Intrusion Detection with Genetic Algorithms and Fuzzy Logic

Emma Ireland
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
irela065@morris.umn.edu

ABSTRACT

Intrusion detection systems provide one way of detecting attacks on systems by monitoring network activities for malicious or abnormal behaviors. This paper describes two ways of training an intrusion detection system to recognize possible attacks on a system: genetic algorithms and fuzzy logic. I will describe an approach to using fuzzy genetic algorithms and compare those results with results obtained using a decision tree. I will then describe the results from using a traditional genetic algorithm and compare those with the winning entry of the KDD99 Classifier Learning Contest, as well as with the fuzzy genetic algorithm. These results show that the use of genetic algorithms and fuzzy logic in intrusion detection are effective ways of detecting attacks.

Keywords

Intrusion detection, genetic algorithms, fuzzy logic, KDD99, RLD09, computer security

1. INTRODUCTION

The CSci computer lab at the University of Minnesota, Morris gets large numbers of login attempts that are attempts at intrusion. An attacker could be trying to gain root access to the system, which would then give them the ability to do things like delete files of other users and change user passwords. If the university had an intrusion detection system (IDS), it would be possible to classify those attempts into legitimate and illegitimate attempts to login. Then it would be possible to block IP addresses that are generating large numbers of attacks.

An attack on systems is a concern for many people, so it is important to have a way to detect and analyze these attacks. Intrusion detection systems provide one way of detecting attacks by monitoring network activities for malicious or abnormal behaviors and then producing reports, alerts, and actions [6]. There are various ways of training an IDS about possible threats. Two approaches that I will talk about in this paper are genetic algorithms and fuzzy logic.

UMM CSci Senior Seminar Conference, December 2013 Morris, MN.

In Section 2 I will give some background on intrusion detection. I will describe the main types of networking attacks in Section 2.1, and ways of detecting attacks in Section 2.2. In Section 2.3 I will describe two data sets that are commonly used in intrusion detection. After that, in Section 2.4 I will describe the measures that are used in determining the accuracy of an IDS. I will then provide an overview on genetic algorithms in Section 3. In Section 4 I will describe the use of a fuzzy genetic algorithm in an IDS, and in Section 5 I will describe the results of using a traditional genetic algorithm. Section 6 provides some conclusions about the use of genetic algorithms and fuzzy genetic algorithms in intrusion detection systems.

2. BACKGROUND

2.1 Types of Networking Attacks

There are four main types of networking attacks that this paper will address: *denial of service attacks*, *remote to user attacks*, *user to root attacks*, and *probe attacks*. Each attack that happens on a network can be placed into one of these categories. [6]

Denial of service (DoS) attacks happen when an attacker makes a machine inaccessible to a user by making it too busy to serve legitimate requests. For example, many systems lock out a user from an account after a certain number of failed login attempts. An attacker would be able to use this to prevent legitimate users from logging in, by intentionally failing to log in enough times to lock the account [2]. Remote to user, also known as remote to local (R2L) attacks happen when an attacker sends packets to a machine over the network in order to gain access to things a local user would have on the machine. An example is when an attacker tries to gain access to a machine by guessing possible usernames and passwords. User to root (U2R) attacks happen when an attacker starts out with access on the machine and then tries to gain root access to the system. If an attacker has access to an account on the machine, they could potentially run programs that take advantage of operating system weaknesses to gain root access. Probe attacks happen when an attacker examines a machine in order to collect information about weaknesses or vulnerabilities that in the future could be used to compromise the system. For example, the attacker could be trying to determine what version of a software is being run on that machine, and if that version has a known issue then that allows them to attempt to attack that. [6, 9]

2.2 Detection Methodologies and Rules

There are two different ways of detecting attacks: *signature-based detection* and *anomaly-based detection.* A signature is a pattern that corresponds to a known attack. Signature-based detection compares well-known patterns of attacks that are already known to the IDS against captured events in order to identify a possible attack. It is a simple and effective way to detect known attacks, but is ineffective against new kinds of unknown attacks. Signature-based detection is also called *knowledge-based detection* or *misuse detection.* [10]

Anomaly-based detection looks for patterns of activity that are rare and uncommon. It is harder to do than signature-based detection, but it can be an effective way to detect new, unknown attacks. Anomaly-based detection is also called *behavior-based detection.* [6]

A commonly used approach for detecting intrusions is to use rules. Rules are represented by if-then statements in the following format: If (*condition*) then (*consequence*). The condition part of the rule is composed of one or more features, and the consequence of the rule says if it is an intrusion or not. For example: if (*duration* = 1, and *src_bytes* = 0, and *serror_rate* = 50) then *intrusion.* [4]

2.3 Data Sets

Two different data sets are used in this paper to evaluate the performance of intrusion detection systems: KDD99 and RLD09. KDD99 is a benchmark data set that was generated by simulating a military network environment in 1999, and it has long been a standard data set for intrusion detection. The data was processed into five million records, where a record is a sequence of TCP packets, between which data flows to and from a source IP address to a target IP address. Each record in the data set is classified as either normal or attack activity. KDD99 uses 41 *features*, which are properties of a record that are used to describe the activity and help to distinguish normal connections from attacks. [5]

In the research discussed in Section 4 [7, 8], eight of the 41 KDD99 features were used based on research done in [4]. These eight features are:

1. duration: the length of the record in seconds.
2. src_bytes: the number of bytes sent from source to destination. Source is the user who may or may not be an attacker, and destination is the server being potentially attacked.
3. num_failed_logins: the number of failed login attempts in this record.
4. root_shell: returns 1 if root shell is obtained, else it returns 0.
5. num_access_files: the number of operations on access control files. Access control files specify which users are granted access to objects and what operations are allowed on the objects. An example would be (Alice, delete), which would give Alice permission to delete the file. [11]
6. srv_count: the number of connections in other records.
7. serror_rate: the percentage of connections that have "SYN" errors. When a client attempts to connect to a server, it first sends a SYN (synchronize) message to the server. The server then acknowledges the request by sending a SYN-ACK to the client. The connection is established when the client sends an ACK back to the server. A SYN error is a failure to get an ACK back. [13]
8. same_srv_rate: the percentage of connections to the same service.

The KDD99 data set is 14 years old, and newer attack types are not included in it because of its age. Because of this, the authors of [7, 8] created their own data set, RLD09, to use in their experiments. To create the data set, the authors captured network data from a university in Bangkok, Thailand. As well as normal network activity, RLD09 has 17 different types of attacks which can be classified as either denial of service attacks or probe attacks. RLD09 uses 12 features. For further information about the features, see [7]. In the research discussed in Section 5, the authors of [6] used only the numerical features of KDD99 (34 out of 41).

In a machine learning experiment, a common technique is to divide the data set into two subsets, a *training set* and a *testing set.* The given algorithm is then trained on the training set to look for patterns. These patterns are then verified using the test set. [14] This approach is used in [7, 8] and [6].

2.4 Determining the Accuracy of an Algorithm

Four measures are used to determine the accuracy of an algorithm. The *false negative* (FN) rate is the percentage that attacks are misclassified from the total number of attack records. The *false positive* (FP) rate is the percentage that normal records are classified as attacks from the total number of normal records. The *true negative* (TN) rate is the percentage that normal records are classified correctly from the total number of normal records. The *true positive* (TP) rate is the percentage that attacks are classified correctly from the total number of attack records. [7, 14]

3. GENETIC ALGORITHMS

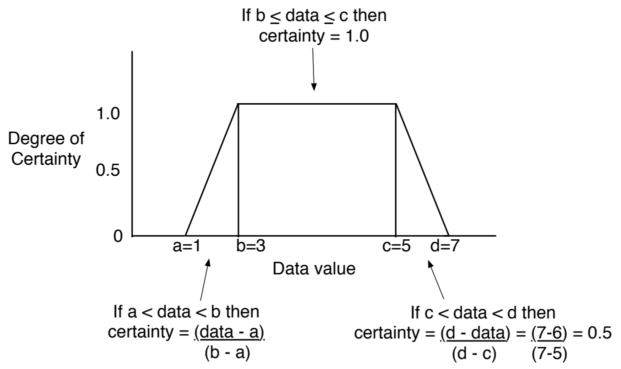
Genetic algorithms (GAs) are a search technique used to find solutions to problems. Operations analogous to biological *mutation*, *selection*, and *crossover* are used to evolve and improve solutions.

Possible solutions to problems can be represented in a variety of problem dependent ways, such as bit strings. For example, IDS rules can be represented as bit strings; see Section 4.1 for more information. First, a randomly generated population of potential solutions is created. Then mutation, crossover, and selection are applied to each generation until an acceptable solution is found or some time limit is exceeded.

Mutation is where random bits in an *individual*, or possible solution, are randomly changed. Crossover is where two individuals swap sequences of bits to form two new individuals. For example, in an IDS mutation takes the bits of a rule and changes them to form a slightly different rule. Crossover takes two rules and creates new rules by swapping the bits of the old rules.

Selection is where individuals that have better *fitness* are chosen to be parents. The fitness of an individual is specified by the *fitness function*, which determines the quality of a particular individual. For example, in an IDS the fitness measures how well a rule classifies records as either attacks or normal activity. Selection combined with a fitness function directs the search towards an effective solution. [1]

Figure 1: Illustration of fuzzy rule used to find the degree of certainty of a record being an attack. Also see Algorithm 1. This is used in [7, 8].



If b ≤ data ≤ c then certainty = 1.0
Degree of Certainty
1.0
0.5
0
a=1
b=3
c=5
d=7
Data value
If a < data < b then certainty = (data - a)/(b - a)
If c < data < d then certainty = (d - data)/(d - c) = (7-6)/(7-5) = 0.5

Algorithm 1 Fuzzy Algorithm that is based on an algorithm used in [7, 8], but a few corrections have been made. Also see Figure 1.

```
if $b \leq$ data $\leq c$ then
    certainty = 1.0
else if $a <$ data $< b$ then
    certainty = (data $- a)/(b - a)$
else if $c <$ data $< d$ then
    certainty = $(d -$ data$)/(d - c)$
else
    certainty = 0.0
end if
```

4. USING FUZZY GENETIC ALGORITHMS

The IDS used in [7, 8] is able to identify normal network activity as well as attacks using a fuzzy genetic algorithm. This kind of algorithm is able to learn to recognize new attacks.

4.1 Fuzzy Logic Rules and Algorithm

Attacks on systems do not always have a fixed pattern, so fuzzy logic is used to detect patterns that have a behavior that is between normal and unusual. Fuzzy logic rules are similar to the rules described in Section 2.2, except that *consequence* is a certainty factor. For example, if (*duration* = 6) then (*the degree of certainty of the record being an attack is 0.5*).

To find the degree of certainty of a record being an attack, a trapezoidal shape was used in [7, 8]; this is shown in Figure 1. The trapezoidal shape has four parameters: a, b, c, and d. Algorithm 1 calculates the certainty of a record being an attack.

An example of using the fuzzy logic approach is: suppose that the feature is duration, and suppose it is 6 seconds, so then data = 6. Suppose that $a = 1$, $b = 3$, $c = 5$, $d = 7$. Because data (6) is between c and d, then the degree of certainty of it being an attack is equal to

$$\frac{d - \text{data}}{d - c} = \frac{7-6}{7-5} = 0.5.$$

The four parameters a, b, c, and d are encoded into blocks of binary strings, where each block is a feature with values between 0 and 7. A rule has one block for each of 12 features followed at the end by a marker indicating the type of attack. An example of this is shown in Figure 2. The authors of [7, 8] compute the degree of certainty for each of the 12 blocks, and if the sum of those is greater than a threshold, then it will be declared as an attack.

Algorithm 2 Fuzzy GA that is based on an algorithm used in [7, 8], but a few corrections have been made.

```
for each rule do
    for each record do
        for each feature do
            certainty = fuzzy(); // Algorithm 1
            total = total + certainty;
        end for
        if total > threshold then
            class is attack;
        else
            class is normal;
        end if
    end for
    compare the predicted result with actual result
    find $A$, $B$, $\alpha$, and $\beta$
    // $A$ is the total number of attack records. $B$ is the
    total number of normal records. $\alpha$ is the total number
    of attack records correctly identified as attack. $\beta$ is the
    total number of normal records incorrectly classified as
    attack.
end for
calculate fitness
//create next generation
preserve_best()
crossover()
mutation()
```

4.2 Algorithm Overview

The algorithm in [7, 8] first randomly generates rules. Then the rules are improved in the training phase, which can be seen in Algorithm 2, which describes the fuzzy genetic algorithm that is used. One record (either an attack or normal activity) is passed into a rule. Each feature in a record is matched to one block of the rule. The parameters of each block measure the degree of certainty of an attack using the trapezoidal fuzzy rule shape. The sum of the degrees of certainty from each block are then compared with a threshold to determine if the record represents an attack or normal behavior.

The fitness function to be maximized in Algorithm 2 is:

$$\text{fitness function} = \frac{\alpha}{A} - \frac{\beta}{B}$$

where A is the total number of attack records, B is the total number of normal records, α is the total number of attack records correctly identified as attack, and β is the total number of normal records incorrectly classified as attack.

A population size of 10 was used for each generation. The two best individuals from the present generation are preserved for the next generation. The other individuals in the new generation come from mutation and crossover.

Figure 2: A rule with 12 blocks of features used in [7, 8].

010	011	100	101	......	010	011	101	111	DoS
a=2	b=3	c=4	d=5	......	a=2	b=3	c=5	d=7	
		Block 1				Block 12			Type

Algorithm 3 This algorithm was used to identify attacks and normal activity in [7].

```
if dos_rule = yes or probe_rule = yes then
    This record is an attack;
else
    This record is normal;
end if
```

4.3 Experimental Design and Results

A variety of experiments were run in [7, 8]. Two experiments used just RLD09, and three experiments used KDD99 and RLD09 together.

4.3.1 Experiments using Only RLD09

The experiments using only RLD09 that [7] performed used a total of 16,000 records of normal activity and 10,500 records of attack activity. Of the attack records, 4,000 were denial of service attacks and 6,500 were probe attacks.

In the first experiment, the fuzzy genetic algorithm was used to create separate denial of service and probe detection rules. Both of the denial of service and probe rules were then used together in the testing process to identify attacks from the testing data set; this is shown in Algorithm 3. 10,000 records were used for the training set and all 26,500 records were used for the testing set. (The authors of [7] really should not have reused the training data in the testing set because it inflates their results.)

To evaluate the accuracy of the fuzzy genetic algorithm, detection rate (DR) was used. Detection rate is defined in [7, 8] as the percentage of normal and attack activity correctly classified from the total number of data records. The detection rate of DoS attacks in training was 91.64% and the detection rate of probe attacks in training was 94.79%. The detection rate of the testing data set increased to 97.92%. Results from this experiment are in Table 1.

In the second experiment, [7] pulled some types of attacks out of the training set and kept them for unknown data testing. This was to test that the fuzzy genetic algorithm could detect unknown attacks. In this experiment, seven tests were run. For each test case there were 13 attack types plus normal activity that were in the training data set. Three attack types were used for the unknown testing data set. For example, test case 1 used the training data set that does not have Advance Port Scan, Ack Scan, and Xmas Tree, which are all probe attacks. These three attacks were then used for the testing data set. Table 2 shows the results from the fuzzy genetic algorithm and a decision tree algorithm, which is another common algorithm for classification problems. For further information on decision trees, see [12], and for further information on the types of denial of service and probe attacks, see [9]. When compared with the decision tree algorithm, the fuzzy genetic algorithm has a higher detection rate in all cases except 5 and 7, however, no information on statistical significance was given.

Table 2: Unknown attack experiment, using only the RLD09 data set [7].

Test Case	Unknown Attacks	Decision Tree DR (%)	Fuzzy GA DR (%)
1	Adv Port Scan (Probe) Ack Scan (Probe) Xmas Tree (Probe)	Avg = 98.33	Avg = 100
2	HTTP Flood (DoS) IP Scan (Probe) Null Scan (Probe)	Avg = 88.4	Avg = 95.30
3	Smurf (DoS) Port Scan (Probe) Connect Scan (Probe)	Avg = 97.65	Avg = 99.15
4	UDP Flood (DoS) Host Scan (Probe) UDP Scan (Probe)	Avg = 46.65	Avg = 99.80
5	Jping (DoS) Syn Scan (Probe) Fin Scan (Probe)	Avg = 99.70	Avg = 98.75
6	UDP Flood (DoS) RCP Scan (Probe) Fin Scan (Probe)	Avg = 70.35	Avg = 98.15
7	HTTP Flood (DoS) RCP Scan (Probe) Fin Scan (Probe)	Avg = 99.94	Avg = 97.50

Table 3: KDD99 and RLD09 results from the first experiment in [8].

Data set	Attack	Normal	FP(%)	FN(%)	DR(%)
KDD99	160,117	39,337	0.13	1.55	98.72
RLD09	10,500	16,000	1.14	3.39	97.97

4.3.2 Experiments using Both RLD09 and KDD99

The authors of [8] also ran experiments that used both the RLD09 data set and the KDD99 data set in order to compare how the fuzzy genetic algorithm would perform on both. They used a subset of the KDD99 data set for both the training data set and testing data set.

The first experiment used the fuzzy genetic algorithm to classify normal activity and attacks from both data sets. The authors of [8] first trained and tested the fuzzy genetic algorithm (Algorithm 2) with the KDD99 data set. There were 6 different types of denial of service attacks and 4 different types of probe attacks. The detection rate of the KDD99 data set was 98.72%. Then 26,500 records of the RLD09 data set were used as the training set. The detection rate was 97.97%. The results of this experiment are shown in Table 3.

The next experiment used the fuzzy genetic algorithm to classify types of attacks in the KDD99 data set. They used the KDD99 training set, with 158,597 records of denial of

Table 1: **Results from Experiment 1, using only RLD09 [7].**

	Attack	Normal	Total Records	FP (%)	FN (%)	DR (%)
DoS Training	1499	8501	10000	1.46	47.50	91.64
Probe Training	2496	7504	10000	1.83	15.38	94.79
Testing	10500	16000	26500	1.13	4.10	97.92

Table 4: Results for KDD99 with Certain Attacks. 10 tests were run in total, 5 are shown here. [8]

Test	Attack	Type	FP(%)	FN(%)	DR(%)
1	Back	DoS	85.33	0.00	16.56
2	Smurf	DoS	0.76	0.10	99.73
3	Neptune	DoS	0.15	0.34	99.75
4	Portsweep	Probe	6.40	0.00	93.66
5	Satan	Probe	0.74	3.75	99.22

Table 5: Results for RLD09 with Certain Attacks. 17 tests were run in total, 6 are shown here. [8]

Test	Attack	Type	FP(%)	FN(%)	DR(%)
1	HTTP Flood	DoS	0.36	3.5	99.46
2	Smurf	DoS	0.02	0	99.98
3	UDP Flood	DoS	11.06	0	89.59
4	Fin Scan	Probe	2.58	0	97.50
5	IP Scan	Probe	13.01	16.4	86.89
6	Syn Scan	Probe	0.65	4.2	99.24

service attacks and 1,500 records of probe attacks. Ten tests were run, and Table 4 shows the accuracy of detecting some of the cases. The results showed that the detection rate of eight out of ten cases were greater than 93%. There were only two cases that had low detection rates, one of which is case 1 in Table 4. The two low detection rates were 16.56% and 15.58%, both of which were denial of service attacks.

The final experiment that was run used only the RLD09 data set with the fuzzy genetic algorithm to classify types of attacks. 17 tests were run, and Table 5 shows the accuracy of detecting some of the cases. The results showed that the detection rate of 15 out of 17 cases were greater than 97%. Again, there were only two test cases that had low detection rates, cases 3 and 5 in Table 5. The two low detection rates were 89.59% (denial of service), and 86.89% (probe).

5. USING GENETIC ALGORITHMS

The authors of [6] used a genetic algorithm to develop an IDS. They didn't explain their research that well and left a lot of things out, so I will just be describing their results.

The KDD99 data set was used in the experiments. Standard subsets of set were used for training and testing. The training set had a total of 494,021 records, 396,741 of which were attacks. The test set had a total of 311,029 records, 250,436 of which were attacks. Table 6 shows the distribution of each type of attack, as well as normal activity, that were in the training and test sets.

The results from running the genetic algorithm are shown in Table 7. Detection rate is defined in [6] as the ratio between the number of correctly detected intrusions and the total number of intrusions. Denial of service attacks had the

Table 6: Training and testing records used in [6].

	Training	Testing
Normal	97,280	60,593
DoS	391,458	229,853
R2L	1,124	16,189
U2R	52	228
Probe	4,107	4,166
Total Attacks	396,741	250,436
Total Records	494,021	311,029

Table 8: Accuracy of the GA in [6].

	Predicted	
Actual	Normal	Attack
Normal	TN: 42,138. 69.5%	FP: 18,455. 30.5%
Attack	FN: 12,528. 5%	TP: 237,908. 94.9%

highest detection rate at 99.4%. A high detection rate on denial of service attacks isn't surprising because there were a lot of denial of service records used compared with the other types of attacks. The accuracy of the genetic algorithm is shown in Table 8. The true negative rate was 69.5%. The false positive rate was 30.5%. The false negative rate was 5%. The true positive rate was 94.9%.

The authors of [6] compared their results (Table 7) with the winning entry of the KDD99 Classifier Learning Contest [3], which can be seen in Table 9. The winning entry used a decision tree algorithm. The winning entry had a higher detection rate for normal activity, probe attacks, and remote to user attacks. The authors of [6] found that they had a better detection rate for denial of service and user to root attacks than the winning entry. For the winning entry, the detection rate of denial of service was 97.1%, and for user to root it was 13.2%. In [6] the detection rate of denial of service was 99.4% and for user to root it was 18.9%. The authors of [6] provide no information about the statistical significance of these improvements.

The detection rate (as defined in Section 4.3 and [8]) of the traditional genetic algorithm in [6] was 90% (see Tables 8 and 6). This means that it correctly classified 90% of the test records. The detection rate of the fuzzy genetic algorithm in [8] was 99% (see Table 3). The fuzzy genetic algorithm in [8] is not the same as the traditional genetic algorithm in [6] with fuzzy logic added to it, but it can be said that the system in [8] is better than [6] however it was built.

6. CONCLUSIONS

The fuzzy genetic algorithm that was used in [7, 8] had a higher detection rate than a decision tree algorithm in most cases, and it was good at detecting unknown attacks.

Table 7: Results for GA Experiment in [6].

	Predicted					
Actual	Normal	Probe	DoS	U2R	R2L	% Correct
Normal	42138	1421	15835	486	713	69.5
Probe	398	2963	654	2	149	71.1
Dos	921	432	228489	1	10	99.4
U2R	146	21	8	43	10	18.9
R2L	11191	578	3398	141	881	5.4
% Correct	76.9	54.7	92.0	6.4	50.0	

Table 9: Results for the Winning Entry of the KDD99 Classifier Learning Contest.

	Predicted					
Actual	Normal	Probe	DoS	U2R	R2L	% Correct
Normal	60262	243	78	4	6	99.5
Probe	511	3471	184	0	0	83.3
Dos	5299	1328	223226	0	0	97.1
U2R	168	20	0	30	10	13.2
R2L	14527	294	0	8	1360	8.4
% Correct	74.6	64.8	99.9	71.4	98.8	

It had a higher detection rate than the traditional genetic algorithm that was used in [6]. The genetic algorithm in [6] had a high detection rate for denial of service attacks. When compared with the winning entry of the KDD99 Classifier Learning Contest, it was shown to have a better detection rate for both denial of service and user to root attacks. This paper showed that the use of genetic algorithms and fuzzy genetic algorithms in intrusion detection are effective ways of detecting attacks.

References

[1] R. Borgohain. FuGeIDS: Fuzzy Genetic paradigms in Intrusion Detection Systems. *CoRR*, abs/1204.6416, 2012.

[2] CERT. Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html, 2001. [Online; accessed 26-October-2013].

[3] C. Elkan. Results of the KDD99 Classifier Learning Contest. http://cseweb.ucsd.edu/ elkan/clresults.html, 1999. [Online; accessed 27-October-2013].

[4] T. P. Fries. A fuzzy-genetic approach to network intrusion detection. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 2141–2146, New York, NY, USA, 2008. ACM.

[5] S. Hettich and S. D. Bay. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/ kddcup99.html, 1999. [Online; accessed 12-October-2013].

[6] M. S. Hoque, M. A. Mukit, and M. A. N. Bikas. An Implementation of Intrusion Detection System Using Genetic Algorithm. *CoRR*, abs/1204.1336, 2012.

[7] P. Jongsuebsuk, N. Wattanapongsakorn, and C. Charnsripinyo. Network intrusion detection with Fuzzy Genetic Algorithm for unknown attacks. In *Information Networking (ICOIN), 2013 International Conference on*, pages 1–5, 2013.

[8] P. Jongsuebsuk, N. Wattanapongsakorn, and C. Charnsripinyo. Real-time intrusion detection with fuzzy genetic algorithm. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*, pages 1–6, 2013.

[9] K. Kendall. Intrusion Detection Attacks Database. http://www.ll.mit.edu/mission/communications/ cyber/CSTcorpora/ideval/docs/attackDB.html, 1999. [Online; accessed 26-October-2013].

[10] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013.

[11] Wikipedia. Access control list — Wikipedia, The Free Encyclopedia, 2013. [Online; accessed 26-October-2013].

[12] Wikipedia. Decision tree — Wikipedia, The Free Encyclopedia, 2013. [Online; accessed 26-October-2013].

[13] Wikipedia. Transmission control protocol — Wikipedia, The Free Encyclopedia, 2013. [Online; accessed 26-October-2013].

[14] S. X. Wu and W. Banzhaf. The Use of Evolutionary Computation in Knowledge Discovery: The Example of Intrusion Detection Systems. In S. Dehuri and S.-B. Cho, editors, *Knowledge Mining using Intelligent Agents*, volume 6 of *Advances in Computer Science and Engineering*, chapter 2, pages 27–59. WorldSciBook, Dec. 2010.