Genetic Programming Based WiFi Data Link Layer Attack Detection

Patrick LaRoche, A. Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia
B3H 1W5, Canada
plaroche@cs.dal.ca, zincir@cs.dal.ca

Abstract

This paper presents a genetic programming based detection system for Data Link layer attacks on a WiFi network. We explore the use of two different fitness functions in order to achieve both a high detection rate and a low false positive rate. Results show that the detection system developed can achieve a detection rate above 90% and a false positive rate below 1%.

1 Introduction

The wireless network protocol IEEE 802.11, also referred to as WiFi, is a protocol which has been deployed in a growing number of locations and environments. This diversity has resulted in deployments of varying purposes and goals, from providing a household with an inexpensive local area network (LAN), to supporting an entire office building. Due to this growing popularity and exposure, an increasing number of exploits are being discovered, undermining the use of the 802.11 network protocol.

The very nature of a wireless network, no matter the protocol being used, opens the network up to vulnerabilities not present in wired networks. These issues are due to data being transferred over open airwaves, where anyone with the appropriate device can intercept the signal. To solve this *openness*, WiFi networks provide security features, such as encryption and client verification. These measures, being important to the overall trust and use of the WiFi protocol, have received the majority of emphasis in research and development. Unfortunately, these solutions do not cover all the possible weaknesses in the 802.11 protocol. There are other known exploits with the 802.11 protocol that allow users of malicious intent to disrupt the use of the network by attacking the lower layers of the protocol, rendering the network unusable by its clients.

Traditionally, intrusion detection systems (IDSs) are used to detect attacks against the integrity, confidentiality and availability of computer networks [13]. In order to build effective IDSs and automate the detection process, various machine learning and data mining techniques have been proposed. These techniques include neural networks [15], data mining [20], decision trees [18], genetic algorithms (GA) [16, 11], and genetic programming (GP) [19, 12, 3]. In general, data mining techniques are introduced to identify key features and machine learning and AI techniques are introduced to automate the classification of normal and attack traffic / behavior on the network. To the best of our knowledge, works utilizing genetic algorithms and genetic programming were all based on the TCP/IP protocol stack. This corresponds to the third and fourth layers of the network stack. On a WiFi network running a TCP/IP protocol stack, however, there exist attacks based on vulnerabilities in the physical and data link layers, the first and second layers respectively.

Moreover, past work applying evolutionary methods to network intrusion detection has focused on connection-based attacks. In doing so, much progress has been made in providing an efficient and effective intrusion detection system (IDS) using GP as the tool to create the rules in which to detect attacks [19]. This does not necessarily make a GP or GA based IDS effective in detecting network protocol specific attacks at layer one and two, such as in the case of WiFi networks.

In this paper we present our work towards developing a GP based IDS for Layer 2 attacks, using known WiFi network exploits for training and testing. By using past research as a starting point, we aim to build on the past successes (quick training time, transparent solutions) while adapting to the challenges of intrusion detection on the WiFi network.

The remainder of this paper is organized as follows: Section 2. details the background information on WiFi networks. Section 3 describes the GP technique employed.

Section 4 details our approach for developing GP based IDS. Section 5, presents the results and conclusions are drawn in Section 6.

2 WiFi Networks

This work is focused on WiFi networks. Such networks have increased in popularity over the past few years, so much so that their use as a last mile solution for Internet connectivity has become common, with homes and businesses alike. It has become so popular, that in the year 2001, the market for WiFi networks exceeded \$1 Billion Dollars [1]. This wide spread (and growing) deployment of WiFi networks makes them a growing area for research, both in improvement of service, but also in security and reliance of the service they provide. In this section, we discuss the basics of WiFi networks, current security features and the known exploits of the WiFi protocol. This is not an exhaustive description of the WiFi protocol, as that is beyond the scope of this paper.

2.1 Topology of a WiFi Network

WiFi networks, in a broad sense, consist of clients communicating via the wireless connection protocol IEEE Std 802.11b [8]. The clients are anything from laptops with WiFi enabled network interfaces (wireless cards), WiFi enabled PDAs, or even access points that connect the WiFi network to another type of network.

WiFi networks can be classified in two categories:

Ad-Hoc Networks composed solely of stations within mutual communication range of each other via the wireless medium (WM). An ad-hoc network is typically created in a spontaneous manner [8].

Managed (or infra-structured) Networks composed of a distribution system medium (DSM, such as an Ethernet LAN), an access point (AP), and clients [8]. An example could be an access point connected to a LAN, that has zero or more clients connected to it via WiFi. The access point is the central "manager" of the network.

We only mention Ad-Hoc WiFi networks here for completeness, as our work focuses on Managed WiFi networks. It is important to note that the most basic components of a managed network are an access point and its clients. If you are to remove the access point from the network, the clients lose their connection and do not regain the connection through connecting directly to each other (which would make it an Ad-Hoc network). This distinction is important for our work. From this point on, we are discussing managed networks only, even though some statements would hold true for ad hoc networks as well.

2.2 Management Frames

WiFi networks have a series of MAC frame types, as described in the IEEE 802.11 Standard (see [8]). Of concern for our work are the management frames which manage the OSI Layer 2 of the 802.11 protocol. The management frame type allows clients to associate with (or conversely disassociate from) the network via an access point (AP), as well as maintaining a channel for communications to proceed. We focus on a subset of the management frame subtypes; *Association request* , *Deauthentication* and *Disassociation*. These subtypes allow clients to join, leave, and be told to leave WiFi networks.

In order for a client to establish a connection with an existing WiFi network, it first needs to associate with an AP. This *association* is established through searching for an access point on a specific BSSID (basic service set identification, an identifier of a specific WiFi network) on a given channel (usually on a range of channels). Once the client has found an access point on the desired BSSID and channel, the following procedure is used to establish a connection with the AP (simplified from [8]) :

1. The client can transmit an association request to an AP with which that client is authenticated.

2. If an Association Response frame is received with a status value of successful, the client is now associated with the AP.

3. If an Association Response frame is received with a status value other than successful or a timeout value passes, the client is not associated with the AP.

This procedure relies on a one-way trust, the client trusting the validity of the AP, not visa versa. This distinction is important. At no time does the client require that the AP prove that it is a valid AP.

A procedure exists for a client to *disassociate* itself from a network. As described in the IEEE 802.11 Standard ([8]), in order to disassociate, the client must send a disassociation subtype management frame. This frame type can be sent in either direction, from the client to the AP or from the AP to the client. The frame contains the hardware address of the client that is being disassociated (the broadcast address in the case of an AP disassociating with all associated clients). It also contains the hardware address of the AP with which the client is currently associated. [8]

Similarly, a client or AP can invalidate an active authenticated connection through the use of *de-authentication* subtype of the management frame type. This frame can again be sent from a client to an AP, an AP to a client, AP to AP, or even client to client. This frame subtype contains the same information as the disassociation subtype.

2.3 Known Exploits

The 802.11 Data Link layer (OSI Layer 2) includes functionality that addresses issues specific to a wireless network [1]. For example, the ability to search for networks, broadcast networks, join and leave networks are all taken care of by Data Link layer frames, specifically management frames. As mentioned earlier (and in other works [1]), there is an implicit trust in the Ethernet address of the sender of a management frame. This implicit trust opens up the door to a group of Denial of Service (DoS) attacks on the 802.11 network. If a malicious client, or hacker, fakes its Ethernet address (a trivial task with most operating systems and a small Internet search) it can then send management frames onto an WiFi network with the network assuming the hardware address is valid. This can cause problems if the hardware address has been set to that of another valid client, or to that of a valid access point.

With the increase in deployment of WiFi networks, the number of tools to exploit them has also seen a rise. Authors are aware of several tools that allow a client to create DoS attacks, specifically Airjack [14], void11 [10] and an entire boot-able CD, Auditor Security Collection [4]. It is important to note that the tools mentioned here are readily available on the Internet, and in all cases have been provided for the use of research, not for malicious use. For example the Auditor Security Collection includes many tools that are for general network monitoring. With the availability of such tools and the relative simplicity of implementing them, however, one can deduce that these types of DoS attacks pose a real threat to the usability of WiFi networks.

DoS attacks are not the only exploits or security weaknesses that exist on WiFi networks. Much work has been done in addressing other security concerns. Papers such as [5] and [2] point out security weaknesses of the encryption algorithms implemented on WiFi networks, and that is assuming the network administrators have enabled encryption. This is not always a fair assumption given the wide spread adoption of WiFi network for home networking, where the network administrator could be assumed to have no networking knowledge at all. This has resulted in a slew of acronyms, protocols and protocol extensions (WEP, WPA, RADIUS, 802.11g, 802.11i, 802.1X) that have all tackled issues from speed to security concerns; however, to the best of our knowledge none of these deal with ensuring that the network remains *usable*. By this we mean that it is still possible to perform simple forms of DoS attacks even on the newest and latest protocol version of 802.11. For this purpose, we have chosen to focus on a specific subset of DoS attacks, described below.

2.4 DoS Attacks

The DoS attack we focus on in this work is that caused by the attacker sending de-authentication frames onto the wireless network (as shown in Figure 1) . An attacker chooses a target on the network (which has been gathered by actively monitoring the WiFi network traffic using a tool such as kismet [9]) and then spoofs their Ethernet address. At this point, the attacker then sends a de-authentication attack to the accesspoint in which the target is associated. This causes the accesspoint to send a de-authentication frame back at the target, removing the target from the network, preventing it from sending or receiving any further communications. The duration of which the target remains removed from the network depends on the frequency in which it attempts to regain network access.

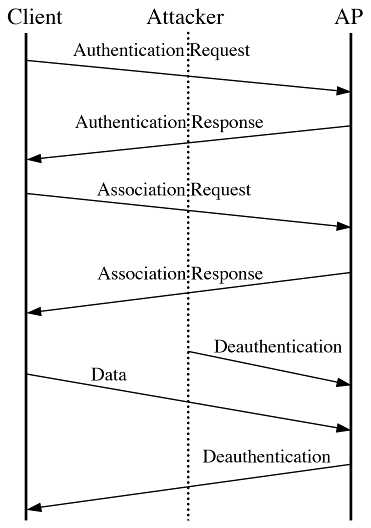

Figure 1. Graphical depiction of a de-authentication attack. [1]

This attack, simple in its implementation, is effective for several reasons. The first reason is that the attacker can choose the scope of the attack, be it a single network client, several, or an entire network (by choosing the access point itself as the spoofed address). The result is that an attacker can completely disrupt all communications on a single network, no matter how many clients the network may have. Secondly, if the attacker has targeted a single client, once the client is removed from the network the attacker can then continue with several other attacks, such as a man in the middle attack [17].

With WiFi networks, a client does not require a fixed, physical link to the network, only to be in range of the networks signal in order to gain a connection is enough to launch an attack. This freedom allows client machines to attempt malicious behavior from locations that are harder to pinpoint (i.e they could be outside the building, etc), rendering the network even more vulnerable to attacks. Due to this added vulnerability, combined with the simplicity of the de-authentication attack, we consider this to be a real security threat, one that requires a system to detect and notify the network administrators in a quick and reliable manor.

3 GP Based Intrusion Detection Systems

Due to previous success in application of GP based IDSs to higher level attacks [19], we have chosen to take the same approach in creating an IDS for layer two attacks. In previous work, a page based linearly structured GP was employed [7] as well as the utilization of a RSS-DSS algorithm which scales GP to data sets consisting of hundred of thousands of exemplars [19]. For the work we present here, we have used a similar approach, but concentrated on the devopment of an appropriate fitness function and feature set using common DoS attacks on WiFi networks for training and testing of the system.

For this work, our goal is to not only to develop a machine learning technique to detect attacks exploiting known layer 2 faults with the WiFi protocol with a high accuracy and low false positive rate, but also be able to develop these solutions in a quick, transparent matter. By doing so, the solution in which the GP develops can easily be justified through domain knowledge and be deployed for use as signatures even in standard IDSs such as Snort-wireless.

3.1 Linear Page Based Genetic Programming

Linear Page Based GPs consist of a sequence of integers that once decoded, form the basis of a program in which the output is taken from the best performing register, as defined by the fitness function. In order to decode this linear set of instructions, each integer is mapped to a valid instruction from the defined instruction set. The instruction set consists of operands and either a source or destination register. The operands in our work are a set of register arithmetic functions, while the source and destinations are a set of valid general-purpose registers. The decoding of a sequence then creates a program that consists of simple register level transformation [19]. Upon completion of execution of the program, the output is taken from the best performing register.

The sequence of integers are grouped in *pages*, each page consisting of the same number of integers (therefore the same number of instructions). The crossover operation performs a crossover on an entire page, preserving the total number of pages in an individual. The mutation operator selects one instruction with uniform probability and performs an Ex-OR operation between this and a bit sequence created with uniform probability. A second crossover operator performs a swap of two instructions within the same individual (selected again with uniform probability) [7]. The page size itself, which controls the number of instructions per individual, is dynamically modified depending on the fitness level of the population. If the fitness level has not changed for a specified window, the page size is increased. This pattern will continue until a maximum page size is reached, at which point the page size is dropped back down to the initial starting page size. This entire process is continued until the GP has reached either optimal fitness, or some sort of previously set stopping criteria. Results show that the dynamic page size algorithm is significantly more efficient then a fixed page size [7].

3.2 RSS-DSS Algorithm

The Random Subset Selection - Dynamic Subset Selection (RSS-DSS) algorithm mentioned above is a technique implemented in order to reduce the computational overhead (therefore time to train the GP) involved with applying GPs to large data sets. To do so, the RSS-DSS algorithm utilizes a hierarchical sampling of training exemplars, dividing the problem into two levels [19]. We present here a brief overview of how the algorithm functions for completeness, as we have implemented it in our GP, but it is not the focus of our work.

The first level of RSS-DSS divides the training set into blocks of equal size, the second level chooses (stochastically) a block and places it in memory (RSS). Level 2 performs the DSS step, as it dynamically selects a subset of the set in memory (the tournament selection). The dynamic selection is based on two metrics the GP maintains, the age of the exemplar and the apparent difficulty of the exemplar [6]. The tournament individuals are then trained on the current subset, genetic operators are applied, and then placed back in the subset. This DSS is continued until a maximum number of DSS iterations or a stopping criteria is met, then the algorithm returns to the RSS step, selecting another block to place in memory and repeats DSS. This entire process continues until a maximum number of RSS iterations or the stop criteria has been met.

The RSS-DSS algorithm removes the requirement to train on the entire data set, instead using only a small subset of the data set that represents the more difficult or least recently encountered exemplars. This allows the GP to train more efficiently then standard techniques, with results being comparable or better than more common GP training techniques [19].

4 Approach

Our L-GP based IDS requires training and testing on labeled data sets. That is to say, we require data files that have a fixed number of input fields per line, with a corresponding output field indicating wether the line represents an attack (binary 1) or normal (binary 0) exemplar. Each exemplar represents a management packet on a WiFi network. To the best of authors' knowledge, no known public database exists of wireless network traffic to use in training IDSs based on learning algorithms. To this end, we first set upon creating such a data set in order to train and test our L-GP based IDS.

In order to create data sets, we attacked a test network consisting of 7 clients, one access point, a hacker laptop and a monitoring machine. The AP was the latest Apple Airport Base Station, the clients were a Macintosh Mini, Macintosh iBook (both running the Mac OS X 10.4.1 operating system) and 5 Palm Tungsten C PDAs (running Palm OS 5.2.1). The clients connected to the AP via an 802.11b network on channel 6. The attack machine was an HP Tablet PC using a Prism 2 based WiFi card running the Auditor Security Collection (a Knoppix variant) operating system. The monitoring machine was an Intel based Desktop computer running Debian and using Kismet for monitoring the wireless network of the AP.

Using this network we implemented a DoS attack directed at the AP. The packet stream gathered via the monitoring machine indicated that the attack required a stream of management frames of subtype 12 (indicating a de-authentication frame) with the source and BSSID Ethernet addresses to be that of the target, and the destination address to be that of the broadcast address (ff:ff:ff:ff:ff:ff). The frequency and duration of the transmission of the attack frames depended upon the desired persistence of the attacker, in our experiments we applied a number of attacks, varying in duration (in both time and number of attack frames sent).

Upon completion of the attacks on the test network, the packet dump file was replayed through Snort (with the snort-wireless patches applied) in order to validate (and for later comparison) our attack procedures. Snort was chosen as it is both a common IDS as well as open source. This allows us to easily implement the system as well as analyze how it detects the de-authentication attack. Snort uses a signature based on certain user defined metrics to detect de-authentication attacks. The two most crucial metrics are the number of de-authentication frames that are to be considered an attack, and the time window in which this number must be met. By defining these numbers appropriately, we can configure Snort to detect all of the attacks we implemented on the network (or conversely if we cannot choose appropriate metrics, Snort can detect none of them). The output of Snort on our testing data set is shown below:

```
11/30-17:02:26.730859 Deauth flood
reported
Assoc.  Req.  0:3:93:EC:64:55 -       >
FF:FF:FF:FF:FF:FF
bssid:  0:3:93:EC:64:55 Flags:  Wep Ord

11/30-17:03:10.655207 End Deauth flood
:  -> Addr dst:  ff:ff:ff:ff:ff:ff,
3971 deauth frame reported.
```

This indicates that Snort successfully detected our attack (and created no false positive, i.e did not indicate an attack where one did not exist). It is important to note, that had we defined the Snort configuration file with different metrics for the attack, it would have missed the attack. In this work, our objective is to eliminate this reliance on *a priori* knowledge so that a solution developed by a GP based IDS would not require it.

As mentioned earlier, the data we use is made up of management frame packets. A management frame on a WiFi network consists of several fields. For our L-GP based IDS, we choose to train and test on the following features extracted from the management frames:

1. Frame Control - indicates the subtype of the frame
2. DA - destination address of the packet
3. SA - sender address of the packet
4. BSSID - Ethernet address of the access point
5. Fragment Number - from the sequence control field
6. Sequence Number - from the sequence control field
7. Channel - the channel the transmission is occurring over

In total, this gives us seven inputs, and one output (attack label). This feature set from each packet was chosen based on our knowledge of the attack type. Our work here is to see if the GP can use this information to then learn to detect the attack.

4.1 Fitness Function selection

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

(i) Detection rate (DR)

(ii) False Positive rate (FP rate).

A high DR and low FP rate would be the desired outcomes. In the instance of an unbalanced data set (more of one type of exemplar then the other, in our case more normal than attack) an evolved solution can survive by simply learning to label all of the exemplars as the larger type in the data set. This survival technique will provide a high DR, but also high FP rate, an undesirable result.

$$DR = 1 - \left(\frac{\#FNClassifications}{TotalNumberofAttackConnections}\right) \quad (1)$$

$$FPRate = \left(\frac{\#FPClassifications}{TotalNumberofNormalConnections}\right) \quad (2)$$

To this end, we implemented two different fitness functions based on techniques given in [19]. The first is defined as a *switching fitness function* that will punish the GP depending on whether the GP has had a false positive or a false negative result. Two different costs will be associated with the switch depending on the makeup of the data set itself. If the individual has resulted in a false positive, the individual is awarded a cost equal to the error over the number of normal packets in the data set (Equation 3). Similarly, if the individual has resulted in a false negative, it is awarded the cost of the error over the number of attack packets in the data set (Equation 4). A higher cost is deemed a poorer performance then a lower cost.

$$Fitness1+ = \frac{1}{TotalNumberofNormalConnections} \quad (3)$$

OR:

$$Fitness1+ = \frac{1}{TotalNumberofAttackConnections} \quad (4)$$

Given the success in [19], we also chose to implement an equally weighted fitness function, as describe in Equation 5.

$$Fitness2+ = \frac{(1 - DR) + FPRate}{2} \quad (5)$$

5 Results

Using the data generated as described above, we conducted 40 runs of the IDS, each run differing only in the random seeds that are used for the initial population creation. Each run consisted of 1000 evolutionary cycles (max RSS iterations). The control parameters used during the running of the GP are shown in Table 1 and were chosen due to work in [19]. This process was performed both with the *switching* (fitness #1) and *equally weighted* (fitness #2) fitness functions. We present the results here.

5.1 Fitness #1

Using the switching fitness function (equations 3 and 4) resulted in 32 outlier solutions out of 40. We define outlier results as individual solutions that labeled all the exemplars as either attack or normal. That is to say the outlying solutions evolved to either labeling everything as attack or everything as normal, not the desired result. As these individuals represent degenerate solutions, they are removed before statistical analysis.

After removing the outliers, our IDS produced the results shown in Table 2 which lists the first, second (median) and third quartiles for the time taken to train the solution, detection rate (DR) and FP rate, time being in minutes, the remaining in percentages.

In Figure 2 we have plotted the solutions, the y axis being the detection rate, the x axis being the false positive rate. This allows us to see the low variance in solutions with respect to DR and FP rate, while also allowing us to identify the best performing solutions (highlighted in Table 3).

Table 1. Parameter Settings for Dynamic Page Based Linear GP

Parameter	Setting
Population size	125
Maximim 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	$\{0, ..., 255\} \bigcup \{r_0, ..., r_7\}$
RSS subset size	5000
DSS subset size	50
RSS iteration	1000
DSS iteration	100

Table 2. Fitness #1 Results

	Performance of De-Authentication Attacks		
	Time	Detection Rate	FP
1^{st} Quartile	40.003	87.500%	0.865%
Median	44.810	90.769%	1.250%
3^{rd} Quartile	50.152	96.154%	1.446%

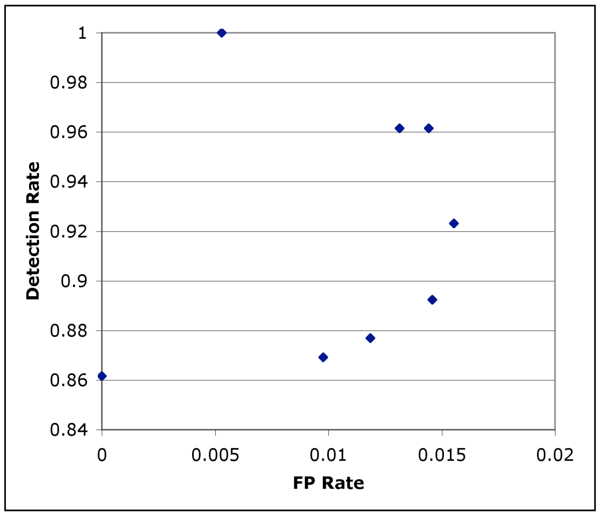

Figure 2. Detection Rate and FP for 40 runs using Fitness #1

Table 3. Fitness #1 Best Results

Best Performer with respect to DR		
Time	Detection Rate	FP
44.098	100%	0.529%
Best Performer with respect to FP Rate		
Time	Detection Rate	FP
40.838	86.154%	0.000%

5.2 Fitness #2

After implementing the equally weighted fitness function (equation 5), and removing the two outlier solutions (as defined above), the results are shown in Table 4 and Figure 3. As with Fitness #1, Table 4 describes the first, second and third quartiles for time of training (in minutes), DR (percentage) and FP rate (percentage). The increased times for training compared to Fitness #1 are not significant, due to variance in background system load during the two different experiments. The times are significant, however, when compared to other runs within this fitness function experiment. Figure 3, similarly to Figure 2, plots DR versus FP rate.

In Table 5 we have highlighted the two best individual solutions found using Fitness #2. These best solutions are given with respect to detection rate, then the false positive rate.

Table 4. Fitness #2 Results

	Performance of De-Authentication Attacks		
	Time	Detection Rate	FP
1st Quartile	56.854	80.769 %	0.465%
Median	68.475	82.308%	0.513%
3rd Quartile	73.130	82.308%	0.513%

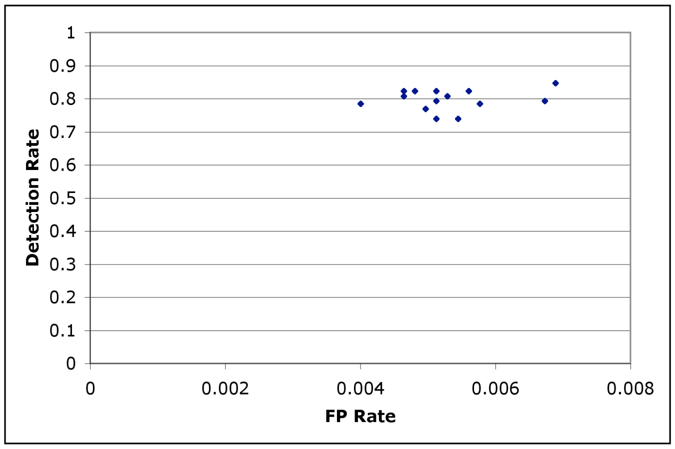

Figure 3. Detection Rate and FP for 40 runs using Fitness #2

6 Conclusion / Future Work

In this work we designed, developed and tested a Linear-GP based IDS on Data Link Layer attacks on Wifi networks. To this end, we developed a feature set and 2 fitness functions for our IDS as well as generated a data set to train and test it. To the best of our knowledge this is the first time such a L-GP based IDS system and training/testing data set has been developed. The first fitness function led to an IDS that achieved a 100% detection rate and a 0.529% false positive rate. The second fitness function resulted in a best solution with a 84% detection rate and a 0.689% false positive rate.

Our results show that the first fitness function results in a large number of outlier results, indicating that it may not be effective when applied to other unbalanced data sets. However, the small number of solutions that were capable of detecting both normal and attack exemplars had a high detection rate while still providing a low false positive rate. The second fitness function resulted in more consistent results, with very few outlier solutions. The solutions found with the second fitness function provided very low false positive rates, but at the cost of lower detection rates. The more consistent results of the second fitness function does indicate

Table 5. Fitness #2 Best Results

Best Performer with respect to DR		
Time	Detection Rate	FP
85.581	84.615%	0.689%
Best Performer with respect to FP Rate		
Time	Detection Rate	FP
74.150	78.461 %	0.401 %

that it encourages the evolving of solutions that can handle the unbalanced nature of our data set.

Compared to implementing Snort for detecting this DoS attack, the resulting IDS from our work does not require a user to set a threshold count of de-authentication frames nor a maximum time window size for this count to be met. The use of a threshold limit to trigger an alarm is also used in other work, such as [17]. Our system eliminates this requirement, providing a more generic tool for detecting the DoS attack.

Our future work will be to explore the use of larger data sets for training and testing our L-GP based IDS. This will allow us to verify the effectiveness of our work over larger networks as well as a varied number and length of DoS attacks. Also, we plan on applying the same approach described here on other WiFi attacks, with the goal of developing an IDS that can be used to detect a variety of attacks.

7 Acknowledgments

This research is supported by the NSERC Discovery and the CFI New Opportunities grants. This work is conducted as part of the Network Information Management and Security (NIMS) project at `http://www.cs.dal.ca/projectx/`.

References

[1] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In *USENIX Security Symposium*, pages 15–28, 2003.

[2] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11, 2001.

[3] M. Crosbie and E. H. Spafford. Applying genetic programming to intrusion detection. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8, MIT, Cambridge, MA, USA, 10–12 1995. AAAI.

[4] R. Exploit. Auditor Security Collection http://new.remote-exploit.org/, 2005.

[5] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. *Lecture Notes in Computer Science*, 2259:1–24, 2001.

[6] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem, 9-14 1994. Springer-Verlag.

[7] M. I. Heywood and A. N. Zincir-Heywood. Dynamic page based crossover in linear genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics: Part B - Cybernetics*, 32(3):380–388, 2002.

[8] IEEE-SA Standards Board. *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*. IEEE, New York, NY, USA, 1999.

[9] M. Kershaw. Kismet http://www.kismetwireless.net/, 2005.

[10] R. Lfoeter. Wireless Lan Security Framework htttp://www.wlsec.net/void11/, 2002.

[11] W. Li. Using genetic algorithm for network intrusion detection. Kansas City, Kansas, May 2004. United States Department of Energy Cyber Security Group 2004 Training Conference.

[12] W. Lu and I. Traore. Detecting new forms of network intrusion using genetic programming. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 2165–2172, Canberra, 8-12 2003. IEEE Press.

[13] E. Lundin and E. Jonsson. Survey of intrusion detection research. Technical Report 02-04, Department of Computer Engineering, Chalmers University of Technology, Goteborg, 2002.

[14] M. Lynn and R. Baird. Advanced 802.11 attack. *Black Hat Briefings*, July 2002.

[15] A. Mukkamala, S. Sung. A comparative study of techniques for intrusion detection, 15th ieee international conference on tools with artificial intelligence. *15th IEEE International Conference on Tools with Artificial Intelligence – ICTAI*, pages 570 – 577, Nov 2003.

[16] P. Ren Hui Gong; Zulkernine, M.; Abolmaesumi. A software implementation of a genetic algorithm based approach to network intrusion detection. *Sixth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing - SNPD/SAWN 2005*, pages 246 – 253, May 2005.

[17] T. Schmoyer, Y.-X. Lim, and H. Owen. Wireless Intrusion Detection and Response: A case study using the classic man-in-the-middle attack. In *IEEE Wireless Communications and Networking Conference*, Atlanta Ga., March 2004.

[18] C. Sinclair, L. Pierce, and S. Matzner. An application of machine learning to network intrusion detection. In *Computer Security Applications Conference*, pages 371–377. ACSAC '99, 1999.

[19] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, 2005.

[20] T. Xia, G. Qu, S. Hariri, and M. Yousif. An efficient network intrusion detection method based on information theory and genetic algorithm. *Performance, Computing, and Communications Conference, 2005. IPCCC 2005*, pages 11 – 17, April 2005.