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

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 that is on a single host or multiple host systems, such as the contents of file systems and system calls to determine if there is a possible threat to the system. Network based intrusion detection looks at information that is from the network. Network based is useful for analyzing the traffic of multiple systems all at once. This type of intrusion detection system analyzes packets that travel through the network. [2]

2.2 Types of Networking Attacks

There are four different types of networking attacks: *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.

Denial of service attacks happen when an attacker makes a machine inaccessible to a user by making it too busy to serve legitimate network requests. Remote to user attacks happen when an attacker sends packets to a machine over the network in order to get access to things a local user would have on the machine. User to root attacks happen when an attacker starts out with access on the machine and then tries to gain root access to the system. Probing happens when an attacker examines a machine in order to collect information about weaknesses or vulnerabilities that in the future could be used in such a way that they compromise the system. [2]

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. A disadvantage of it is that it is an ineffective way of detecting unknown attacks. Signature-based detection is also called *knowledge-based detection* or *misuse detection*. [5]

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 vulnerabilities. Some disadvantages of this type of system is that they are expensive and they can detect an intrusive behavior as being normal because of insufficient data. Anomaly-based detection is also called *behavior-based detection*. [2]

2.4 Genetic Algorithm

2.5 Fuzzy Logic

2.6 Rules

Rules are represented by IF-THEN statements in the following format:

If <condition> then <action>

Write more here

2.7 KDD99 Data Set

Knowledge Discovery in Databases (KDD) is the process of extracting useful and interesting information from data in order to understand patterns in the data, with a limited amount of human intervention. Data is collected from several different sources and then is analyzed in order to produce patterns or models over the data. [6]

The KDD99 data set is a benchmark data set that was simulated in a military network environment in 1999 [4]. It is composed of 41 *features*, which are things that help to distinguish normal connections from attacks. The experiments that [4, 2] ran, which I will describe in Sections 3 and 4, used the KDD99 data set.

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

In an experiment, the researchers divide the data set into two subsets, a *training set* and a *testing set*. Algorithms are trained on the training set to look for patterns. These patterns are then verified when the testing set is used. [6] Four different measures are then used to determine the accuracy of an algorithm, which are described in Secion 2.8.

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

False positives and *False negatives* are two values that are used to evaluate the accuracy of an intrusion detection system. A false positive happens when an intrusion detection system incorrectly identifies normal activity as being an attack. A false negative happens when an intrusion detection system fails to identify harmful activity. [5]

Write more here about True Pos, True Neg, Detection Rate

3. FUZZY GENETIC ALGORITHM IMPLEMENTATION

Figure out better title for Section 3

3.1 Setup

The authors of [3, 4] used 8 out of the 41 KDD99 features in their system: duration, src_bytes, num_failed_logins, root_shell, num_access_files, srv_count, serror_rate, same_srv_rate.

Duration is the length of the connection in seconds.

src_bytes is the number of bytes sent from source to destination. num_failed_logins is the number of failed login attempts. root_shell returns 1 if root shell is obtained and 0 otherwise. num_access_files is the number of operations on access control files. srv_count is the number of connections to the same service as the current connection in the past two seconds. serror_rate is the percentage of connections that have "SYN" errors. What are SYN errors? same_srv_rate is the percentage of connections to the same service. [1]

3.2 Fuzzy Algorithm

The focus of [3, 4] 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 system used is evaluated in terms of detection speed, detection rate, and false alarm rate.

In order to measure the probability of an attack, a trapezoidal shape was used in the algorithm. The trapezoidal shape has four parameters: a, b, c, and d. Put in trapezoidal shape figure. Algorithm 1 calculates the probability of being attacked.

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
```

Figure 1: Fuzzy encoding for a feature

010	011	100	101
a	b	c	d

The four parameters a, b, c, and d are encoded into blocks of binary strings, where each block is a feature with parameters between 0.0 and 7.0. See Figure 1 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.

3.3 Algorithm Overview

The algorithm that is used in [3, 4] first randomly generates a rule. Then the rule is improved in the training phase. After that, the rules are used to classify the data into classes 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 combined by taking the average of the probabilities and comparing them with a threshold to determine if the record is an attack class or normal class.

Algorithm 2 Fuzzy Genetic Algorithm

```
for each record do
  for each rule do
    for each feature do
      prob = fuzzy();
      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 the 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 single-point crossover.

3.4 Experimental Design and Results

3.4.1 RLD09 Data Set

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 [3, 4] 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, as well as normal network activity. The attacks can be divided into denial of service attacks and probe attacks. 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. The features are the number of TCP, UDP, and ICMP packets, the number of TCP and UDP source ports, the number of TCP and UDP destination ports, and the number of TCP fin flags, syn flags, push flags, ack flags, urgent flags.

3.4.2 Experiments using Only RLD09

The experiments that the authors of [3, 4] performed used a total of 16,000 records of normal activity and 10,500 records of attack activity. Of the attack activity, 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 from the training set. Both of these rules were then used in the testing process to identify attacks from the testing data set. Algorithm 3 was used in the testing process.

Algorithm 3

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

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%. Detection of an attack happened 2-3 seconds after the packet data arrived at the intrusion detection system. Results from this experiment are shown in Table 1.

Fix tables

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 UDP Flood, Host Scan, and UDP Scan, (the first is DoS, and the last two are probe). 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. The fuzzy genetic algorithm has an accuracy of at least 95% in identifying unknown attacks. When compared with the decision tree algorithm, the fuzzy genetic algorithm has a higher detection rate in all cases except 2 and 4. It can be seen that in cases 1 and 3 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	UDP Flood Host Scan UDP Scan	Avg = 46.65	Avg = 99.80
2	Jping Syn Scan Fin Scan	Avg = 99.70	Avg = 98.75
3	UDP Flood RCP Scan Fin Scan	Avg = 70.35	Avg = 98.15
4	Http Flood RCP Scan Fin Scan	Avg = 99.94	Avg = 97.50

Table 3: Number of Records for KDD99

Attack	Training	Testing
Normal	39,387	39,337
DoS	158,597	158,503
Probe	1,550	1,674
Total	199,534	199,514

3.4.3 Experiments using Both RLD09 and KDD99

The authors of [3, 4] also ran experiments that used and compared the RLD09 data set with the KDD99 data set. They used the KDD99 10% version file for both the training dataset and testing dataset. The number of records of normal and attack activity for the KDD99 data set is shown in Table 3, and the number of records of normal and attack activity for the RLD09 data is shown in Table 4.

The authors of [3, 4] 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 5.

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 6 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 distinc-

Table 4: Number of Records for RLD09

Attack	Training	Testing
Normal	8,000	16,000
DoS	2,400	4,000
Probe	3,900	6,500
Total	14,300	26,500

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	1600	26500	1.13	4.10	97.92

Table 5: KDD99 and RLD09 Results

Data set	Testing Attack	Testing Normal	False Positive	False Negative	Detection Rate
KDD99	160,117	39,337	0.13	1.55	98.72
RLD09	10,500	16,000	1.14	3.39	97.97

Table 6: 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

tive from normal activity. There were only two cases that had low detection rates, one of which is case 1 in Table 6.

The final experiment that was run used only the RLD09 data set with the fuzzy genetic algorithm. 17 tests were run, and Table 7 shows the accuracy of detecting some of the cases. The results showed that the detection rate of a 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 7.

4. GENETIC ALGORITHM IMPLEMENTATION

Figure out better title for Section 4

4.1 Algorithm Overview

The authors of [2] 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 4 is used in the precalculation phase.

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 5 is used in the detection phase.

4.2 Experimental Design and Results

The authors of [2] used the KDD99 data set. For the training data set, the KDD99 10% version file was used, and for the testing data set, the KDD99 corrected version file was used. Write more here

Table 7: 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

Algorithm 4 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
```

Algorithm 5 Major steps in detection

```
Initialize the population
crossoverRate = 0.15, mutationRate = 0.35
while number of generation is not reached do
  for each chromosome in the population do
    for each precalculated chromosome do
      Find fitness
    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
```

5. CONCLUSIONS

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] S. Hettich and S. D. Bay. Kdd cup 99 task description. http://kdd.ics.uci.edu/databases/kddcup99/task.html, 1999.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.