JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN COMPUTER ENGINEERING

INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM

[1]MIT HTESHBHAI DAVE

[1]M.Tech [Information Technology] Student, Department of Information Technology,
NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY,
Raisen Road, Bhopal, Madhya Pradesh, India.

m.dave2005@gmail.com

ABSTRACT: *Intrusion Detection Systems (IDS's) try to detect network anomalies and maintain the secure state of network hosts. They have a long history [1] but even nowadays their efficiency is not 100% and correlates proportionally to the number of detected false positives. Nevertheless, IDS's are considered useful especially when new community hacking tools are emerging. This allows for greater number of users to experiment with modern exploits and increases the average security risk of every online system. Snort is an open source tool for network intrusion detection. In this research, we will focus on the Genetic algorithm technique and how it could be used in Intrusion Detection Systems giving some examples of systems and experiments proposed in this field. The purpose of this research is to give a clear understanding of the use of Genetic Algorithm in IDS. This research will mainly focus on using Genetic Algorithms with SNORT and categorize all the rules of Snort based on their functionality that will make them faster in detecting the attacks. The traditional approach of detecting attacks is very time consuming and also facing difficulties in automating the detection of intruders. It is therefore one of the focus points of this research to improve the efficiency of IDS using Genetic algorithms with snort which will decrease the amount of time for checking the rules of intrusion detection.*

KEY WORDS: *IDS (Intrusion Detection System), SNORT, G.A. (Genetic Algorithm).*

1. INTRODUCTION

Because of increased network connectivity, computer systems are becoming increasingly vulnerable to attack. These attacks often exploit laws in either the operating system or application programs. The general goal of such attacks is to subvert the traditional security mechanisms on the systems and execute operations in excess of the intruder's authorization. These operations could include reading protected or private data or simply doing malicious damage to the system or user files. The degree of protection from such malicious actions depends on the amount of time and effort spent building and maintaining the system's security defenses. By building complex tools, which continually monitor and report activities, a system security operator can catch potentially malicious activities as they occur. However, this involves a large expense in terms of time and money in both building and maintaining such a monitoring system. The monitoring will also impose a performance penalty on the system being Protected - something which the users may object to.

This research proposes a mechanism for building such a monitoring system which does not involve a significant process operates independently of the other agents, but they all cooperate in monitoring the system. This approach has significant advantages in terms of overhead, scalability and flexibility.

1.1 Intrusion and Intrusion Detection

An intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource and they can be categorized into two main classes:

Misuse intrusions: They are well defined attacks on known weak points of a system. They can be spotted by watching for certain actions being performed on certain objects.

Anomaly intrusions: These are based on observations of deviations from normal system usage patterns. They are caught by examining log messages resulting from system calls. This can be done using a pattern matching approach such as in [3]. However, anomaly intrusions are harder to detect. There are no fixed patterns that can be monitored for and so a more ―fuzzy" approach must be taken. Ideally we would like a system that combined human-like pattern matching capabilities with the vigilance of a computer program. Thus it would always be monitoring the system for potential intrusions, but would be able to ignore spurious false intrusions if they resulted from legitimate user actions. It would rely on heuristics to decide this - they could either be pre-specified (by a human operator) or learned by the system over time. However heuristics will not always guarantee perfect accuracy, so another goal is to minimize the probability of incorrect classification.

Intrusion Detection: An intrusion detection system (IDS) must [2] identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems. An intrusion detection system does not attempt to stop an intrusion as it occurs. Its role is to alert a system security officer that a potential security violation is occurring. As such it is a reactive, rather than proactive, form of system defense. An intrusion detection system can either be host-based or network-based. Often a system is a hybrid of the two approaches. A host based system will monitor all the activity on a single host computer. It will ensure that no user operations are violating the site security policy. A network based system monitors on a net-wide basis -it will consider actions occurring on the network and analyze them as to whether they constitute potential security violations.

In either case, the resulting system is often a large monolithic module. This module performs all of the monitoring, data gathering, data manipulation and decision making for the whole system. It either sits in, or on top of, the system kernel. Depending on the data being gathered, it can monitor system audit logs, user activities and system state. From these observations, it will deduce some metrics about the system's overall security state, and decide whether an intrusion is currently occurring. The most apparent problem with this approach to building IDS is the overhead it imposes on the system being protected. Often, a single system is analyzed by another system due to the added overhead of the IDS. If audit logs are being analyzed, the kernel must generate audit information for all the actions it performs.

This results in a large amount of information, which must be stored at least semi-permanently on some storage medium. Generating these detailed logs consumes both disk-space and CPU time. Once these logs are generated, the IDS must read and interpret them, attempting to correlate activities with other system information. IDS may also perform its own system monitoring - it may keep aggregate statistics which give a system usage profile. These statistics can be derived from a variety of sources - CPU usage, disk I/O, memory usage, activities by users, number of attempted logins etc. These statistics must be continually updated, and correlated with an internal model. This model may describe a set of intrusion scenarios or perhaps it encodes the profile of a ―clean" system. Either way, significant processing must occur in matching the observed profile to an internal model**.**

Types of IDS

1) **Network-based intrusion detection system (NIDS)**

It is an independent platform that identifies intrusions by examining network traffic and monitors multiple hosts. Network intrusion detection systems gain access to network traffic by connecting to a network hub, network switch configured for port mirroring, or network. In a NIDS, sensors are located at choke points in the network to be monitored, often in the demilitarized zone (DMZ) or at network borders. Sensors capture all network traffic and analyze the content of individual packets for malicious traffic. An example of a NIDS is SNORT.

2) Host-based intrusion detection system (HIDS*)*

It consists of an agent on a host that identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability databases, Access control lists, etc.) and other host activities and state. In a HIDS, sensors usually consist of a software agent. Some application-based IDS are also part of this category. Examples of HIDS are Tripwire and OSSEC.

3) Stack-based intrusion detection system (SIDS)

This type of system consists of an evolution to the HIDS systems. The packets are examined as they go through the TCP/IP stack and, therefore, it is not necessary for them to work with the network interface in promiscuous mode. This fact makes its implementation to be dependent on the Operating System that is being used. Intrusion detection systems can also be system-specific using custom tools and honey pots.

1.2 SNORT

Snort is a free and open source network intrusion prevention system (NIPS) and network intrusion detection system (NIDS) created by Martin Roesch in 1998. Snort is now developed by Sourcefire, of which Roesch is the founder and CTO. In 2009, Snort entered InfoWorld's Open Source Hall of Fame as one of the "greatest [pieces of] open source software of all time".

Snort's open source network-based intrusion detection system (NIDS) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching, and content matching. The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans.

Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyze it against a rule set defined by the user. The program will then perform a specific action based on what has been identified.

1.3 Genetic Algorithm

A Genetic Algorithm (GA) is a programming technique that mimics biological evolution as a problem-solving strategy. It is based on Darwinian's principle of evolution and survival of fittest to optimize a population of candidate solutions towards a predefined fitness.

GA uses an evolution and natural selection that uses a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. The process usually begins with randomly generated population of chromosomes, which represent all possible solution of a problem that are considered candidate solutions. Different positions of each chromosome are encoded as bits, characters or numbers. These positions could be referred to as genes. An evaluation function is used to calculate the goodness of each chromosome according to the desired solution; this function is known as ―Fitness Function. 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.

The following figure shows the structure of a simple genetic algorithm. Starting by a random generation of initial population, then evaluate and evolve through selection, recombination, and mutation. Finally, the best individual (chromosome) is picked out as the final result once the optimization meets its target. GA has some common elements and parameters which should be defined:

Fitness Function is defined as a function which scales the value individual relative to the rest of population. It computes the best possible solutions from the amount of candidates located in the population.

GA Operators according to the figure, the selection, mutation and crossover are the most effective parts in the algorithm as they are they participate in the generation of each population.

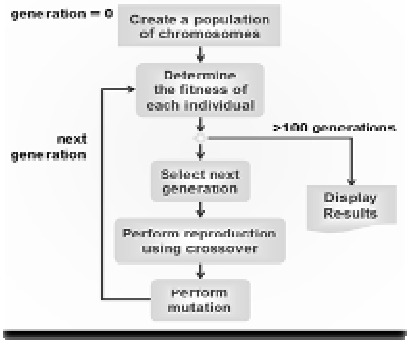

Fig 1:- Selection of fittest Chromosome in Genetic Algorithm

Selection is the phase where population individuals with better fitness are selected, otherwise it gets damaged.

Crossover is a process where each pair of individuals selects randomly participates in exchanging their parents with each other, until a total new population has been generated.

Mutation flips some bits in an individual, and since all bits could be filled, there is low probability of predicting the change.

To implement new IDS, that will be Combination of SNORT and Genetic Algorithm resulting in fast processing and detection rate.

2. COMPARING SNORT & SNORT WITH G.A.

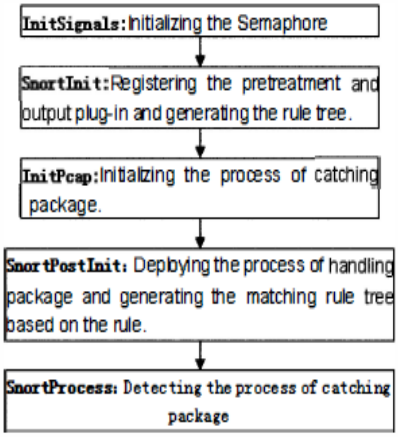

Fig 2:-The Working Process of SNORT [10]

F.Alserhana at el. [5] evaluated the performance of snort in high speed network. They were having two cases:

- Snort and attacker on different operating system platforms.
- Snort and attacker on same operating system platform.

In the first case snort only detect 20% of attack at 1.0 gbps speed of network traffic, when snort installed on Window XP SP2 and was generated from Linux 2.6. In second case snort detect 100% attack up to 400 mbps but only capture 30% of attack at 1.0 gbps. CPU utilization by snort is 80% at 500 mbps input traffic but only 30% at 1.0 gbps. So in this way performance of snort degrade as we increase network traffic. Different problem arise in existing system [5]. These problems are: Fidelity problem is caused when data packets traverse through long path and it can be modified by an attacker. Resource usage problem caused because component of IDS has to be run whole time while there is no intrusion occurred. Reliability problem occurred because the component of intrusion detection system is implemented as separated programs, they are susceptible to tempering and an intruder can disable or modify them.

Figure 2 Shows Traditional SNORT approaches which checks each and every rule at both sending and receiving time of package. Rather than inserting new rules in the SNORT tree Mapping of the rules smartly in such a way that doesn't compromises against

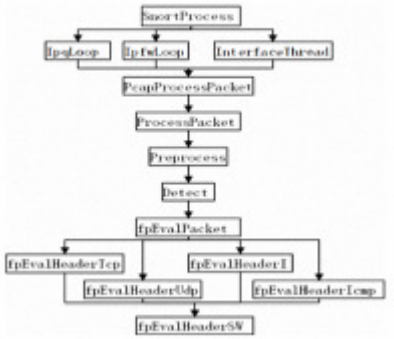

Fig. 3:- The Function Call Detecting Package [10]

Attacks and though having less amount of rules will reduce amount of time [9]. Figure 3 shows how SNORT rule tree matches the data packets.

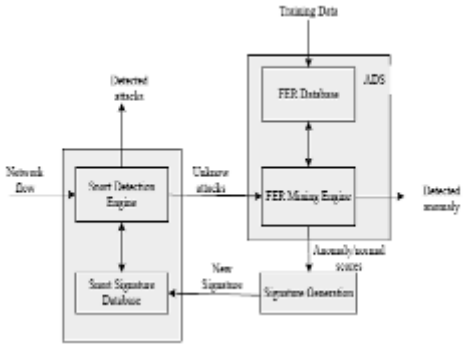

Fig 4:-The Architecture of HIDS with Training Data Set of ADS [9]

Instead of using simple SNORT rule set yu-xin ding, min xiao, ai-wu liu used a Hybrid Architecture with ADS rule set which checks for unknown attacks to SNORT and if there any then it will be sent to FER mining engine which checks for existing FER Database if there is no records that matches with new attack then FER mining engine creates new signature then it adds it to snort signature database[9],such system will produce more and more rules in SNORT signature database and will detect 90% above attacks on SNORT[9].Table 1 and 2 Shows the statistics of HIDS which was working on FER and ADS[9].

Times	connections	No. Of. FERs	Time(s)
1	2000	125	55
2	10000	1523	682
3	20000	2213	1125

Table 1:-Training Phase of ADS [9]

Times	NO. of Records	No. Of. FERs	Time (s)	Detected Events	Detection Rate (%)
1	2000	125	71	1914	95.7
2	10000	1523	871	9432	94.32
3	20000	2213	1512	18436	92.18

Table 2:-The Verification of FER Algorithm [9]

Though FER and ADS Approach of HIDS detects high amount of attacks but they became slow in processing and not detecting the attacks at real time [9].The main problem in this system was there are signature Database of SNORT is too big which was checked by SNORT [10], So most of the time is consumed in checking of rule set.

The Proposed system will use Genetic algorithm for minimization of snort rule set which will reduce the time amount in the whole process and also will stand against new attacks at real time. The proposed architecture is shown in Fig.5.

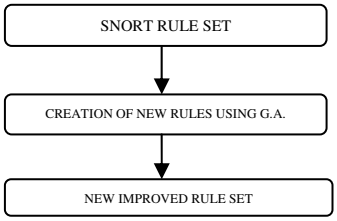

Fig.5 Architecture of SNORT using GA

So, Main Objectives of the new system are

- To categorize rules into based on its similarities in detecting the vulnerability and merging them formulating new improved rule set.
- To Integrate Genetic Algorithm with existing SNORT which may reduce the amount of time consumed in detecting the vulnerability.
- To implement new IDS, that will be Combination of SNORT and Genetic Algorithm resulting in fast processing and detection rate.

3. PROPOSED METHODOLOGY

The proposed Method will take the Genetic Algorithm technique performed by Guong [7] and proposed by Li and implement it in SNORT using the same DARPA datasets for testing. Using SNORT with genetic algorithms creates a unique situation. In both Li and Guong's work, they use the DARPA BSM format for training and testing. With SNORT, it reads raw network data, libpcap format. For the genetic algorithm to work in training mode with SNORT, the test data and the training data have attributes and an indicator of what kind of record it is. Since SNORT is rule based and has plug-in, I will develop a program or plug-in that will evaluate the DARPA training data using the same attributes identified by Li and Guong and integrate the results

into SNORT. Second, the testing part, I will write a module that will process the equivalent libpcap format DARPA data and determine how well the rules created from the training part correctly identify the equivalent malicious DARPA records in the test set.

4. FUTURE WORK

Li points to future work in creating standard test data, yet Guong continued the implementation of Li's proposed methodologies. Farschi notes the need for more Artificial Intelligence techniques for use in intrusion detection. The SNORT tool has proven its self popular with the security community and SPADE provides evidence that Artificial Intelligence methodologies can be integrated into SNORT. The mentioned items naturally lead to the idea of better integrating genetic algorithms in SNORT.

5. CONCLUSION

This paper evaluated that SNORT architecture can be made better by optimizing its rule set and also creating new rules with automatic detection of unknown attacks.

HIDS is usefull for automatic detection and creation of new rules for unknown attacks but it takes high amount of time in checking rule set as it adds new rules in snort signature database [9].

By using G.A. we can reduce the rule set of SNORT which leads to, Better utilization of SNORT by means of time and also not compromising security risk as it reduce the SNORT rule set based on similarities.

6. REFERENCES

[1]Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman Lincoln Laboratory MIT, 244 Wood Street, Lexington

[2] Efficient Snort Rule Generation using Evolutionary computing for Network Intrusion Detection Raghavan Muthuregunathan1, Siddhartha S2, Srivathsan R3, Rajesh SR4 1,2,3,4 Undergraduate students, Madras Institute of Technology, Anna University, Chennai 2009 First International Conference on Computational Intelligence, Communication Systems and Networks

[3] Implementation of Simple SNORT Processor for Efficient Intrusion Detection Systems 2009 IEEE Ehsan Azimi Dept. of CE. School of Engineering Islamic Azad University, Science and Research Branch Tehran, Iran M.B. Ghaznavi-Ghoushchi Dept. of EE. School of Engineering Shahed University Tehran, Iran Amir Masoud Rahmani Dept. of CE. School of Engineering Islamic Azad University, Science and Research Branch Tehran, Iran

[4] 2010 International Conference on Networking and Digital Society The Study on Network Intrusion Detection System of Snort Zhou Zhimin, Chen Zhongwen , Zhou Ti echeng, Guan Xiaohui Department of Computer Science Zhejiang Water Conservancy And Hydropoeer College Hangzhou, China

[5]2011 The 6th International Forum on Strategic Technology Research on Network Intrusion Prevention System Based on Snort Jiqiang Zhai, Yining Xie Computer Science & Technology College,Harbin University of Science and Technology Harbin, China.

[6] Using Genetic Algorithm for Network Intrusion Detection, Wei Li Department of Computer Science and Engineering Mississippi State University, Mississi ppi State, MS 39762

[7] Ren Hui Gong, Mohammad Zulkernine, Purang Abolmaesumi, A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection, Accessed 2005.

[8] Luke, Sean et al., A Java-based Evolutionary Computation (ECJ) and Genetic Programming ResearchSystem,http://cs.gmu.edu/~eclab/projects/ecj/,Accessed August 2009.

[9] Research and Implementation on SNORT-Based Hybrid Intrusion Detection system yu-xin ding, min xiao, ai-wu liu Key Laboratory of Network Oriented Intelligent Computation, Department of Computer Sciences and Technology, Harbin Institute of Technology Shenzhen Graduate School, China

[10]The Study on Network Intrusion Detection System of Snort Zhou Zhimin, Chen Zhongwen , Zhou Ti echeng, Guan Xiaohui,Department of Computer Science Zhejiang Water Conservancy And Hydropoeer College Hangzhou, China