802.11 De-authentication Attack Detection using Genetic Programming

Patrick LaRoche and 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 approach to detect deauthentication attacks on wireless networks based on the 802.11 protocol. To do so we focus on developing an appropriate fitness function and feature set. Results show that the intrusion system developed not only performs incredibly well - 100 percent detection rate and 0.5 percent false positive rate - but also developed a solution that is general enough to detect similar attacks, such as disassociation attacks, that were not present in the training data.

1 Introduction

As computer networks have become more prevalent, in varying forms, different protocols are being developed in order to support alternative networking under different environments. One such protocol is 802.11, where this describes a standard for communication over wireless connections. This protocol, also known as WiFi, has been deployed in a growing number of locations resulting in a diversity of purposes, from providing a household with a wireless network, to supporting an entire office building. Due to this growing popularity and exposure, more and more 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 arise from the fact that data is being transfered over the air, where anyone with the appropriate device can intercept. For this reason the 802.11 network provides many security features, such as encryption and client verification. These measures, as important as they are, have received the majority of emphasis. Unfortunately, they do not cover all the possible weaknesses in the 802.11 protocol. In our work we look at the issue of maintaining the availability of the network itself. That is to say there are well known exploits with the 802.11 protocol that allow users of malicious intent to disrupt the use of the network, either for a specific client or the entire network. This in return disrupts the availability of the network for its users; whereas the availability and ease of use represent one of the main reasons behind the widespread deployment of WiFi networks.

Traditionally, intrusion detection systems (IDSs) are used to detect attacks against the integrity, confidentiality and availability of computer networks [1]. In order to build effective IDSs and automate the detection process, various machine learning and artificial intelligence techniques have been proposed. These techniques include neural networks [2], data mining [3], decision trees [4], genetic algorithms (GA) [5, 6], and genetic programming (GP) [7–9]. 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 authors' 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, their exist attacks based on vulnerabilities in the physical and data link layers, i.e. the first and second layers respectively.

Thus, 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 genetic programming as the tool to create the rules in which to detect attacks [7]. 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 WiFi networks. 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 802.11 network. The remainder of this paper first details background information on 802.11 networks in Section 2, followed by a description of the GP that we have implemented in Section 3. In Section 4, we detail our approach for developing our GP based IDS, followed by Section 5, in which we list the performance of our GP based IDS, and concluding with Section 6 where we discuss our findings and how we will be proceeding with our work.

2 802.11 - WiFi Networks

The network protocol we focus on in our work is that of the 802.11b network (WiFi). WiFi networks have increased in popularity over the past few years, so much so that their use as a last mile solution for Internet connection has become common place, not only with homes but businesses alike. It has become so popular, that in the year 2001, the market for 802.11 based networks exceeded $1 Billion Dollars [10]. 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 Management Frames

WiFi networks have a series of MAC frame types, as described in the IEEE 802.11 Standard (see [11]). Of concern for our work are the management frames. 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 must 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 [11]) :

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 ([11]), 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. [11]

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.

2.2 Known Exploits

The 802.11 MAC layer includes functionality that addresses issues specific to a wireless network [10]. For example, the ability to search for networks, broadcast networks, join and leave networks are all taken care of by MAC layer frames, specifically management frames. As mentioned earlier (and in other works [10]),

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 802.11 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.

DoS attacks are not the only exploits or security weaknesses that exist on 802.11 networks. Much work has been done in addressing other security concerns. Papers such as [12] and [13] point out security weaknesses of the encryption algorithms implemented on 802.11 networks, and that is assuming the network administrators have even enabled encryption. This is not always a fair assumption given the wide spread adoption of 802.11 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, etc) 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.

2.3 De-Authentication 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 . An attacker chooses a target on the network (which has been gathered by actively monitoring the 802.11 network traffic using a tool such as kismet [14]) and then spoofs their ethernet address. At this point, the attacker then sends a de-authentication attack to the access point in which the target is associated. This causes the access point 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.

This attack, simple in its implementation, is quite powerful for several reasons. The first being 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 being 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 [15].

3 GP Based Intrusion Detection Systems

In order for us to be able to detect attacks that exploit the above mentioned vulnerabilities in the 802.11 standard, we have applied a genetic programming based intrusion detection system (IDS). Due to previous success in application of GP based IDSs to wired networks [7], we have chosen to take the same approach in creating an IDS for 802.11 networks. In previous work, a page based linearly structured GP was employed [16] as well as the utilization of a RSS-DSS algorithm which scales GP to data sets consisting of hundred of thousands of exemplars [7]. 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. For this work, our goal is to not only to develop a machine learning technique to detect network intrusions on 802.11 networks with a high accuracy and low false positive rate, but to also be able to develop these solutions in a quick, transparent matter.

3.1 Linear Page Based Genetic Programming

Linear Page Based GPs (L-GP) 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 *a priori* 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 [7]. 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) [16]. 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 [16].

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 [7]. 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 [17]. 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 stoping 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 only uses 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 [7].

4 Approach

Due to the above mentioned ability of L-GP when applied to Intrusion Detection [7], we felt that this previous work would make for a strong foundation for applying GP techniques to WiFi Intrusion Detection. To this end, we have developed an appropriate fitness function and feature set for the detection of the de-authentication attack on 802.11b networks. As mentioned, this attack is both real and easily performed, and has damaging effects on network usability.

4.1 Data Set Creation

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 creating an exemplar, with an output field that indicates that it is either an attack (binary 1) or normal (binary 0). For this realm, 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 of our L-GP based IDS. Our approach to this task

was both practical (to be feasible within our research facilities ability) as well as appropriate (large enough data set to be considered useful for training a learning algorithm).

A data set collection of 12 different network traffic dumps were collected on a real WiFi test network(performed using kismet [14] which dumps network traffic in a pcap format, a standard format for network traffic). These *dump files* were grouped into two different sets, the *large* and *small* sets, as six data sets were considerably larger then the other six. File statistics are shown in Table 1 (the original file size is equal to the normal packet count).

These data sets were assumed to have no attacks existing within them (a fair assumption given they were developed from load testing networks). From this, we developed a series of scripts that allowed us to inject simulated deauthentication attacks into the network stream. In order to create these attacks, we attacked our own small test network consisting of one client, one access point, a hacker laptop and a monitoring machine. The AP was the latest Apple Airport Base Station, the client a Mac Mini. The client connected to the AP via an 802.11b network on channel 6. The attack machine was an Intel based laptop running a Prism 2 based WiFi card running the Auditor Security Collection operating system. The monitoring machine was a 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 both the client and then 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). It is important to note that this attack can be performed with slight variations of this frame, but this format seemed the most effective. The frequency and duration of the transmission of the attack frames depended upon the desired persistence of the attacker, we found that continuous frame transmissions for a period of thirty frames (minimum) was required to have the desired DoS result on the target. Due to these findings, our injected de-authentication attacks vary in length between 30 and 100 attack frames at random locations within the provided network streams. Our test "injected" data sets then consisted of a varying number of actual attacks, with each attack varying in length. The statistics of these data sets are listed in Table 1.

A second pair of data sets were also created in order to validate the above approach. For these data sets, we implemented a small test network (the same as described as above, with 5 more clients added to the network, all generating web traffic) then attacked the network (using the techniques described above). Thus the result being two data sets of varying length, with two different attack placements and durations. These data sets are then used as a training testing pair, as they represent *live* network traffic with an inline attack (file statistics in Table 3).

Table 1. Injected Network Stream File Statistics

File	Normal Packets	Attack Packets	Total Packets	% Attack/Normal
1	155079	61959	217038	0.400
2	152759	126399	279158	0.827
3	156559	184159	340718	1.176
4	158399	241399	399798	1.524
5	153839	309079	462918	2.009
6	154679	364559	519238	2.357
7	44879	119	44998	0.003
8	44959	119	45078	0.003
9	44999	239	45238	0.005
10	45119	399	45518	0.009
11	45079	1759	46838	0.039
12	44999	3079	48078	0.068

4.2 Feature selection

The data files listed in Table 1 are made up of management frame packets. A management frame on a 802.11 network consists of several fields. For our L-GP based IDS, we chose to train and test on the following features of the frame:

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 selection of features to select from each packet was chosen based on a priori 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.

4.3 Fitness Function selection

The fitness function (or cost function) we chose for our GP was based on our goals of having a high detection rate (DR), as well as low false positive (FP) and false negative (FN) rates, defined as follows;

$$DetectionRate = 1 - (\frac{\#FalseNegativeClassifications}{TotalNumberofAttackConnections}) \quad (1)$$

$$FalsePositiveRate = (\frac{\#FalsePositiveClassifications}{TotalNumberofNormalConnections}) \quad (2)$$

To this end, we define 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.

$$Performance of individual+ = \frac{Error}{Total Number of Normal Connections} \quad (3)$$

$$Performance of individual+ = \frac{Error}{Total Number of Attack Connections} \quad (4)$$

The result of this switching cost function is that an individual will get awarded costs that are proportional to how *easy* it is to achieve either a false positive or a false negative. For example, if the individual results in a false negative, and there are less normal packets then attack packets in the data set (say a ratio of 3 to 1), then the cost associated with this will be larger then if it were a false positive.

Table 2. Disassociation Injected Network Stream File Statistics

File	Normal Packets	Attack Packets	Total Packets	% Attack/Normal
1	45040	0	45040	0
2	44960	120	45080	0.003
3	45120	160	45280	0.004
4	45120	440	45560	0.010
5	45200	880	46080	0.019
6	45120	1800	46920	0.038
7	45320	2800	48120	0.058
8	45400	5840	51240	0.114

5 Results

We ran the GP over all 12 data sets (Table 1), using each data set as a training data set, as well as testing data sets. That is to say, we would run the GP on data set 1, then test on data sets 2-12, etc. This insured that the GP was being evaluated on an array of different data sets, with different ratios of attack to normal exemplars, different placements and durations of attacks, as well as the

size of the data sets themselves. We then ran each one of the above experiments 10 times, with different random seeds used for the initial population creation. The parameters used during the running of the GP are shown in Table 6, the results are shown in Table 4.

Table 3. Live Network Data Set File Statistics

File	Normal Packets	Attack Packets	Total Packets	% Attack/Normal
Training	5890	450	5440	0.083
Testing	6370	130	6240	0.021

Table 4. Results

Performance:	De-Authentication Attacks			Disassociation Attacks		
	Run Time	DR	FP Rate	Run Time	DR	FP Rate
1^{st} Quartile	19.9983	100.00%	0.00	18.5455	100.00%	0.00
Median	23.5468	100.00%	0.00	26.145	100.00%	0.00
3^{rd} Quartile	27.712	100.00%	0.00	33.9506	100.00%	0.00

As seen from the results, our L-GP based IDS performed incredibly well. Thus for verification purposes, two more experiments were conducted. The first being to create another 8 data sets (file statistics show in Table 2). This collection of data sets were injected with a similar attack to the de-authentication attack: the disassociation attack [10], which is different only in the type of frame that is sent by the attacker. We tested the previously trained GPs (trained with the de-authentication attack) on these newer data sets. The intent was to determine if the GP had over specified its training; if so, it would miss the very similar attack. The results, shown in Table 4, indicate that the GP had developed a solution that was general enough to detect this new (admittedly similar) attack.

The second additional experiment used the data set pair created with live network traffic and inline attacks (described in 4.1). These data sets represent a real WiFi network that had been attacked using the DoS attack, with the network traffic before, after and during the attack recorded and labeled for use in training and testing the L-GP based IDS. The goal of this experiment is to show that the response of the network to the attack (such as the rejoining of the clients to the network after being de-authenticated) would not cause difficulty to the IDS. The GP was run 40 times, using different initial seeds for the initial population generation, which resulted in two best performing solutions shown in Table 5. As shown, even on live network traffic, the L-GP based IDS performs incredibly well. The best solution with respect to the detection rate, maintains

a 100% DR, while still providing a very low FP rate, as well the best solution with respect to the false positive rate can achieve a 0% FP rate, with a high DR.

Table 5. *Live* Network Resulting Solutions

Best Performer	with respect to DR	with respect to FP Rate
Time	44.098	40.838
Detection Rate	100.000%	86.154%
FP Rate	0.529%	0.000%

Table 6. Parameter Settings for Dynamic Page Based Linear GP

Parameter	Setting	Parameter	Setting
Population size	125	Number of registers	8
Maximim number of pages	32	Function set	{+,-,*,/}
Page size	8 instructions	Terminal set	{0, ..., 255} $\bigcup$ {i_0, ...,i_{63}}
Maximum working page size	8 instructions	RSS subset size	5000
Crossover probability	0.9	DSS subset size	50
Mutation probability	0.5	RSS iteration	1000
Swap probability	0.9	DSS iteration	100
Tournament size	4		

6 Discussion / Future Work

We have successfully shown that a L-GP based IDS can be applied to attacks that are unique to the realm of WiFi networks. Our results show that the GP can be trained on one of the most common attacks to 802.11 networks, with an outcome of 100% detection rate with a 0.529% false positive rate. We have also shown that the GP, even when trained on such a specific attack type, can provide a solution that is generalized enough to detect similar attacks.

We hypothesize that the original collection data sets did not simulate the real reaction of the network from the attack, thus providing the GP with a clean separation between the attack packets and the network background traffic. By then running our IDS on live network traffic during a DoS attack, and using this as our training and testing data sets, we have shown that we can still achieve such high DR and low FP rates. Our future work will explore larger live network data sets including both DoS attacks and more complicated attacks such as Man in the Middle 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 NIMS project at `http://www.cs.dal.ca/projectx/`.

References

1. Lundin, E., Jonsson, E.: Survey of intrusion detection research (2002)
2. Mukkamala, S. Sung, A.: 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 (2003) 570 – 577
3. Xia, T., Qu, G., Hariri, S., Yousif, M.: An efficient network intrusion detection method based on information theory and genetic algorithm. Performance, Computing, and Communications Conference, 2005. IPCCC 2005 (2005) 11 – 17
4. Sinclair, C., Pierce, L., Matzner, S.: An application of machine learning to network intrusion detection. In: Computer Security Applications Conference, ACSAC '99 (1999) 371–377
5. Ren Hui Gong; Zulkernine, M.; Abolmaesumi, P.: 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 (2005) 246 – 253
6. Li, W.: Using genetic algorithm for network intrusion detection, Kansas City, Kansas, United States Department of Energy Cyber Security Group 2004 Training Conference (2004)
7. Song, D., Heywood, M.I., Zincir-Heywood, A.N.: Training genetic programming on half a million patterns: an example from anomaly detection. IEEE Transactions on Evolutionary Computation **9**(3) (2005) 225–239
8. Lu, W., Traore, I.: Detecting new forms of network intrusion using genetic programming. In Sarker, R., Reynolds, R., Abbass, H., Tan, K.C., McKay, B., Essam, D., Gedeon, T., eds.: Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, IEEE Press (2003) 2165–2172
9. Crosbie, M., Spafford, E.H.: Applying genetic programming to intrusion detection. In Siegel, E.V., Koza, J.R., eds.: Working Notes for the AAAI Symposium on Genetic Programming, MIT, Cambridge, MA, USA, AAAI (1995) 1–8
10. Bellardo, J., Savage, S.: 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In: USENIX Security Symposium. (2003) 15–28
11. IEEE-SA Standards Board: ANSI/IEEE Std 802.11, 1999 Edition (R2003). IEEE, New York, NY, USA (1999)
12. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the key scheduling algorithm of RC4. Lecture Notes in Computer Science **2259** (2001) 1–24
13. Borisov, N., Goldberg, I., Wagner, D.: Intercepting mobile communications: The insecurity of 802.11. http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html (2001)
14. Kershaw, M.: Kismet http://www.kismetwireless.net/ (2005)
15. Schmoyer, T., Lim, Y.X., Owen, H.: 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. (2004)
16. Heywood, M.I., Zincir-Heywood, A.N.: Dynamic page based crossover in linear genetic programming. IEEE Transactions on Systems, Man, and Cybernetics: Part B - Cybernetics **32**(3) (2002) 380–388
17. Gathercole, C., Ross, P.: Dynamic training subset selection for supervised learning in genetic programming. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature III. Volume 866 of LNCS., Jerusalem, Springer-Verlag (1994) 312–321