Robust Learning Intrusion Detection for Attacks on Wireless Networks

Adetokunbo Makanju*, A. Nur Zincir-Heywood, Evangelos E. Milios
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, B3H 1W5.
Canada.
T: +1-902-494-2093
F: +1-902-492-1517
{makanju, zincir, eem}@cs.dal.ca

May 27, 2010

Abstract

We address the problem of evaluating the robustness of machine learning based detectors for deployment in real life networks. To this end, we employ Genetic Programming for evolving classifiers and Artificial Neural Networks as our machine learning paradigms under three different Denial-of-Service attacks at the Data Link layer (De-authentication, Authentication and Association attacks). We investigate their cross-platform robustness and cross-attack robustness. Cross-platform robustness is the ability to seamlessly port an Intrusion Detector trained on one network to another network with little or no change and without a drop in performance. Cross-attack robustness is the ability of a detector trained on one attack type to detect a different but similar attack on which it has not been trained. Our results show that the potential of a machine learning based detector can be significantly enhanced or limited by the representation of the training data for the learning algorithms.

*Corresponding author

Keywords: Intrusion Detection, Wireless Networks, Machine Learning, Dependability, Robustness.

1 Introduction

It is no longer news to state that the use of Wireless Fidelity (WiFi) networks introduces security risks to which traditional wired networks are not susceptible. Despite these well known vulnerabilities [1], the use and deployment of WiFi networks has continued to grow. This has made the IEEE 802.11 protocol, on which WiFi networks are based, by far the most widely used wireless networking standard in the world today.

While there are several classes of WiFi specific attacks that exploit these vulnerabilities, our work focuses on Denial of Service (DoS) attacks, which exploit Management Frames at the Media Access Control (MAC) or Data Link layer. This attack class is important because such attacks are relatively easy to implement and can cause a WiFi network to become unusable. The Data Link layer is the second layer of the Open System Interconnect (OSI) protocol stack and it is above the physical layer. Unless new security features are introduced into the 802.11 protocol [2], the only means of mitigating this class of attacks is through the use of Intrusion Detection Systems (IDSs). The current set of security features incorporated into the 802.11 protocol such as data encryption and client authentication are not able to guard against these attacks. Thus, in this work, we focus specifically on the three most popular attack types from this category. These are: de-authentication flood attack, authentication flood attack and association flood attack. These attacks are named after the subset of the IEEE 802.11 Management Frames, which they exploit.

Recent research has proposed machine learning solutions for data link layer attacks, including DoS attacks, in 802.11 networks. Machine learning techniques include Genetic Programming (GP) [3], and Artificial Neural Networks (ANNs) [4]. Moreover, it has been shown that GP based solutions are more adaptable than conventional IDSs such as Snort-Wireless to modified versions of known attacks [5]. Specifically, it was shown that a machine learning (GP)based IDS was very successful under stealth conditions, whereas Snort-Wireless, a well known conventional IDS, failed to perform under the same conditions [5]. While these results are valuable, more research still needs to be done to ascertain the robustness of the proposed machine learning based solutions.

Thus, in this paper, we investigate the robustness of a machine learning based detector in terms of cross-platform robustness and cross-attack robustness. By cross-platform robustness, we refer to the ability of seamlessly porting an IDS solution, without a drop in performance, from one physical network to another with little or no change to the IDS. Conventional signature based IDSs such as Snort-Wireless and Kismet that can be used for detecting data link layer attacks are cross-platform robust because anyone on any network can simply install and use them to monitor a given network without changing any attack signatures..

On the other hand with cross-attack robustness, we mean the ability for a detector trained on one attack type to detect a different but similar attack, which it is not trained on or seen before. For example the de-authentication and dissociation wireless DoS attacks are very similar in implementation but differ only in the management frame exploited. Even though these attacks are

quite similar, the signatures for the De-Authentication attack in a conventional IDS like Snort-Wireless will not detect the dissociation attack, simply because it uses a different management frame. Thus, to this end, the possibility of building GP based IDSs, which are cross-attack robust, has been investigated in [3], the results showed promise.

With this in mind, our objective in this work is to investigate the cross-platform robustness and cross-attack robustness of these, extending beyond the results presented in [3] and [6]. We focus on the applicability of two machine learning techniques namely, GP and ANN, to this problem. These two techniques are chosen because to the best of our knowledge they are the only machine learning techniques previously applied to Wireless network IDS in the literature [3, 4, 5, 6].

The remainder of this paper is organised as follows. Section 2 discusses WiFi networks and data link layer attacks. Section 3 discusses different methods of detecting data link layer attacks investigated. Section 4 outlines the problem and the proposed solution. Section 5 outlines the MAC address mapping techniques used in our experiments in detail. Section 6 outlines our methodology and experimental setup while Section 7 presents the results. Finally, conclusions are given and future work is discussed in Section 8.

2 Data Link Layer Attacks and WiFi Networks

2.1 WiFi Networks

WiFi networks generally consist of one or more Access Points (APs) and a number of clients, the clients can be any device from laptop computers to wireless Personal Digital Assistants (PDAs). These networks communicate over a wireless medium using the IEEE 802.11 standard. Variants based on the IEEE 802.11 standard include 802.11b, 802.11g and others. These variants differ from each other, amongst other things, by the frequency at which they operate and the bandwidth that they are able to deliver. In this paper, we deal specifically with 802.11b networks but our results can be generalized to the other variants of the standard, as the difference between these variants exist more at physical layer of the 802.11 protocol[7].

WiFi Access Points (APs) act as base stations or servers for Wireless Local Area Networks (WLANs). Using Beacon Frames, they periodically broadcast their Service Set Identifier (SSID), which is a character string identifying the AP. This way, any authorised client machine that is within the range of the AP and that can pick up the SSID signal can choose to join the network.

However, security is of great concern in WiFi networks. Several protocols, which use authentication and cryptographic techniques like Wireless Encryption Protocol (WEP) and WiFi Protected Access (WPA), as well as wireless Virtual Private Networks (VPN), have been used to ameliorate these vulnerabilities. These protocols, however, do not deal with attacks that target the physical and/or data link layers of the network protocol stack. Some of these attacks

are DoS attacks, which usually exploit data link layer management frames. These attacks cause the network to become unusable or inaccessible to legitimate clients.

Management frames are used by stations to establish and maintain connections. It is important to note that all the procedures used by the stations to establish and maintain their connections rely on an explicit trust, the client trusting the validity of the AP and vice versa. It is for this reason that 802.11 management frames can be exploited to craft novel DoS attacks, which aim to make a WiFi network unusable. Types of management frames include: Association, Disassociation, Authentication, De-authentication, Beacon and Probe frames. Full discussion on the uses of these frames is beyond the scope of this paper, however the reader can refer to [7] for a more detailed explanation of these frame types.

WiFi specific data link layer attacks are very easy to implement. An attacker simply eavesdrops on a network and gathers information about the stations on the network. The attacker then uses this information to create forged management frames with a spoofed data link layer (MAC) address of a station or an AP on the network. If the attacker chooses to target a specific client, it creates a management frame with the address of the target client as the destination and the address of the AP as the source. The attacker can also choose to vary the scope of the attack i.e. by focusing on the AP to take down the entire network, or by targeting a specific client or group of clients. Our work utilises three well-known data link layer attacks i.e. De-authentication, Authentication and Association attacks. Fig. 1 visually depicts how these attacks can be executed on a WiFi network.

2.2 Attacks and Void11

Void11 is a free software implementation of some common 802.11b attacks, such as the De-authentication flood, Authentication flood and Association flood [8]. The basic implementation works in a command line Linux/Unix environment, though it has a GUI implementation called gvoid11, too. Table 1 shows the default values for the parameters used to launch an attack. For void11 to work on a computer, the computer must have a prism based wireless Network Interface Card (NIC) and must have hostap drivers installed. The hostap drivers allow the machine to act as a wireless AP [9].

Void11 implements the three data link layer attacks that we utilize in this work[1]. The basic goal of each of the attacks is to flood the network with management frames causing random clients to loose their connection with the AP or keep the AP busy dealing with client requests, which slows down the network. The end result of each of these attack types differs based on the rate of injection of the frames and on the client involved. All the data link layer attacks, which are launched to create the datasets used in our experiments are

[1]The command syntax for using the void11 tool to launch an attack is: ***void11penetration -D -t[type of attack] -d[delay] -s[station MAC] -B[BSSID] [interface]***

executed using void11, with the default values of its command line arguments, see Table. 1. It should be noted that void11 does not simulate the DoS attacks it implements but mounts the actual attacks.

3 Detecting Data Link Layer Attacks

Intrusion Detection Systems (IDSs) are used to detect attacks against the integrity, confidentiality and availability of computer networks [10, 3]. They are analogous to burglar alarms, which monitor the premises to find evidence of break-ins. These operations aim to catch attacks and log information about the incidents such as source and nature of an attack. An IDS can be a combination of software and hardware, which collects and analyzes data collected from a network(s) and/or a host(s).

In this section, we briefly discuss the intrusion detection systems utilised in this work i.e. Snort-Wireless, Artificial Neural Network (ANN) based IDS and Genetic Programming (GP) based IDS.

3.1 Snort-Wireless Based Detection

There are several open source and commercial IDSs available in the market today but Snort stands out as being one of the most popular. Developed in 1998 by Martin Roesch, Snort is an open source, real-time intrusion detection system [11]. Using signature based metrics, it detects and prevents attacks by utilizing a rule-driven language. It is the most widely deployed open source IDS in industry and research.

Snort-Wireless [12], which is utilized in our work is an extension of Snort, with signatures which allow the detection of WiFi specific attacks included.

Snort-Wireless is employed in this work for two reasons:

(i) To put our proposed IDS system into context with the existing technology.

(ii) To validate our datasets (both normal and attack traffic generated) and to compare to other systems.

To achieve this, we set up physical networks and attacked them using void11. The network traffic from these physical networks was logged using the traffic logging features of Kismet. This logged network traffic data files, which consists of frames which the machines on network send when communicating with each other, were later replayed to Snort-Wireless in their raw tcpdump format.

3.2 Machine Learning Based Systems

Unlike their conventional counterparts, machine learning based detectors utilize detection signatures, which are not produced by human experts. Their signatures are learnt automatically through a learning phase, which is done using labelled network traffic data. Recent research has focused on the use of

machine learning solutions in the detection of 802.11 data link layer attacks [3, 4, 5]. Specifically Genetic Programming (GP) and Artificial Neural Networks (ANNs) have been used. Also in [5], a GP based IDS which was evolved to classify between normal and intrusion states for the de-authentication attack was compared against Snort-Wireless. The paper showed that a GP based IDS was capable of detecting the de-authentication attack, even when the attacker stealthily injects attack frames into the network traffic. The stealthy injection of packets is done to hide the fact that an attack is in progress. While the GP based IDS was still able to detect the attack, the stealth attack was not detected by Snort-Wireless.

This success in the use of machine learning based solutions in the detection of 802.11 data link layer attacks in part forms the motivation for our work. If machine learning based solutions are to be seriously considered as replacements for conventional detectors like Snort, more research needs to be carried out to ascertain their robustness. So far not much has been done in this respect and our work attempts to fill this gap. Under cross-platform and cross-attack conditions, we ascertain the robustness of machine learning based IDSs, hereby shedding more light on their strengths and weaknesses.

4 Proposed System

In this section we discuss the Machine Learning (ML) based IDSs proposed. We provide details of the methods utilized in data preprocessing and the parameter values utilized during training of the proposed IDSs. These details allow for the reproducibility of the results achieved in the testing our proposed IDSs.

4.1 Machine Learning (ML) Based IDSs

The proposed ML based IDS, as with all machine learning based solutions, relies on the use of labelled network traffic data. Collecting such data is usually the first step in building most ML based classifiers. Thus, in this work, the network data was collected on live networks, which were set up in our laboratory. A detailed description of our laboratory test network is given in Section 6.4.

Once this data is collected, it can then be used to train the ML solution to produce rules that detect the network attacks. The network traffic data used in training is usually not used in its native format during the training and testing phases of most ML based solutions and this is no different in our case. A set of relevant features needs to be extracted and presented in a vector space model to the classifier. Section 6.3 explains the feature set used in our work in more detail. In some cases, based on the data type of features chosen, an appropriate mapping needs to be performed. For example, GP and ANN classifiers can only work on numerical values, so all non-numeric features need to be mapped to numeric values. This problem reared its head with regard to data link layer (MAC) addresses in our feature set, our work shows that this mapping is important for the production of robust solutions. This forms the

rationale for the address mapping techniques proposed in section 5 of our work. The mapping techniques in sections 5.2 and 5.3 are our novel contributions, while the technique highlighted in section 5.1 is from previous work [3, 5].

In this work, the input data for the training does not consist of individual frames from the network traffic data (network data collected on live networks in the lab) but each input exemplar comprises of a collection of frames stringed together in arrival order. The trained solutions produced by the classifiers are then tested for efficiency using another partition of the network traffic data, which was not used in the training phase. This testing phase validates the rules produced by the system for correctness. The remainder sub-sections detail how the GP and ANN classifiers work and the parameter settings used by the classifiers during the rule learning phase of developing the signatures automatically.

4.2 GP Based IDS Training Parameters

GP is an extension of the Genetic Algorithm (GA); which is an evolutionary computation (EC) method proposed by John H. Holland [13]. GP extends the GA to the domain of evolving complete computer programs [14]. Using the Darwinian concepts of natural selection and fitness proportional breeding (fitness is determined using a fitness function), populations of programs are genetically bred to solve problems. These populations of programs can either be represented as tree like LISP structures or as binary strings, which represent integers . These integers are then mapped onto an instruction set and a set of source and destination registers. Each individual can thus be decoded into a program, which takes the form of assembly language type code for a register based machine. This is known as the Linear Page Based GP (L-GP)[13]. Our work utilizes the L-GP approach.

L-GP alongside the Random Subset Selection - Dynamic Subset Selection (RSS-DSS) algorithm [15] has been used successfully by other researchers in the realm of IDSs, based on its successful use in detecting the deauthentication attack [3] and other higher level attacks [16]. Moreover, because of its performance in the face of stealth attacks [5] and the transparency of its solutions, it is also employed in this work.

The parameter settings for the GP employed in this work in all cases are given in Table 2. In addition to the GP parameters, the fitness function utilised in this work is the switching fitness function [3]. The switching fitness function assigns credit to a member of the population depending on whether the execution of the individual on an exemplar produces a false positive, Eq. (1) or a false negative, Eq. (2). During each generation, the variables $Fitness(n)$ and $Fitness(n+1)$ represent the fitness value of an individual before an evaluation and the fitness value of an individual after an evaluation, respectively. A generation proceeds by consecutively testing each member of the population against the exemplars in the traffic log dataset, if the individual incorrectly classifies an exemplar, its fitness value is incremented using either Eq. (1) or Eq. (2). A run of the GP would consist of a predetermined number of generations. A higher credit value (Fitness) assignment at the end of the run indicates a poor

performing individual.

$$Fitness(n+1) = Fitness(n) + \frac{1}{TotalNo.OfNormalConnections} \quad (1)$$

$$Fitness(n+1) = Fitness(n) + \frac{1}{TotalNo.OfAttackConnections} \quad (2)$$

4.3 ANN Based IDS Training Parameters

An ANN can be described as a massively parallel distributed processor, which consists of connected individual processing units called nodes, which is capable of storing experiential knowledge it gets from its environment in internode connectors as weights [17]. An ANN also has the property of being able to make its acquired knowledge available for use. An ANN, specifically a Dynamically Growing Neural Network (DGNN) was used in the training of an anomaly based wireless IDS in [4]. Liu et. al., utilized the Improved Winner Takes It All (IWTA) algorithm to successfully select a feature set and train an anomaly based detector for wireless attacks [4].

In order to employ an ANN based IDS, we use the ANN implementation from WEKA[18]. WEKA is a suite of Machine Learning and Data Mining algorithms developed at the University of Waikato by Ian H. Witten and Eibe Frank. The neural network algorithm used in these experiments consists of a very simple multilayer perceptron feed forward network for building the classifier. The multilayer perceptron has an input layer, one hidden layer and an output layer. The output layer has two outputs indicating whether the exemplar is an attack or normal. The network uses a sigmoid function as its activation function and back propagation as its learning algorithm. The parameter settings for the ANN in all cases are given in Table 3. These parameters are the default values for multilayer perceptrons in WEKA. Results using a simple ANN are included in our discussion to reinforce the fact that the issues investigated in this work potentially pertain to all Machine Learning based IDSs.

5 Data Link Layer (MAC) Address Mapping Techniques

This section explains the mapping schemes used for the three data link layer identifier/address fields used in our feature set. These identifier fields are the Destination Address, the Source Address and the Basic Service Set Identifier (BSSID).

The data link layer (MAC) address is a unique address attached to every Network Interface Card. The MAC address enables each station to have a unique identifier on a network. Although a machine on a network may identify itself differently depending on the network layer at which the communication

is taking place, the MAC address acts as the identifier at Layer-2. Indeed other attributes such as IP addresses and hostnames used at higher layers of the protocol stack, all map back to MAC addresses. MAC addresses are usually represented with the hexadecimal equivalent of each octet separated by a dash or colon. The following are valid MAC addresses: FF-FF-FF-FF-FF-FF, FF:FF:FF:FF:FF:FF. While the other fields of the frame used in our feature set are numerical variables, the MAC addresses are nominal variables. However, the machine learning algorithms i.e. GP and ANN, employed in this work, can only deal with numeric variables. Thus, this makes it imperative for us to come up with an address mapping scheme, which can represent MAC addresses by unique numbers in the datasets while still providing meaningful patterns that can be deciphered by the learning algorithm in the training and testing phases.

In our work, we compare five different techniques for mapping MAC addresses. For a machine learning solution to be able to detect a data link layer attack effectively, it must be able to discriminate between each participant (host) on the network and differentiate them based on what role they play on the network, information that is also important to the formulation of the attack. It is shown in [6] that the Simple MAC address mapping technique can capture this information when used within one given platform, i.e. network, but fails when used across different platforms since the mapping is not consistent i.e. in this case, a physical machine will be mapped to a different numerical value each time its physical network location changes. With this intuition, in this work, we investigate more appropriate mapping techniques, which can ensure that a MAC address can be mapped to the same numerical value, irrespective of its physical network location or the role it plays on a given physical network.

With this in mind, we designed two novel mapping techniques. These are: (i) The *Decimal-Sum* technique, a hashing function, which produces a numeric representation for each MAC address based on the MAC address itself; and (ii) The *Role-Based* technique, which maps the MAC addresses based on the role they play in the network. It should be noted here that we designed three variants of the Role-Based technique. The results of our first set of experiments are obtained using these different techniques, which are highlighted below.

5.1 Simple Mapping

This is the mapping scheme used in [5]. It maps the identifiers based on the ordinal position of the MAC addresses in a sorted list. The simple MAC mapping technique is detailed in Algorithm 1. It is shown in [6] that this technique does not lead to cross-platform robustness for the IDS.

Algorithm 1 Simple Mapping

Input: Array $X[]$ containing all MAC addresses in dataset .
Output: Array $Y[]$ containing integer mappings of the MAC addresses in $X[]$.
{Mapping of $X[i] = Y[i]$}

```
Sort(X)
i = 1
for every macaddr in X do
    Y[i] = i
    i++
end for
Return(Y)
```

5.2 Decimal-Sum Mapping

Each MAC identifier is unique i.e. theoretically no two Network Interface Cards have the same MAC identifier. A MAC identifier consists of two parts, the Organisationally Unique Identifier (OUID) and the Local Identifier. Thus, the combination of these two numbers makes the identifier unique. The decimal-sum technique is therefore designed to give a unique decimal number representation for each hexadecimal MAC identifier. Thus, the decimal-sum mapping technique is designed to achieve a balance between mapping the MAC addresses to large integers and the need to achieve a perfect hash. The scheme will map the addresses to integers in the range of 0 - 765,765 and has a $2 \cdot 90 \times 10^{-12}$ chance of a collision occurring. The mapping scheme is provided in Algorithm 2 and is also a diagram is shown in Figure 4.

Algorithm 2 Decimal-Sum Mapping

Input: Array $X[]$ containing all MAC addresses in dataset .
Output: Array $Y[]$ containing integer mappings of the MAC addresses in $X[]$.
{Mapping of $X[i] = Y[i]$}

```
Sort(X)
i = 1
for every macaddr in X do
    j = 1
    for each Octet in macaddr do
        dec_octet[j] =ConvertTODecimal(Octet)
    end for{There are six octets in each MAC address, FOR loop starts at the 1st octet }
    oui_sum =Sum(dec_octet[1],dec_octet[2],dec_octet[3])
    org_sum =Sum(dec_octet[4],dec_octet[5]dec_octet[6])
    dec_sum = oui_sum.org_sum {dec_sum is an integer formed as the concatenation of string equivalents of oui_sum and org_sum}
    Y[i] = dec_sum
    i++
end for
Return(Y)
```

5.3 Role Based Mapping

This novel technique maps MAC addresses based on the role, which the machine of origin plays on the network in question or the role of the MAC address (i.e. special/reserved). The recognized roles in our scheme are:

- Broadcast
- Access Point
- Station/Client
- Host
- Other

We designed three variants of this technique. Each one of the role based techniques maps a MAC address to a positive integer based solely on its role in the network.

The first variant, Algorithm 3, assigns a fixed integer to each role and maps each MAC address to this number. This setup potentially leads to a loss of information, as it becomes difficult to differentiate between two stations on the network where they share the same role without performing an analysis of the fragment and sequence numbers. For this reason the second variant, Algorithm 4, is designed. This variant introduces elements of the Simple addressing technique into the Role-Based technique by mapping MAC addresses to numeric

representations, which are based on a combination of both its role and its ordinal position in a sorted list of MAC addresses of the same role. Creating a sorted list of MAC addresses might be difficult to implement in an on-line network, as wireless networks are very dynamic, especially with regards to the clients. For this reason a third variant, Algorithm 5, of the Role-Based technique is designed. This variant simply takes the second variant a step further by using the Decimal-Sum technique for clients alone, and maintaining the role based mapping for other roles.

Algorithm 3 Role-Based Mapping I

Input: Array $X[]$ containing all MAC addresses in dataset .
Output: Array $Y[]$ containing integer mappings of the MAC addresses in $X[]$.
{Mapping of $X[i] = Y[i]$}

```
1: Sort(X)
2: i = 1
3: for every macaddr in X do
4:    Role = DetermineRole(macaddr) {Role can either be Broadcast, Access_Point, Host, Station or Other}
5:    if Role = Broadcast then
6:       Y[i] = 1
7:       i++
8:    end if
9:    if Role = Access_Point then
10:      Y[i] = 2
11:      i++
12:   end if
      {Next IF is included only if data is been processed for a Host Based IDS}
13:   if Role = Host then
14:      Y[i] = 3
15:      i++
16:   end if
17:   if Role = Station then
18:      Y[i] = 4
19:      i++
20:   else
21:      Y[i] = 5 {Assumed that Role = Other}
22:      i++
23:   end if
24: end for
25: Return(Y)
```

Algorithm 4 Role-Based Mapping II

Input: Array $X[]$ containing all MAC addresses in dataset .

Output: Array $Y[]$ containing integer mappings of the MAC addresses in $X[]$. {Mapping of $X[i] = Y[i]$}

```
Sort(X)
i = 1
client_no = 0 {Stores a count of the number of clients seen so far}
ap_no = 0 {Stores a count of the number of Access Points seen so far}
for every macaddr in X do
    Role = DetermineRole(macaddr) {Role can either be Broadcast, Access_Point, Host, Station or Other}
    if Role = Broadcast then
        Y[i] = 1
        i++
    end if
    if Role = Access_Point then
        Y[i] = 2.ap_no {Y[i] is an integer formed from the concatenation of the string equivalents of 2 and ap_no}
        i++
        ap_no++
    end if
    {Next IF is included only if data is been processed for a Host Based IDS}
    if Role = Host then
        Y[i] = 3
        i++
    end if
    if Role = Station then
        Y[i] = 4.client_no {Y[i] is an integer formed from the concatenation of the string equivalents of 2 and client_no}
        i++
        client_no++
    else
        Y[i] = 5 {Assumed that Role = Other}
        i++
    end if
end for
Return(Y)
```

Algorithm 5 Role-Based Mapping III

Input: Array $X[]$ containing all MAC addresses in dataset .

Output: Array $Y[]$ containing integer mappings of the MAC addresses in $X[]$.
{Mapping of $X[i] = Y[i]$}

```
1: Sort(X)
2: i = 1
3: ap_no = 0 {Stores a count of the number of Access Points seen so far}
4: for every macaddr in X do
5:    Role = DetermineRole(macaddr) {Role can either be Broadcast, Access_Point, Host, Station or Other}
6:    if Role = Broadcast then
7:       Y[i] = 1
8:       i++
9:    end if
10:   if Role = Access_Point then
11:      Y[i] = 2.ap_no {Y[i] is an integer formed from the concatenation of the string equivalents of 2 and ap_no}
12:      i++
13:      ap_no++
14:   end if
      {Next IF is included only if data is been processed for a Host Based IDS}
15:   if Role = Host then
16:      Y[i] = 3
17:      i++
18:   end if
19:   if Role = Station then
20:      Y[i] = DecimalSum(macaddr)
21:      i++
22:   else
23:      Y[i] = 5 {Assumed that Role = Other}
24:      i++
25:   end if
26: end for
27: Return(Y)
```

6 Experimental Setup

Our experiments require labelled (i.e., each packet is labelled as attack vs normal) wireless network traffic datasets. To the best of our knowledge there is no such dataset that is publicly available. Thus, we had to generate such datasets in a lab environment. To this end, we set up three separate physical networks i.e. Network-I, Network-II and Network-III, as shown in Fig. 2 . The components of these networks include APs, a number of PDAs, laptops and desktop machines. While Network-I and Network-II are set up in the same manner, see

Fig. 2 (a), Network-III is set up as an ESS (Extended Service Set) with two APs, see Fig. 2 (b). The only difference between Network-I and Network-II are their APs. In Network I, an Airport based AP is employed, whereas in Network II a Cisco based AP is employed. In doing so, our aim is to create two different network environments. An AP is central to any infrastructure based wireless network, creating two networks with different APs implies different network environments. In an infrastructure based wireless network, the AP acts as the central server. For this reason all communication must pass and all traffic must be routed through the AP. This fact makes two networks with different APs sufficiently different enough to be considered as being different platforms. By setting up Network-III, our aim is to create a network environment, which involves more than one AP, and therefore represents more closely a real-world enterprise network. All the clients on all three networks are connected to the APs via 802.11 connections on channel 6. Attacks are generated on all networks using void11 installed on the attack machine. Data is collected on the monitoring machine using the data logging features of Kismet Wireless [19].

All the datasets collected on Network-I and Network-II have approximately 30 minutes worth of WiFi traffic data logged in them, while those in Network-III had approximately 10-mins worth of data in them. During the traffic logging period, we attacked the network twice. Each attack lasting for approximately between 2 -5 minutes.

6.1 Normal Traffic Generation

To ensure that normal traffic is also generated on our test networks while we collect our data, we implemented a web crawler using the Java 2 Platform, Micro Edition (J2ME) and installed it on all the clients on the network. This web crawler ensures a continuous stream of web browsing requests from the clients as the background normal traffic. The queue of web browsing requests is generated from parsing the html code of each site visited. This stream of web requests continued throughout the data collection period and was only interrupted during the course of an attack.

6.2 Attack Traffic Generation

The attack traffic consisted of the intermittent release of a stream of management frames into the network traffic. In all cases the source and the target MAC addresses of the frames are set to that of an AP, a client or the broadcast address **(ff:ff:ff:ff:ff:ff)**, depending on the scope of the attack. See Table 1 for an explanation of the parameters used by void11.

6.3 Feature Selection

Tcpdump traffic log files collected by Kismet wireless can be automatically replayed through Snort-Wireless but requires further processing before they can be employed for training and testing on the GP and ANN based IDSs. To this

end, an appropriate feature set is employed. 802.11 frames consist of several features but not all of them are relevant to this attack. Thus, based on the feature selection in previous work [3, 5], the following subset of features are used:

1. **Frame Control** - Defines the protocol version,
   type/subtype of a frame and flags. We are only interested in the frame type or frame subtype in this feature. The frame type or subtype consists of binary number representations of the different frame types/subtypes defined in 802.11 standard.

2. **Destination Address** - MAC address of the destination of a frame. Like the source address and Basic Service Set Identifier, the destination address field is a MAC address. The MAC address is a nominal variable with 22^6 possible values, as explained in Section 5.

3. **Source Address** - MAC address of the source of a frame.

4. **Basic Service Set Identifier (BSSID)**- MAC Address of an Access Point.

5. **Fragment Number** - Defines the fragment number in a particular sequence of frames. Like the Sequence number, this is a numeric variable with no specific range, these numbers are attached sequentially to each frame by the sending machine. The information contained in these features is utilised in the reassembling of the frames by the receiving machine.

6. **Sequence Number** - Defines the sequence number of a frame.

7. **Channel** - The transmission channel used for communication. This is a numeric variable with values between 0 and 13. Each number is a discrete representation of the frequency range blocks defined by the 802.11 standard.

All the features mentioned above were preprocessed and were converted to integers before being presented as input to the machine learning algorithms during the training phase employed in this work.

6.4 Data Set Generation

A total of 26 tcpdump traffic log files were collected during the course of our work. These log files were then passed through several processing stages, which include feature extraction, feature mapping for appropriate data types and the grouping of individual frames to form sessions. The processing of the dataset files was achieved by passing the tcpdump files through a number of scripts. Table 4 gives a summary of the resulting 26 dataset files. The table outlines the attack type, the physical network on which the original log files were collected, the number of files collected and the experiment set in which they were

employed. Each line in the datasets consists of the features extracted from 8 consecutive frames. Moreover, when at least one of these frames was part of the attack, the exemplar is labeled as attack, otherwise it was labeled normal. For more detailed statistics on our datasets, please see the appendix. In our first set of experiments, the objective is to investigate the cross-platform robustness of the aforementioned Machine Learning Based IDSs, while the objective of the second set of experiments is to test for cross-attack robustness.

7 Results

In intrusion detection, two metrics are typically used to quantify the performance of the IDS,

(i) Detection Rate (DR)

(ii) False Positive Rate (FP)

which are Eq. (3) and Eq. (4) respectively. A high DR and a low FP rate would be the desired outcomes. Evaluation of our results is based on the above criteria.

$$DR = 1 - \frac{\#FalseNegativeClassifications}{TotalNumberOfAttackConnections} \tag{3}$$

$$FP = \frac{\#FalsePositiveClassifications}{TotalNumberofNormalConnections} \tag{4}$$

7.1 Snort-Wireless Validation

We replayed the raw tcpdump files, which were preprocessed to produce the datasets in Table 4 through Snort-Wireless. This was done to validate the data. Snort-Wireless was able to detect the attacks that we launched using void11 against the networks in the datasets. The following is an example of Snort-Wireless de-authentication alert:

```
[**] [211:1:1] (spp_deauthflood) Deauthflood
 detected! Addr src: 00:03:93:ec:64:55 ->
 Addr dst: ff:ff:ff:ff:ff:ff,
 Bssid: 00:03:93:ec:64:55. [**]
```

The result of these tests show that Snort-Wireless, the conventional IDS, is robust in terms of cross-platform but is not robust in terms of cross-attack unless it is faced with a known attack.

7.2 Cross-Platform Robustness

Recent research has shown that machine learning based IDS for 802.11 data link layer attacks unlike their conventional counterparts are indeed susceptible

to diminished performance when used on networks other than that on which they were trained if no attention is paid to the representation of features employed [6]. In [6], it was shown that from one network to another, the performance of both Artificial Neural Network(ANN) and GP based IDSs can drop to 46% and 75% respectively, if no attention is paid to the feature representation technique, especially with the MAC addresses. In other words, the way a feature set is preprocessed for presentation to the learning algorithms plays an important role in cross-platform robustness. This forms the rationale behind the various map addressing schemes we developed in our research.

Our work in investigating cross-platform robustness was carried out by performing a 20-fold cross validation using the 20 datasets collected on Network-I and Network-II, see Table 4. This means that results were produced from all possible combinations of training and testing pairs of the datasets. Given the small size of our datasets, we believe that such an approach eliminates the effect of any artifacts of the datasets. The cross validation was performed using both GP and ANN algorithms. The final results were divided into four groups for each learning algorithm:

(a) Group-1: Results of Testing On Network-I datasets using solutions trained on Network-I datasets

(b) Group-2: Results of Testing On Network-II datasets using solutions trained on Network-II datasets

(c) Group-3: Results of Testing On Network-I datasets using solutions trained on Network-II datasets

(d) Group-4: Results of Testing On Network-II datasets using solutions trained on Network-I datasets

Group-1 and Group-2 are called within-platform results, whereas Group-3 and Group-4 are cross-platform results. The experiments were carried out using each one of the MAC address mapping techniques and the results are presented in the following sections.

7.2.1 Simple MAC Address Mapping

All the runs of the GP and ANN algorithms using this MAC mapping technique are able to produce best case solutions that achieved a 100% detection rate and a 0% FP rate, however we present the mean in Tables 5 and 6.

An analysis of the results show that a difference can be noticed in the detection rates from one platform to another. Within-Platform the mean DRs for both GP and ANN solutions are above 99% but this reduces to 75% and 46% respectively, for cross-platform experiments. Thus, we observe a significant drop in performance. These results show that the Machine Learning based solutions would have to be trained specifically for each network on which they run and re-trained every time there is a major change to the configuration of the network, if they were to be employed on a real-life network.

7.2.2 Decimal-Sum

The results for the mean performance using Decimal-Sum mapping technique with GP and ANN can be found in Tables 5 and 6 respectively. These results show that this technique is able to increase the DR across platform for GP and ANN solutions. However, Fig. 3 shows that Decimal-Sum mapping performs worse than Simple mapping technique within platform. We speculate that this might be as a result of the large numerical values produced by this mapping technique.

7.2.3 Role-Based MAC Address Mapping

The mean results for the three variants of this mapping technique can be found in Tables 5 and 6. These techniques all map based on the role of the MAC addresses on the network. We see that FP rates remain pretty much constant for all solutions within or across platforms. We however do not notice the significant drop in performance when comparing DRs within-platform with those across platform as observed with the Simple mapping technique. The average DRs remain relatively constant whether solutions are used within-platform or across platform. The mean DRs of all the three Role-Based techniques were higher than the mean DR for the Decimal-Sum technique both within-platform and across platform conditions.

In order to investigate if the results obtained using the different mapping techniques have any statistical significance, we performed an ANOVA (Analysis of Variance) Test [20]. The results of the ANOVA test shows that we can claim a statistically significant difference in the mean DRs of the different mapping techniques with 99% confidence. See Tables 12 and 13 in the Appendix for more detail on the results of our ANOVA test. Finally, Table 7 gives an overview of our evaluation of the mapping techniques used here according to the criteria of information loss, cross-platform robustness and whether they are implementable on an on-line network. It is important to take information loss into consideration because mapping several MAC addresses to the same numerical value in the preprocessed datasets makes it difficult to differentiate between two stations on the network, which share the same role, a situation, which is undesirable. On the other hand, creating a sorted list of all MAC addresses on a network might be difficult on an online wireless network due to their dynamic nature, so it is important to consider how easy it will be to implement the mapping technique on an online network. A fact that may be forgotten when dealing with previously collected network traffic data. It is our opinion that the Role Based techniques show the best promise, with the Role Based III having the edge.

Further experiments were carried to ascertain cross-platform robustness with GP learning algorithm using the 6 datasets collected on Network-III, Table 2. In these experiments, using the Role-Based III mapping scheme also achieves 99.9% detection rate with 0.3% false positive rate as its mean performance. In summary, we believe Role-Based III mapping technique is very successful

in terms of providing cross-platform robustness in both single or multiple AP networks. While the ANN based solutions perform better than the GP based solutions for cross platform results (the difference was not statistically significant), we did not pursue ANN solutions further for the following reasons: First GP based solutions produce more transparent solutions, which can be analysed manually, if desired and second, the GP based solutions required significantly less time for training as shown in Tables 5 and 6.

7.3 Cross-Attack Robustness

The Association Flood and the Authentication Flood are very similar attacks, differing only in the management frame exploited. Thus, we also investigate whether a machine learning based IDS solution trained on the Authentication attack can detect the Association attack and vice versa, because this can demonstrate its dependability and survivability under similar DoS attacks.

The results presented here followed the same training and testing procedures outlined in our previous set of experiments. The only difference is that in this case, only the four Association and Authentication datasets presented in Table 4 are used.

Table 8 shows the results for the FP and DRs, using the Role-Based III mapping scheme. We are able to achieve above 98% DR when employing the authentication solution against the association attack and 71% DR vice versa. Keeping in tune with idea of paying attention to feature representation, the Authentication and Association subtypes were represented as 11(1011) and 13(1101) respectively, in the datasets used in these experiments. This potential to be able to detect similar unseen attacks is an advantage for machine learning based detectors, conventional detectors like Snort-Wireless are incapable of detecting any exploit other than the one for which they have a signature, even if the new exploit is similar. Indeed, further experiments are needed to improve and analyse the implications of these results.

8 Conclusion and Future Work

In this work, we investigate the robustness of machine learning based 802.11 IDSs by building on previous work, which showed that machine learning based IDSs are more adaptable than conventional systems in the case of stealth attacks. In our first set of experiments, using only the de-authentication attack and using GP and ANN based classifiers, we ascertained that issues exist with cross-platform robustness in machine learning based solutions for 802.11 Link Layer DoS Attacks if simple representation techniques are used for the features employed. By focusing on the feature set, we identified the MAC address mapping techniques in the feature set as a significant part of the problem. Our solution involved a number of novel schemes for mapping the MAC addresses. Our results show that the Role-Based mapping paradigm proposed is not only able

to maintain detection capability within-platform but also significantly improves the detection rate in cross-platform situations in terms of the mean performance.

In our second set of experiments, we extended our analysis further by testing our techniques on data sets collected on larger, more realistic real-world networks and using other link layer attacks. Our results showed that GP-based classifiers in such environments achieve a mean performance of 99% on cross-platform experiments. Moreover, cross-attack environment results also show that a GP based IDS, trained on one type of DoS attack can even detect a different DoS attack from the one it was trained on. These results show that a machine learning IDS, such as the one evolved by GP, not only has high dependability but also has survivability capabilities against cross and stealth attacks, unlike conventional IDS such as Snort-Wireless.

Our work has ascertained that using a Role Based paradigm in the mapping of MAC address information in data used for training machine learning based detectors is imperative to producing robust detectors in WiFi networks. This conclusion is not limited to MAC addresses or WiFi networks but can be extended to other network addresses like IP addresses and to other types of computer networks. We also ascertain that machine learning based detectors are capable of detecting DoS attacks that they were not trained to detect when the mode of launching the attack is sufficiently similar. These conclusions imply that the goal of developing a commercially viable machine learning based IDS that can be used to detect a variety of WiFi attacks is achievable. Future work would explore not only similar experiments on bigger datasets (if available) but also would evaluate the limits of cross-attack robustness on attacks other than DoS.

Acknowledgements

This research is supported by the NSERC Discovery and the CFI New Opportunities grants. The authors would also like to acknowledge the staff of Palomino System Innovations Inc., based in Toronto, Ontario and Telecoms Applications Research Alliance (TARA), based in Halifax, Nova Scotia for their support in completing this work. This work is conducted as part of the Dalhousie NIMS Lab at http://www.cs.dal.ca/projectx/.

References

[1] W. A. Arbaugh, N. Shankar, Y. Wan, and K. Zhang, "Your 802.11 wireless network has no clothes," *IEEE Wireless Communications*, vol. 9, pp. 44 – 51, December 2002.

[2] M. Malekzadeh, A. Azim, Z. Zulkarniam, and Z. Muda, "Security improvements for management frames in ieee 802.11 wireless networks," *International Journal of Computer Science and Network Security*, vol. 7, no. 6, pp. 276 – 284, June 2007.

[3] P. LaRoche and A. N. Zincir-Heywood, "Genetic programming based wifi data link layer attack detection," in *CNSR 2006*. Los Alamitos, CA 90720-1314: IEEE Computer Society, May 2006, pp. 285 – 292.

[4] Y. Liu, D. Tian, and B. Li, "A wireless intrusion detection method based on dynamic growing neural network," *1st International Multi-Symposium on Computer and Computational Sciences*, 2006.

[5] A. Makanju, P. Laroche, and N. Z. Heywood, "A comparison between signature and GP-Based IDSs for link layer attacks on WiFi networks," in *Proceedings of the 2007 IEEE Symposium Series on Computational Intelligence*, April 2007.

[6] A. Makanju and A. N. Zincir-Heywood, "Investigating cross-platform robustness in machine learning based IDSs for 802.11 networks." *International Journal of Computer Science and Network Security*, pp. 1 –9, June 2007.

[7] I.-S. S. Board, *ANSI/IEEE Std. 802.11*, 1999th ed., IEEE, New York, NY, USA, 2003.

[8] R. Floeter, "Void11: A free implementation of some basic 802.11b attacks, http://www.wirelessdefence.org/contents/void11main.htm," Retrieved from the Web., September 2007.

[9] J. Malinen, "Host AP Driver , http://hostap.epitest.fi," Retrieved from the Web., September 2007.

[10] M. Crosbie and E. Spafford, "Applying genetic programming to intrusion detection," in *AAAI Symposium on Genetic Programming*, J. K. E.V. Siegel, Ed., AAAI. Cambridge, MA, USA: MIT, 1995, pp. 1 – 8.

[11] Sourcefire-Inc, "Snort - the de facto standard for intrusion detection/prevention, http://www.snort.org," Retrieved from the Web., September 2007.

[12] A. Lockhart, "Snort wireless, http://www.snort-wireless.org," Retrieved from the web., September 2007.

[13] J. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbour, Michigan, USA: University of Michigan Press, 1975.

[14] J. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems," Computer Science Department , Stanford University, Tech. Rep., 1990.

[15] C. Gathercole and P. Ross, "Dynamic training subset selection for supervised learning in genetic programming," *Parallel Problem Solving from Nature III*, vol. 866, pp. 312 – 321, 1994.

[16] D. Song, M. Heywood, and A. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, pp. 225 – 239, 2005.

[17] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1999.

[18] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[19] M. Kershaw, "Kismet wireless, http://www.kismetwireless.net," Retrieved from the Web., September 2007.

[20] C. U. ISERP, "About the anova test, http://ccnmtl.columbia.edu/projects/qmss/anova_about.html," Retrieved from the Web, September 2007.

Appendix

Client De-authentication, Authentication and Association Attacks on Network-III

Our rationale for running this new set of experiments using the all the datasets collected on Network-III is based on the following reasons:

- Our experiments are performed using only a broadcast de-authentication flood attack (for cross-platform robustness)as a base line for the attack class we are investigating. Thus, the natural next step is to perform other attack types from this class. The new attacks used are
  Client De-authentication Flood, Authentication Flood and Association Flood.

- We have only trained network based IDSs. Client de-authentication attacks can present a chance to test our role based mapping framework on the host side. Thus, Client de-authentication attack files are all preprocessed to be employed on the host side, too.

- Except in the case of small home networks, most real world 802.11 networks contain more than one AP. We therefore capture our second set of datasets on a bigger test network i.e. Network-III.

- Data link layer wireless DoS attacks cannot be directed at an entire wireless network (except if several attacks are launched simultaneously), they can only be launched against a BSS (or the clients in the BSS) within an ESS. This gives us an opportunity to check for robustness between attacks on different BSSs within the same network.

In this case, we only present the results of the Role Based III mapping technique here because we observe it is the best option among the other mapping techniques based on our evaluation criteria. The results presented here followed the same training and testing procedure outlined in our previous set of experiments. Final results are divided into four groups:

(a) Group-1: Results of testing on datasets that contain attacks on AP1 (or clients attached to AP1) using solutions trained on datasets that contain attacks on AP1 (or clients attached to AP1).

(b) Group-2: Results of testing on datasets that contain attacks on AP2 (or clients attached to AP2) using solutions trained on datasets that contain attacks on AP2 (or clients attached to AP2).

(c) Group-3: Results of testing on datasets that contain attacks on AP1 (or clients attached to AP1) using solutions trained on datasets that contain attacks on AP2 (or clients attached to AP2).

(d) Group-4: Results of testing on datasets that contain attacks on AP2 (or clients attached to AP2) using solutions trained on datasets that contain attacks on AP1 (or clients attached to AP1).

Groups (1) and (2) are called results within-platform, whereas (3) and (4) are called cross-platform. Table 9 shows the mean detection and false positive rates for each of the three attack types investigated. The results show that the GP solutions are able to detect all the attack types both within platform and across platform (as defined for this experiment).

Dataset Statistics

The tables in this section i.e. Table 10 and Table 11, give details of the datasets which were used to train and test our GP and ANN IDSs.

Anova Results

The tables in this section i.e. Table 12 and Table 13, provide detail results of the Two-Factor ANOVA tests which we performed on our results for GP classifier and the ANN classifier respectively.

Tables

Table 1: Void11 general parameters and default values.

Parameter	**Meaning**	**Default Value**
-t	Type of attack	Deauthentication Attack
-d	Delay	10,000μsecs
-s	Station MAC Address	ff:ff:ff:ff:ff:ff
-S	Service Set Identifier (SSID)	“ ”
-B	Network MAC Address	Scan for networks

Table 2: GP Parameters employed in this work for all cases studied.

Parameter	**Setting**
Population Size	125
Maximum Number of Pages	32
Page Size	8 Instructions
Maximum Working Page Size	8 Instructions
Crossover Probability	0.9
Mutation Probability	0.5
Swap Probability	0.9
Tournament Size	4
Number of Registers	8
Function Set	(+,-,*,/)
Terminal Set	(0,...,255) $\cup$ (*r0*,...,*r7*)
RSS Subset Size	5000
DSS Subset Size	50
RSS Iteration	1000
DSS Iteration	100

Table 3: Artificial Neural Network Parameters employed in this work for all cases studied.

Parameter	**Setting**
Momentum	0.2
Learning Rate	0.3
No. Epochs	500
Random Seed for Weights	0
No. of hidden nodes	30

Table 4: Dataset Summary

Attack Type	**Network**	**# of Files**	Experiment
De-authentication	I	10	Set-I
De-authentication	II	10	Set-I
De-authentication	III	2	Set-II
Authentication	III	2	Set-II
Association	III	2	Set-II

Table 5: Mean Performance Using Genetic Programming

Within Platform			
	FP	DR	TIME
Simple	0.01	0.99	42.28
Decimal Sum	0.06	0.90	26.27
Role Based I	0.01	0.98	26.14
Role Based II	0.01	0.97	25.64
Role Based III	0.02	0.97	26.68

Across Platform			
Simple	0.02	0.75	41.81
Decimal Sum	0.06	0.81	25.28
Role Based I	0.01	0.98	26.32
Role Based II	0.01	0.98	25.68
Role Based III	0.02	0.96	26.69

Table 6: Mean Performance Using Artificial Neural Networks

Within Platform			
	FP	DR	TIME
Simple	0.02	1.00	83.05
Decimal Sum	0.01	0.98	87.66
Role Based I	0.00	0.99	92.39
Role Based II	0.01	0.99	83.14
Role Based III	0.01	0.98	83.14

Across Platform			
Simple	0.05	0.46	83.05
Decimal Sum	0.01	0.97	87.66
Role Based I	0.00	0.99	92.39
Role Based II	0.00	0.99	83.14
Role Based III	0.01	0.98	83.14

Table 7: MAC Address Mapping Scheme Evaluation

	Info. Loss	**Cross-Platform Robustness**	**Implementable Online**
Simple	No	No	No
Decimal_Sum	No	Yes(Minimal)	Yes
Role Based I	Yes	Yes	Yes
Role Based II	No	Yes	No
Role Based III	No	Yes	Yes

Table 8: Mean Performance Using Genetic Programming with Role-Based III Mapping

Within Platform			
	FP	DR	TIME
Auth. vs. Assoc.	0.01	0.98	33.85
Assoc. vs. Auth.	0.00	0.67	32.85
Across Platform			
Auth. vs. Assoc.	0.01	0.94	36.81
Assoc. vs. Auth.	0.00	0.71	32.05

Table 9: Mean Performance Using GP with Role Based III Mapping

Within Platform			
	FP	DR	TIME
Client De-Auth.	0.00	0.95	29.99
Authentication	0.00	0.9991	32.81
Association	0.00	1.00	33.50

Across Platform			
Client De-Auth.	0.00	0.95	31.60
Authentication	0.00	0.99	36.99
Association	0.00	0.99	34.25

Table 10: Experiment Set-I Data Sets

File	**Size**	**# of Attacks**	**Type**	**% Attack**	Network
A1	23960	3302	De-authentication	13.78	I
A2	20960	3440	De-authentication	16.41	I
A3	15880	3472	De-authentication	21.86	I
A4	15640	3258	De-authentication	20.83	I
A5	23160	3526	De-authentication	15.22	I
A6	17280	3505	De-authentication	20.28	I
A7	19120	3048	De-authentication	15.94	I
A8	18560	3835	De-authentication	20.66	I
A9	22080	3294	De-authentication	14.91	I
A10	21680	3253	De-authentication	15.00	I
C1	20160	3685	De-authentication	18.27	II
C2	24320	3669	De-authentication	15.08	II
C3	19040	3928	De-authentication	20.63	II
C4	27120	3890	De-authentication	14.34	II
C5	23880	3801	De-authentication	15.91	II
C6	24000	3466	De-authentication	14.44	II
C7	16040	4001	De-authentication	24.94	II
C8	23360	3297	De-authentication	14.11	II
C9	18760	3403	De-authentication	18.14	II
C10	17360	3086	De-authentication	17.77	II

Table 11: Experiment Set-II Data Sets

File	Size	# of Attacks	Type	% Attack	Network
D1	11000	4253	De-authentication	38.66	III(Access Point I)
AU1	12520	6273	Authentication	50.10	III(Access Point I)
AS1	13280	3449	Association	25.97	III(Access Point I)
D2	10040	5120	De-authentication	50.99	III(Access Point II)
AU2	8360	3355	Authentication	40.13	III(Access Point II)
AS2	7640	3017	Association	39.48	III(Access Point II)

Table 12: ANOVA Two-Factor With Replication: Genetic Programming

Network-I						
Source of Variation	SS	df	MS	F	P-value	F crit
Sample	2.52	1	2.52	46.28	1.88E-11	6.66
Columns	2.83	4	0.71	13.01	2.68E-10	3.34
Interaction	3.12	4	0.78	14.34	2.36E-11	3.34
Within	48.40	890	0.05			
Total	56.87	899				

Network-II						
Source of Variation	SS	df	MS	F	P-value	F crit
Sample	0.69	1	0.69	21.28	4.48E-06	6.66
Columns	2.86	4	0.71	22.18	1.47E-17	3.34
Interaction	0.79	4	0.20	6.13	7.11E-05	3.33
Within	31.93	990	0.03			
Total	36.27	999				

Table 13: ANOVA Two-Factor With Replication: Artificial Neural Network

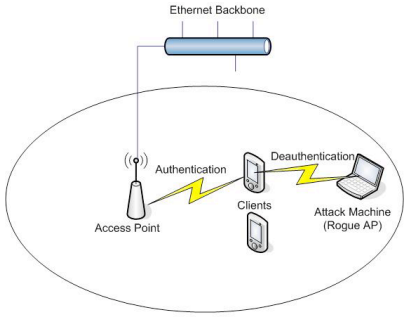
Network-I						
Source of Variation	SS	df	MS	F	P-value	F crit
Sample	2.52	1	2.52	46.28	1.88E-11	6.66
Columns	2.83	4	0.71	13.01	2.68E-10	3.34
Interaction	3.12	4	0.78	14.34	2.36E-11	3.34
Within	48.40	890	0.05			
Total	56.87	899				

Network-II						
Source of Variation	SS	df	MS	F	P-value	F crit
Sample	0.69	1	0.69	21.28	4.48E-06	6.66
Columns	2.86	4	0.72	22.18	1.47E-17	3.33
Interaction	0.79	4	0.20	6.13	7.11E-05	3.34
Within	31.93	990	0.03			
Total	36.27	999				

Figure Captions

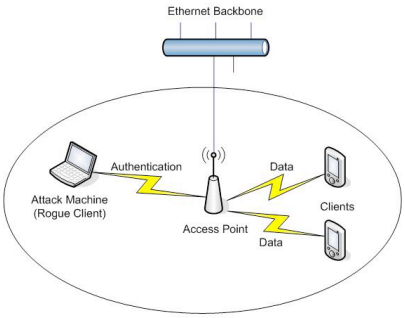
Fig. 1: 802.11 data link layer DoS Attack Types (a) De-authentication Flood (b) Authentication Flood (c) Association Flood.
Fig. 2: Setup of Test Networks (a) Network-I and Network-II (b) Network-III.
Fig. 3: Within-Platform DR for GP Using Simple and Decimal-Sum Mapping Techniques.
Fig. 4: Decimal-Sum Mapping Technique.

Figures



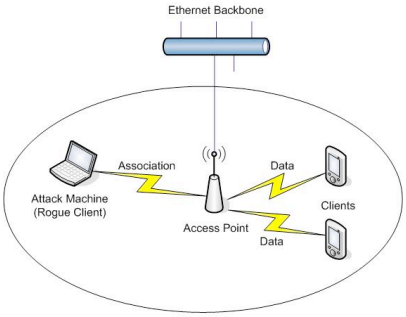
De-authentication Attack

(a) De-authentication Attack



Authentication Attack

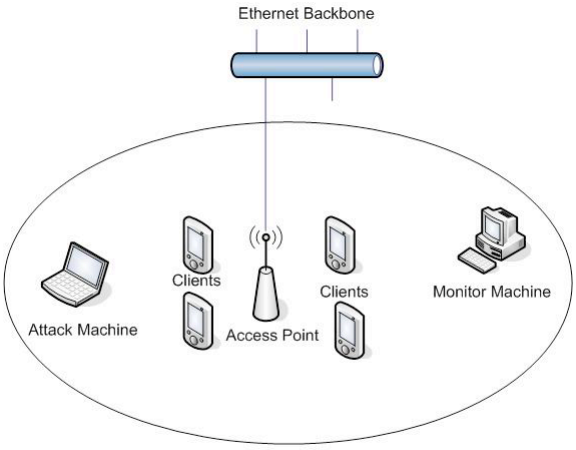
(b) Authentication Attack



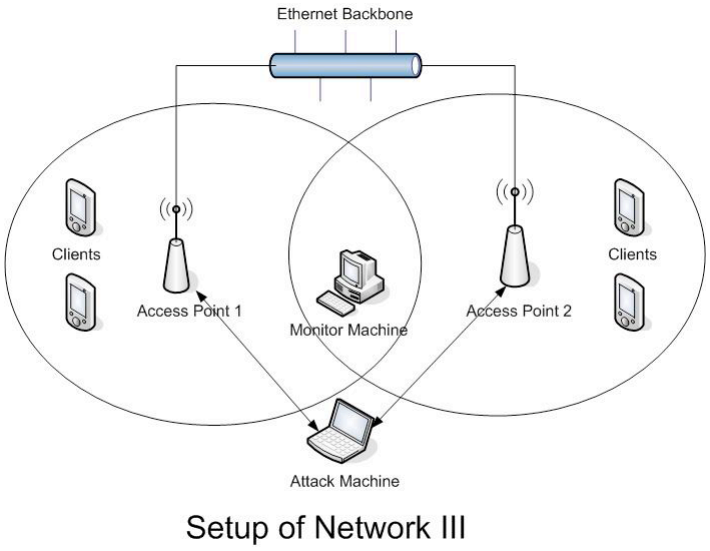
Association Attack

(c) Association Attack

Figure 1: 802.11 data link layer DoS Attack Types (a) De-authentication Flood (b) Authentication Flood (c) Association Flood



(a) Network-I and Network-II



(b) Network-III

Figure 2: Setup of Test Networks (a) Network-I and Network-II (b) Network-III.

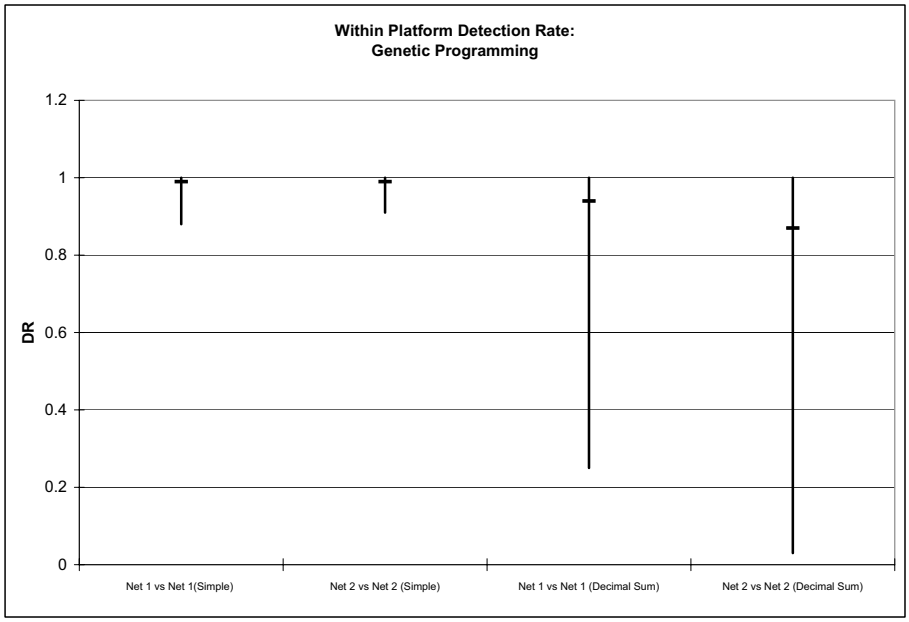


Figure 3: Within-Platform DR for GP Using Simple and Decimal-Sum Mapping Techniques.

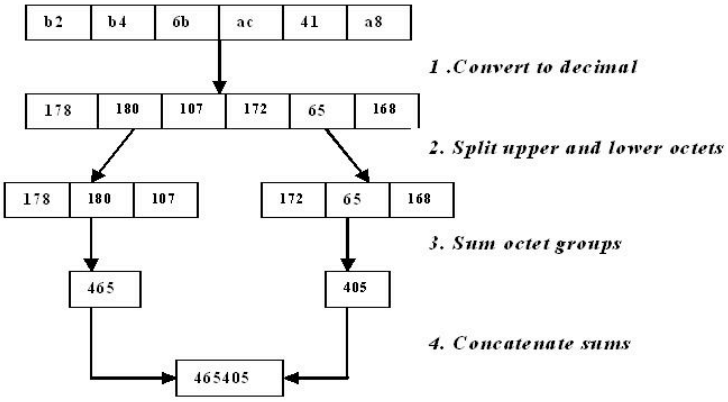


Figure 4: Decimal-Sum Mapping Technique.