Asian Journal of Information Technology 6 (8): 896-901, 2007
ISSN: 1682-3915

Rule Based Network Intrusion Detection System Based on Crossover and Mutuation

[1]S. Jeya and [2]K. Ramar
[1]K.S.R. College of Engineering, Thiruchengode, Namakkal, Tamil Nadu, India
[2]Department of HOD/CSE, National Engineering College, Kovilpatti, Tamil Nadu, India

Abstract: Quickly increased complexity, openness, interconnection and interdependence have made computer systems more vulnerable and difficult to protect from malicious attacks. Network intrusion detection system plays a vital role in today's network. In this system using Genetic Algorithm (GA), crossover, mutation and other related techniques to used to detect network intrusion system. As the transmission of data over the Internet increases, the need to protect connected systems also increases. Although, the field of IDSs is still developing, the systems that do exist are still not complete, in the sense that they are not able to detect all types of intrusions. Some attacks which are detected by various tools available today cannot be detected by other products, depending on the types and methods that they are built on. Network Intrusion Detection Systems based on crossover and mutation is the latest technology used for this purpose. The network intrusion detection system should be adaptable to all type of critical situations arise in network. This is helpful for identification of complex anomalous behaviors. This research is focused on the TCP/IP network protocols.

Key words: Genetic algorithm, crossover, mutation, network sniffer, misuse detection, anomaly detection, niching technique and fitness value

INTRODUCTION

The study describes the design of an intrusion detection system, which can provide active detection and automated responses during intrusions. It is designed to be a sense and response system that can monitor various activities on the network (i.e. looks for changes such as malfunctions, faults, abnormalities, misuse, deviations, intrusions, etc.). Due to advances in information communication technology, Intrusion Detection Systems (IDSs) have become essential tools for the security of computer systems. Generally, IDSs can be divided into two categories: misuse-detection systems and anomaly-detection systems. Misuse-detection systems use knowledge about known attacks and attempt to match current behavior against those attack patterns (Kemmerer and Vinga, 2002). Most commercial IDSs use this approach because known attacks can be detected economically and reliably. A shortcoming of this approach is that it cannot detect unknown attacks. Anomaly detection, which uses knowledge about normal behaviors and attempts to detect intrusions by noting significant deviations, has been studied actively.

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network (Yu *et al*., 2006). Intrusions are caused by attackers accessing the systems from the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized and authorized users who misuse the privileges given them. Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process. Intrusion detection allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. Intrusion Detection Systems (IDS) attempts to detect intrusion through analyzing observed system or network activities. It may also report, intrusions by emailing or paging system administrator and even disconnect intrusion connection locally.

In the near future, sensor technology will be integrated into our everyday computing environment. We've seen something similar with firewalls, which are now an integral part of operating systems: Both UNIX and Windows provide some form of host-based fire walling. It's now time for operating systems and network software to integrate intrusion detection sensors. Intrusion detection will no doubt become a default feature, rather than an esoteric option. Intrusion detection systems perform the following functions well.

Corresponding Author: S. Jeya, K.S.R. College of Engineering, Thiruchengode, Namakkal, Tamil Nadu, India

- Monitoring and analysis of system events and user behaviors
- Testing the security states of system configurations
- Recognizing patterns of system events that correspond to known attacks
- Recognizing patterns of activity that statistically varies from normal activity
- Managing operating system audit and logging mechanisms and the data they generate
- Alerting appropriate author by appropriate attacks are detected

There are three fundamental functions of IDS: Monitoring, analysis, response and generating reports (Chaker, 2006). The different sources of event information can be drawn from different levels of the system, with network, host and application monitoring system. Analysis makes sense of the events derived from the information sources, deciding when those events indicate that intrusions are occurring or have already taken place. Responses can be generated involving some automated intervention on the part of the system and involving reporting IDS findings to humans, who are then expected to take action based on those reports.

We have used Genetic algorithm, rule based system, expert system to detect the intrusive behavior of the system. Moreover, using the effective technique crossover and mutation is the additional feature of the proposed system.

Learning the behavior of a program is a well-known and widely used intrusion-detection paradigm. Several kinds of intrusion can occur by inducing program misuse that exploits the bugs of the specific program. Victim programs behave differently from normal execution programs (Mishra *et al*., 2004). Therefore, learning normal behaviors and recognizing significant deviations can be efficient for anomaly detection. Though, there are many ways to observe the behavior of a program, capturing the system call sequences is a typical and efficient method.

In this way, a normal profile can be built by learning the patterns of the short system-call sequences. Programs that show sequences that deviate from normal sequence profiles are considered to be victims. Namely, intrusion detection by learning the behavior of a program can be transformed to the problem of learning and classifying temporal sequence data. Machine-learning techniques that are known for good solutions for this kind of problem have been applied. ADAM (Audit Data Analysis and Mining) is an online network based IDS which uses association rules algorithm in detection (Vigairagvan *et al*., 2007).

PROPOSED SYSTEM

In the proposed system contain GA, crossover and mutation, expert system and rule based system. The GA generated more effective standard rules for detecting intrusion using crossover and mutation. These rules are stored in a rule based system for expert systems reference.

Genetic algorithm: Genetic Algorithms (GAs) represent a class of general-purpose adaptive search methods that mimic the metaphor of natural biological evolution. Genetic algorithms operate on a population of candidate solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. In general, in order to successfully employ GA to solve a problem, one needs to identify the following four components:

- Syntax of the chromosome-represent the problem space.
- Interpretation of the chromosome-decoding and checking the feasibility.
- Fitness Measure of the chromosome-determine the quality of solutions.
- Genetic Operators-crossover and mutation operators are used to manipulate candidate solutions.

Genetic algorithm works with a population of candidate solutions (chromosomes). The fitness of each member of the population (point in search space) is computed by an evaluation function that measures how well an individual performs with respect to the problem domain. The constraints can be incorporated in the evaluation function in the form of penalty terms. The better-performing members are rewarded and individuals showing poor performance are punished or discarded. So starting with an initial random population, the genetic algorithm exploits the information contained in the present population and explores new individuals by generating offspring in the next generation using genetic operations. By this selective breeding process it eventually reaches to a near-optimal solution with a high probability. Several different genetic operators have been developed, but usually two (crossover and mutation) are extensively used to produce new chromosomes. The crossover operator exchanges portions of information between the pair of chromosomes. The mutation operator is a secondary operator, which randomly changes the values at one or more genes of a selected chromosome in order to search for unexplored space. The workings of simple GAs have been described in detail elsewhere (Sang *et al*., 2004; Sarasamma *et al*., 2005).

In this system GA work on intrusion within a computer simulation, a population of many individuals is created, each individual representing a possible mathematical model. Each individual has one or more chromosomes (set of rules) that function as basic instructions to the individual in a cause and effect manner. An individual is measured by the aggregate performance of its chromosomes. An initial population is created by complete randomization of the chromosomes and individuals of subsequent generations go through mutations, which are also randomized.

The performance of an individual is measured by a fitness function. A fitness function rates the performance of an individual in its environment by comparing the results of the individual's chromosomes with the desired results of the problem as defined by the author of the algorithm (Dong *et al*., 2005).

The Algorithm is as follows:

- Randomly generate an initial population M (0)
- Compute and save the fitness u (m) for e
- Each individual m in the current population M (t)
- Define selection probabilities p (m) for each individual m in M (t) so that p (m) is proportional to u(m)
- Generate M (t+1) by probabilistically selecting individuals from M (t) to produce offspring via genetic operators.
- Repeat step 2 until satisfying solution is obtained.

The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representation of the problem to be solved. According to the attribute of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are sometimes 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.

Genetic algorithms can be used to evolve simple rules for network traffic. These rules are used to differentiate normal network connections from anomalous connections. These anomalous connections refer to events with probability of intrusions. The rules stored in the rule base are usually in the following form:

if { condition } then { act }

For example, a rule can be defined as:

if {the connection has following information: source IP address 124.12.5.18; destination IP address:130. 18.206.55; destination port number: 21; connection time: 10.1 seconds } then {stop the connection}

The following steps are used to calculate the evaluation function. First the overall *outcome* is calculated based on whether a field of the connection matches the pre-classified data set and then multiply the weight of that field. The Matched value is set to either 1 or 0.

$$\text{Outcome} = \sum_{t=1}^{57} \text{Matched} * \text{weight}$$

The absolute difference between the outcome of the chromosome and the actual suspicious level is then computed using the following equation

$$\Delta = \text{Outcome-Suspicious level}$$

Once a mismatch happens, the penalty value is computed using the absolute difference. The ranking in the equation indicates whether or not an intrusion is easy to identify.

$$\text{Penalty} = \Delta * \text{ranking}/100$$

The fitness of a chromosome is computed using the above penalty:

$$\text{Fitness} = 1 - \text{penalty}$$

Obviously, the range of the fitness value is between 0 and 1. An evaluation function is used to determine the fitness of each population.

One network connection and its related behavior can be translated to represent a rule to judge whether or not a real-time connection is considered an intrusion. These rules can be modeled as chromosomes inside the population. The population evolves until the evaluation criteria are met. The generated rule set can be used as knowledge inside the IDS for judging whether the network connection and related behaviors are potential intrusions.

Crossover and mutation: Crossover is the main operator used for reproduction. It combines portions of 2 parents to create two new individuals, called offspring, which inherit a combination of the features of the parents. Recombination of genes on the same chromosome is accomplished by crossing over. Mutation is an incremental change made to each member of the population with a very small probability. Mutation enables new features to be introduced into a population. Without mutation all genes would exist in only one form.

The genetic algorithm starts with a population that has randomly selected rules. The population can evolve by using the crossover and mutations operators

(Ortiz *et al*., 2005). Due to the effectiveness of the evaluation function, the succeeding populations are biased toward rules that match intrusive connections. Ultimately as the algorithm stops, rules are selected and added into the IDS rule base (Fig. 1).

Traditional genetic algorithms have been used to identify populations of candidate hypotheses to a single global optimum. For this problem, a set of rules is needed as a basis for the IDS. As mentioned earlier, there is no way to clearly identity whether a network connection is normal or anomalous just using one rule. Multiple rules are needed to identify an unrelated anomaly, which means that several good rules are more effective than a single best rule. Another reason for finding multiple rules is that because there are so many network connection possibilities, a small set of rules will be far from enough. Using the genetic algorithm, we need to find local maxima as opposed to the global maximum. The niching techniques can be used to find multiple local maxima. It is based on the analogy to nature in that within each environment, there are different subspaces that can support different types of life (Sang and Aung, 2006). In a similar manner, genetic algorithm can maintain the diversity of each population in a multimodal domain, which refers to domains requiring the identification of multiple optima.

Algorithm of Crossover and Mutation

```
PROCEDURE evoNN ( PopSize, MaxGeneration, DataSet)
        NEURALNETWORK Pop[PopSize]
        NUMBER Fitness [PopSize]
        NUMBER Generation := 0
        Initialize(PopSize)
        WHILE Generation < MaxGeneration
            PartialLearning(Pop)
            FOR EACH Individual in Pop
                Individual. Fitness := Test (Indevidual, Dataset)
            EBDFIR
            NEURALNETWORK Next Pop[PopSize]
            Next Pop := Selection (Pop)
            Crossover (Next Pop)
            Mutation (Next Pop)
            Pop := Next Pop
            Generation += 1
        ENDWHILE
END PROCEDURE
```

Two basic methods, crowding and sharing can be used for niching. The crowding method uses the most similar member for replacement to slow down the population to converge towards a single point in the following generations. The sharing method reduces the fitness of individuals that have highly similar members and forces individuals to evolve to other local maxima that may be less populated (Ortiz *et al*., 2005). The similarity metrics used in these techniques can be phenotype to genotype similarity such as Hamming distance between bit representations, or phenotype similarity such as the relation between two network connections in this problem. The latter one is more fitful for finding rules used in IDS.

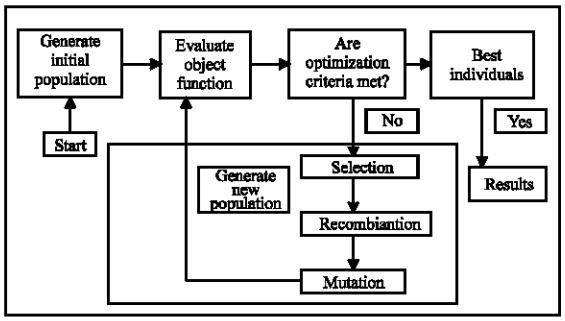

Fig. 1: Evaluation of crossover and mutation

Rule-based system: Most known network attacks can be characterized by a sequence of events. In rules-based detection systems, organizations form models of high-level system state changes or audit-log events that occur during attacks. These models form rules bases. Rules-based detection systems monitor system logs and resources, searching for models that match an attack profile. Administrators or security experts must regularly update the rules base to reflect newly discovered attack methods. Because rules-based systems monitor for known attack patterns, they generate very few false alarms (Fig. 2).

Rule-based system analyzes the traffic data passing through it and differentiates the traffic behaviors to be intrusive or normal. It use the rules stored in its knowledge base to detect and take actions when anomaly occurs in the traffic and undergoing some unauthorized activities. In brief, the rule set is applied to compare the patens in the traffic data with the patters customarily found during attack attempts. The IDS will alert the network administrator or security officer if the network is, or probably will be, under attack. The alerts are usually sent together with detail information of the suspected intrusion (Jungck and Shim, 2004). In general, the domain experts define the rules used in the rule-based IDS, by their experiences and observations. They observe the behavior of individual attack and define the corresponding rules to detect the attack. Moreover, the subsequence actions of network administrator to deal with the attack are also defined.

We need to collect enough historical data that includes both normal and anomalous network connections. This is the first part of the current system.

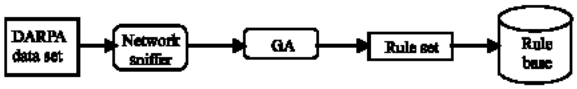

Fig. 2: Architecture of rule based system

The network sniffers analyze this data set and results are fed into GA for fitness evaluation. Then the GA is executed using crossover and mutation selecting the best rule, the rule set is generated. These rules are stored in a database to be used by the IDS. The degree of importance is hypothesized based on the domain knowledge. The purpose is to generate rules from a general knowledge base designed by experts. Though the accuracy of this knowledge base will result in more realistic actions, the heuristic rule set that we used can provide similar detection ability.

Expert system: The rule-based system is based on an expert system. An expert system is a computer program dedicated to solving problems and giving advice within a specialized area of knowledge. A good system can match the performance of a human specialist. The field of expert systems is the most advanced part of artificial intelligence. In the proposed system depending on an expert system for judge the intrusive behaviors of the network. In the architecture of rule based system we can create more useful rules using an expert system. The basic components of an expert system are a "knowledge base". The information stored in the expert system into a collection of rules, typically of "if-then" structure. The knowledge base of an expert system is small and therefore manageable--a few thousand rules at most. This techniques of detecting intrusion or anomalies usually involve monitoring the network and system parameters continuously over a period of time and collecting information about the network's normal behavior (Geer, 2006; Jungck and Shim, 2004; Leckie and Yasinsac, 2004). Accordingly, some parameters of the system are identified as the important indicators of abnormalities. The detection is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of the system usage. Our prototype system collects historical data and determines the normal (activities) usage of the system resources based on various monitored parameters.

RESULTS AND DISCUSSION

Intrusion detection system based on crossover and mutation results obtained; it is evident that the genetic algorithm designed for this experiment was successfully able to generate a security with the desired over training

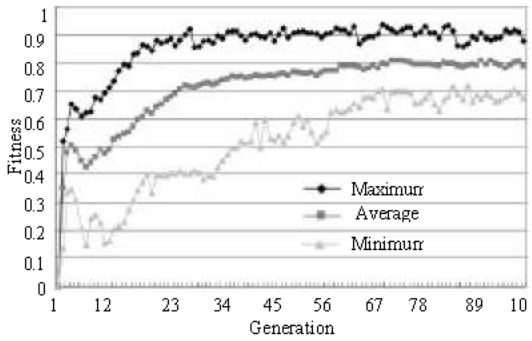

Fig. 3: Best individual's fitness value calculated

data. The genetic algorithm was able to perform the mutation and evolution strategies according to the fitness function. The genetic algorithm then successfully applied what it had learned to a real-world test case.

The Fig. 3 shows the evolution of the best individual of each generation. The fitness value of the best individual for each generation had an approximate steady increase, at which point it is apparent that the best possible individual possible by the current methods had been created. This demonstrated the ability of the crossover and mutation to successfully evolve an individual's model. Fitness increases as evolution proceeds and the maximum fitness converges to 0.9. This shows that we can find better network intrusion by evolution and that the evolved networks can classify the training data at about 91% accuracy.

However, without using crossover and mutation the largest intrusion detection rate is only 0.6899% of attacks. Using network intrusion detection system based on crossover and mutation the largest detection rate is 0.834% of attacks. The results presented in this paper show that "Network intrusion detection system based on crossover and mutation" are a promising method for the detection of malicious intrusions into computer systems.

CONCLUSION

Intrusion detection is a viable and practical approach for providing a different notion of security in our huge and existing infrastructure of computer and network systems. This system should be more helpful for identification of network anomalous behaviors. Using Genetic Algorithm, crossover and mutation technique we can create more useful rules to prevent network intrusions. The proposed method not only improved the detection performance but also reduced the time requirements.

Future work includes creating a standard test data set for the genetic algorithm proposed in this study and applying it to a test environment. It is necessary to find network structures that are particularly good for intrusion detection by analyzing the evolved structures. Further improvement of detection performance can be expected. Detailed specification of parameters to consider for genetic algorithm should be determined during the experiments. Combining knowledge from different security sensors into a standard rule base is another promising area in this work.

ACKNOWLEDGEMENT

The authors would like to thank the Mother Teresa Women's University and their colleague's for the useful comments, which improve this study. S. Jeya would like to especially thank my husband Mr. I. Murugan for his encouragement and heart full support.

REFERENCES

Geer, D., 2006. Behavior based Network Security goes mainstream. IEEE. Comp. Soc., 39: 14-17.

Jungck, P., Shim ssy, 2004. Issues in high speed internet security. IEEE. Comp. Soc., 37: 36-42.

Katar, C., 2006. Combining Multiple Techniques for Intrusion Detection. International Journal of Computer Science and Network Security, Vol. 6 No. 2B.

Kin, D.S., H.N. Ngugen and J.S. Park, 2005. Genetic Algorithm to improve SVM based Network Intrusion Detection System. Adv. Inform. Networking and Application, 2: 155-158.

Leckie, T. and A. Yasinsac, 2004. Metadata for Anomaly based Security Protocol attack Detection. IEEE. Trans. Knowledge Data Eng., 16: 1157-1168.

Mishra, A., K. Nadkarni and A. Patcha, 2004. Intrusion Detection in wireless ad-hoc Network. IEEE. Wireless Commun., 4: 48-60.

Ortiz Boyer, D., C. Hervas Martinez and N. Garcia Pedrajas, 2005. C1XL2: A crossover operator for evolutionary algorithms based on population features. J. Artificial Intelligence Res., 24: 1-48.

Sang Jun Han and Sung Bae Cho, 2006. Evolutionary neural network for anomaly detection based on the behaviour of a program. IEEE. Trans. Syst. Man and Cybernetics, Part B, 36: 559-570.

Sang Long Pao, T. and P.W. Wang, 2004. Net flow based Intrusion Detection System. IEEE. Int. Conf. Networking Sensing and Control, 2: 731-736.

Sarasamma, S.T., Q.A. Zhu and J. Huff, 2005. Hierarchical Kohonenen net for anomaly detection in Network Security. IEEE. Trans. Man and Cybernetics, 35: 302-312.

Vijairagavan, V., D. Shah, P. Galgali, D. Shah, V. Srinivasan and L. Bhatia, 2007. Marking Technique to isolate boundary router and attacker. IEEE. Comput. Soc., 40: 54-58.

Yu F Laksman, T.V., M.A. Motoyarna and R.H. Kata, 2006. Efficient Multi match Packet classification for Network Security Application. IEEE. J. Selected areas Commun., 24: 1805-1816.