Control Theory and Informatics
ISSN 2224-5774 (Paper) ISSN 2225-0492 (Online)
Vol.4, No.1, 2014
www.iiste.org

Application of Genetic Algorithm in Intrusion Detection System

Omprakash Chandrakar (Corresponding author)
Associate Professor, Department of Computer Science and Technology
Uka Tarsadia University, Bardoli, Gujrat, India.
Mob. No.:09374279151, E-mail : opchandrakar@gmail.com

Rekha Singh
Disha College, Kota, Ram Nagar, Raipur, India
Mob. No.: 09827103180, E-mail : rekhac.mca@gmail.com

Dr. Lal Bihari Barik
Faculty of Computing & Information Technology-Rabigh, King Abdulaziz University
Email: lalbihari@gmail.com

Abstract

The existing NIDSs involve various mechanisms in order to identify the patterns related to the network problems. In this paper we described the implementation of genetic algorithm using steady-state selection mechanism. We have performed various test cases in order to analyze the effect of varying iterations and varying initial chrome length along with different fitness functions.

The analyzed result would be helpful to have quicker generation of reliable rule sets for Network Intrusion Detection System using less/more number of iterations, depending on the varying initial chrome length.

Keywords: Data Mining, Genetic Algorithm, Network Intrusion, Network Intrusion Detection System.

1. Introduction

Intrusion Detection System is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. Incidents have many causes, such as malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized. Although many incidents are malicious in nature, many others are not; for example, a person might mistype the address of a computer and accidentally attempt to connect to a different system without authorization.

There exist a number of mechanisms in order to identify the patterns related to the network problems. One of them is Genetic Algorithm, in which some basic operations are performed in order to generate the patterns (also termed as Chromosomes or RULE SETS). These generated patterns can be used along with the audit dataset as input into the Network Intrusion Detection System, in order to achieve the kind of intruder in the audit dataset.

2. Network Intrusion Detection System

2.1 Introduction

An intrusion detection system (IDS) is software that automates the intrusion detection process. An intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents. This section provides an overview of IDS and IPS technologies as a foundation for the rest of the publication. It first explains how IDS and IPS technologies can be used. Next, it describes the key functions that IDS and IPS technologies perform and the detection methodologies that they use. Finally, it provides an overview of the major classes of IDS and IPS technologies.

2.2 Components of NIDS

An intrusion detection system normally consists of three functional components:

The first component of an intrusion detection system, also known as the event generator, is a data source. Data sources can be categorized into four categories namely Host-based monitors, Network-based monitors, Application-based monitors and Target-based monitors.

The second component of an intrusion detection system is known as the analysis engine. This component takes information from the data source and examines the data for symptoms of attacks or other policy violations.

The analysis engine can use one or both of the following analyzing approaches:

2.2.1 Misuse/Signature-Based Detection

This type of detection engine detects intrusions that follow well-known patterns of attacks (or signatures) that exploit known software vulnerabilities. The main limitation of this approach is that it only looks for the known weaknesses and may not care about detecting unknown future intrusions.

2.2.2 Anomaly/Statistical Detection

An anomaly based detection engine will search for something rare or unusual. They analyses system event streams, using statistical techniques to find patterns of activity that appears to be abnormal. The primary disadvantages of this system are that they are highly expensive and they can recognize an intrusive behavior as normal behavior because of insufficient data.

The third component of an intrusion detection system is the response manager. In basic term, the response manager will only act when inaccuracies (possible intrusion attacks) are found on the system, by informing someone or something in the form of a response.

2.3 Types of Network Attacks

2.3.1 Denial of Service (DoS)

A DoS attack is a type of attack in which the hacker makes a computing or memory resources too busy or too full to serve legitimate networking requests and hence denying users access to a machine e.g. apache, smurf, neptune, pingof death, back, mail bomb, UDP storm etc. are all DoS attacks.

2.3.2 Remote to User Attacks (R2L)

A remote to user attack is an attack in which a user sends packets to a machine over the internet, which s/he does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer.

2.3.3 User to Root Attacks (U2R)

These attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain super user privileges.

2.3.4 Probing

Probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system. This technique is commonly used in data mining.

2.4 Types of NIDS

2.4.1 Host Based Intrusion Detection (HIDS)

HIDSs evaluate information found on a single or multiple host systems, including contents of operating systems, system and application files.

2.4.2 Network Based Intrusion Detection (NIDS)

NIDSs evaluate information captured from network communications, analyzing the stream of packets which travel across the network.

3. Genetic Algorithm (GA)

Genetic algorithm is a family of computational models based on principles of evolution and natural selection. This algorithm converts the problem specific domain into a model, using a chromosome-like data structure. Chromosome-like data structure is evolved by performing selection, recombination, and mutation operators over the various chromosomes.

3.1 Components of GA

There are mainly 6 components in Genetic Algorithm System.

3.1.1 Evaluation function (or fitness function)

A fitness function is a particular type of objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims.

3.1.2 Population size, Crossover Rate & Mutation Rate

Population size: Good population size is about 20-30, however sometimes sizes 50-100 are reported as best. Some research also shows that best population size depends on encoding, on size of encoded string. It means, if you have chromosome with 32 bits, the population should be say 32, but surely two times more than the best population size for chromosome with 16 bits.

Crossover rate: Crossover rate generally should be high, about 80%-95%. (However some results show that for some problems crossover rate about 60% is the best.)

Mutation rate: On the other side, mutation rate should be very low. Best rates reported are about 0.5%-1%.

3.1.3 Encoding Mechanism

Binary Encoding: In binary encoding every chromosome is a string of bits, 0 or 1.

Permutation Encoding: In this encoding mechanism, every chromosome is a string of numbers, which represents number in a sequence

Value Encoding: In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.

Tree Encoding: In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language.

3.1.4 Parent Selection Mechanism

Roulette Wheel Selection: Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have.

Rank Selection: Rank selection first ranks the population and then every chromosome receives fitness from this ranking.

Steady-State Selection: For every generation a few (good - with high fitness) chromosomes are selected for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

Elitism: Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way.

3.1.5 Variation Operators (crossover and mutation)

Crossover selects genes from parent chromosomes and creates a new offspring.

Mutation changes randomly the new offspring.

There are various ways to perform cross-over & mutation, as per the kind of encoding implementation listed using a table(ref. Table 1)

3.1.6 Survivor Selection Mechanism (replacement)

The survivor selection mechanism is normally based on the type of “parent selection mechanism” been selected. This operation is basically used for deciding which of the existing chromosome should be replaced with the new generated (off-spring) chromosome.

4. Proposed System

We have chosen Genetic Algorithm to make our own intrusion detection system. This section gives an overview of the algorithm and the system. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. The range of the applications that can make use of genetic algorithm is quite broad. In computer security applications, it is mainly used for finding optimal solutions to a specific problem.

The process of a genetic algorithm usually begins having a randomly selected set of chromosomes of a specific size, acting as an input. These chromosomes are representations of the problem to be solved. According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are referred to as genes and are changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation function is used to calculate the “goodness” of each chromosome. During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes.

Figure 1 shows the structure of a simple genetic algorithm. It starts with a randomly generated population, evolves through selection, recombination (crossover), and mutation. Finally, the best individual (chromosome) is picked out as the final result once the optimization criterion is met (ref. Fig.1).

Proposed Algorithm

Algorithm:

1. Input the value for population size and Iteration Count
2. Initialize the new generated chromosome as per binary encoding
3. Set the genes (chg, gb) to the value "false".
4. Calculate fitness value for each generated initial chromosome
5. Now start on a loop for specified iteration.
6. For each iteration perform below operations:
   - **Sorting:** Sort the chromosome set as per fitness value
   - **Selection:** Select the 2 chromosomes such that they have the highest fitness value
   - **Crossover:** Generate new off-spring using the combination of the selected chromosomes (single point crossover is performed, selecting the crossover point randomly)
   - Assign the new generated chromosome, replacing the chromosome having least fitness value in the chromosome set
   - Set the gene(chg) value to "true", for this chromosome, indicating that the chromosome has been modified (not as initial)
   - **Mutation:** Change the value for any single gene for the generated off-spring (randomly)
   - Calculate fitness value for the new off-spring
   - Set value for gene(gb), if its fitness value is at least half the highest fitness value in the set chromosome
7. Display the chromosome set obtained after the specified iterations, once sorted again.

References

Ren Hui Gong, Mohammad Zulkernine, Purang Abolmaesumi, "A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection", School of Computing, Queen's University, Kingston, Ontario, Canada, 2005.

Mohammad Sazzadul Hoque1, Md. Abdul Mukit2 and Md. Abu Naser Bikas, "AN IMPLEMENTATION OF INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM", International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.2, March 2012

Adhitya Chittur, "Model Generation for an Intrusion Detection System Using Genetic Algorithms", Ossining High School Ossining, NY, November 27, 2001

Brian Lavender, "Implementation of Genetic Algorithms into SNORT, a Network Intrusion Detection System"

Wei Li, "Using Genetic Algorithm for Network Intrusion Detection", Department of Computer Science and Engineering Mississippi State University, Mississippi State, MS 39762

Data Mining for Network Intrusion Detection: Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, Jonathan Tivel The MITRE Corporation1820 Dolley Madison Blvd.McLean, VA 22102 (703) 983-5274

Genetic Algorithm Tom V. Mathew Assistant Professor, Department of Civil Engineering, Indian Institute of Technology Bombay, Mumbai-400076.

M. Revathi 1 1Department Of Information Technology, Bharathiar University, Coimbatore, Tamilnadu, India T.Ramesh2 Assistant Professor Department Of Information Technology, Bharathiar University, Coimbatore, Tamilnadu, India

Modeling An Intrusion Detection System Using Data Mining And Genetic Algorithms Based On Fuzzy Logic by G.V.S.N.R.V. Prasad Y.Dhanalakshmi Dr.V.Vijaya Kumar Dr I.Ramesh Babu Professor Scholar Professor &Dean Professor Dept. of CSE Dept of CSE Dept. of CSE &IT Dept. of CSE Gudlavalleru Engg.College A.N.U G.I.E.T A.N.U Gudlavalleru Guntur Rajamandry Guntur

Importance of Intrusion Detection System (IDS) by Asmaa Shaker Ashoor (Department computer science, Pune University) Prof. Sharad Gore (Head department statistic, Pune University)

Guide to Intrusion Detection and Prevention Systems: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

Understanding Intrusion Detection Systems:

http://www.sans.org/reading_room/whitepapers/detection/understanding-intrusion-detection-systems_337

Using Genetic Algorithm for Network Intrusion Detection: http://www.security.cse.msstate.edu/docs/Publications/wli/DOECSG2004.pdf

Understanding-Intrusion-Detection-Systems_337

Intelligent Network Intrusion Detection Using DT and BN Classification Techniques

A genetic algorithms based approach for conflicts resolution in requirement.

Experiment

For a sample test, we have taken a chromosome structure as below:

- 6 genes representing the network data
- 1 gene representing the change in chromosome (1=changed, 0=unchanged)
- 1 gene representing the chromosome satisfying the condition : fitness value > (highest fitness value)/2 (1=satisfied, 0=unsatisfied)

We have under gone two fitness functions (x^2-5x and x^2+12x-5), along with the population size of 4 & 15.

We have performed up to 14 iterations for each population size, each fitness functions.(ref. Table 2)

The resultant output is as shown in classified into 2 groups depending on the iteration cycle: Group-1: 1-7 iteration, Group-2:8-14 iterations.

Above table, provides the total number of changed chromosomes for each chromosome size (4 & 15), depending on the each classified group.

Ref.figures5-8 shows the experiments.

Conclusion

From the above experiment, we have finally reached to two different conclusions, for the case of genetic algorithm being implemented using steady–state selection mechanism.

Conclusion-1

If there are less number of initial chromosomes, the no. of iteration to be performed in order to have new reliable chromosomes, will be less.

Similarly, for higher number of initial chromosomes, it requires more number of iterations to be performed to have new reliable chromosomes (ref. Table 3).

Conclusion-2

If there is less number of iterations, then the no. of new reliable chromosomes being generated goes on decreasing with an increase in length of initial chromosomes being given.

Similarly, if we keep on increasing the iterations then the resultant new reliable chromosomes scope is not much affected due to varying chromosome length.

Table. 1. Variation Operators Table

Encoding Type	**Cross-Over Type**	**Mutation Type**
Binary	Single Point Cross-Over Two Point Cross-Over Uniform Cross-Over Arithmetic Cross-Over	Bit Inversion
Permutation	Single Point Cross-Over	Order Changing
Value	Same as for Binary	Add/Subtract or Replace by Random number
Tree	Tree Cross-Over	Changing Operator, number

Table. 2. Outcome of the Experiment

Population Size	**Total Number of Iterations**	
	Group-1	**Group-2**
4	31	42
15	57	145

Table. 3. Conclusion Table

Population Size	**Total Number of Iterations (% change)**			
	Group-1	**Group-2**		
4	55.36%	75%	Likely Similar	**Conclusion 1**
15	27.14%	69.05%	Much Differing	
	Nearly Half	**Slightly changed**		

Figure. 1. Coding in GA

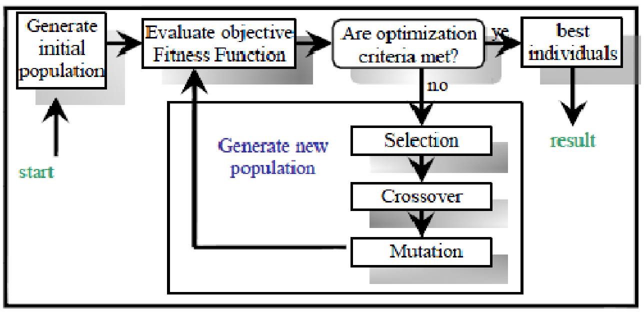


Figure. 2. Flow Chart for the proposed Algorithm

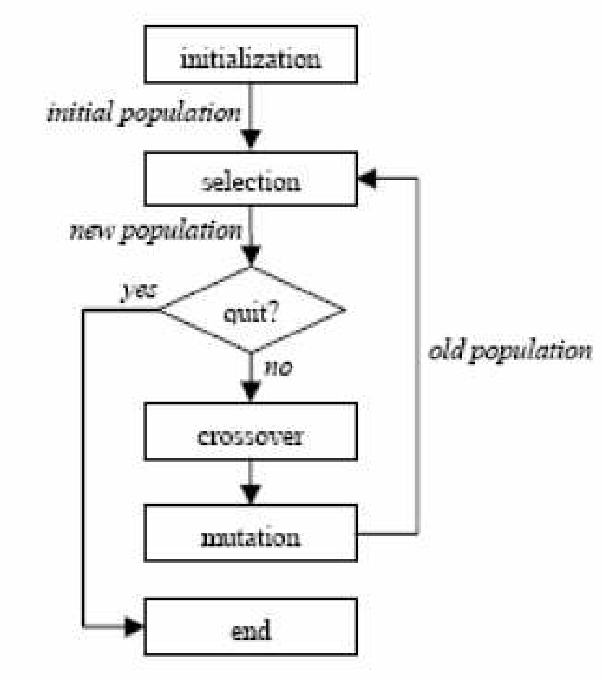
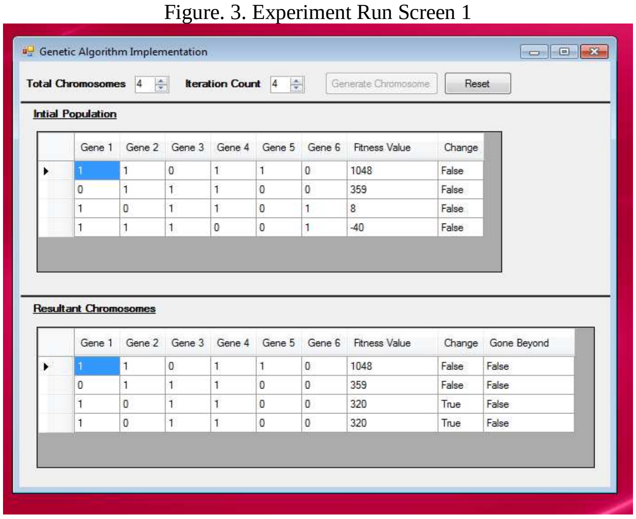


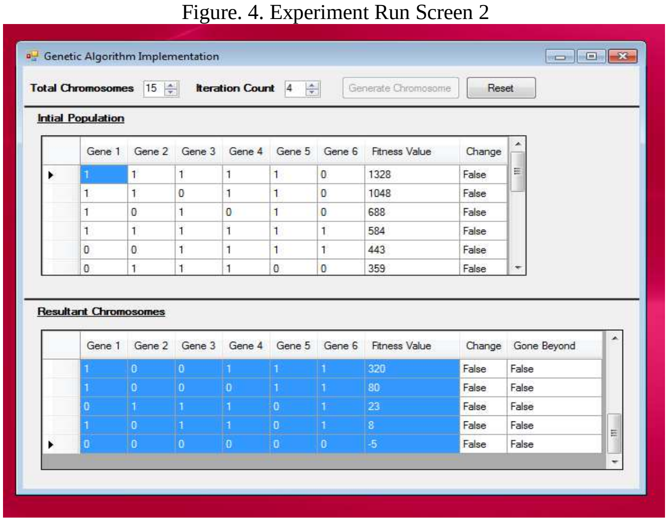
Figure. 3. Experiment Run Screen 1



Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Fitness Value	Change
1	1	0	1	1	0	1048	False
0	1	1	1	0	0	359	False
1	0	1	1	0	1	8	False
1	1	1	0	0	1	-40	False

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Fitness Value	Change	Gone Beyond
1	1	0	1	1	0	1048	False	False
0	1	1	1	0	0	359	False	False
1	0	1	1	0	0	320	True	False
1	0	1	1	0	0	320	True	False

Figure. 4. Experiment Run Screen 2



Genetic Algorithm Implementation

Total Chromosomes 15 Iteration Count 4 Generate Chromosome Reset

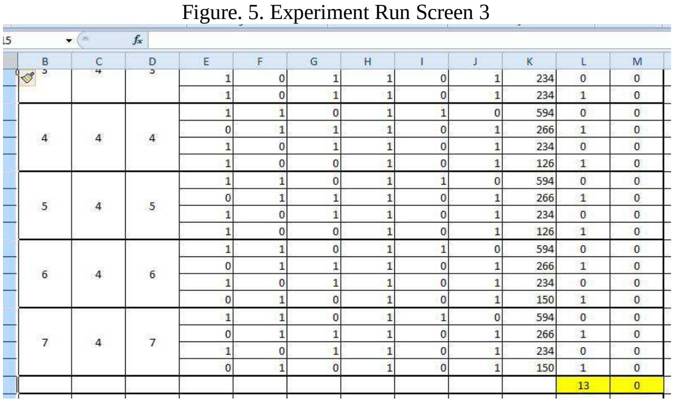
Intial Population

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Fitness Value	Change
1	1	1	1	1	0	1328	False
1	1	0	1	1	0	1048	False
1	0	1	0	1	0	688	False
1	1	1	1	1	1	584	False
0	0	1	1	1	1	443	False
0	1	1	1	0	0	359	False

Resultant Chromosomes

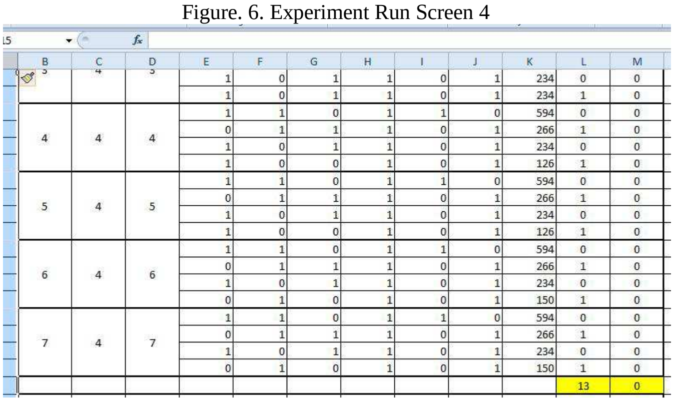
Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Fitness Value	Change	Gone Beyond
1	0	0	1	1	1	320	False	False
1	0	0	0	1	1	80	False	False
0	1	1	1	0	1	23	False	False
1	0	1	1	0	1	8	False	False
0	0	0	0	0	0	-5	False	False

Figure. 5. Experiment Run Screen 3



B	C	D	E	F	G	H	I	J	K	L	M
3	4	3	1	0	1	1	0	1	234	0	0
			1	0	1	1	0	1	234	1	0
4	4	4	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			1	0	0	1	0	1	126	1	0
5	4	5	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			1	0	0	1	0	1	126	1	0
6	4	6	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			0	1	0	1	0	1	150	1	0
7	4	7	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			0	1	0	1	0	1	150	1	0
										13	0

Figure. 6. Experiment Run Screen 4



B	C	D	E	F	G	H	I	J	K	L	M
3	4	3	1	0	1	1	0	1	234	0	0
			1	0	1	1	0	1	234	1	0
4	4	4	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			1	0	0	1	0	1	126	1	0
5	4	5	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			1	0	0	1	0	1	126	1	0
6	4	6	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			0	1	0	1	0	1	150	1	0
7	4	7	1	1	0	1	1	0	594	0	0
			0	1	1	1	0	1	266	1	0
			1	0	1	1	0	1	234	0	0
			0	1	0	1	0	1	150	1	0
										13	0