:

For the runtime analysis of the approach, a simple conjunctive rule based modelled as describe by Satam et.al in [80] was used. The performance of the IDS was tested with the attacks listed in table 4.6. The IDS was tested over a period of two days and was able to detect all the attacks successfully with no false positives or negatives (on the tested attacks).

The results of this test may seem in contradiction with the results shown in table 4.8, but can be justified by the fact that, attacks can span over multiple flows. Say an attack spans over two flows, and the IDS detects only one of the flows as abnormal and classifies the second flow as normal, the IDS still detected that attack.

Table 4.8: Attacks used to test the runtime performance of the Wi-Fi IDS



Sr. No.	Attack Name
1.	Deauthentication attack
2.	Fake Authentication
3.	Minimal Deauthentication attack
4.	Disassociation attack
5.	Man in the middle attack

iv. Experiment 4: Runtime analysis of the IDS with high interference:

The goal of this experiment was to measure the performance of the IDS when facing high frame drop rates. To perform this experiment, the sniffer module was configured to drop frames in flows following a gaussian distribution. Figure 4.31 and 4.32 shows the performance of these experiments.

Figure 4.31 shows that the intrusion detection system gives zero false positives till 80% frame drop rate. As the frame drop rate increases beyond 80% to 93%, there is a steady increase in the false positives till the system stops performing at 93% frame drop rate.

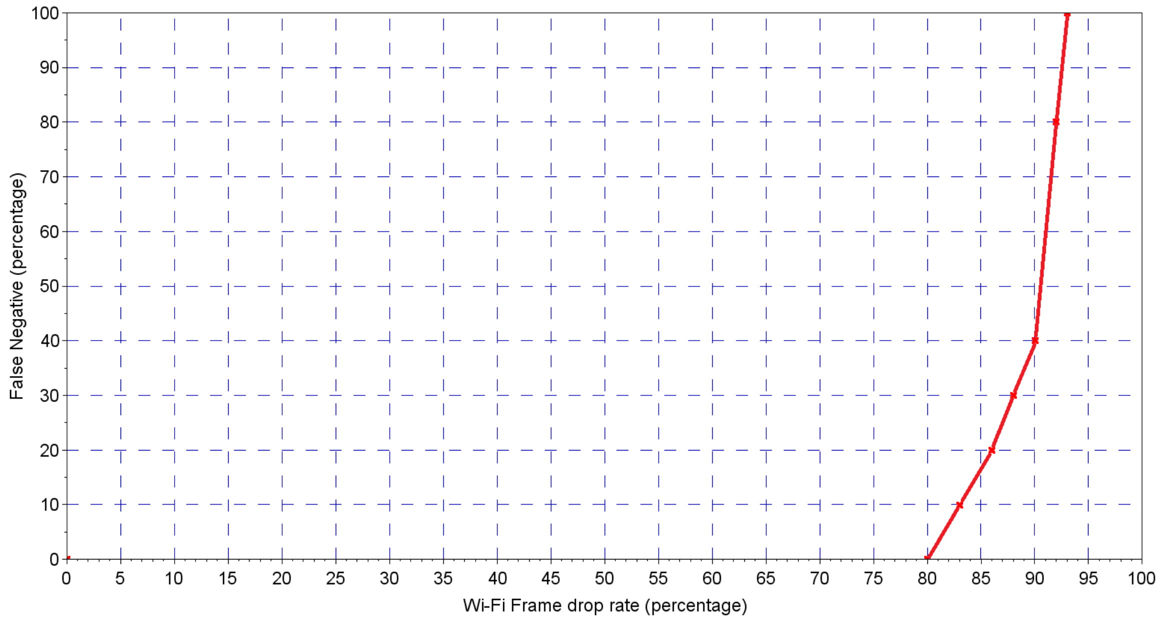


Figure 4.31: False Negative versus Frame drop rate

Figure 4.32 shows that that the algorithm can classify all the attacks correctly at all times till the frame drop rate is as high as 66%. Beyond 66% frame drop rate we observed a steady decline in the algorithms ability to classify the attacks accurately.

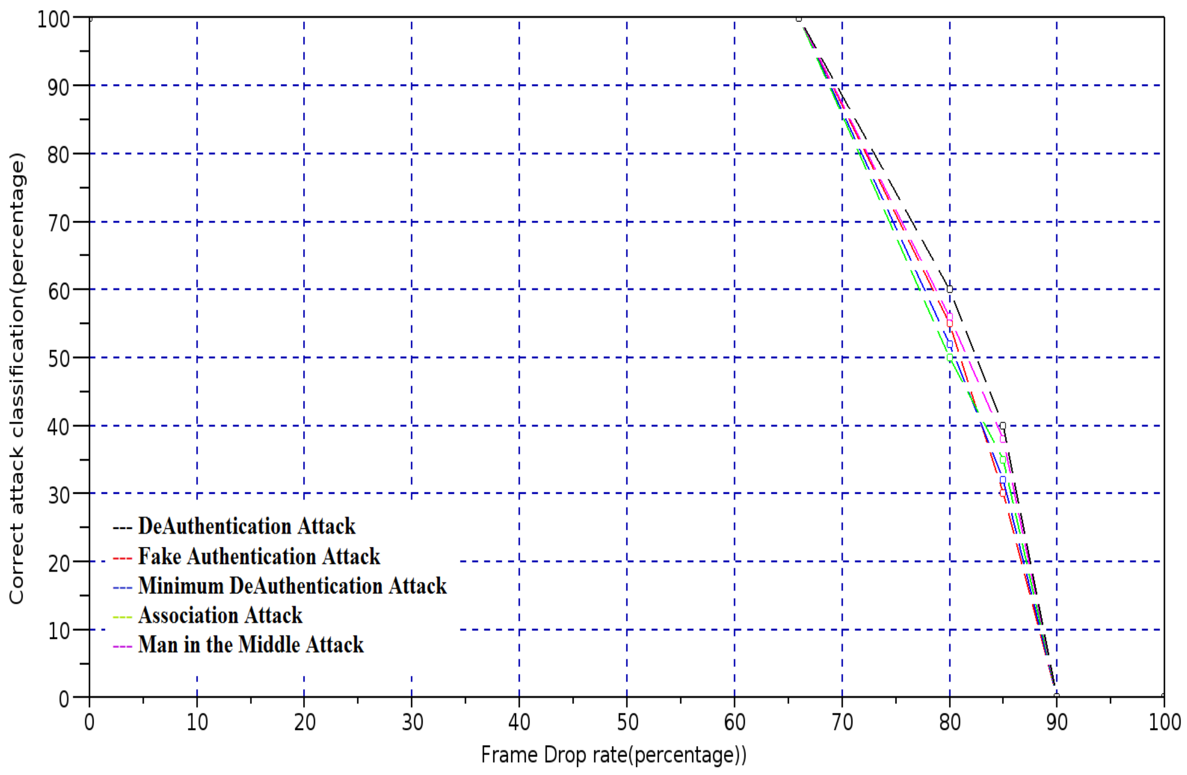


Figure 4.32: Correct Attack Classification (percentage) versus Frame drop rate

Chapter 5: Designing AB-IDS for the DNS Protocol

Domain Name System (DNS) is used to associate a domain name with an IP address. It is one of the most prominently used protocols over the internet, used by for domain name resolution by users during user to user communication and server to server communication. With the current trends in the growth of Internet of Things (IoT), wherein all user appliances ranging from cars to microwave ovens will be connected to the internet, the DNS protocol is set to play an even more important role, making it important to secure it. In this chapter, we apply the IDS design methodology to the DNS protocol.

5.1 DNS Protocol

DNS [85] [86] is a distributed naming system that utilizes a set of hierarchically connected DNS servers while maintaining a client server model. It is used to translate the human readable domain names to IP addresses. The DNS passes the queries hierarchically over a tree like domain space network and the query for a particular domain travels up the network till it reaches the authoritative server for the requested domain. Then the server sends a reply down the tree to the requesting source with a DNS reply. This tree network is subdivided into smaller networks called zones beginning at the root zone. Each zone has at least one authoritative name server that provides authoritative replies for that zone. A zone may contain multiple domains. An authoritative server may sometimes delegate its task of answering queries for a domain in its zone to some other DNS server that is authoritative to a sub zone. Generally, the host of a network has a list of root authoritative servers. These servers are used as a query initiation point. These servers then pass the query up the tree until it reaches the correct authoritative DNS server which then generates a response to the query and sends it back to the local DNS server which in turn sends the response back to the requesting machine’s resolver. Generally depending on the Time to Live (TTL) of the DNS reply and the type of the DNS server involved, the reply that a server receives is cached by the DNS server for future query replies. The TTL of the cached records is reduced with time. This cached record of DNS query-reply is flushed out when the TTL of that particular record reduces to zero. A DNS server can be configured to query recursively. It can also be configured to just forward the queries that it receives. Figure 5.1 gives an example of DNS query sample process. In figure 5.1, the user sends query for the ece.arizona.edu to the next hop DNS server. As the local DNS system does not have the answer cached locally, the query is sent up and down the DNS chain till the authoritative DNS server for the domain, ece.arizona.edu is reached. The authoritative DNS server responses to the query with a DNS answer packet directed to the User.

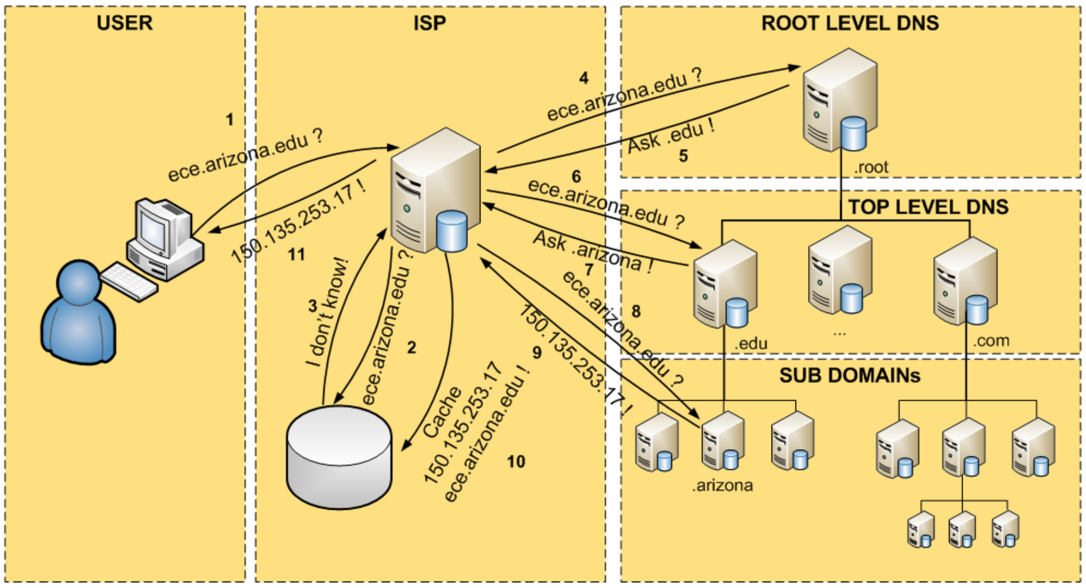


Figure 5.1: A sample Domain Name resolution

Generally, a DNS query is of a particular type (example Type A- 32 bit IP address). A list of the general queries in a network is given in Table 5.1. The reply to a query can be of any size. If the reply to a query is more than the size of the MTU then the reply is fragmented into multiple packets. At the user end, a DNS resolver is responsible for resolving the DNS information for a particular system. A resolver may be implemented by an operating system or locally by a web browser. A resolver is generally configured to cache the replies to the queries that it sends over the network for future use till the TTL for the reply is greater than zero.

Table 5.1: Common DNS message Types

Type	Description
A	A 32-bit IPv4 address, most commonly used to map hostnames to the IP address of the host
AAA	A 128-bit IPv6 address, most commonly used to map hostnames to the IP address of the host
NS	Specifies the authoritative name servers for related zones.
CNAME	Alias of one name to another canonical name. The DNS lookup will continue by retrying the lookup with the new canonical name
MX	Maps a domain name to a list of message exchange agents for that domain.
SOA	Specifies authoritative information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.
TXT	Originally for arbitrary human-readable text in a DNS record.
PTR	Pointer to a canonical name. Unlike a CNAME, in PTR DNS processing does not proceed, just the name is returned. The most common use is for implementing reverse DNS lookups.

Typically, the DNS protocol operates over the UDP. The DNS packet has a header as shown in Figure 5.2.

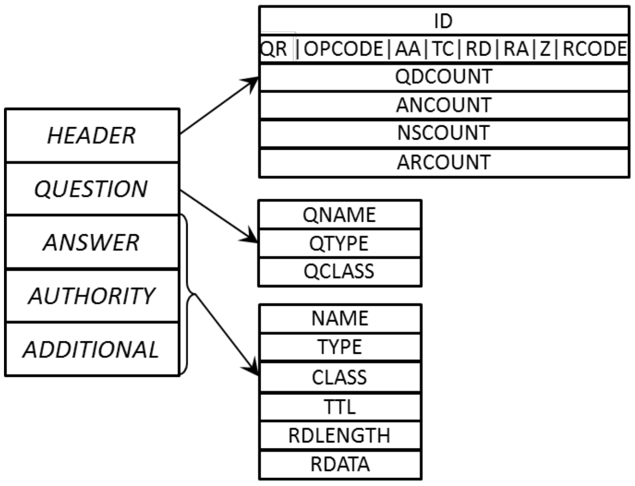


Figure 5.2: DNS message format

5.2 Related Work

In general, there are two main approaches to protect the DNS protocol against cyber-attacks. The first approach is the use of DNSSEC [87] in which the DNS protocol has been significantly changed by adding security features The second approach is to use an intrusion detection system (IDS) to detect attacks that are launched over the network [88][89][90]. The use of DNSSEC to secure the DNS protocol has certain problems like broken validations which give results similar to self DoS [8], and the system generates large number of DNS responses for a small increase in the DNS protocol security [91].

The goal of the IDS methods is to analyze normal and abnormal behavior of the DNS traffic to detect threats against the DNS protocol [92][93][94][95]. Since our approach focuses on detecting attacks against the DNS protocol, we will focus our analysis at different levels of the Domain Name hierarchy during a period of time in order to model the temporal transitions of normal DNS traffic. Other techniques focused on developing statistical models for the DNS traffic without any detection approach. But there are other methods [92][93][95] which have proposed anomaly detection mechanisms for DNS protocol. In what follow, we highlight the main approach used by these techniques.

In [92], normal traffic of DNS protocol is monitored and some statistical thresholds for each protocol parameter are calculated during the training phase. Then during the detection phase, values of these parameters will be compared with the specified thresholds to detect abnormal traffic. This approach will be heavily influenced by the traffic context. For example, considering the time and the location of the deployment of the system the traffic of the system will change. Consequently, it is difficult to come up with a fixed threshold for the system and that results in high false positive alarm rate.

In [93], the authors describe an approach in which they monitor the header information in order to evaluate the statistical properties of various header parameters with respect to an n-dimensional normal distribution. The covariance of the parameters is compared with a threshold value to measure the anomalous behavior in case of an attack. They also perform payload scanning and analysis to detect attacks that are exploiting protocol vulnerabilities. As discussed previously this approach will be affected by the traffic context. Moreover, the value of the threshold that is difficult to determine the optimal value.

In [95] the authors presented an approach in which the flow of DNS traffic between the source and the destination DNS server is used to detect attacks. As a part of their approach, the authors present different models that can detect attacks on the DNS protocol. For instance, to detect cache poisoning attacks they use the flow based approach to separate the queries and requests in a flow and calculate if the count is below a threshold. To detect tunneling attacks using DNS they measure the traffic statistics at standard DNS ports which is indicative of tunneling activity. Also, they present Cross-Entropy Anomaly detection model which they use to detect anomalies in DNS packet sizes. Where they use cross entropy to detect changes in the sizes of packets given the normal distribution of packets. They perform decision fusion of these different models to give one conclusive result of the alerts received from the model.

In [96] the authors present methods to monitor DNS traffic in large networks. They use the concept of standard flow and extended flow to detect DNS attacks on large networks. They present the use of access control lists in combination with the standard knowledge presented in a DNS flow to detect malware infected devices that operate rouge DNS resolvers. They also present a statistical approach to detect changes in networks DNS behavior patterns.

5.3 Using The Methodology To Design AB-IDS for the DNS Protocol

- **DNS state machine**

The proposed system is developed based on the assumption that DNS follows a stateful approach in which generally consecutive DNS queries and replies from a source are dependent on each other and the protocol moves through a well-defined state machine to perform its request and reply messages. The DNS protocol in none of its implementations, keeps track of the protocol state machine. As shown by [47], the DNS state machine can be built dynamically by observing and analyzing the DNS messages as shown in Figure 5.3.

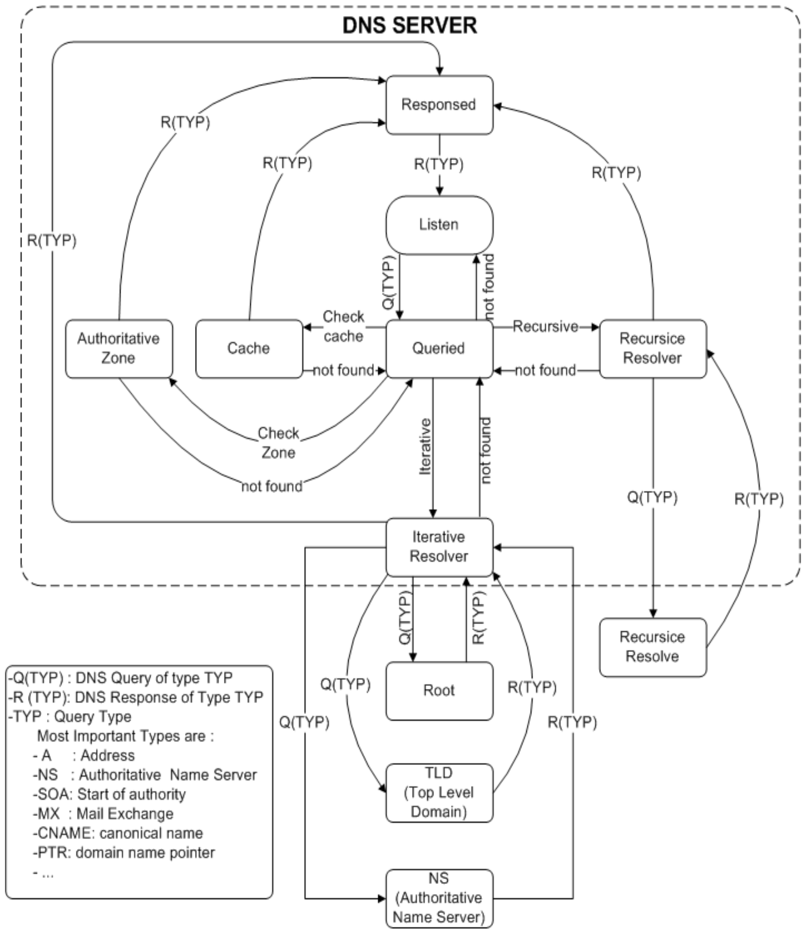


Figure 5.3. DNS protocol state diagram

- **Identification of Attack surfaces, impact and mitigation analysis of the DNS protocol**

As explained in Section 5.1, DNS protocol is a Query response protocol, where the user sends a query for a domain name, and the system responds back with an answer. The user side application that frames the question and interprets the answer is called DNS resolver or resolver. The DNS resolvers use the DNS packet fields QNAME, destination and source port numbers and DNS transaction IDs to match a DNS query to the DNS response. This is the primary attack vector to target the DNS protocol. As the DNS protocol trusts all the responses/answers it receives from the network (as long as the ports and transaction ids match), leaving room for attacker to craft DNS responses to a server of their choice.

- **Feature extraction**

Figure 5.4 demonstrates how the features QNAME (which is used to identify the session), and QType, Type (which identify the state transitions) are extracted from the DNS message, this approach is the same as the approach used by Hamid et.al in his Ph.D. dissertation and Satam et.al in [47].

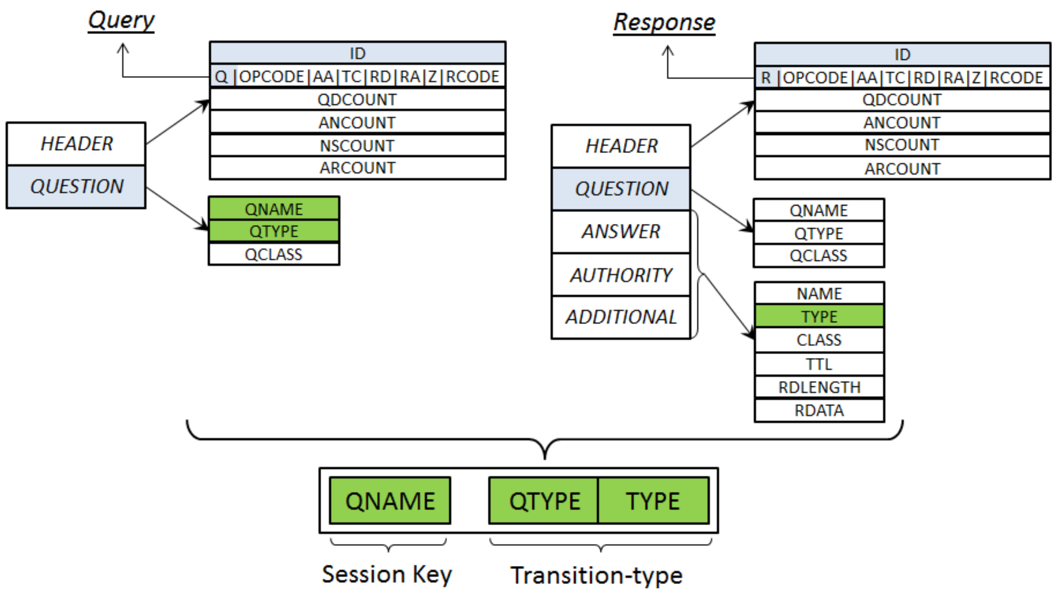


Figure 5.4: DNS message Feature Extraction

i. Session Generation:

DNS sessions are created by sampling DNS traffic in windows of t seconds based on Qname and resolver IP address. Packet with no resolver IP address are treated as a separate session.

ii. n-gram formation

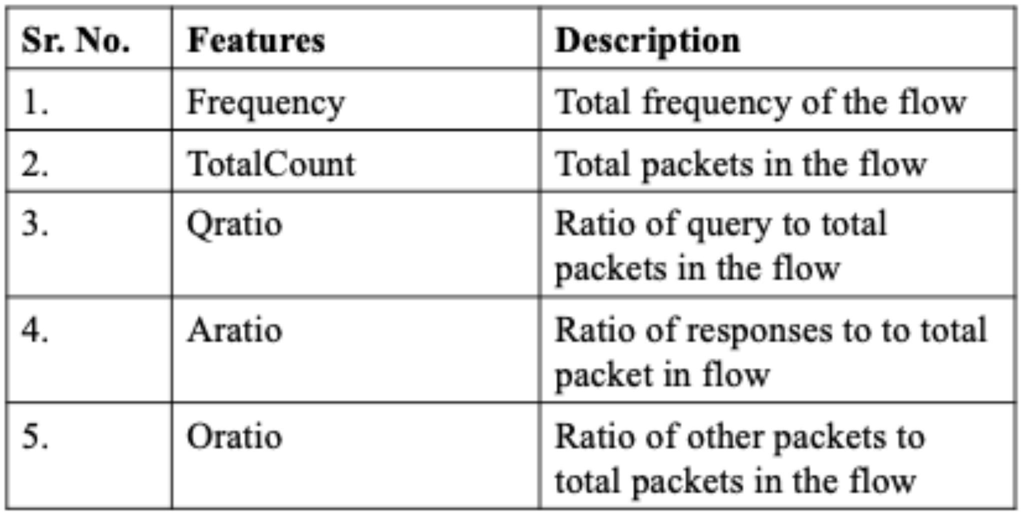
DNS sessions are converted into a flow of n-grams by matching queries to responses based on transaction ids, source and destination ports and IP addresses. The type of the query and the type of response are combined together to get the type of the gram. In case of responses with multiple records, the type of the first record is used to obtain the type of the gram. N-grams are ordered using the query timestamp or response timestamp when query time stamp is unavailable for that particular packet. In case a query receives two separate responses, the first response is combined with the query to get the first gram, while the seconds response is treated as response to a null query to get the second gram.

iii. n-gram counter

During the initial n-gram learning phase, we try to learn all possible normal n-grams, the n-gram counter counts the frequency of each unique n-gram observed.

iv. Features to build the machine learning model

Figure 5.5 shows the feature set used to build the machine learning models. A set of unique n-grams and their frequencies is collected. This set of unique n-grams and their frequencies is used to obtain the values of features shown in figure 5.5. For a DNS flow, the frequency is obtained by adding the observed frequencies of all n-grams in that flow. Totalcount is the total number of packets in the DNS flow. Qratio is the ratio of DNS question packets to the total number of packets in the flow. Aratio is the ratio of DNS answer packets to the total number of packets in the flow. Oratio is the ratio of number of non-question and answer packets to the total number of packets in the flow.



Sr. No.	Features	Description
1.	Frequency	Total frequency of the flow
2.	TotalCount	Total packets in the flow
3.	Qratio	Ratio of query to total packets in the flow
4.	Aratio	Ratio of responses to to total packet in flow
5.	Oratio	Ratio of other packets to total packets in the flow

Figure 5.5 Features used to build the machine learning model

- **Building behavior analysis models using machine learning**

Machine learning algorithms are used to build models based on the feature set shown in Figure 5.5. Figure 5.6 shows the architecture of the DNS IDS what was designed using this approach. The IDS has a packet sniffer module that monitors the network and collects the DNS traffic and stores it in a database. The behavior analysis unit (BAU), converts these packets in flows. The flows are then converted to n-grams and fed to the machine learning algorithm to classify if the event is normal.

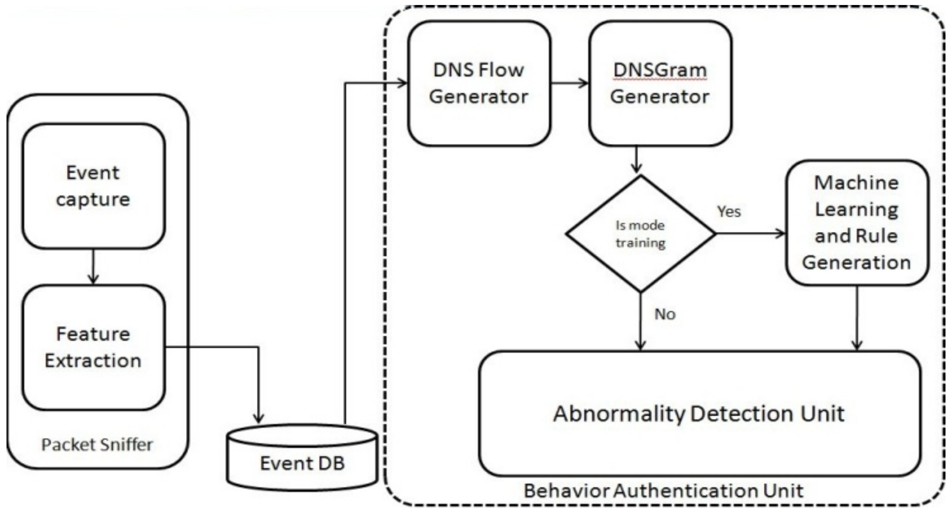


Figure 5.6: System Architecture

5.4 ATTACKS ON THE DNS PROTOCOL

• **Birthday Attack**

This attack uses the Birthday paradox [97] to target the DNS server, by increasing the chances of a spoofed DNS response being accepted by the DNS server. Figure 5.7 highlights the Birthday attack [97]. In the first step, the attacker sends a large number of queries to the nameserver for a particular domain name. In the second step, at the same time, the attacker spoofs responses to the queries he sent nameserver. At some later time, the victim sends a query for the target domain name, to which the Nameserver responds with the fake information he received from the attacker.

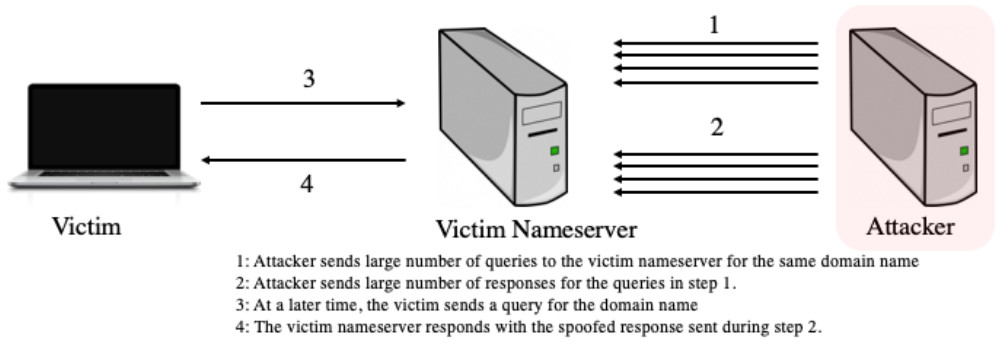

1
2
3
4
Victim
Victim Nameserver
Attacker
1: Attacker sends large number of queries to the victim nameserver for the same domain name
2: Attacker sends large number of responses for the queries in step 1.
3: At a later time, the victim sends a query for the domain name
4: The victim nameserver responds with the spoofed response sent during step 2.

Figure 5.7: Birthday Attack

- **DNS Amplification Attack**

The DNS amplification [98] is a Distributed Denial of Service (DDoS) [99] attack, that exploits open DNS resolvers to overwhelm a target server or network with an amplified amount traffic. The DNS Amplification attack takes advantage of the fact that DNS queries are smaller in size compared to DNS answers. The attacker is able to generate a large amount of traffic in the form of DNS responses for the small number of DNS queries he generates. Figure 5.8 show a DNS amplification attack. The attacker begins the attack by spoofing DNS request with the victim's credentials to open DNS resolvers, which answer the queries with responses to the target of the attack.

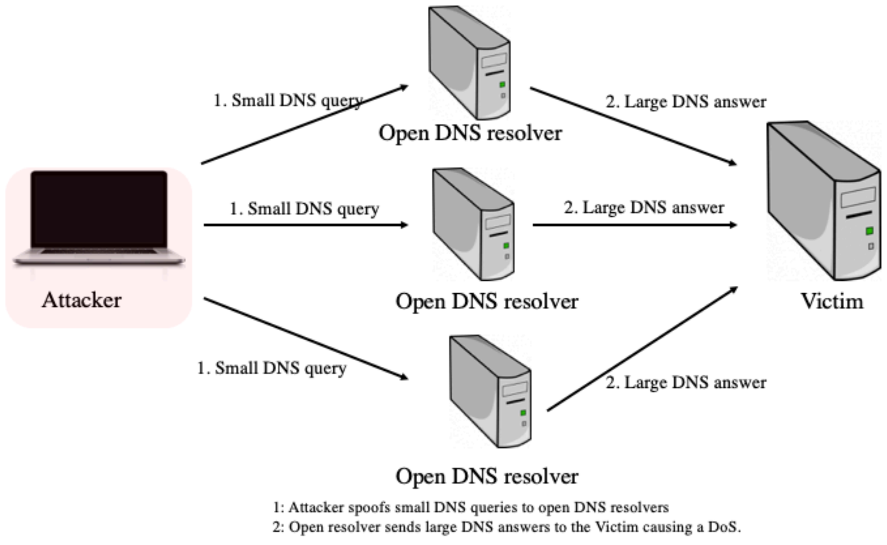


Figure 5.8: DNS Amplification Attack

- **DNS Hijacking Attack**

Figure 5.9 shows the DNS hijacking attack [97]. In DNS hijacking attack, the attacker is a computer on the local network of the victim. As shown in figure 5.9, the attacker sees the victim send a query for a domain name. The attacker responds to the query with his own response which is accepted by the victim as a legitimate response. The legitimate response arrives after the attacker's response and is ignored by the victim.

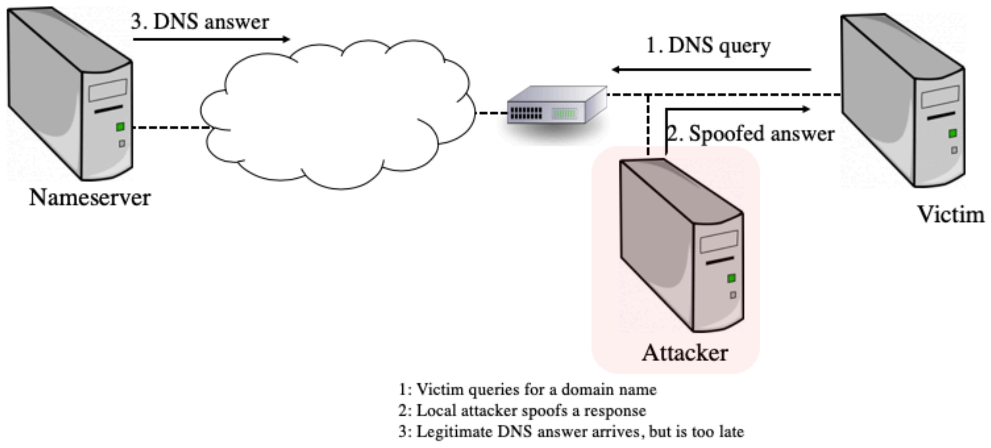

3. DNS answer
1. DNS query
2. Spoofed answer
Nameserver
Victim
Attacker
1: Victim queries for a domain name
2: Local attacker spoofs a response
3: Legitimate DNS answer arrives, but is too late

Figure 5.9: DNS Hijacking Attack

- **DNS Flooding Attack**

DNS flooding attack [97] is DDoS attack, where the attacker floods the DNS servers of a target domain name to disrupt the ability of the DNS server to service legitimate DNS queries. Figure 5.10 shows the DNS flood attack. DNS flood attacks are gaining prominence with the rise of IoT devices as it is easy to infect IoT devices with malware to generate DNS flooding attacks using botnet, Mirai [100] [101] malware and the attack on Dyn in October 2016 [102] being perfect examples.

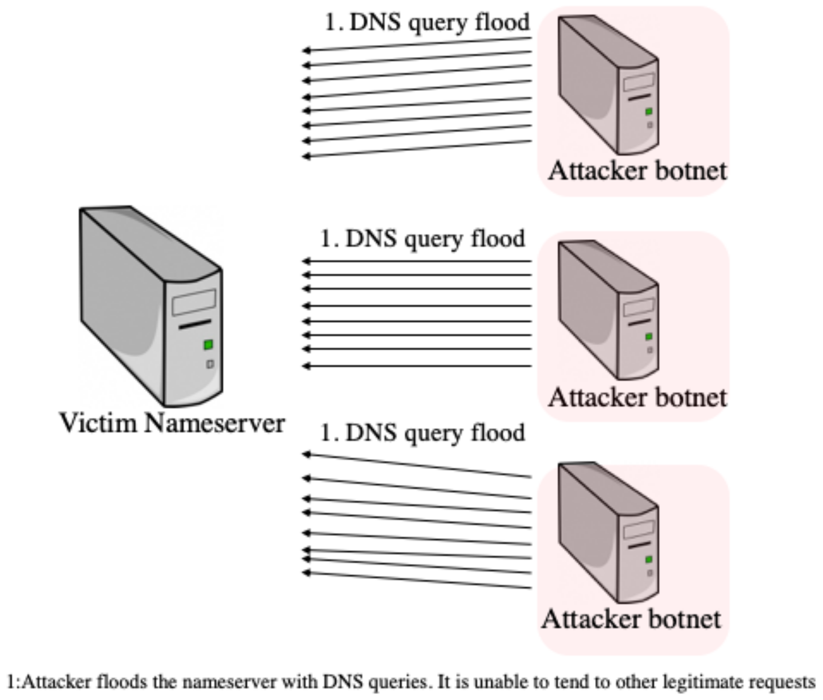


Figure 5.10 DNS Flooding Attack

5.5 Experimental Evaluation

This section presents our experimental evaluation of the approach.

- **DNS Testbed**

The network traffic collection and experimentation of this approach happened on two different testbeds shown in figure 5.11 and 5.12. As the work presented in this chapter is an extension of the work presented by Hamid et.al in his Ph.D. dissertation, collected traffic, and some analysis performed in that work were reused while extending this work.

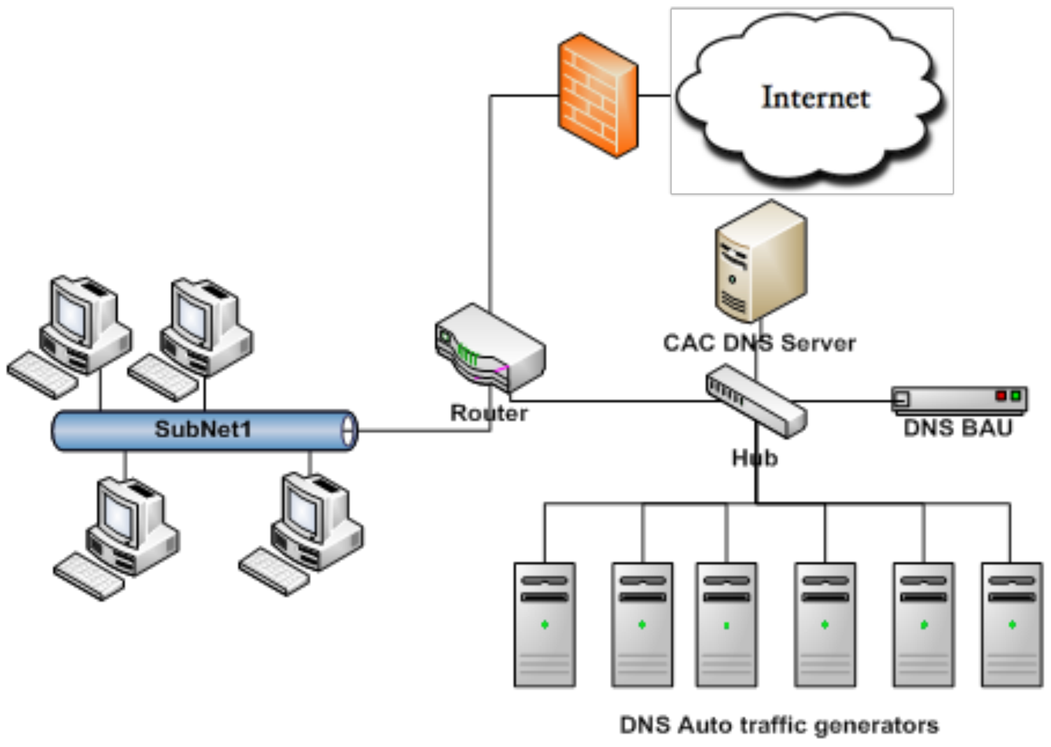


Figure 5.11: Test Bed 1 topology for UACAC-DNS dataset

Figure 5.11 shows the DNS testbed used to collect traffic by Hamid et.al in his Ph.D. thesis. This dataset consists of real network traffic traces combined with auto generated traffic that is collected by monitoring the local recursive name server at the NSF center for Cloud and Autonomic Computing (NSFCAC). This dataset was used for obtaining the set of unique n-grams and getting the n-gram frequencies. Table 5.2. and 5.3 highlight some characteristics of the dataset collected on the testbed in Figure 5.11 over a month. As shown in Figure 5.11, 6 Linux machines were used to generate normal traffic in addition to the traffic generated by the users of subnet 1. The traffic generators used Perl scripts, to mimic real traffic by visiting the top 10 domains of that time. As this traffic was collected on a local network secured by a firewall, this traffic is considered to be normal.

Table 5.2 Number of queries and replies in normal dataset 1

Week	DNS-CAC		
	Queries	Replies	Names
1st	3460433	3481879	74077
2nd	2912931	2866395	74407
3rd	3534062	3467415	70344
4th	1266596	1241167	37562
total	11174022	11056856	88269

Table 5.3 Percentage of message types in normal dataset 1

Message Type	UACAC-DNS
A	41.23%
AAAA	16.12%
ANY	3.30%
CNAME	9.27%
MX	5.24%
NS	6.7%
OPT	0.74%
PTR	-
SOA	16.27%
Other	1.13%

The second testbed shown in Figure 5.12 consisted DNS traces obtained from normal internet usage of members of NSF center for Cloud and Autonomic Computing (NSFCAC).

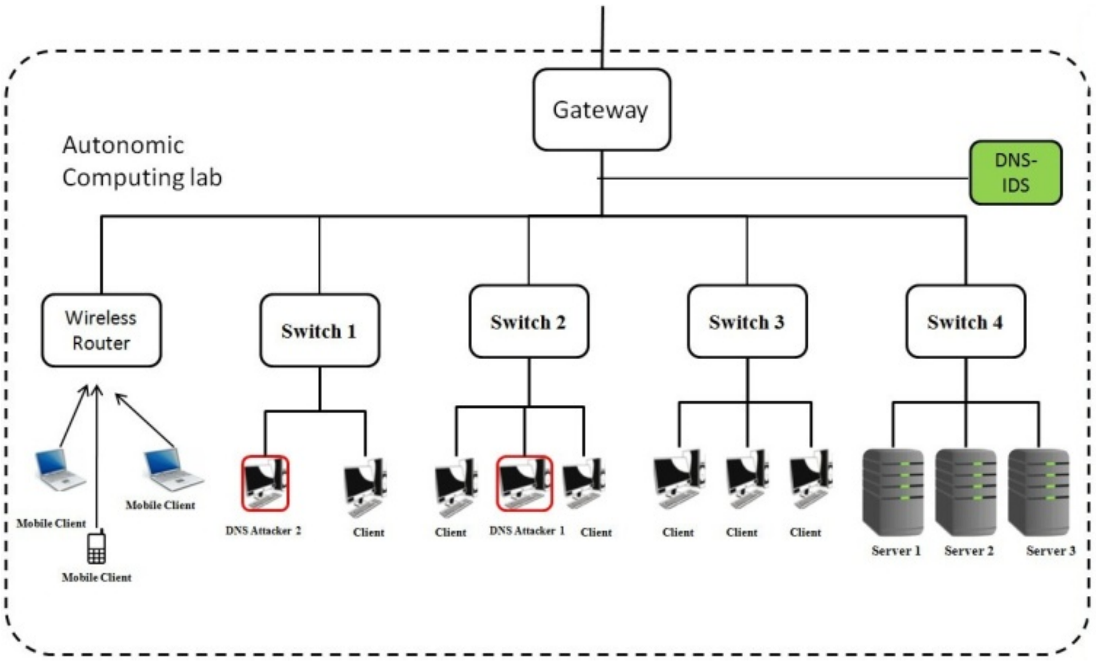


Figure 5.12: Test Bed

The network on an average consisted of 16 desktop computers, a number of mobile devices connected to the local wireless local area network, and the traffic from the center's private cloud severs (at any time, at least 20 virtual machines are connected to the network). Tables 5.4 and 5.5 show characteristics of the dataset collected from this testbed. This dataset was used to extend work presented in [47] to include machine learning algorithms for detection of normal and abnormal behavior.

Table 5.4 Number of queries and replies in normal dataset 2

DNS-CAC dataset 2		
Total packets	**Queries**	**Replies**
721355	253422	467933

Table 5.5 Percentage of message types in normal dataset 2

Message Type	DNS-CAC dataset 2
A	96.78%
AAA	0.32%
CNAME	0.77%
MX	-
OPT	0.76%
SOA	1.35%

- **Attack Dataset**

The attack dataset was collected by executing the attacks mentioned in section 5.4 on different physical and virtual machines in the NSFCAC network. Table 5.6 shows some characteristics of the attack dataset.

Table 5.6 Number of queries and replies in attack dataset

DNS-CAC Attack dataset		
Total packets	Queries	Replies
708401	397698	310703

- **Experiment 1- N-gram size analysis**

This experiment was performed by Hamid et.al as a part of his Ph.D. dissertation and is being presented here to ensure completeness of the experimental analysis. This experiment helps answer the question, “How much training is required for the DNS protocol?” and justifies picking of a n-gram size.

For all experiments pertaining to the DNS protocol here on, the flow or session size will be 10 seconds.

Figure 5.13 presents the total number of unique n-grams observed in comparison to total extracted n-grams, where we can see that for different n-gram sizes, there is a limiting value of total number of unique n-grams seen (that is less the total number of permutations for that n-gram size).

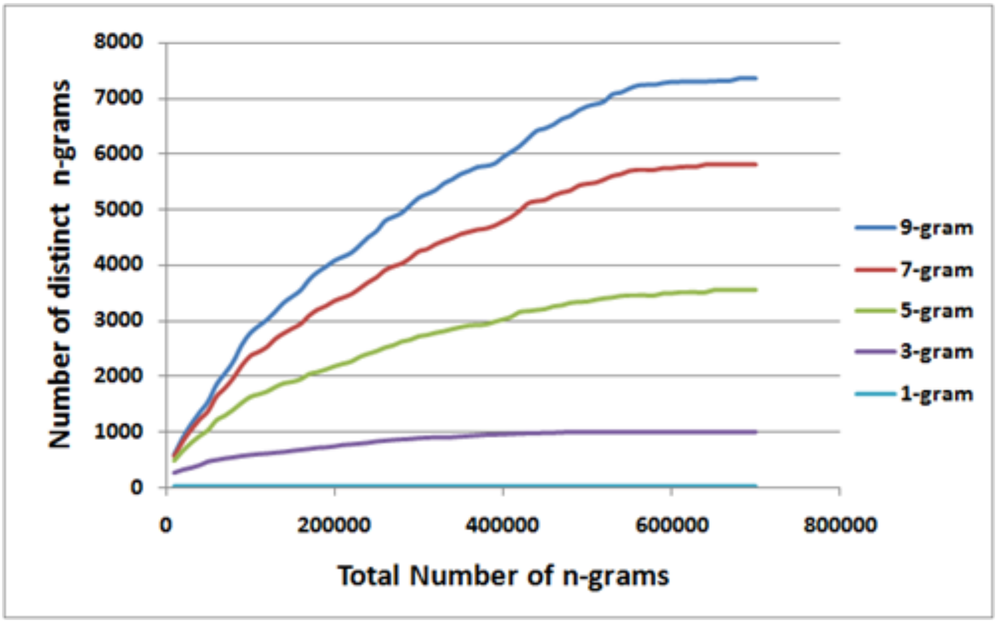


Figure 5.13 Total number of unique n-grams observed compared to total extracted n-grams

Figure 5.14 presents the ratio of new distinct n-grams seen over a week. From the graph in figure 5.14 we can observe that for the lower n-gram sizes (1-3ngrams), most of the new n-grams observed on the first day of training, corelating with the graph in figure 5.13.

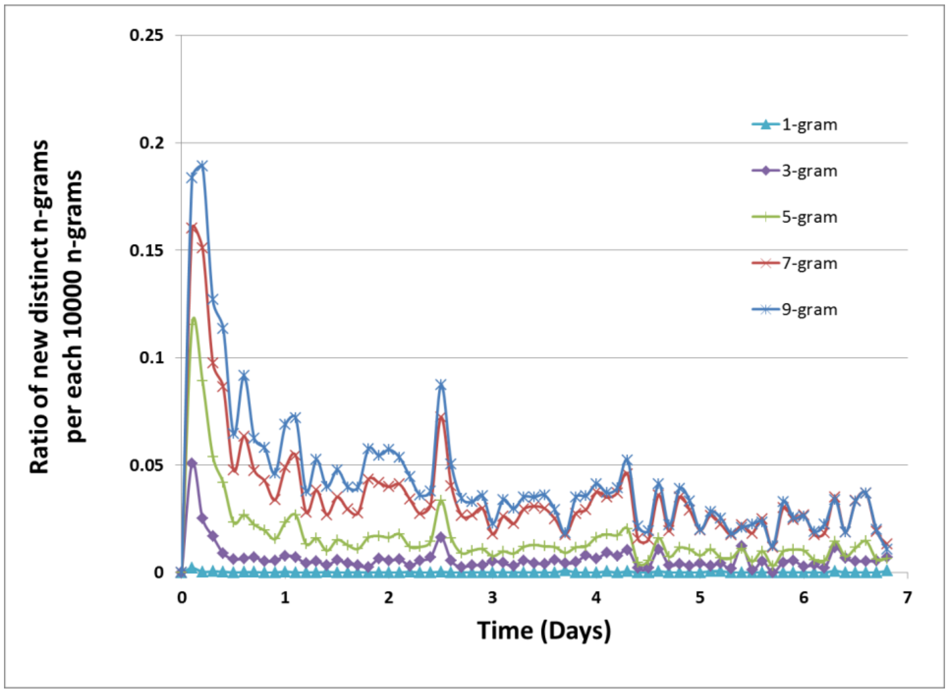


Figure 5.14: Ratio of new distinct n-grams seen over a week

- **Experiment 2- Measurement of performance of the models**

In this experiment, we measure the performance of the machine learning models built during the training phase. We decided to make an n-gram choice of 5 n-grams for this analysis. Table 5.7 shows the results of this experiment. The performance was measured using 10-fold cross validation.

Table 5.7 Performance of the machine learning models

Algorithm	TP rate	FP rate	Precision	Recall	Class
Ripper	0.97	0.00	0.980	0.972	Abnormal
	1.00	0.028	1.00	1.00	Normal
C4.5	0.965	0.00	0.978	0.965	Abnormal
	1.00	0.035	1.00	1.00	Normal
Hoeffding tree	0.870	0.00	0.988	0.87	Abnormal
	1.00	0.130	1.00	1.00	Normal
Random Forest	0.972	0.00	0.989	0.972	Abnormal
	1.00	0.028	1.00	1.00	Normal
Random tree	0.980	0.00	0.980	0.980	Abnormal
	1.00	0.020	1.00	1.00	Normal
AdaBoostM 1	0.972	0.00	0.980	0.972	Abnormal
	1.00	0.027	1.00	1.00	Normal

- **Experiment 3-Birthday Attack**

As a part of this experiment, Birthday attack [97] was launched in the network. In this attack, the attacker sends a large number of queries to the local DNS server followed by a large number of replies to these DNS queries. This attack takes advantage of the birthday paradox to successfully execute the attack. The intensity of the birthday attack was varied from 30 queries to 1000 queries. The attack was successfully detected for all the cases.

- **Experiment 4- DNS Amplification Attack**

In this experiment, DNS amplification attack [98] was launched in the network. The DNS amplification is a DDoS attack in which the objective of the attacker is to flood the system being attacked with replies from various DNS servers, that results into a denial of service attack. The DNS amplification attack intensity varied from 10 queries to 10000 queries. The attack was successfully detected in each case.

- **Experiment 5- DNS Hijacking Attack**

Under this experiment, DNS hijacking attack [97] was launched in the network. In this attack, the attacker listens onto the local network and whenever it sees the local DNS server send out a query, it immediately replies to that query with a predetermined answer. Thus resulting in the domain being hijacked. When the authentic reply arrives, it is ignored by the local DNS server. This attack was performed on the network a number of times and a 100% detection rate for this attack.

- **Experiment 6- DNS Flooding**

As a part of this experiment, a DNS tunneling attack [103] was executed in the network. It was observed that the attack was detected in each case.

- **Experiment 7- Runtime Analysis of the IDS**

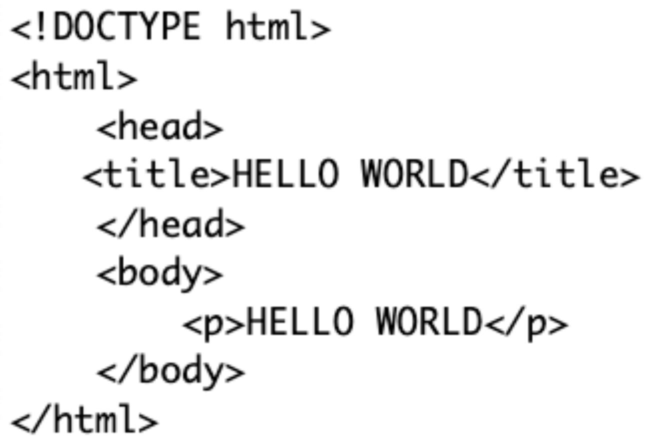
As a part of this experiment, we allowed the IDS to run on the testbed for two days, while performing attacks at random intervals on it. It was observed that the IDS had an attack detection rate of 97% with low false positive alarm rate of 0.01397% and false negatives of 3%.

CHAPTER 6: DESIGNING AB-IDS FOR THE HTML PROTOCOL

The world wide web has grown exponentially over the previous decade in terms number of websites hosted on the world wide web, as well as the number of users accessing these websites. In fact, web usage has become pervasive to touch all aspects of our life, economy and education. These rapid advances have also significantly increased the vulnerabilities of websites. According to White Hat security's "2015 Website Security Statistics Report" more than 86% of all websites have one or more critical vulnerability and the likelihood of information leakage is 56%. In this chapter we apply our AB-IDS methodology to design IDS' to detect malicious HTML files.

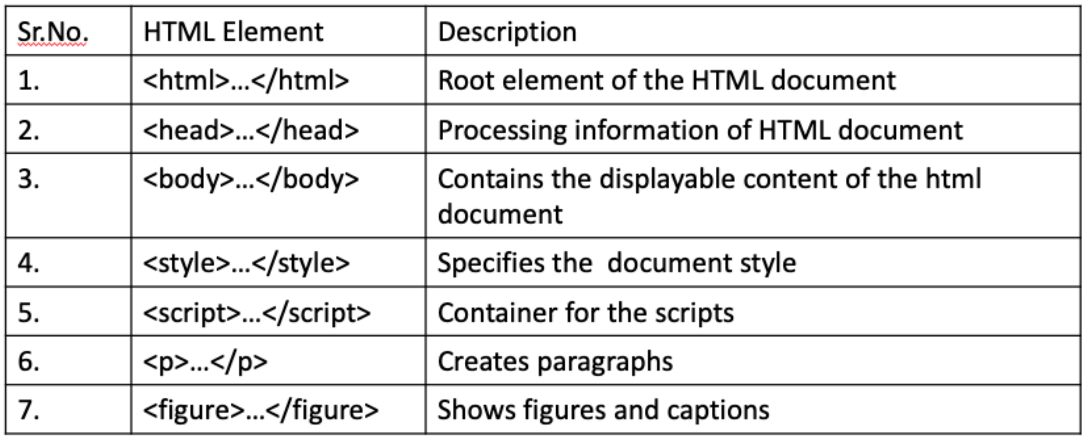
6.1 HTML PROTOCOL

The internet's information services have become pervasive and touch all aspects of our life, economy, education, entertainment, and more. The internet currently hosts more than a billion websites and has an even more in the number of daily users [104]. A wide range of heterogeneous devices (mobile or stationary) access the internet for various functionalities and with the introduction of Internet of Things (IoT), the number is expected to grow to more than 50 billion devices [105]. Most of the content on the internet is hosted on websites which are basically Hyper Text Markup Language (HTML) webpages. Web browsers which are applications to access the world wide web, request web servers for HTML pages when they visit a website. HTML pages allow the content provider to give the access to a variety of content including images, data, files, videos etc. HTML pages are composed of HTML elements, that give it the uncanny flexibility to host a wide variety of data and file types. In an HTML page, multiple HTML elements are organized in a tree structure with the root element always being the 'html element'. An HTML element always begins with a start tag and ends with an end tag. The start tag is indicated by "<elementname>" and the endtag is indicated be "</elementname>". HTML has different elements defined for different datatypes that can be included in the HTML file. Figure 6.1 shows an example HTML file, and figure 6.2 lists popular HTML elements. The web browser receives HTML files like the one shown in Figure 6.1 as a document from the webserver. The HTML file is parsed by the web browser to convert it into a document object model (DOM). The screen rendering of the content is performed on the DOM.



```
<!DOCTYPE html>
<html>
    <head>
   <title>HELLO WORLD</title>
    </head>
    <body>
        <p>HELLO WORLD</p>
    </body>
</html>
```

Figure 6.1: HTML file example



Sr.No.	HTML Element	Description
1.	<html>…</html>	Root element of the HTML document
2.	<head>…</head>	Processing information of HTML document
3.	<body>…</body>	Contains the displayable content of the html document
4.	<style>…</style>	Specifies the document style
5.	<script>…</script>	Container for the scripts
6.	<p>…</p>	Creates paragraphs
7.	<figure>…</figure>	Shows figures and captions

Figure 6.2: Popular HTML elements

6.2 RELATED WORK

Xu et.al in [106] conducted a survey of different techniques to obfuscate malicious javascripts, performing a detailed study of performing obfuscation on malicious javascripts. They have also performed a study by selecting top 20 anti-virus software and observed that they had an average detection rate of 86.5% for non-malicious javascripts, but the performance dropped down to 55.3% when randomization obfuscation was performed, 45.7% when data obfuscation was performed, 0% when encoding obfuscation was performed when different obfuscation techniques were used.

Canfora et.al. in [107] proposed an approach which builds machine learning models on features extracted by performing static and dynamic analysis to detect malicious javascripts. The authors extract features like script execution time, calls to the javascript functions, and number of function calls made by the javascript code via dynamic analysis using Chrome developer. They also extract static analysis features like the number of urls accessed by the javascript from the file. Using these features, they build machine learning models using J48, LADTree, NBTree, Random Forest, Random tree, and RepTree.

Yoo et.al. in [108] present a two phase approach to detect malicious webpages. The first phase involves use of misuse detection to detect known malicious webpages, and depends on a one class svm based model to detect new malicious webpages in phase 2. They report to have a significantly high malicious webpage detection rate at the cost of higher false positives on the normal webpages.

Likarish et.al in [109], presented a classification-based approach to detect malicious javascripts. In [109], the authors used unigram and bigram-based feature extraction approach; similar to natural language text processing, to extract features like %of human readable characters, %whitespace etc. to train models using Naïve Bayes, ADTree, SVMs and RIPPER algorithm. They were able to train a model with a precision of 0.92 and recall of 0.742.

6.3 Attacks on HTML Files

In general, there are four common ways for inserting malicious code into otherwise legitimate HTML files and XML Files:

- **Hidden Iframes**

An Iframe (Inline Frame) is a way of loading one web page inside another, usually from a different server. It can be useful in creating online applications. Malware writers can make the included page hidden by making the Iframe as small as 0 pixels square, give them coordinates that make the frame off the page, or set their property to “hidden”. Then anything can be run inside the Iframe such as JavaScript or a remote malicious web page downloading code. The following example demonstrates hiding an Iframe by making it one-pixel square:

```
<iframesrc="http://www.MaliciousWebsite.com"
width="1" height="1"></iframe>
```

Another interesting example of a hidden Iframe involves redirecting visitors to a website hosting a Java exploits (103.27.108.45) which downloads and decodes a variant of Poison Ivy hosted at: hxxp:img//103.27.108.45//js.php. Notice the iframe height and width is “0”.

```
<div class=”views-field views-field-body”>
<divclass=”field-content”><p>

<iframe                                    height=”0”
src=http://103.27.108.45/img/js.php width=”0”>
</iframe></p>
```

- **Malicious Reference**

Malicious reference is a method used to link one page to a malicious page or to download a malicious file when clicked on. There are several tags that can be modified in either the HTML or CSS file that could cause these actions. The first method could be accomplished using the "meta" tag inside the HTML Head. This would look something like the following for malicious redirects:

```
<meta                              http-equiv="location"
content="url=http://www.MaliciousSite.com" />
```

The second method is the ***<a>*** tag (hyperlink) which would be inserted into the HTML body or CSS file. There are several attributes for the ***<a>*** tag that would be useful for malicious intent. The first would be the ***href*** attribute (link's destination). This will try to open the remote page. The following example will try to open "www.maliciousSite.com" when you click the "Wonderful Website" hyperlink.

```
<ahref="http://www.maliciousSite.com"?>Wonderful
Website</a>
```

The third method also uses the ***<a>*** tag, but with the download attribute. This will cause the filename following the attribute to be downloaded onto the client's computer. The following example will download ***attack.php*** from the remote host and rename it to "***UsefulFile***". Notice that either an ***IP*** or ***URL*** can be used as the location of the file.

```
<ahref=http://213.171.193.5?20/MaliciousAttk/attack.php
download=”UsefulFile”>
```

- **Malicious Scripts**

There are many ways to use ***JavaScript*** for malicious intent. The code in a ***JavaScript*** function is not executed when the function is defined. It is executed when the function is invoked. There are almost unlimited ways to use ***JavaScript*** maliciously and a few of them are in the following list:

1. Read files from local drive
2. Fill up a local drive
3. Access or replace files on the local machine
4. Close or open windows
5. Launch an application
6. If the browser allows, read browser history or cookies
7. Exploit bugs in a browser. If the browser allows, load another document or script from a different domain

- **Abnormal Construction**

This category contains several different malicious activities such as: seldom used tag names, HTML structure inconsistencies, and malicious obfuscation.

6.3 HTML Protocol Analysis

We have used two methods to perform the data analytics: Static Analysis and Dynamic Analysis. Figure 6.3 shows our approach to build the HTML IDS.

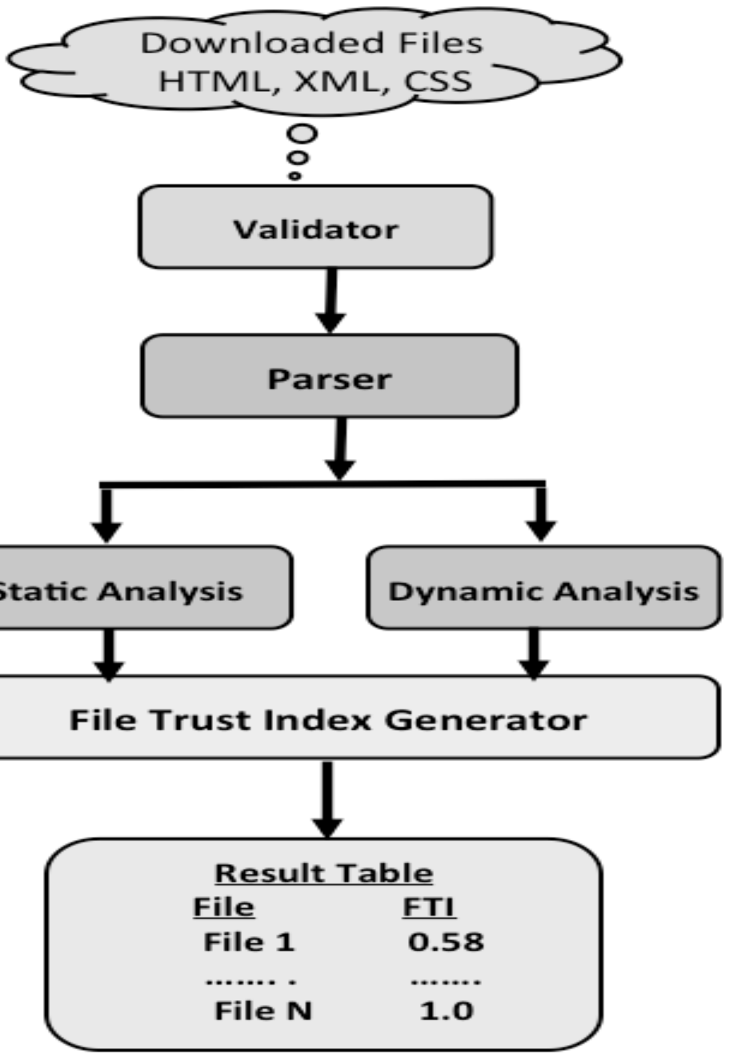


Figure 6.3 HTML IDS architecture

6.2.1 STATIC ANALYSIS OF HTML FILES

Static analysis involves analyzing the file without executing it. This analysis is easy to perform when the source code is available. HTML files are transferred as documents to the client, making them an ideal target for static analysis. In static analysis, the structural features like the number of ‘a’ tags in the HTML file, the Keyword to work ratio etc. help differentiate between the normal HTML files and the malicious HTML files. In the following sections, we present our IDS that performs static analysis of HTML files.

6.2.1.2 USING THE METHODOLOGY TO DESIGN AB-IDS FOR THE HTML PROTOCOL TO PERFORM STATIC ANALYSIS

- **Feature selection methodology**

The literature review and an understanding of the HTML files presented us with a large set of features that could be collected from an HTML file as shown in table 6.1.

Table 6.1: Features List

Feature Name	Associated Category	Feature Type
Minimum Length of 'a' Tags	Malicious References	Discrete
Average Length of 'a' Tags	Malicious References	Discrete
Total External 'a' Tags	Malicious References	Discrete
Total 'a' Tags	Malicious References	Discrete
Total Harmful Links	Malicious References	Discrete
Total Redirects	Malicious References	Discrete
Length of 'script' tags	Malicious Script	Discrete
Length of stings in 'script' tags	Malicious Script	Discrete
Maximum Entropy	Malicious Script	Continuous
Maximum Length of 'a' Tags	Malicious Script	Discrete
Minimum Entropy	Malicious Script	Continuous
Total 'script' Tags	Malicious Script	Discrete
Total External 'script' Tags	Malicious Script	Discrete
Total Entropy	Malicious Script	Continuous
Total Obfuscated HTML Tags	Malicious Script	Discrete
Total DOM Modification functions	Malicious Script	Discrete
Total String Modification Functions	Malicious Script	Discrete
Total Native Functions	Malicious Script	Discrete
Total SetTimeout calls	Malicious Script	Discrete
Ratio of Key Words to Words	Malicious Script	Continuous
Ratio of Whitespace	Malicious Script	Continuous
Total 'iframe' Tags	Malicious Iframe	Discrete
Total External 'iframe' Tags	Malicious Iframe	Discrete
Total Hidden iframes	Malicious Iframe	Discrete
Total 'form' Tags	Miscellaneous Features	Discrete
Total 'object' Tags	Miscellaneous Features	Discrete
Total Characters	Miscellaneous Features	Discrete
Total Event Attachments	Miscellaneous Features	Discrete
Total External 'form' Tags	Miscellaneous Features	Discrete
Total Interaction Events	Miscellaneous Features	Discrete
Total Tags	Miscellaneous Features	Discrete
Unique Tags	Miscellaneous Features	Discrete

We decided to perform feature selection on this feature set to reduce the number of features to be extracted from the HTML file. The feature selection algorithm used as a part of this approach is based on the feature extraction algorithm (FEA) presented by Qu et al.[110]. In this algorithm, information theory is used to identify the most important and relevant features. There are several contributing factors that can be used in information theory to extract relevant features and can be summarized as follows:

i. Entropy

Entropy is a measure of uncertainty of the random variable. It is determined by the equation:

$$H(x) = -\sum_{x \in \psi} p(x) \log p(x) \tag{6.1}$$

ii. Conditional Entropy

Conditional entropy quantifies the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable X is known. Given discrete random variables X with domain X and Y, the conditional entropy of Y given X is defined as:

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x) \tag{6.2}$$

iii. Mutual Information

Mutual information is a measure of the amount of information that one random variable contains about another random variable. Mutual information is used to reduce the uncertainty about one random variable given knowledge of another. The larger the mutual information, the more information the feature can provide. The definition of mutual information can be described as:

$$I(X;Y) = H(X) - H(X|Y) \tag{6.3}$$

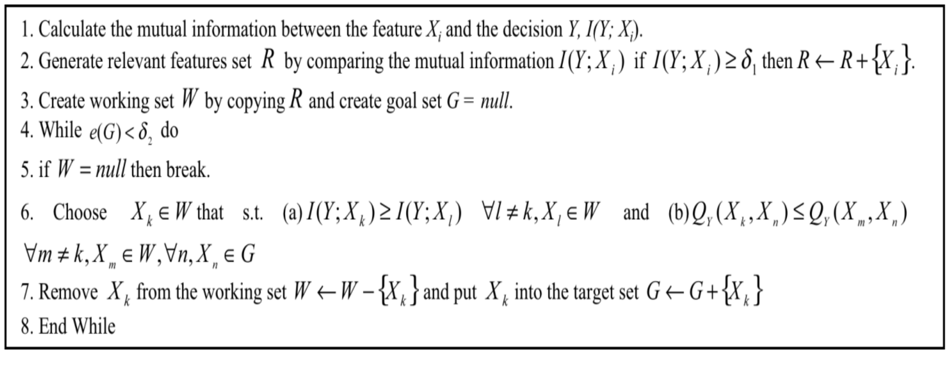
iv. Decision Independent Correlation (DDC)

It is the measure of the redundancy of information between two features X_i and X_j with respect to an analytic objective Y. It can be computed as [110],

$$Q_Y\left(X_{i,}X_j\right) = \frac{I(Y;X_i) + I\left(Y;X_j\right) - I(Y;X_i,X_j)}{H(Y)} \tag{6.4}$$

In the feature selection algorithm shown in Figure 6.4, we use the evaluation factor ***e(s)*** to guide the selection process. This measure specifies a subset in which the mutual information with respect to the decision function is incentivized but decision dependent correlation between the features is penalized.

$$e(S) = \frac{\sum_{\forall j \in I_m} I(Y;X_j)}{H(Y)} - \sum_{\forall i,j\ i \neq j\ i,j \in I_m} Q_Y(X_i\,,X_j) \tag{6.5}$$



```
1. Calculate the mutual information between the feature $X_i$ and the decision $Y$, $I(Y; X_i)$.
2. Generate relevant features set $R$ by comparing the mutual information $I(Y;X_i)$ if $I(Y;X_i) \geq \delta_1$ then $R \leftarrow R + \{X_i\}$.
3. Create working set $W$ by copying $R$ and create goal set $G = null$.
4. While $e(G) < \delta_2$ do
5. if $W = null$ then break.
6.  Choose  $X_k \in W$ that  s.t.  (a) $I(Y;X_k) \geq I(Y;X_l)$  $\forall l \neq k, X_l \in W$  and  (b) $Q_Y(X_k, X_n) \leq Q_Y(X_m, X_n)$
    $\forall m \neq k, X_m \in W, \forall n, X_n \in G$
7. Remove $X_k$ from the working set $W \leftarrow W - \{X_k\}$ and put $X_k$ into the target set $G \leftarrow G + \{X_k\}$
8. End While
```

Figure 6.4: Feature selection algorithm.

The FSA algorithm shown in Figure 6.4 consists of two functional modules. The first one is focused on removing irrelevant features. We use a user-defined threshold $\delta_1$to determine which feature is relevant to the final decision (lines 1 and 2). In this part of the algorithm, irrelevant features are removed from the original feature set. The second part focuses on eliminating redundancy from the features to be selected (lines 4 to 8). We quantify a final state criterion as the distance of the subset evaluation metric $e(S)$ from the user-defined threshold δ_2 (line 5). For each pass, the feature X_k is chosen which satisfies two conditions simultaneously. The first one is that feature X_k should be the most relevant one compared with the rest of features in the working set (line 6 (a)). The second one is that feature X_k should have the least correlation with all the features in the goal set G when compared with the other features in the working set W (line 6 (b)).

- **IDS for HTML static analysis**

Figure 6.5 shows the general approach for anomaly analysis of HTML files. Below we will be describing the main modules shown in Figure 6.5:

i. Parser

The parser receives the html file as an input. It is the job of the parser to verify that the input file is an html file. The parser then parses the html file and extracts the data pertaining to the features from the html file. The extracted data is then passed to the validator block.

ii. Validator

The validators job is to verify the data that has been extracted from the parser block is valid and correct and has the correct format.

iii. Classifier

The classifier module utilizes several classification models that are obtained as a result of machine learning on the data collected during the training phase. The classifier classifies the output from the validator as either normal or abnormal. We will be discussing the classifiers in detail in the experimental results section.

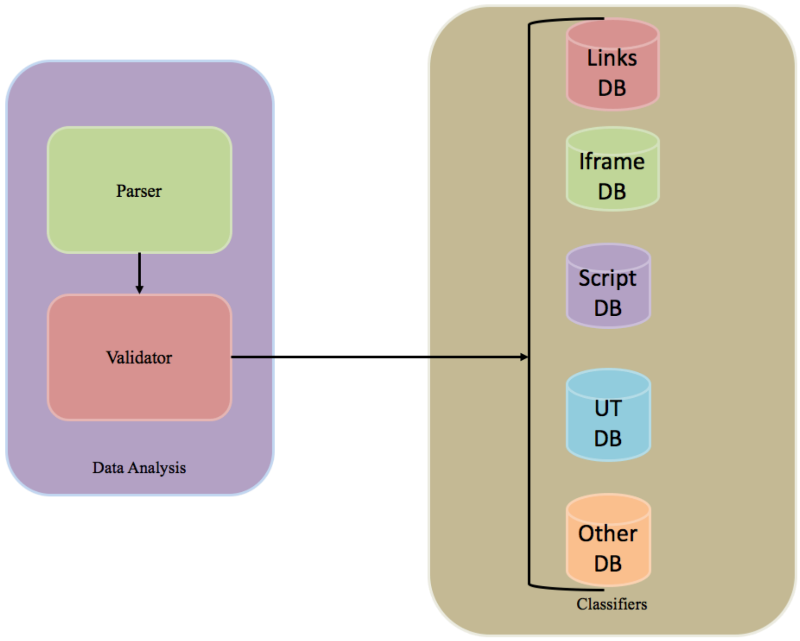


Figure 6.5: General architecture of HTML file static analysis

The IDS operates in two distinct phases: Training phase and Operational phase.

i. Training phase:

In the training phase the html file data is used to develop models to classify the normal structure of the html files. This phase mainly involves, feature extraction, data processing and model development.

ii. Operational phase:

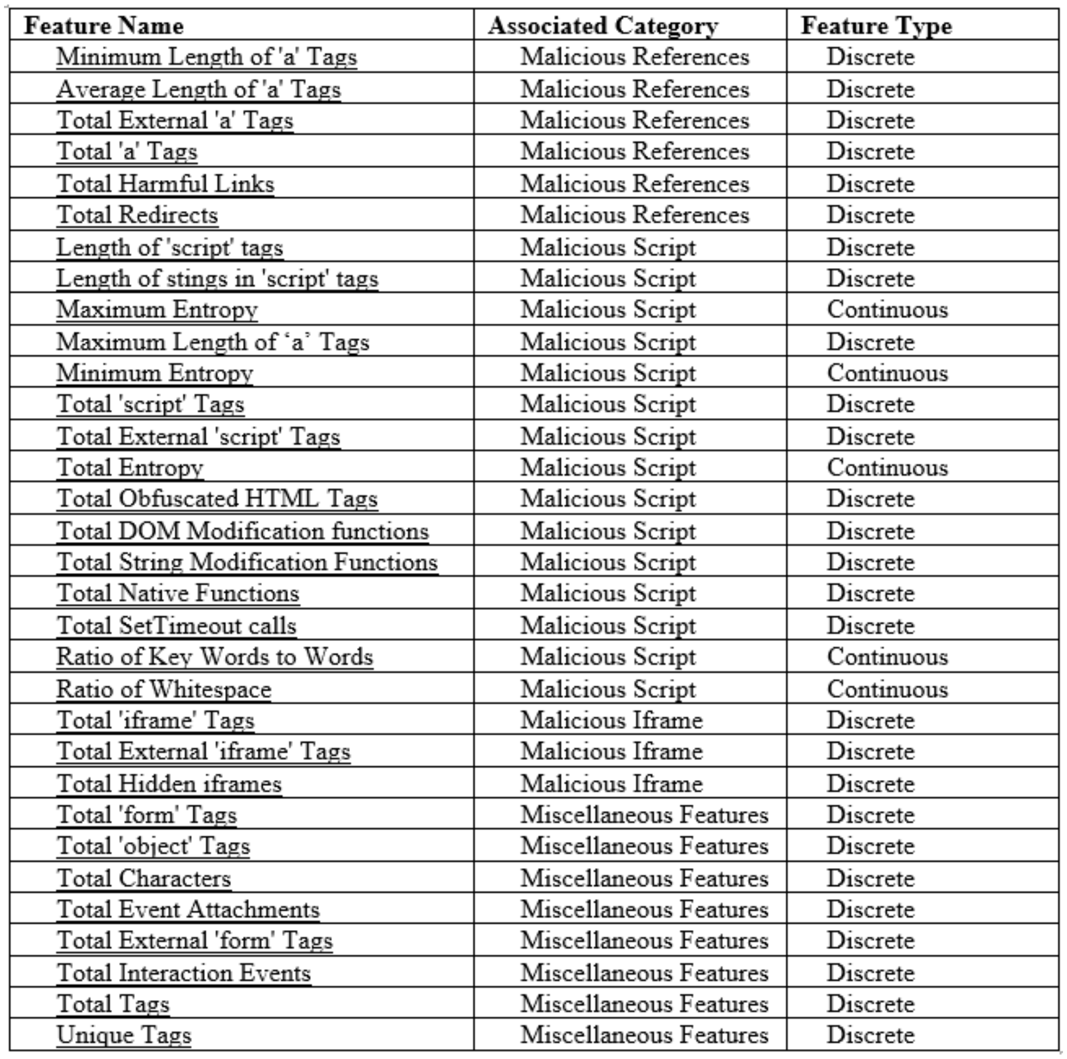
During the operational phase, the models that have been developed during the training phase are used to determine if the html file observed is normal or abnormal. In the next section, we present our experimental results and evaluation of the approach when applied to more than 10,000 HTML files.

6.2.1.4 EXPERIMENTAL EVALUATION AND RESULTS

A. Feature description

Based on literature review and initial pre-analysis of 10,000 normal and 103 malicious files, we identified 32 features shown in Table 6.2. As shown in the table, there are two types of features: Discrete and Continuous features. Discrete features are those that can be represented by a finite set of values. Continuous features have numerical values representing a large numerical range. For each discrete feature, a computable value was assigned. But for continuous features, it is infeasible to use the continuous values in our data analytics and consequently, the continuous parameters are discretized.

Table 6.2: Features List



Feature Name	Associated Category	Feature Type
Minimum Length of 'a' Tags	Malicious References	Discrete
Average Length of 'a' Tags	Malicious References	Discrete
Total External 'a' Tags	Malicious References	Discrete
Total 'a' Tags	Malicious References	Discrete
Total Harmful Links	Malicious References	Discrete
Total Redirects	Malicious References	Discrete
Length of 'script' tags	Malicious Script	Discrete
Length of stings in 'script' tags	Malicious Script	Discrete
Maximum Entropy	Malicious Script	Continuous
Maximum Length of ‘a’ Tags	Malicious Script	Discrete
Minimum Entropy	Malicious Script	Continuous
Total 'script' Tags	Malicious Script	Discrete
Total External 'script' Tags	Malicious Script	Discrete
Total Entropy	Malicious Script	Continuous
Total Obfuscated HTML Tags	Malicious Script	Discrete
Total DOM Modification functions	Malicious Script	Discrete
Total String Modification Functions	Malicious Script	Discrete
Total Native Functions	Malicious Script	Discrete
Total SetTimeout calls	Malicious Script	Discrete
Ratio of Key Words to Words	Malicious Script	Continuous
Ratio of Whitespace	Malicious Script	Continuous
Total 'iframe' Tags	Malicious Iframe	Discrete
Total External 'iframe' Tags	Malicious Iframe	Discrete
Total Hidden iframes	Malicious Iframe	Discrete
Total 'form' Tags	Miscellaneous Features	Discrete
Total 'object' Tags	Miscellaneous Features	Discrete
Total Characters	Miscellaneous Features	Discrete
Total Event Attachments	Miscellaneous Features	Discrete
Total External 'form' Tags	Miscellaneous Features	Discrete
Total Interaction Events	Miscellaneous Features	Discrete
Total Tags	Miscellaneous Features	Discrete
Unique Tags	Miscellaneous Features	Discrete

A set of 13242 normal and 133 abnormal html files were collected from different sources as a part of the training and testing of our approach. Of the 13242 normal html files and 133 abnormal html files, 1000 normal and 30 abnormal files were used in the training phase and the remaining files were used to evaluate the detection rate and accuracy of our approach.

The feature selection algorithm discussed in 6.2.1.2 under feature selection methodology is used on the complete dataset (13242 entry points) in order to reduce the number of features to be analyzed. Table 6.3 shows the mutual information of each of the features with respect to determining whether or not a file being evaluated is normal or abnormal while Table 6.4 shows the reduced feature set obtained after running the FSA algorithm.

Table 6.3: Features with mutual information

umber	Features	Mutual Information
0	url_total_forms	0.01986
1	url_external_forms	0.05322
2	url_total_links	0.60971
3	url_external_links	0.34647
4	url_max_length_links	0.56511
5	url_min_length_links	0.37321
6	url_ave_length_links	0.27798
7	url_unique_tags	0.19475
8	url_total_tags	0.61442
9	url_total_scripts	0.23598
10	url_external_scripts	0.17694
11	url_obfuscated_html	0.25174
12	url_native_functions	0.07446
13	url_set_timeout	0.06745
14	url_total_iframes	0.04181
15	url_hidden_iframes	0.02192
16	url_external_iframes	0.05045
17	url_total_objects	0.00188
18	url_keyword_count	0.33242
19	Keywords to Words Ratio	0.64045
20	White Space Ratio	0.73919
21	url_script_length	0.8289
22	url_string_modification	0.02299
23	url_dom_modification	0.07651
24	url_interaction_events	0.01004
25	url_event_attachments	0.26367
26	url_total_redirects	0.12739
27	url_string_length	0.5767
28	url_max_entropy	0.79029
29	url_min_entropy	0.08365
30	url_total_entropy	0.85154

Table 6.4: Reduced feature set

Number	Features	Mutual Information
2	url_total_links	0.60971
3	url_external_links	0.34647
4	url_max_length_links	0.56511
7	url_unique_tags	0.19475
8	url_total_tags	0.61442
9	url_total_scripts	0.23598
10	url_external_scripts	0.17694
18	url_keyword_count	0.33242
30	url_total_entropy	0.85154

B. Machine learning model training

The training data set (containing 1000 data points) obtained after feature extraction is processed for data discretization. Some of the features used in the characterization of the html file are continuous and hence the values of these features need to be discretized. The discretization is performed by using an unsupervised attribute discretization filter [111][112][113]. The discretized training data set is then normalized by the use of a normalization filter [114][115]. Various classification-based machine learning algorithms were used to obtain the appropriate machine learning models that can be used to classify the normal and abnormal files. The best results were obtained using a classifier that fuses the results of multiple classifiers. The results from the different classification models will be discussed in the experiments section.

i. Support vector machine (SVM) classifier:

A linear support vector machine (svm) based classification model was built using the Spark engine [116]. The weights that were used for the convex function f are shown in Table 6.5. Table 6.5 also lists the accuracy of the classification algorithm on the test case. Where the convex function f is given in Equation 6.6,

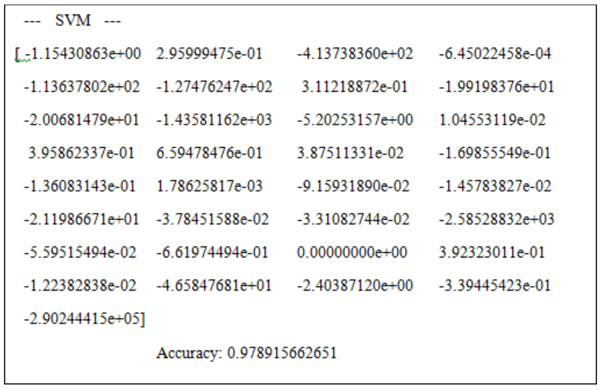
$$\mathrm{f(w)} \coloneqq \lambda \mathrm{R(w)} + \frac{1}{n}\sum_{i=1}^{n} \mathrm{L(w; x_i, y_i)} \qquad (6.6)$$

$$where\ y = result\ and\ x_i is\ the\ vector$$

$$L(w; x, y) \coloneqq \max\{0, 1 - y * w^T x\} \qquad (6.7)$$

$$R(w) \coloneqq \frac{1}{2}||w||_2^2 \qquad (6.8)$$

Table 6.5: Support Vector Machine Results



```
 ---  SVM  ---
[-1.15430863e+00  2.95999475e-01   -4.13738360e+02  -6.45022458e-04
 -1.13637802e+02  -1.27476247e+02   3.11218872e-01  -1.99198376e+01
 -2.00681479e+01  -1.43581162e+03  -5.20253157e+00   1.04553119e-02
  3.95862337e-01   6.59478476e-01   3.87511331e-02  -1.69855549e-01
 -1.36083143e-01   1.78625817e-03  -9.15931890e-02  -1.45783827e-02
 -2.11986671e+01  -3.78451588e-02  -3.31082744e-02  -2.58528832e+03
 -5.59515494e-02  -6.61974494e-01   0.00000000e+00   3.92323011e-01
 -1.22382838e-02  -4.65847681e+01  -2.40387120e+00  -3.39445423e-01
 -2.90244415e+05]
                  Accuracy: 0.978915662651
```

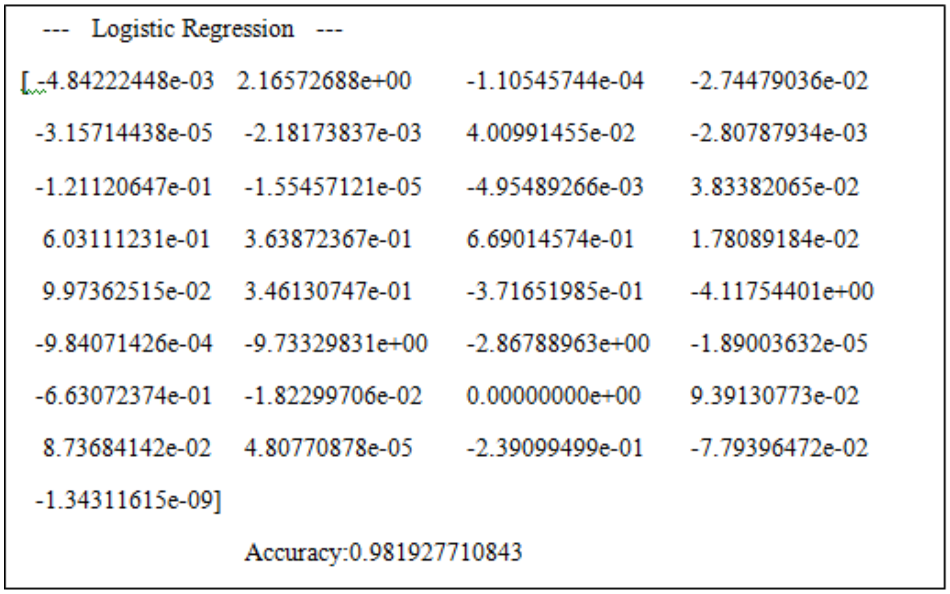
ii. Logistic regression classifier:

The logistic regression-based model was built using logistic regression classifier provided by the spark engine. The weights used by the classifier are presented in Table 6.6. This algorithm performed with an accuracy of 0.9819. The logistic regression classifier used the following equations,

$$f(z) = \frac{1}{1+e^{-z}} \tag{6.9}$$

$$where z = w^T x$$

Table 6.6: Logistic Regression Results



```
 ---  Logistic Regression  ---
[ -4.84222448e-03  2.16572688e+00   -1.10545744e-04   -2.74479036e-02
  -3.15714438e-05  -2.18173837e-03   4.00991455e-02   -2.80787934e-03
  -1.21120647e-01  -1.55457121e-05  -4.95489266e-03    3.83382065e-02
   6.03111231e-01   3.63872367e-01   6.69014574e-01    1.78089184e-02
   9.97362515e-02   3.46130747e-01  -3.71651985e-01   -4.11754401e+00
  -9.84071426e-04  -9.73329831e+00  -2.86788963e+00   -1.89003632e-05
  -6.63072374e-01  -1.82299706e-02   0.00000000e+00    9.39130773e-02
   8.73684142e-02   4.80770878e-05  -2.39099499e-01   -7.79396472e-02
  -1.34311615e-09]
                  Accuracy:0.981927710843
```

iii. C4.5 Classifier:

The java based implementation of the C4.5 algorithm provided by the weka machine learning tool [117][115][118][113][119] called j48 was used to build a c4.5 based classifier model. The best c4.5 based model had a true positive rate (TPR) of 89.3% while a false positive rate (FPR) of 0.004 with respect to classification of abnormal files, further details have been presented in Table 6.7.

iv. Regression classifier:

The java-based implementation of regression classifier provided by the machine learning tool weka was used to build this model. The regression classifier used m5 [8] algorithm to build models for each of the individual classes. This classification model had a true positive rate (TPR) of 43.7% and a false positive rate (FPR) of 0% with respect to the classification of abnormal files, further details have been provided in the table 6.7 below.

v. Bagging Classifier:

The java-based implementation of the bagging classifier provided by the machine learning tool weka was used to build this model. This classification model had a true positive rate (TPR) of 50.5% and a false positive rate (FPR) of 0% with respect to the classification of abnormal files, further details have been provided in Table 6.7.

vi. Fusion-based Data Analytics Algorithm

In our effort to improve the overall accuracy and efficiency of our data analytics algorithms, we have developed a model to fuse the results of different classifiers as shown in Figure 6.6. A combined classification model was built using the classifications results of the C4.5 classifier, regression classifier, and bagging classifier. The fusion of the classifier results was performed in a manner such that the results of each of the classifiers acted as new features to the data set and the resultant data set was classified using a C4.5 based classification model. This combined classification model had a true positive rate (TPR) of 99% and a false positive rate (FPR) of 0.8% with respect to the classification of abnormal files, further details have been provided in Table 6.7.

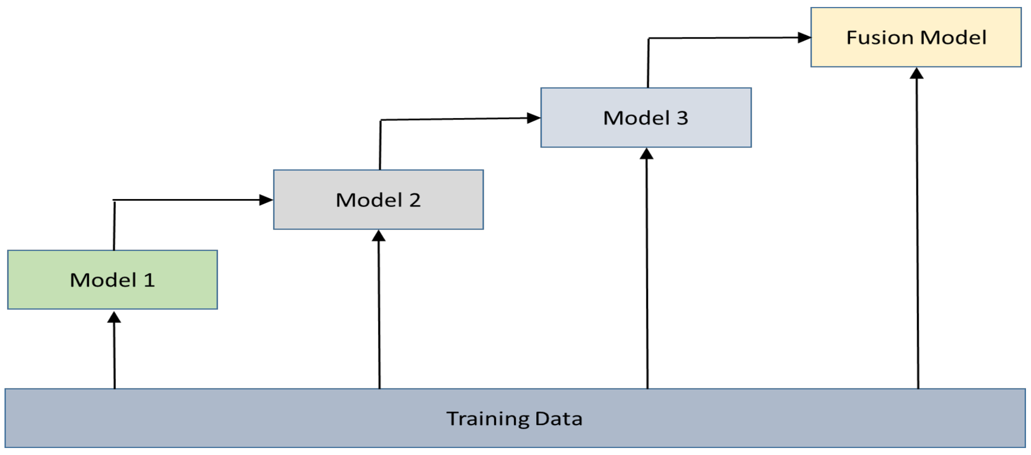
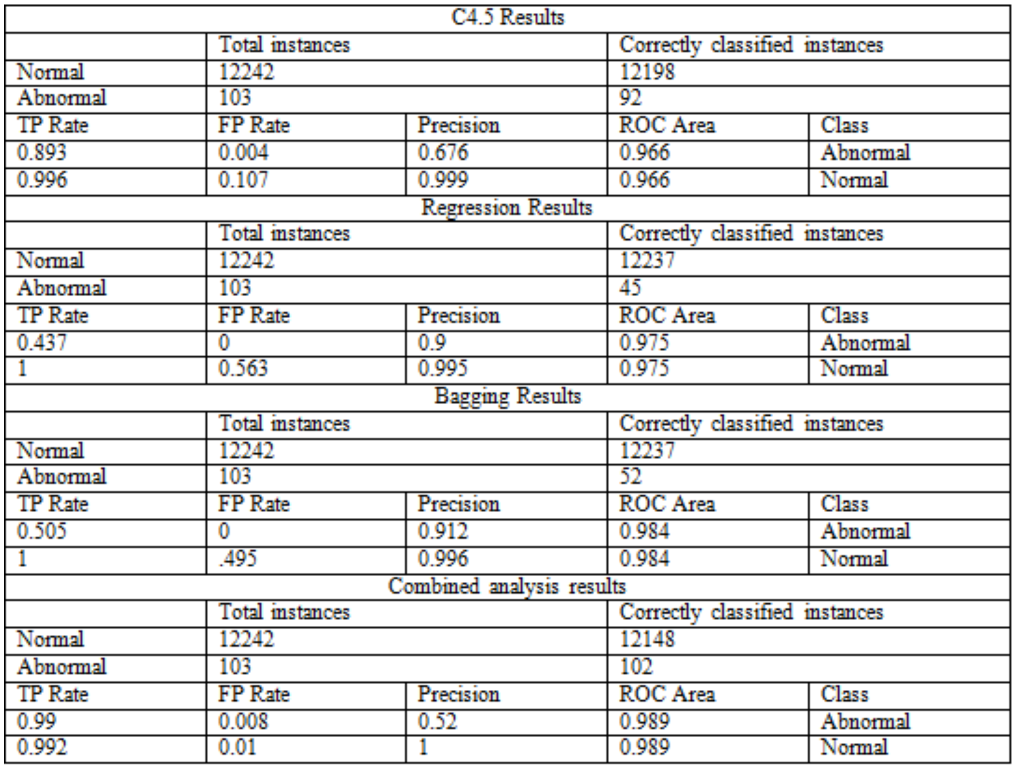


Figure 6.6: Fusion stair structure

Table 6.7: Classifier Results



C4.5 Results				
	Total instances		Correctly classified instances	
Normal	12242		12198	
Abnormal	103		92	
TP Rate	FP Rate	Precision	ROC Area	Class
0.893	0.004	0.676	0.966	Abnormal
0.996	0.107	0.999	0.966	Normal
Regression Results				
	Total instances		Correctly classified instances	
Normal	12242		12237	
Abnormal	103		45	
TP Rate	FP Rate	Precision	ROC Area	Class
0.437	0	0.9	0.975	Abnormal
1	0.563	0.995	0.975	Normal
Bagging Results				
	Total instances		Correctly classified instances	
Normal	12242		12237	
Abnormal	103		52	
TP Rate	FP Rate	Precision	ROC Area	Class
0.505	0	0.912	0.984	Abnormal
1	.495	0.996	0.984	Normal
Combined analysis results				
	Total instances		Correctly classified instances	
Normal	12242		12148	
Abnormal	103		102	
TP Rate	FP Rate	Precision	ROC Area	Class
0.99	0.008	0.52	0.989	Abnormal
0.992	0.01	1	0.989	Normal

Table 6.8 compares the results of all the classifiers. The fusion classifier performs the best with true positive rate of 99% with false positive rate of 0.8%.

Table 6.8: Classifier comparison

Classifier	Accuracy	TP-rate	FP-rate
Support vector machine	0.9789	-	-
Logistic regression classifier	0.9819	-	-
C4.5 classifier	0.9964	0.893	0.004
Regression classifier	0.9995	0.437	0
Bagging classifier	0.9995	0.505	0
Fusion-based analytics algorithm	0.9923	0.99	0.008

6.2.2 DYNAMIC ANALYSIS OF HTML FILES

Dynamic analysis analyzes the actions performed by the HTML file once it is opened in a web browser, allowing us to perform a detailed functional analysis on the file's actions. In our approach, we open HTML files in a sandboxed environment. Sophisticated HTML scripts might have embedded malicious JavaScripts which have hyperlinks pointing to phishing websites or perform iframe attacks. Fine grained data analytics can detect these types of sophisticated attacks by analyzing the execution traces of the HTML file when it is running within a browser. Figure 6.7 shows our implementation (architecture) of the dynamic analysis of HTML files that was carried out in an isolated sandbox using Cuckoo sandbox tool [120].

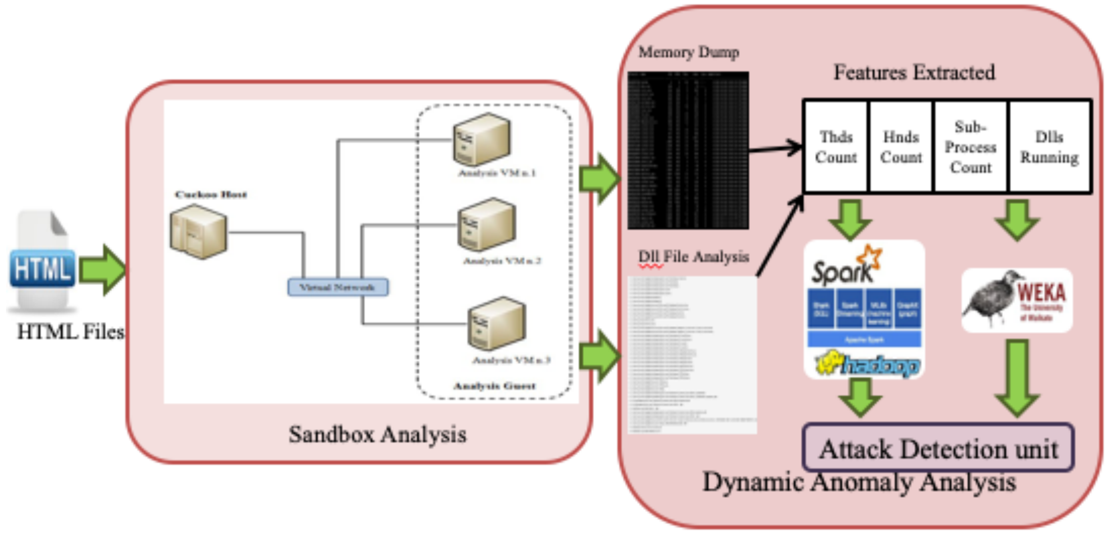


Figure 6.7: HTML Dynamic Analysis.

For experimental purposes, we chose the sandbox to host a Windows VM. In this approach, we analyze the temporal behavior and transitions of the HTML execution modules when they run within a given operating system and Internet browser environment. We believe this approach is the strongest detection technique because we analyze the binary version of the executable file where malicious codes, iframes or JavaScript obfuscations cannot be hidden from the dynamic data analysis.

The effects of HTML files on the Windows operating system are examined by observing the running time of DLL file calls. These DLL file calls are generated by the web browser upon execution of the HTML file. This approach involves analyzing the statistics of normal execution flows of DLL calls when the web browser is viewing non-malicious HTML files. We used the methodology presented in chapter 3 to create an on the fly a state machine that represents all possible DLL call sequences, that is invoked by the browser for normal HTML files. Consequently, any abnormal DLL call transitions, which are triggered by the existence of malicious codes, will be immediately detected by the dynamic analysis. Figure 6.8 shows the architecture of a general web browser, such as the Microsoft Internet Explorer.

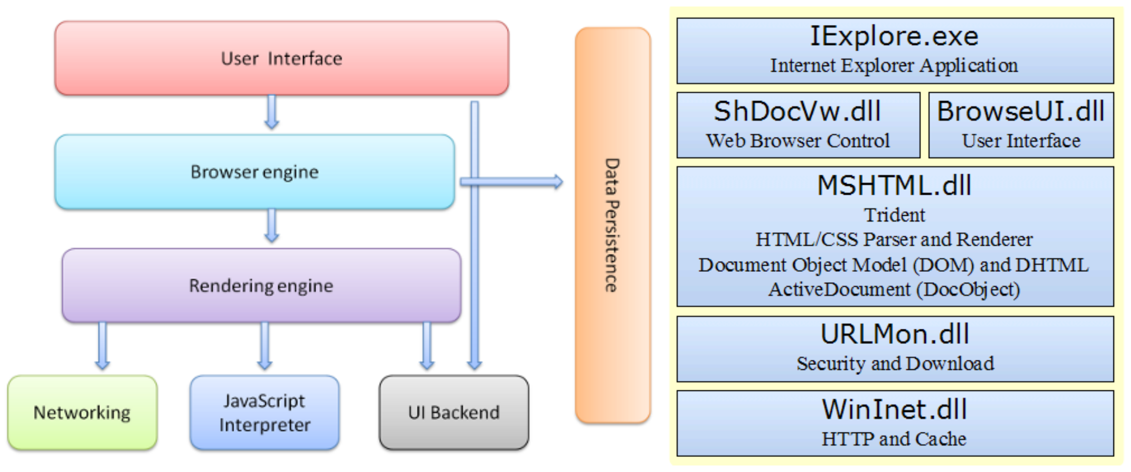


Figure 6.8. The architecture of an IE web browser

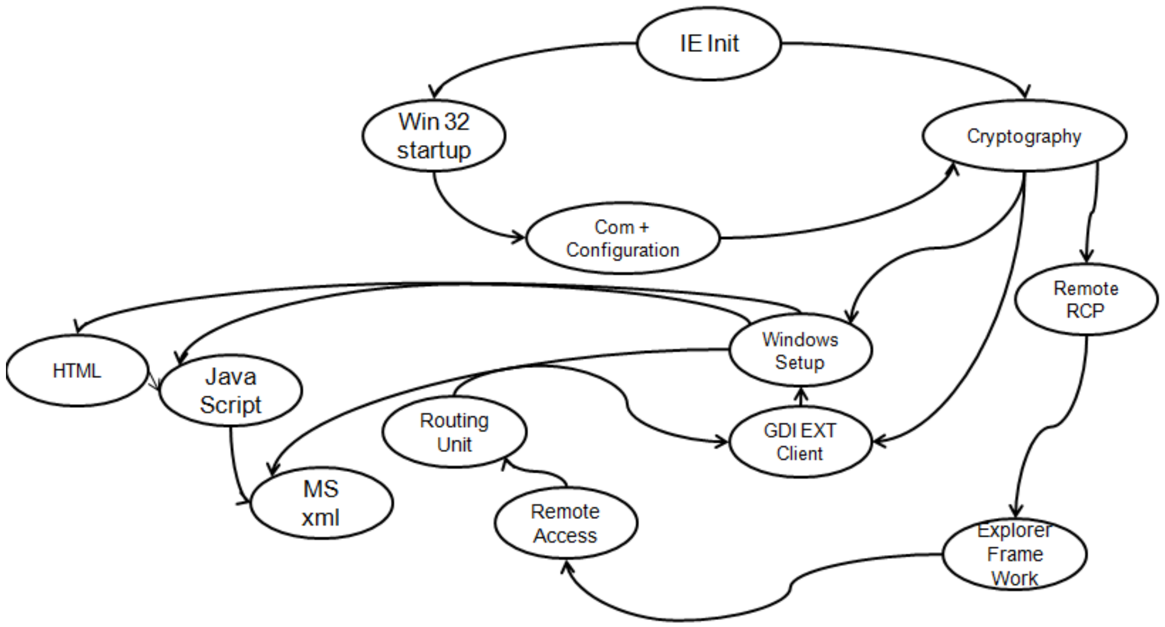


Figure 6.9. Dll State transition diagram for the Internet Explorer 8

Upon viewing a normal HTML file, a web browser will need to initialize the execution environment, setup the appropriate cryptography, desktop environment, and then proceed to display the content of the HTML file. We have developed a tool based on (Cuckoo) to obtain all the Dll calls and their temporal sequences that are invoked to view a normal HTML file as shown in Figure 6.9. However, upon execution of a malicious HTML file, the web browser will deviate from the normal state machine pathway, ultimately resulting in abnormal behavior that can be detected by the dynamic analysis tool.

As a part of a test to better understand the behavior of the dynamic model when HTML files are run on the system, we carefully analyzed the mapping over the state transitions of 3 normal and 3 abnormal files. The state transitions for the normal files are shown in figure 6.10.

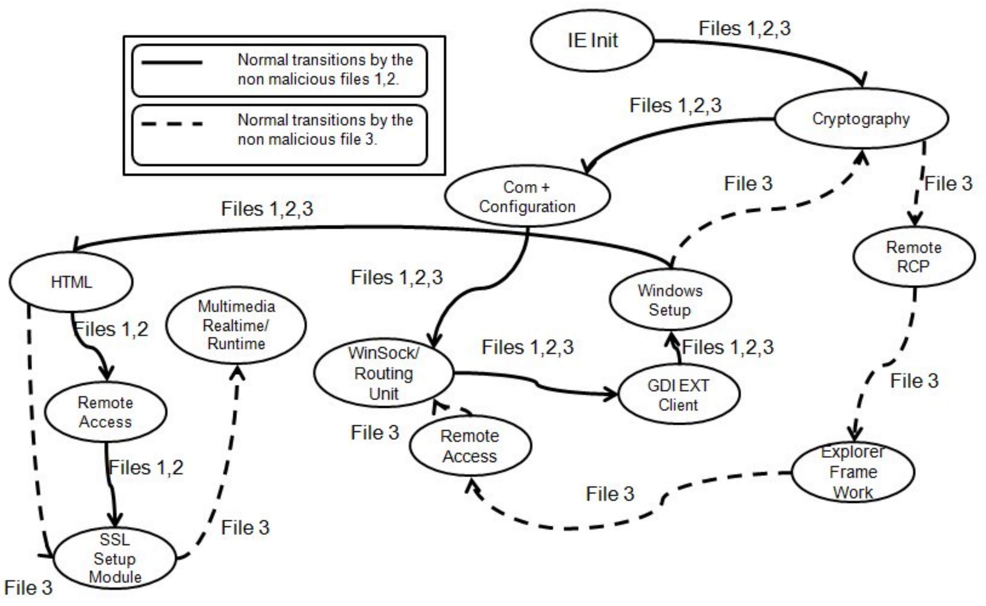


Figure 6.10: Normal State Transitions

It was observed in figure 6.11 that all 3 malicious files made the same transitions.

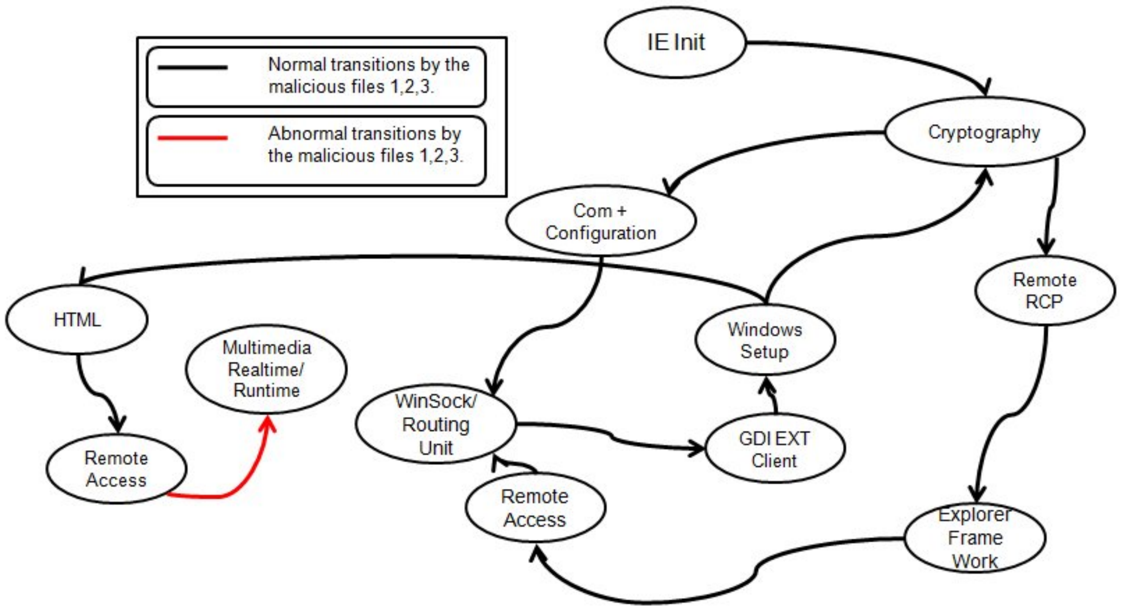


Figure 6.11: State transitions by malicious files.

The system was trained using DLL flows from 1000 normal HTML files and 40 abnormal files with a variety of malicious behavior including malicious javascripts, malicious links, hidden i-frames etc. A classification model was developed using an implementation of Adaboost.

The obtained classification model was tested on a test set which consisted of 4623 normal HTML files and 119 HTML abnormal files. The overall accuracy was observed to be 99.7259%. Table 6.9 highlights the observed results. The high false negatives can be attributed to the inability of the dynamic analysis approach to classify HTML pages with malicious links as malicious. Because we do not parse the HTML pages recursively, the malicious HTML page is never opened resulting in a page that contains a link having a normal behavior.

Table 6.9: Detailed Accuracy Summary

True Positive	False Positive	Precision	Class
95.8%	.2%	93.4%	Abnormal
99.8%	4.2%	99.9%	Normal

CHAPTER 7: CONCLUSION AND FUTURE WORK

In this chapter we summarize the AB-Analysis methodology developed I this thesis and its main research contributions and discuss future work.

7.1 RESEARCH SUMMARY

In this dissertation we reviewed the concepts of machine learning, Intrusion detection systems, and Machine learning. We then presented an architecture for IoT and cyber physical devices. Using this architecture, we presented our threat modelling methodology. We then used our methodology for designing Intrusion Detection Systems for IoT/networking protocol.

The methodology has following steps: 1. Threat modelling analysis of the protocol; 2. Feature selection and protocol foot printing to characterize the behavior of the protocol; 3. Using the correct features develop and test machine learning models that characterize the normal behavior. We then applied this methodology to three protocols: 1. Wi-Fi protocol, 2. DNS protocol, 3. HTML protocol.

For the Wi-Fi protocol, the methodology was used to analyze the attack vectors for the Wi-Fi protocol and select features to model the normal behavior of the Wi-Fi protocol. N-grams and observation flows were used to model the normal behavior of the Wi-Fi protocol. Machine learning models were built to differentiate normal Wi-Fi flows from abnormal Wi-Fi flows. During the runtime analysis and evaluation of the Wi-Fi IDS, it was shown that the IDS had no false positive and false negatives.

For the DNS protocol, the methodology was used to analyze the attack vectors for the DNS protocol and select features to model the normal behavior of the DNS protocol. N-grams and observation flows were used to model the normal behavior of the DNS protocol. Machine learning models were built to differentiate normal DNS flows from abnormal DNS flows. During the runtime analysis and evaluation of the DNS IDS, it was shown that the IDS had an attack detection rate of 97% with low false positive alarm rate of 0.01397% and false negatives of 3%.

For the HTML protocol, the methodology was used to analyze the attack vectors for the HTML protocol. Based on this analysis, we designed two IDS' to detect threats on the HTML protocol. The first IDS performed static analysis on the HTML file, by extracting static features from the HTML file. Machine learning algorithms were used to train machine learning models to detect normal and malicious files. We showed that our approach detected malicious HTML files with a true positive rate of 99% and a false positive rate of 0.8% for malicious files. The second IDS performed dynamic analysis on the HTML file, by opening the HTML file in a sandboxed environment. The state transitions made by the browser were tracked, can machine learning models were built to classify normal HTML files from the abnormal. We showed that the dynamic analysis IDS had a true positive rate of 95.8% and false positive rate of 0.2% for abnormal files and 99.8% true positive rate and 4.2% false positive rate for Normal files.

7.2 Research Contributions

- We developed a methodology to design micro IDS' to detect attacks on IoT/ networking protocols, with low false positives and false negatives.
- We developed a new novel data structures that we refer to as observation flows and n-grams that were used to accurately characterize the normal behavior of the IoT/ networking protocol.
- We developed intrusion detection systems that can detect attacks with high accuracy on Wi-Fi, DNS and HTML protocol. Our experimental results and evaluation showed that our IDS' can detect known and unknown attacks with high detection rates and low false alarms.

7.3 Future Work:

To improve the IDS design methodology presented in this dissertation, we propose to perform the following research tasks:

- **Root cause analysis**

Once an IDS detects an attack, steps have to be performed to locate the attack, identify its type and then stop or mitigate the impact of the detected attack. Root cause analysis focuses on analyzing and identifying the essential factors that caused the attack. We need to develop a methodology to perform automated root cause analysis on the intrusions detected by the ML-IDS. This methodology will identify the vulnerabilities that are exploited by the attack, establish a timeline of all the steps taken by the attack, and then extract the causal factors attributing to the attack.

The challenges involved in designing of the root cause analysis methodology are many folds. The primary challenge is to analyze large amounts of data in a short amount of time to extract causal relationships between the data and the observed behavior of the detected attacks.

- **Attack Resolution and Mitigation**

This dissertation presented a methodology to design intrusion detection systems for IoT/network protocols. The next logical step is to develop a methodology to prevent attacks on these protocols. The Intrusion Protection System (IPS) will take advantage of the root cause analysis methodology to identify the main cause of the detected attack and the steps to stop the attack and/or prevent its successful completion. The development of an IPS methodology will result in increased network availability, faster remediation of attacks on the IoT devices and networks, flexibility in system deployment, and a comprehensive threat protection.

- **Proactive actions and counter measures**

The wide spread acceptance of IoT and the rise of medical IoT devices or IoMT, has brought the need for securing these devices 24 by 7 (not just using reactive approaches like IDS/IPS). ISO/IEC 27000 and ISO/IEC 27030 present cyber security best practices and Internet of Things standards for best practices. There is a critical need to design a methodology that will check and ensure compliance of IoT devices with these standards.

- **Unsupervised and Adaptive Machine learning**

The current work involved collection of data and training the system in a supervised fashion. But the normal behavior of protocols can change over time. This change is especially noticeable with the continuous improvement in the IoT technology and their capabilities; for instance the use of HTML files has changed over the years and the normal behavior models need to account for these changes. The current work needs to be extended to include unsupervised and adaptive learning, to enable the normal behavior models to change as the usage of the protocols change.

- **Integrate network threat detection with other systems to provide all around cyber security**

As a part of this research, we used protocol state machines and data structures like n-grams and flows to characterize the normal behavior of the networking protocols. But, modern computing infrastructure exposed to a wide range of external sources over communication networks and internal threats. We can argue that modern infrastructure is more susceptible to insider attacks, making it necessary to secure the infrastructure from both external and internal threats. The footprints developed to detect threats on networks (n-grams and flows) need to be extended to build a comprehensive footprint (Cyber DNA), that can characterize the normal behavior of the complete system; allowing detection of not just network threats and attacks, but also host based attacks, malwares and user impersonation attacks.

References

1. https://www.internetworldstats.com/stats.htm[accessed: October 2018]
2. Al-Fuqaha, Ala, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. "Internet of things: A survey on enabling technologies, protocols, and applications." IEEE Communications Surveys & Tutorials 17, no. 4 (2015): 2347-2376.
3. Whitmore, Andrew, Anurag Agarwal, and Li Da Xu. "The Internet of Things—A survey of topics and trends." Information Systems Frontiers 17, no. 2 (2015): 261-274.
4. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54, no. 15 (2010): 2787-2805.
5. Lin, Jie, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications." IEEE Internet of Things Journal 4, no. 5 (2017): 1125-1142.
6. Yang, Nan, Lifeng Wang, Giovanni Geraci, Maged Elkashlan, Jinhong Yuan, and Marco Di Renzo. "Safeguarding 5G wireless communication networks using physical layer security." *IEEE Communications Magazine* 53, no. 4 (2015): 20-27.
7. Satam, Pratik. "An Anomaly Behavior Analysis Intrusion Detection System for Wireless Networks." (2015).

8. Gaur, Padmini, and Mohit P. Tahiliani. "Operating systems for IoT devices: A critical survey." In *Region 10 Symposium (TENSYMP), 2015 IEEE*, pp. 33-36. IEEE, 2015.

9. https://assistant.google.com/platforms/speakers/ [accessed: October 2018]

10. https://developer.amazon.com/alexa [accessed: October 2018]

11. https://thejournal.com/articles/2013/10/07/212-billion-devices-to-make-up-the-internet-of-things-by-2020.aspx [accessed: October 2018]

12. https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world [accessed: October 2018]

13. Navetta, David, Boris Segalis, Erin Locker, and Andrew Hoffman (May, 2017) “Large Ransomware Attack Affects Companies in Over 70 Countries”. [online] Available on: http://www.dataprotectionreport.com/2017/05/large-ransomware-attack-affects-companies-in-over-70-countries/

14. Y Al-Nashif, A Kumar, S. Hariri, Y. Luo, F Szidarovsky, G. Qui, “Multi-Level Intrusion Detection System (ML-IDS),” Autonomic Computing, 2008. ICAC ’08, 131-140, 10.1109/ICAC.2008.25.

15. Axelsson, Stefan. *Intrusion detection systems: A survey and taxonomy*. Vol. 99. Technical report, 2000.

16. https://www.ept.ca/features/speed-development-wearable-devices-solid-targeted-platform/ [accessed: October 2018]

17. L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P/ Dokas, “The MINDS- Minnesota Intrusion Detection System”, Next Generation Data Mining, MIT press 2004.

18. D. Denning, “An intrusion-detection model”, IEEE Transactions on Software Engineering, SE-13:222-232, 1987.

19. H. Javitz and A. Valdes, “The nides statistical component: Description and justification”, Technical report, computer Science Laboratory, SRI International, 1993.

20. P. Garcia-Teodoro, J. Diaz-Verdejo, G. Marcia-Fernandez, E. Vazquez, “Anomaly-based network intrusion detection: Techniques, system and challenges”, Computers & Security, Volume 28, Issues 1–2, February–March 2009,18–28.

21. M. Roesch, “Snort- Lightweight Intrusion Detection for Networks”, 13th Systems Administration Conference- LISA 1999.

22. K. Ilgun, “USTAT: A Real- Time Intrusion Detection for UNIX”, Master Thesis, University of California, Santa Barbara, November 1992.

23. G. Vigna and R Kemmerer, “ NetSat: A Network-Based Intrusion Detection Approach”, In proceedings of the 14 Annual Information Theory: 50 Years of Discovery Computer Security Application Conference, Dec 1998.

24. U. Lindquist, P. A. Porras, “Detecting Computer and Network Misuse through the Production-Based Expert System Toolset(P-BEST)”, In proceeding of the 1999 IEEE Symposium on Security and Privacy. 146-161.

25. D. Barbara, N. Wu, and S. Jajodia, "Detecting novel intrusions using bayes estimators", In Proceedings of First SIAM Conference on Data Mining, Chicago, IL, 2001.

26. M. Joshi and V. Kuma, "Credos: classification using ripple down structure (a case for rare classes)", In Proceedings of 19th International Conference on Data Engineering, Bangalore, India, 2003.

27. R. Agarwal and M. V. Joshi, "PNrule: A new framework for learning classifiers models in data mining (a case-study in network intrusion detection)", Technical Report TR 00-015, Department of Computer Science, University of Minnesota, 2000.

28. M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements", In ICDM, 257-264, San Jose, CA, 2001.

29. A. Lazarevic, N. V. Chawala, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving the prediction of minority class in boosting", Technical Report 2002-136, AHPCRC, 2002.

30. Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413-422. IEEE, 2008.

31. Quinlan, J. Ross. "Bagging, boosting, and C4. 5." In AAAI/IAAI, Vol. 1, pp. 725-730. 1996

32. Quinlan, John R. "Learning with continuous classes." In 5th Australian joint conference on artificial intelligence, vol. 92, pp. 343-348. 1992

33. Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." *R news* 2, no. 3 (2002): 18-22.

34. Rätsch, Gunnar, Takashi Onoda, and K-R. Müller. "Soft margins for AdaBoost." *Machine learning* 42, no. 3 (2001): 287-320.

35. Cohen, William W. "Fast effective rule induction." In *Machine Learning Proceedings 1995*, pp. 115-123. Morgan Kaufmann, 1995.

36. Hulten, Geoff, Laurie Spencer, and Pedro Domingos. "Mining time-changing data streams." In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97-106. ACM, 2001.

37. Pfahringer, Bernhard, Geoffrey Holmes, and Richard Kirkby. "New options for hoeffding trees." In *Australasian Joint Conference on Artificial Intelligence*, pp. 90-99. Springer, Berlin, Heidelberg, 2007.

38. Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." *Neural processing letters*9, no. 3 (1999): 293-300.

39. Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." *ACM transactions on intelligent systems and technology (TIST)* 2, no. 3 (2011): 27.

40. Breiman, Leo. "Bagging predictors." *Machine learning* 24, no. 2 (1996): 123-140.

41. Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.

42. Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.

43. Raza, Shahid, Linus Wallgren, and Thiemo Voigt. "SVELTE: Real-time intrusion detection in the Internet of Things." *Ad hoc networks* 11, no. 8 (2013): 2661-2674.

44. Sangkatsanee, Phurivit, Naruemon Wattanapongsakorn, and Chalermpol Charnsripinyo. "Practical real-time intrusion detection using machine learning approaches." *Computer Communications* 34, no. 18 (2011): 2227-2235.

45. Kenkre, Poonam Sinai, Anusha Pai, and Louella Colaco. "Real time intrusion detection and prevention system." In *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, pp. 405-411. Springer, Cham, 2015.

46. Alipour, Hamid, Youssif B. Al-Nashif, Pratik Satam, and Salim Hariri. "Wireless anomaly detection based on IEEE 802.11 behavior analysis." *IEEE transactions on information forensics and security* 10, no. 10 (2015): 2158-2170.

47. Satam, Pratik, Hamid Alipour, Youssif B. Al-Nashif, and Salim Hariri. "Anomaly Behavior Analysis of DNS Protocol." *J. Internet Serv. Inf. Secur.* 5, no. 4 (2015): 85-97.

48. Shao, Sicong, Cihan Tunc, Pratik Satam, and Salim Hariri. "Real-time irc threat detection framework." In *2017 IEEE 2nd International Workshops on*

Foundations and Applications of Self Systems (FAS* W)*, pp. 318-323. IEEE, 2017.

49. Satam, Pratik, Shalaka Satam, and Salim Hariri. "Bluetooth Intrusion Detection System (BIDS)." In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1-7. IEEE, 2018.
50. Satam, Shalaka Chittaranjan. "Bluetooth Anomaly Based Intrusion Detection System." PhD diss., The University of Arizona, 2017.
51. Satam, Pratik, Jesus Pacheco, Salim Hariri, and Mohommad Horani. "Autoinfotainment Security Development Framework (ASDF) for Smart Cars." In *Cloud and Autonomic Computing (ICCAC), 2017 International Conference on*, pp. 153-159. IEEE, 2017.
52. Jin, Jiong, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. "An information framework for creating a smart city through internet of things." IEEE Internet of Things Journal 1, no. 2 (2014): 112-121.
53. Hiro Gabriel Cerqueira Ferreira, Edna Dias Canedo, Rafael Timóteo de Sousa Junior, "IoT Architecture to Enable Intercommunication Through REST API and UPnP Using IP, ZigBee and Arduino"
54. M. Soliman, T. Abiodun, T. Hamouda, J. Zhou1, C.Lung, "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing", IEEE 5th International Conference on Cloud Computing Technology and Science, 2013.

55. Reddy, B. Raghavendhar, and E. Mahender. "Speech to text conversion using android platform." *International Journal of Engineering Research and Applications (IJERA)* 3, no. 1 (2013): 253-258.
56. https://cmusphinx.github.io [accessed: October 2018]
57. https://cloud.google.com/speech-to-text/ [accessed: October 2018]
58. IEEE Std 802.11-1997 Information Technology- telecommunications And Information exchange Between Systems-Local And Metropolitan Area Networks-specific Requirements-part 11: Wireless Lan Medium Access Control (MAC) And Physical Layer (PHY) Specifications. 1997.
59. "802.11a-1999 High-speed Physical Layer in the 5 GHz band", IEEE, 1999.
60. "802.11b-1999 Higher Speed Physical Layer Extension in the 2.4 GHz band", IEEE, 1999.
61. "IEEE 802.11g-2003: Further Higher Data Rate Extension in the 2.4 GHz Band", IEEE. 2003.
62. "IEEE 802.11n-2009—Amendment 5: Enhancements for Higher Throughput". IEEE-SA. 29 October 2009. doi:10.1109/IEEESTD.2009.5307322.
63. "IEEE Std 802.11ac 2013 - 22.5 Parameters for VHT-MCSs" IEEE. 2013.
64. A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker, " MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs", In Proceedings of the 4th international conference on Mobile systems, applications and services, June 19-22, 2006, Uppsala, Sweden .

65. Sheng, Y., Tan, K., Chen, G., Kotz, D. and Campbell, A. 2008. Detecting 802.11 MAC Layer Spoofing Using Received Signal Strength. In Proceeding IEEE 27th Annual Conference on Computer Communications (INFOCOM), April 2008.

66. D. Madory, "New methods of spoof detection in 802.11 wireless networking." Master's thesis, Dartmouth College, Hanover, New Hampshire, June 2006.

67. A.G. Fragkiadakis, V.A. Siris, and A.P. Traganitis, "Effective and robust detection of jamming attacks," Future Network and Mobile Summit, 2010 , vol., no., pp.1-8, 16-18 June 2010.

68. J. Yang, Y. Chen, W. Trappe, J. Cheng, "Detection and Localization of Multiple Spoofing Attackers in Wireless Networks," Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.1, pp.44,58, Jan. 2013.

69. Q. Li, and W. Trappe, "Detecting Spoofing and Anomalous Traffic in Wireless Networks via Forge-Resistant Relationships". Information Forensics and Security, IEEE Transactions on , vol.2, no.4, pp.793-808, Dec. 2007.

70. D. Dasgupta, F. Gonzalez, K. Yallapu. and M. Kaniganti, "Multilevel monitoring and detection systems (MMDS)". In Proceedings of the 15th Annual Computer Security Incident Handling Conference, Ottawa, Canada, 2003.

71. F. Guo, and T. Chiueh, "Sequence number-based MAC address spoof detection" *in Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection* (RAID'05), Seattle, U.S.A., 2005.

72. R. Gill, J. Smith, A. Clark, “Specification-Based Intrusion Detection in WLANs”, Computer Security Applications Conference, 2006.ACSAC '06. 22nd Annual , vol., no., pp.141-152, Dec. 2006.

73. S. Fayssal, S. Hariri, and Y. B. Al-Nashif, “Anomaly-Based Behavior Analysis of Wireless Network Security”, In Proceeding of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking &Services.MobiQuitous 07. Aug. 2007.

74. Airmagnet,(December 2014) http://www.flukenetworks.com/enterprise-network/wireless-network/AirMagnetEnterprise

75. Airdefence, (December 2014) http://www.airdefense.net

76. F. Izquierdo, “Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN”, Wireless Pervasive Computing, pp 1- 6, January 2006.

77. Kolias, Constantinos, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset." *IEEE Communications Surveys & Tutorials* 18, no. 1 (2016): 184-208.

78. Allahdadi, Anisa, and Ricardo Morla. "802.11 Wireless Access Point Usage Simulation and Anomaly Detection." *arXiv preprint arXiv:1707.02933* (2017).

79. Usha, M., and P. Kavitha. "Anomaly based intrusion detection for 802.11 networks with optimal features using SVM classifier." *Wireless Networks* 23, no. 8 (2017): 2431-2446.

80. Satam, Pratik. *An anomaly behavior analysis intrusion detection system for wireless networks*. The University of Arizona, 2015.

81. Sivaraman, Vijay, Hassan Habibi Gharakheili, Arun Vishwanath, Roksana Boreli, and Olivier Mehani. "Network-level security and privacy control for smart-home IoT devices." In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, pp. 163-167. IEEE, 2015.

82. Libpcap, (March 2019) https://www.tcpdump.org

83. Hostapd, (October 2015) https://w1.fi/hostapd/

84. Hping3 (March 2019) http://www.hping.org

85. RFC1034, "Domain Names - Concepts And Facilities", Network Working Group, November 1987.

86. RFC1035, "Domain Names - Implementation And Specification", Network Working Group, November 1987.

87. Y G.Ateniese , S.Mangardm, "A New Approach to DNS Security (DNSSEC)", In Proceedings of the 8th acm conference on Computer and Communications Security,2001.

88. A. Yamada, Y. Miyake, M. Terabe, K. Hashimoto, N. Kato, "Anomaly Detection for DNS Servers Using Frequent Host Selection," 2009 International Conference on Advanced Information Networking and Applications, 2009, pp.853-860.

89. Y. Wang, M. Hue, B. Li, B. Yan, "Tracking Anomalous Behaviors of Name Servers by Mining DNS Traffic", Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops, 2006, pp.351-357.

90. B. Zdrnja, N. Brownlee and D. Wessels "Passive Monitoring of DNS Anomalies" International Conference on Detection of Intrusion and Malware and Vulnerability Assessment, 2007.

91. W Lian, E Rescorla, H Shacham, S Savage, "Measuring the practical impact of DNSSEC deployment.", USENIX Security, 2013.

92. A. Yamada, Y. Miyake, M. Terabe, K. Hashimoto, N. Kato, "Anomaly Detection for DNS Servers Using Frequent Host Selection," 2009 International Conference on Advanced Information Networking and Applications, 2009, pp.853-860.

93. Y. Wang, M. Hue, B. Li, B. Yan, "Tracking Anomalous Behaviors of Name Servers by Mining DNS Traffic", Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops, 2006, pp.351-357.

94. B. Zdrnja, N. Brownlee and D. Wessels "Passive Monitoring of DNS Anomalies" International Conference on Detection of Intrusion and Malware and Vulnerability Assessment, 2007.

95. A Karasardis, K Meier-Hellstern, D Hoeflin, "NIS04-2: Detection of DNS Anomalies using Flow Data Analysis.", Global Telecommunications Conference, 2006, 1-6.

96. Cermak M., Celeda P., Vykopal J., “Detection of DNS Traffic Anomalies in Large Networks,” Advances in Communication Networking,Volume 8846, pp 215-226, December 2014.

97. S. Son, V. Shmatikov, “The Hitchhiker’s Guide to DNS Cache Poisoning”, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Volume 50, 2010, pp 466-483.

98. United States Computer Emergency Readiness Team, “DNS Amplification Attacks”, Internet:https://www.us-cert.gov/ncas/alerts/TA13-088A, July 22, 2013 [June 13, 2015].

99. Lau, Felix, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajkovic. "Distributed denial of service attacks." In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3, pp. 2275-2280. IEEE, 2000.

100. Antonakakis, Manos, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric et al. "Understanding the mirai botnet." In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1093-1110. 2017.

101. Kolias, Constantinos, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. "DDoS in the IoT: Mirai and other botnets." *Computer* 50, no. 7 (2017): 80-84.

102. Hilton, Scott. "Dyn analysis summary of friday october 21 attack." *Dyn Blog, Oct* (2016).

103. T. van Leijenhorst, K. Chin & D. Lowe, "On the viability and performance of DNS tunneling," in International Conference on Information Technology and Applications, 2008, pp. 560-566.

104. Bosch, Antal, ToineBogers, and Maurice Kunder. "Estimating search engine index size variability: a 9-year longitudinal study." Scientometrics 107, no. 2 (2016): 839-856.

105. Evans Dave "The internet of things: How the next evolution of the internet is changing everything." CISCO white paper 1(2011): 1-11.

106. Xu, Wei, Fangfang Zhang, and Sencun Zhu. "The power of obfuscation techniques in malicious JavaScript code: A measurement study." In Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on, pp. 9-16. IEEE, 2012.

107. Canfora, Gerardo, Francesco Mercaldo, and Corrado Aaron Visaggio. "Malicious JavaScript Detection by Features Extraction." e-Informatica Software Engineering Journal 8, no. 1 (2014).

108. Yoo, Suyeon, Sehun Kim, Anil Choudhary, O. P. Roy, and T. Tuithung. "Two-Phase Malicious Web Page Detection Scheme Using Misuse and Anomaly Detection." International Journal of Reliable Information and Assurance 2, no. 1 (2014).

109. Likarish, Peter, Eunjin Jung, and Insoon Jo. "Obfuscated malicious javascript detection using classification techniques." In MALWARE, pp. 47-54. 2009.

110. Qu, Guangzhi, Salim Hariri, and MazinYousif. "A new dependency and correlation analysis for features." Knowledge and Data Engineering, IEEE Transactions on 17, no. 9 (2005): 1199-1207.

111. Bouckaert, Remco R. Bayesian network classifiers in weka. Department of Computer Science, University of Waikato, 2004.

112. Bouckaert, Remco R. "Bayesian network classifiers in weka for version 3-5-7." Artificial Intelligence Tools 11, no. 3 (2008): 369-387.

113. Witten, Ian H., Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham. "Weka: Practical machine learning tools and techniques with Java implementations." (1999): 81-17.

114. Singhal, Swasti, and Monika Jena. "A study on WEKA tool for data preprocessing, classification and clustering." International Journal of Innovative Technology and Exploring Engineering (IJITEE) 2, no. 6 (2013): 250-253.

115. Frank, Eibe, Mark Hall, Len Trigg, Geoffrey Holmes, and Ian H. Witten. "Data mining in bioinformatics using Weka." Bioinformatics 20, no. 15 (2004): 2479-2481.

116. Meng, Xiangrui, Joseph Bradley, BurakYavuz, Evan Sparks, ShivaramVenkataraman, Davies Liu, Jeremy Freeman et al. "Mllib: Machine learning in apache spark." arXiv preprint arXiv:1505.06807 (2015).

117. Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA data mining software: an update." ACM SIGKDD explorations newsletter 11, no. 1 (2009): 10-18.

118. Holmes, Geoffrey, Andrew Donkin, and Ian H. Witten. "Weka: A machine learning workbench." In Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on, pp. 357-361. IEEE, 1994.

119. Garner, Stephen R. "Weka: The waikato environment for knowledge analysis." In Proceedings of the New Zealand computer science research students conference, pp. 57-64. 1995.

120. https://cuckoosandbox.org [accessed March 2019]

121. Jelinek, Frederick and Robert Mercer. Interpolated estimation of Markov source parameters from sparse data. In Proceeding of the Workshop on Pattern Recognition in Practice, Amsterdam, May1980.