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

Categories and Subject Descriptors

[**Security and Privacy**]: Intrusion Detection Systems

General Terms

Keywords

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

1. INTRODUCTION

2. BACKGROUND

2.1 Types of Intrusion Detection Systems

There are two types of intrusion detection systems: *host based* and *network based*. Host based intrusion detection systems look at and analyze information on a single host to determine if there is a possible threat to the system. The information that is analyzed include file systems and system calls. Network based intrusion detection systems look at information from multiple systems all at once. The information that is analyzed is the packets that pass through the network. [7]

2.2 Types of Networking Attacks

There are four main types of networking attacks that this paper will address: *denial of service*, *remote to user attacks*, *user to root attacks*, and *probing*. Each attack that happens on a network can be placed into one of these categories. [10]

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. [1] Remote to user (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. For example, if a program expects a user to input their name, the programmer has to decide how many characters that name buffer will require. Assume the program allocates 20 characters for the name buffer. Suppose the user's name has 35 characters. The last 15 characters will overflow the buffer, which will then overwrite the instructions that are to be executed next. An attacker can cause commands to be executed by manipulating the data that overflows. Probing happens 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. [10, 13]

2.3 Detection Methodologies

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 in the intrusion detection system against captured events in order to identify a possible attack. It is a simple and effective way to detect known attacks. Signature-based detection is also called *knowledge-based detection* or *misuse detection*. [14]

An anomaly is something that deviates from what is normal. Anomaly-based detection looks for patterns of activity that are rare and uncommon. It is an effective way to detect new attacks. Anomaly-based detection is also called *behavior-based detection*. [10]

2.4 Rules

Rules are a way in which elements of one set are separated into different sets, or classes, in order to differentiate between normal connections and attacks. Rules are represented by if-then statements in the following format: If <*condition*> then <*action*>. [6] Rules can specify the details of a packet such as the IP address, port number and protocal. If a packet matches any of the rules in the intrusion detection system then the system will take appropriate action, which may include stopping the connection or logging off the system. [7]

2.5 Fuzzy Logic

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

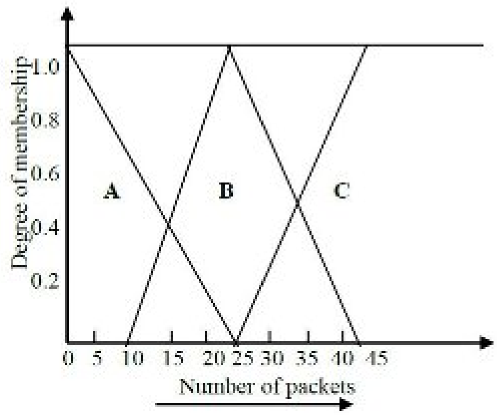

Figure 1: Fuzzy shape with three sets: low (A), medium (B), and high (C) [7]

Attacks on systems do not always have a fixed patter, so fuzzy logic is used to detect patterns that have a behavior that is between normal and unusual. A fuzzy logic rule has the following format: If <*condition*> then <*consequence*>, where *condition* is a fuzzy variable, and *consequence* is a fuzzy set. An example of this is: if the number of packets with the same destination address is high, then the pattern is unusual. To determine what is considered high, the values of the packets are divided into fuzzy sets. In Figure 1 there are three sets: low (region A), medium (region B), and high (region C). The x-axis are the values in the fuzzy set (number of packets), and the y-axis is the membership function, which determines the region. For example, if the number of packets with the same destination address is 15 then this will be considered as low for a degree of 0.4, but will be considered medium or high for other degrees. If the number of packets is determined to be high, then the connection will be aborted. [7]

2.6 Genetic Algorithm

Genetic algorithms are a search technique used to find solutions to problems *Mutation*, *selection*, and *crossover* are used to evolve and improve rules.

Mutation is where random bits in a solution are changed. Selection is where solutions that have a better *fitness* are chosen over the other solutions. The fitness function determines the quality of a particular solution. Crossover is where two solutions swap one of their characteristics with the other to form two new solutions.

The solution to a problem is represented as a chromosome. First, a randomly generated population of chromosomes is created. According to the characteristics of the problem, the positions of each chromosome are encoded as bits, characters, or numbers. Then mutation, selection, and crossover are applied to each generation and eventually the best solution is found. [7]

2.7 KDD99 Data Set

The KDD99 data set is a benchmark data set that was generated by simulating a military network environment in 1999 [12]. It has long been a standard data set for intrusion detection. The data in the set is classified as normal or attack activity. KDD99 uses 41 *features*, which are properties of a *record* (either an attack or normal activity), that are used to describe the activity. The experiments that [12, 10] ran, which I will describe in Sections 3 and 4, used the KDD99 data set.

Say something about the features.

2.8 False Positives, False Negatives, True Positives, True Negatives

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. [15] Four different measures are then used to determine the accuracy of the algorithm in question.

A *false positive* (FP) happens when an intrusion detection system incorrectly identifies normal activity as being an attack. A *false negative* (FN) happens when an intrusion detection system fails to identify harmful activity. A *true positive* (TP) happens when an intrusion detection system correctly identifies activities to be attacks. A *true negative* (TN) happens when an intrusion detection system correctly identifies activities to be normal.

The *detection rate* (DR) of an intrusion detection system is the number of true positives divided by the total number of intrusions that happen. [4]

3. FUZZY GENETIC ALGORITHM IMPLEMENTATION

Figure out better title for Section 3

The focus of [11, 12] is on detecting new or unknown types of attacks in a network. The intrusion detection system used is able to identify normal network activity as well as attacks using a fuzzy genetic algorithm. This kind of algorithm is able to learn new attacks, and has a high detection rate. The authors of [11, 12] used 8 out of the 41 KDD99 features in their system:

1. duration: the length of the normal or attack activity 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.
4. root_shell: returns 1 if root shell is obtained, which means that the user is able to log in as root. This gives them the ability to do things like add accounts and change user passwords. If root shell is not obtained, 0 is returned.
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. [2]
6. srv_count: the number of connections to the same service as the current connection in the past two seconds.

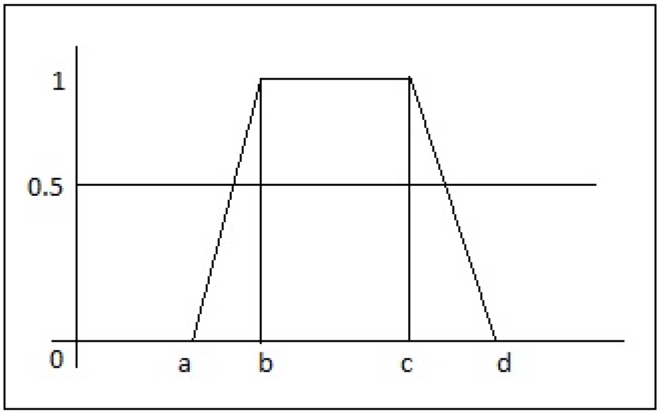

Figure 2: Trapezoidal shape with 4 parameters where $a \leq b \leq c \leq d$

Figure 3: Fuzzy encoding for a feature

010	011	100	101
a=2	b=3	c=4	d=5

7. serror_rate: the percentage of connections that have "SYN" errors. What are 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 that happens early in this process. [5]

8. same_srv_rate: the percentage of connections to the same service.

[9]

3.1 Fuzzy Algorithm

In order to measure the probability that a record is an attack, a trapezoidal shape was used in the algorithm. This is shown in Figure 2. The trapezoidal shape has four parameters: a, b, c, and d. Algorithm 1 calculates the probability of a record being an attack.

Figure out how to float algorithms.

Algorithm 1 Fuzzy Algorithm

if data value is between b and c **then**
  prob = 1.0
else if data value is between a and b **then**
  prob = (data $-$ a)/($b - a$)
else if data value is between c and d **then**
  prob = (d $-$ data)/($d - c$)
else
  prob = 0.0
end if

The four parameters a, b, c, and d are encoded into blocks of binary strings, where each block is a feature with values between 0.0 and 7.0. See Figure 3 for an example of a block. A rule has 12 blocks of features, and at the end of the string is the type of attack. An example of this is Figure 4.

3.2 Algorithm Overview

The algorithm that is used in [11, 12] first randomly generates rules. Then the rules are improved in the training phase, which can be seen in Algorithm 2. After that, the rules are used to classify the data in the testing phase. Algorithm 2 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 probability of an attack using the trapezoidal fuzzy rule shape. The probabilities of each block are then compared with a threshold to determine if the record represents an attack or normal behavior.

Algorithm 2 Fuzzy Genetic Algorithm

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

The fitness function to be maximized is:

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

In this implementation, a population size of 10 was used for each generation. An individual in the population represents a possible detection rule. The two best individuals from a present generation are preserved for the next generation. The other individuals in the new generation come from mutation and single-point crossover.

3.3 Experimental Design and Results

3.3.1 RLD09 Data Set

Possibly move this section to KDD99 section. 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 [11, 12] created their own data set, RLD09, to use in their experiments. To create the data set, the authors captured network data from the Computer Engineering Department at King Mongkut's University of Technology Thonburi, in Bangkok, Thailand. The data has around ten million preprocessed data packets. It has 17 different types of attacks

Figure 4: A rule with 12 blocks of features

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

that can be divided into denial of service attacks and probe attacks. It also has normal network activity. A packet sniffer was used to get information about TCP, UDP, and ICMP headers from protocol packets. Then this information was processed into 12 features:

- Features of TCP

  # of packets

  # of source ports

  # of destination ports

  # of fin flags, syn flags, push flags, ack flags, urgent flags

- Features of UDP

  # of packets

  # of source ports

  # of destination ports

- Features of ICMP

  # of ICMP packets

3.3.2 Experiments using Only RLD09

The experiments that the authors of [11, 12] performed used a total of 16,000 records of normal activity and 10,500 records of attack activity. Of the attack records, 4,000 of them were denial of service attacks and 6,500 were probe attacks.

In the first experiment, the fuzzy genetic algorithm was used to create denial of service and probe detection rules and then the rules were verified with known attack types. 10,000 records were used for the training set and all 26,500 records were used for the testing set. The two steps in the training process were to find a denial of service rule, and find a probe rule. Both of these rules were then used together in the testing process to identify attacks from the testing data set.

The detection rate of denial of service 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 shown in Table 1.

In the second 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 some of the results from the fuzzy genetic algorithm and a decision tree algorithm, which is another common way of addressing these problems. For further information, see [3]. When compared with the decision tree algorithm, the fuzzy genetic algorithm has a higher detection rate in all cases except 3 and 5. It can be seen that in cases 2 and 4 the decision tree has low detection rates, while the fuzzy genetic algorithm has much higher detection rates.

Table 2: Unknown Attack Experiment

Test Case	Unknown Attacks	Decision Tree DR (%)	Fuzzy Genetic DR (%)
1	Advance Port Scan Ack Scan Xmas Tree	Avg = 98.33	Avg = 100
2	UDP Flood Host Scan UDP Scan	Avg = 46.65	Avg = 99.80
3	Jping Syn Scan Fin Scan	Avg = 99.70	Avg = 98.75
4	UDP Flood RCP Scan Fin Scan	Avg = 70.35	Avg = 98.15
5	Http Flood RCP Scan Fin Scan	Avg = 99.94	Avg = 97.50

Table 3: KDD99 and RLD09 Results

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

3.3.3 Experiments using Both RLD09 and KDD99

The authors of [11, 12] also ran experiments that used and compared the RLD09 data set with the KDD99 data set. They used the KDD99 10% version file [9] for both the training dataset and testing dataset.

The authors of [11, 12] first trained and tested the fuzzy genetic algorithm 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 was the use of the KDD99 training set with the fuzzy genetifc algorithm to separate the data into two classes. Then each specific attack type was extracted and combined with normal activity. Ten tests were run, and Table 4 shows the accuracy of detecting some of the cases. The results showed that the detection rate of most of the cases were greater than 93%. The results also showed that the behavior of the attacks were highly distinctive from normal activity. There were only two cases that had low detection rates, one of which is case 1 in Table 4.

The final experiment that was run used only the RLD09 data set with the fuzzy genetic algorithm. 17 tests were run, and Table 5 shows the accuracy of detecting some of the cases. The results showed that the detection rate of a

Table 1: Results from Experiment 1

	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

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	Portsweep	Probe	6.40	0.00	93.66
4	Satan	Probe	0.74	3.75	99.22

Table 5: Results for RLD99 with Certain Attacks

Test	Attack	Type	FP (%)	FN (%)	DR (%)
1	Smurf	DoS	0.02	0	99.98
2	UDP Flood	DoS	11.06	0	89.59
3	Ackscan	Probe	0.03	0	99.97
4	Synscan	Probe	0.65	4.2	99.24

majority of the cases were greater than 97.5%. Again, there were only two test cases that had low detection rates, one of which is case 2 in Table 5.

4. GENETIC ALGORITHM IMPLEMENTATION

Figure out better title for Section 4
Switch Sections 3 and 4

4.1 Algorithm Overview

The authors of [10] used a genetic algorithm to make their intrusion detection system. The system is divided into two phases: a precalculation phase and a detection phase. In the precalculation phase, a set of chromosomes are created using training data. Then this set of chromosomes is used in the detection phase for comparison. Algorithm 3 is used in the precalculation phase.

Algorithm 3 Major steps in precalculation

```
range = 0.125
for each training data do
    if it has neighboring chromosome within range then
        Merge it with the nearest chromosome
    else
        Create new chromosome with it
    end if
end for
```

In the detection phase, a population is created for test data and then the type of the data is predicted. The set of chromosomes that was created in the precalculation phase is used in the detection phase to find the fitness of each chromosome in the population. Algorithm 4 is used in the detection phase.

Algorithm 4 Major steps in detection

```
Initialize the population
while number of generation is not reached do
    for each chromosome in the population do
        for each precalculated chromosome do
            Find fitness // Fitness function is the standard de-
            viation equation with distance
        end for
        Assign optimal fitness as the fitness of that
        chromosome
    end for
    Remove some chromosomes with worse fitness
    Apply crossover to the selected pair of chromosomes of
    the population
    Apply mutation to each chromosome of the population
end while
```

4.2 Experimental Design and Results

The authors of [10] used the KDD99 data set. They used only the numerical features of the KDD99 data set (34 out of 41 total features). For the training data set, the KDD99 10% version file was used, and for the test data set, the KDD99 corrected version file was used. Table 6 shows the number of records of normal and attack activity in the training and test sets.

Table 6: Number of records

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

In the detection phase, an initial population is created and then is compared with each chromosome that was created in the training phase. Crossover and mutation happen in the population in order to create a new population. The results from running the genetic algorithm are shown in Tables 7 and 8. The detection rate was 94.9%. The false positive rate was 30.5%. The false negative rate was 5%. The true negative rate was 69.5%.

The authors of [10] compared their results with the winning entry of the KDD99 Classifier Learning Contest [8], and found that they had a better detection rate for denial of service and user to root attacks than the winning entry.

5. CONCLUSIONS

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] Denial of service attacks. http://www.cert.org/tech_tips/denial_of_service.html, 2001.

Table 7: Results for Genetic Algorithm Experiment

		Predicted label					
		Normal	Probe	DoS	U2R	R2L	% Correct
Actual class	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 8: Results for Genetic Algorithm Experiment

		Predicted label	
		Normal	Intrusion
Actual class	Normal	True Negative (42,138)	False Positive (18,455)
	Probe	False Negative (12,528)	True Positive (237,908)

[2] Access control list. http://en.wikipedia.org/wiki/Access_control_list, 2013.

[3] Decision tree. http://en.wikipedia.org/wiki/Decision_tree, 2013.

[4] Intrusion detection system. http://en.wikipedia.org/wiki/Intrusion_detection_system, 2013.

[5] Transmission control protocol. http://en.wikipedia.org/wiki/Transmission_Control_Protocol, 2013.

[6] V. Bapuji, R. N. Kumar, A. Govardhan, and S. Sarma. Soft computing and artificial intelligence techniques for intrusion detection system. *Network and Complex Systems*, 2(4):24–31, 2012.

[7] R. Borgohain. Fugeids: Fuzzy genetic paradigms in intrusion detection systems. *CoRR*, abs/1204.6416, 2012.

[8] C. Elkan. Results of the KDD99 classifier learning contest. http://cseweb.ucsd.edu/ elkan/clresults.html, 1999.

[9] S. Hettich and S. D. Bay. KDD cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[10] 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.

[11] 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.

[12] 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.

[13] K. Kendall. Intrusion detection attacks database. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/docs/attackDB.html.

[14] 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.

[15] 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.