Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

Detecting a denial of service using artificial intelligent tools, genetic algorithm

Maath. K. Al-anni*[1] and V. Sundarajan[2]
[1]*Computer Science Department;* [2]*Centre for Development of Advanced Computer CDAC, Pune University, India*, ** Baghdad University*
mkalanni@unipune.ernet.in; maadk_anni@live.com

Abstract: This paper describes novel work in using Genetic Algorithm for detecting misuse of programs. A brief overview of Intrusion Detection System, genetic algorithm and related detection techniques is presented. Developing rules manually through incorporation of attack signatures results is meaningful but weak as it is difficult to define thresholds. In this paper the proposition of learning the Intrusion Detection, rules based on genetic algorithms is presented. The experimental results are demonstrated on the KDD cup 99 and UoP intrusion detection data set (in the DARPA evaluations) in our experiments the characters of an attack such as Smurf and Apache2 (Denial of Service Attacks) are summarized through the KDD 99 data set and the effectiveness and robustness of the approach are discussed.

Keywords: Attack Signatures, Intrusion Detection, Genetic Algorithm, KDD Cup Set, Rule set, the 1999 DARPA evaluation.

Introduction

Intrusion detection system (IDS) is an important component of the defence-in-depth or layered network security mechanisms. An IDS collects system and network activity data, e.g., tcpdump (Tacobson *et al*., 1995) data, and analyzes the information to determine whether there is an attack occurring. The two main techniques for intrusion detection (ID) are misuse detection and anomaly detection. Misuse detection (sub) systems, for example, IDIOT (Kumar & Spafford, 1995) and STAT (Ilgun *et al*., 1995), use the "signatures" of known attacks, i.e., the patterns of attack behaviour or effects, to identify a matched activity as an attack instance. Misuse detection is not effective against new attacks, i.e., those that don't have known signatures. Anomaly detection (sub) systems, for example, the anomaly detector of IDES (Lunt *et al*., 1992), use established normal profiles, i.e., the expected behaviour, to identify any unacceptable deviation as possibly the result of an attack. Anomaly detection can be effective against new attacks. However, new legitimate behaviour call also be falsely identified as an attack, resulting in false alarm. In practice, reports of attacks are often sent to security staff for investigation and appropriate actions.

In 1998, DARPA conducted an evaluation to assess the state-of-the-art of ID research. The results showed that the best research systems had detection rates (i.e., the percentages of attack incidents correctly identified) below 70% (Lippmann *et al*., 2000). Most of the missed intrusions were new attacks that can lead to unauthorized user or root access to the mocked military network used in the evaluation. The results of the 1999 DARPA evaluation are even more troubling. With improved capabilities, e.g., the added modules for detecting the attacks missed in the previous evaluation, the research IDSs still had detection rates below 70% because many new attacks (that is, new in the 1999 evaluation) were missed. These evaluations showed that even the cutting-edge ID technology is not very effective against new attacks, and the improvement is often too slow and too little to keep up with the "innovation" by sophisticated attackers.

The IDSs can also be classified into two categories depending on where they look for intrusions. Fig. 1 shows the Computer Network with Intrusion Detection Systems. A host-based IDS monitors activities associated with a particular host, and a network based IDS listens to network traffic.

The research systems in the DARPA evaluations, like most of the leading commercial products, employ mainly misuse detection techniques. The main reason against deploying anomaly detection (sub) systems is that they tend to generate many false alarms and hence compromise the effectiveness of intrusion detection. Given that our adversaries will always develop and launch new types of attacks in an attempt to defeat our deployed intrusion prevention and detection systems, and that anomaly detection is the key to the defence against novel attacks, we must develop significantly better anomaly detection techniques.

In most computing environments, the behaviour of a subject (e.g., a user, a program, or a network element, etc.) is observed via the available audit data logs. The basic premise for anomaly detection is that there is intrinsic characteristic or regularity in audit data that is consistent with the normal behaviour and thus distinct from the abnormal behaviour. The process of building an anomaly detection model should therefore involve first studying the characteristics of the data and then selecting a model that best utilizes the characteristics. However, due to the lack of theoretical understanding and useful tools for characterizing audit data, most anomaly detection models are built based solely on "expert" knowledge or intuition (Lunt, 1993), which is often imprecise and incomplete given the complexities of today's network environments. As a result, the effectiveness of the models is limited. More seriously, a lot of research in anomaly detection (and intrusion detection in general) has been focusing on a specific (and ad hoc) method for a specific environment. The research results often do not contribute to the fundamental understanding of the field nor lend themselves to the broader problem domain.

Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

Our research aims to provide theoretical foundations as well as useful tools that can facilitate the IDS development process and improve the effectiveness of ID technologies. In this paper, we propose to use artificial intelligent tools AI to describe the characteristics of an audit data set (tcpdump data), suggest the appropriate anomaly detection model(s) to be built, and explain the performance of the model(s). We use case studies on *http* and *icmp* system call data, and network *tcpdump* data to illustrate the utilities of these measures.

Related work

Anomaly detection is an important research area in intrusion detection. In earlier systems (Haystack, 1988; Andersoii *et al*., 1995; Los Alamos National Laboratory. Wisdom and sense guidebook), a normal profile for a user or program is usually based on statistical measures of the system features, e.g., the CPU usage, the number of shell commands used, etc. In several recent studies, learning-based approaches were applied to build anomaly detection models using system call data of privileged programs (Forrest *et al*., 1996; Lee & Stolfo, 1998; Ghosh & Schwartzbard, 1999; Warrender *et al*., 1999). Lane and Brodley (1998) proposed a learning algorithm for analyzing user shell command history to detect anomalies. The algorithm attempts to address the "concept drift[1]" problem, i.e., when the normal user behaviour changes. EMERALD (Porras & Neumann, 1997) uses statistical anomaly detection modules to monitor network traffic and a "resolver" to correlate alarms from misuse, and anomaly detectors across an enterprise. While these systems all have some degree of success, they were developed for a particular kind of environment. The fundamental question of "how to build and evaluate anomaly detection model in general" has not been adequately addressed. As a result, the approaches developed in these studies may not be applicable to other environments.

Researchers have begun to develop principles and theories for intrusion detection. Axelsson (2000) pointed out that the established field of detection and estimation theory bears similarities with the IDS domain. For example, the subject of an anomaly detection model corresponds to the "signal source" in detection and estimation theory, the auditing mechanism corresponds to "signal transmission[1]", the audit data corresponds to the "observation space", and in both cases, the task is to derive detection rules. Therefore, the results from detection and estimation, which have been found applicable to a wide range of problems, may be used in the IDS domain. One of the key findings by Axelsson is that when building a detection model, both anomaly and intrusion data is needed to ensure detection performance. In previous work Lee *et al*. (1999) showed that using labelled training dataset with normal and intrusion connections, we can build highly effective classifier for intrusion detention. However, in practice, it is difficult to obtain intrusion data. In this work, we therefore focus on how to build anomaly detection models when only normal data is available for training. Another key finding by Axelsson is that a detection model should be optimized for some utility function, not necessarily statistical accuracy, and instead could be some definition of cost function. We have independently begun to address how to build cost-sensitive IDS, i.e. IDS that provides the best-valued protection (Lee *et al*., 2000).

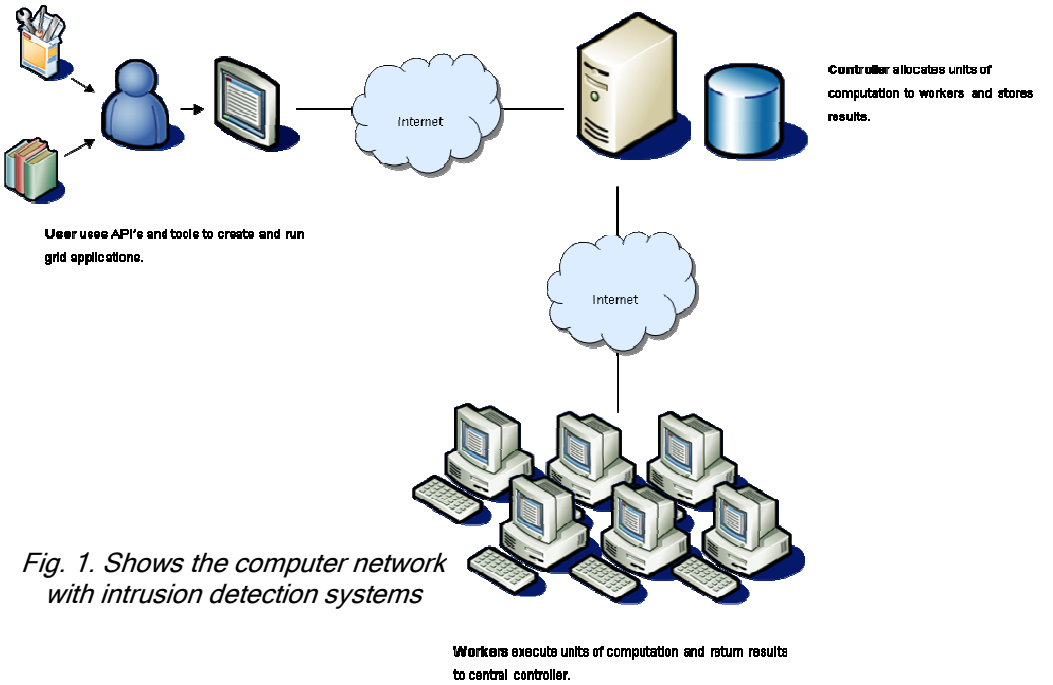


Fig. 1. Shows the computer network with intrusion detection systems

The most related work is by Maxion & Tan (2000), where the relationship between data set and detection performance of anomaly detection model was studied. The study focused on sequence data, and hence rule set is defined as intrusion activities. The key result from experiments on synthetic data is that when an anomaly detection model is tested on data with a range of datasets with varying regularity values, the detection

Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

performance also varies. This suggests that the current practice of deploying a particular anomaly detection system across different environments should be reconsidered. Our study here confirmed this finding that we showed that the expected detection performance can be attained only when the deriving rules of the training and testing datasets are small. Our study is more extensive because we used real system and network audit data in our case studies, but more importantly, we defined more Artificial Intelligent tools and showed how to use them to build anomaly detection models.

Case Studies

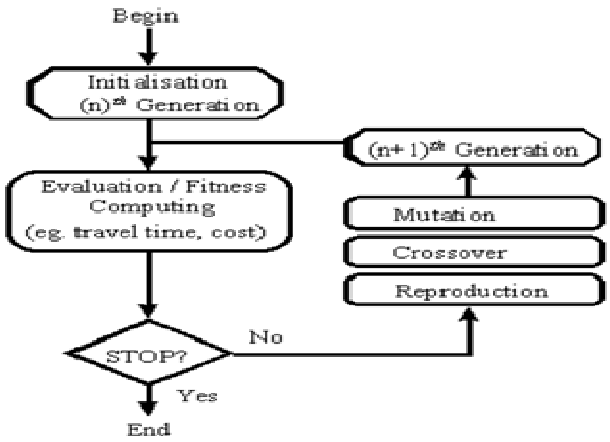
In this section, we describe our experiments on the University of Pune (Uop) *http* and *icmp* system call data, Computer Science Lab (DARPA Evaluation) http and icmp data, and Computer Science Lab *tcpdump* data, to show how to use the Artificial Intelligent tools AI to build anomaly detection models. These case studies are presented in the order of simpler to more complex in high terms of the audit data used. With the UoP system call data, we demonstrate how to use NNs to determine the appropriate data set used for sequencing the system calls to construct anomaly detection models. With the Computer Science Lab data, we show how to use classified correct to determine whether detecting the intrusive activities, e.g., Attack Signature, is likely to improve detection performance. With the Computer Science Lab *tcpdump* data, we show how to use test data to partition the network data into more regular subsets, and how to use rule set to determine the attack signature by which temporal and statistical features can be computed and included in the anomaly detection models.

Genetic Algorithm

Genetic algorithms (Crosbie & Spafford, 1995; Pohlheim, 2005) employ metaphor from biology and genetics to iteratively evolve a population of initial individuals to a population of high quality individuals, where each individual represents a solution of the problem to be solved and is composed of a fixed number of genes. The number of possible values of each gene is called the cardinality of the gene.

Fig. 2 illustrates the operation of a genetic algorithm. The operation starts from an initial population of randomly generated individuals. Then the population is evolved for a number of generations and the qualities of the individuals are gradually improved. During each generation, three basic genetic operators are sequentially applied to each individual with certain probabilities like selection, cross over, and mutation. First, the numbers of best-fit individuals are selected based on a user-defined fitness function. The remaining individuals are selected and paired with each other. Each individual pair produces one offspring by partially exchanging their genes around one or more randomly selected crossing points. At the end, a certain number of individuals are selected and the mutation operations are applied. When a GA is used for problem-solving, three factors will have impact on the effectiveness of the algorithm, they are: 1) the selection of fitness function; 2) the representation of individuals; and 3) the values of the GA parameters.

Fig.2. Apache2R-a-Deny (temporary/administrative)



Description

The Apache2 attack is a denial of service attack against an apache web server where a client sends a request with many http headers. If the server receives many of these requests it will slow down, and may eventually crash (Ghosh & Schwartzbard, 1999).

Simulation Details

This exploit was adapted from C code originally posted to the bugtraq mailing list. A C-shell wrapper was also created which executes the apache2 C program in a loop until the server being attacked is no longer responsive. As soon as the attack was launched the load average (as reported by the "top" program) of the victim server jumped to 5 or more. As more and more requests were submitted to the web server the memory usage and load average of the victim continued to climb until eventually the *httpd* daemon ran out of memory and crashed (shown in Fig.3). At this point the server no longer responded to http requests and the *httpd* daemon needed to be restarted by the super user for service to be restored.

Attack Signature

Every http request submitted as part of this exploit contains many http headers. Although the exact number and value of these headers could be varied by an attacker, the particular version of the exploit which was used in the 1998 DARPA evaluation sent http GET requests with the header "User-Agent: sioux\r\n" repeated 10000 times in each request. The actual content of the header is not important for the exploit– the exploit is only dependent on the fact that http request contains *many* headers. A typical *http* request contains twenty or fewer headers, so the 10000 headers used by this exploit are quite anomalous.

Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

Smurf R-a-Deny(Temporary)

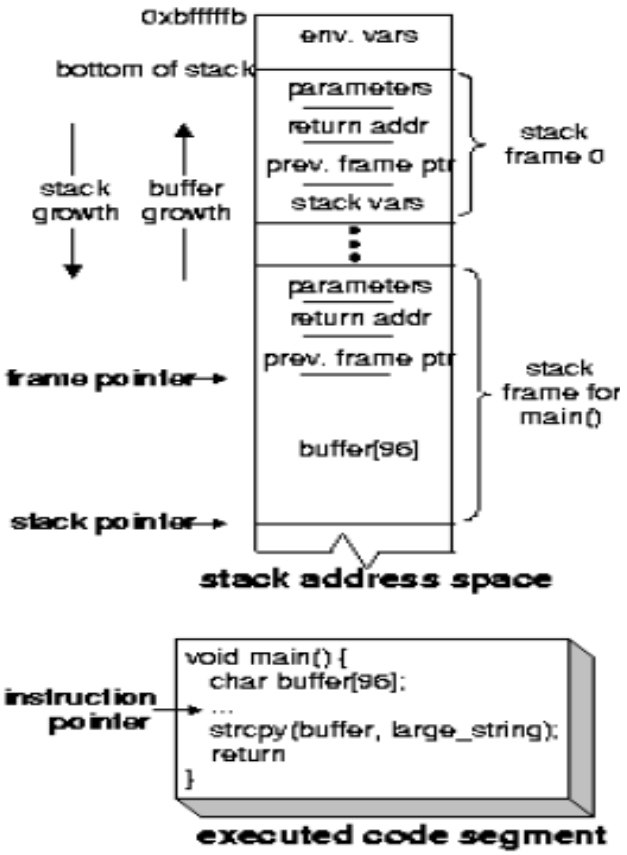
Description

In the "smurf' attack, attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to create a denial-of-service attack. There are three parties in these attacks: the attacker, the intermediary, and the victim (note that the intermediary can also be a victim) (Haystack, 1988). The attacker sends ICMP "echo request" packets to the broadcast address (xxx.xxx.xxx.255) of many subnets with the source address spoofed to be that of the intended victim. Any machines that are listening on these subnets will respond by sending ICMP "echo reply" packets to the victim. The smurf attack is effective because the attacker is able to use broadcast addresses to amplify what would otherwise be a rather innocuous ping flood. In the best case (from an attacker's point of view), the attacker can flood a victim with a volume of packets 255 times as great in magnitude as the attacker would be able to achieve without such amplification. This amplification effect is illustrated by Fig.4. The attacking machine (located on the left of the figure) sends a single spoofed packet to the broadcast address of some network, and every machine that is located on that network responds by sending a packet to the victim machine. Because there can be as many as 255 machines on an Ethernet segment, the attacker can use this amplification to generate a flood of ping packets 255 times as great in size (in the best case) as would otherwise be possible. This figure is a simplification of the smurf attack. In an actual attack, the attacker sends a *stream* of icmp "ECHO" requests to the broadcast address of *many* subnets, resulting in a large, continuous stream of "ECHO" replies that flood the victim.

Simulation Details

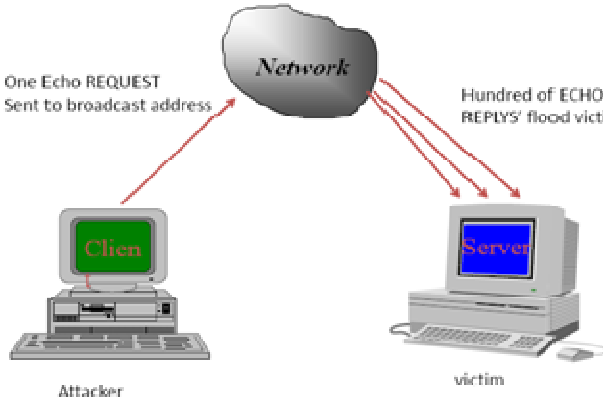
Because the simulation network for the 1998 DARPA evaluation has a flat network topology with only two physical subnets, the smurf attack as described above could not be implemented on the simulation network. For this reason, the "smurfsim" program was developed to recreate the observable effects of a smurf attack. Smurfsim uses the raw socket API to construct ICMP packets with forged source addresses. Smurfsim takes as parameters the IP address of the victim, the number of packets to send, the average percentage of hosts on a subnet that are alive, and a comma-separated list of subnets. The program then randomly constructs a list of hosts that are alive on each of the subnets in the comma-separated list and starts sending "echo reply" packets to the victim, that have been spoofed to look like they originating from the hosts in the list. This behavior is exactly what would occur if an attacker had performed an actual Smurf attack in which "echo request" packets (with the source address spoofed to be that of the victim machine) were sent to the broadcast address of each subnet given in the parameter list. Several different simulated Smurf attacks were included in the evaluation data. In the most extreme case, the smurfsim program was used to simulate a sIn all, this particular attack instance generated over tmurf attack that generated traffic from 100 subnets for a period of one hour. During this period of time the entire simulation network was unresponsive and other network sessions (such as normal users trying to send e-mail, etc) would time out before they could be completed. wo gigabytes of network packets.

Fig.3.The overflow attack



Attack Signature

Fig.4 . The Smurf attack diagram



Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

The Smurf attack can be identified by an intrusion detection system that notices that there are a large number of "echo replies" being sent to a particular victim machine from many different places, but no "echo requests" originating from the victim machine.

GA Applied to ID

By analyzing the dataset, rules will be generated in the rule set. These rules will be in the form of an 'if then' format as follows.

if {condition} then {act}

The condition using this format refers to the attributes in the rule set that forms a network connection in the dataset. The condition will result in a 'true' or 'false'. The attack name will be specified only if the condition is true

If Number of "hot" indicators <=0.0 and Number of connection to the same host as the connection in the last two seconds <= 500.82 and % of connection that have "REJ" errors >0.21 and <= 0.01 and Number of connections to host <= 41.2 and 112.3
Then SMURF attack

If duration > 265 and protocol = tcp and Service = http ˇ https and Src_bytes>265616 ˇ <=283618 and dest_bytes = 0 and hot>0 ˇ <=2 and is_gust_login = 1 and Number of service requested of host <= 25.83
Then Apache attack

Since the GA has to use such rules to detect intrusions, such rules in the rule set will be codified to the GA format. Each rule will be represented in a GA format. The first part of the GA will act as a search algorithm. It will match the rules with any anomalous connections that occur the network to detect an intrusion. Each rule will carry values for the intrusions that they have detected and a value for the intrusion. Each rule will carry values for a false alarm that the rule produces. The second part of the GA is the fitness function. The fitness function F determines whether a rule is good or bad. F is calculated for each rule using the support confidence framework.

Support = |A and B | / N

Confidence = |A and B | / |A|

Fitness = t1 * support + t2 * confidence

Where

N is the total number of records.

| A | stands for the number of network connections Matching the condition A.

| A and B | is the number of records that matches the rule.

t1 and t2 are the thresholds to balance the two terms.

Performance evaluation of proposed rules on the KDD data set

These rules used reduced features from the KDD data set and tested on the KDD training set to observe their performance with respect to detection, false alarm and missed alarm rates. If the rules are well formed, then the detection rate is expected to be high while concurrently achieving low false alarm and missed alarm rates. For each chromosome in the population, the number of network connections and the number of connection that matches the condition is initialized to zero. For each record in the training set, if the record matches the chromosome update the network connection by 1 and if the record only matches the condition part, then update that value A by 1. Then calculate the fitness of each rule and select the best fit rules into new population. Then apply the crossover and mutation operators to each rule in the new population. Finally, decide whether to terminate the training process or to enter the next generation to continue.

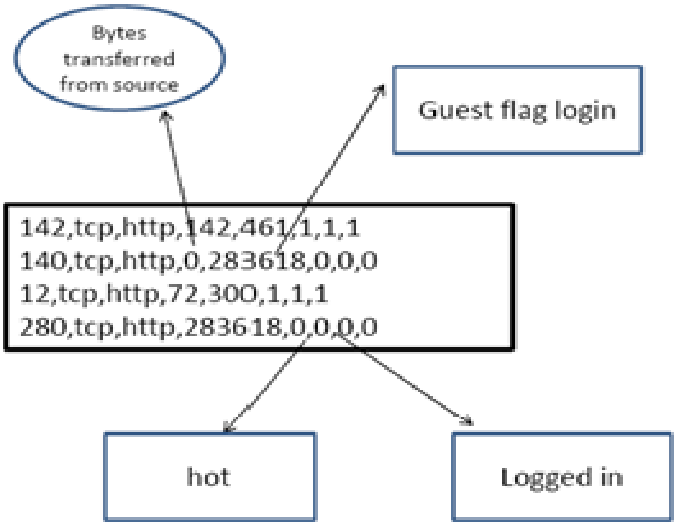
Record Type	Training Set	Testing Set
Normal	96.20%	93.80%
Smurf	90.60%	67.30%
Apache2	92.70%	76.60%

The experimental result (Table) shows that the proposed method yielded good detection rates when using the generated rules to classify the training data itself. Fig. 6 shows missed alarm example of apache2 attack on the KDD data set.

Conclusions

This paper describes novel work in using Genetic Algorithm for detecting misuse of programs. Two important observations result from this work. First, the results demonstrate how misuse of programs can be detected using GA. The results indicate that training set with significantly generated data lend to the best performance in detection of possible novel misuse attempts an area in which most misuse detection ap-

Fig.6. Missed alarm example of apache2 attack on the KDD data set



Indian Journal of Science and Technology Vol.2 No 2 (Feb. 2009) ISSN: 0974- 6846

proaches are weak, contrarily the results with testing set are shown the low performance due to include the novel attacks in the testing set that generates high false alarm, Furthermore, the results show the benefit of applying misused detection at the process level such that abnormal process behaviour can be detected irrespective of individual users' behaviour. This approach abstracts out users' individual behaviour and allows misused detection against the set of all users' behaviour; further research will attempt to identify important attack features for anomalous use of programs. These results may be important in not only detecting attempted misuse of programs, but also erroneous program behaviour.

References

1. Andersoii D, Frivold T and Valdes A (1995) Next-generation intrusion detection expert system (NTDES): A summary. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California.
2. Axelsson S (2000) A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical report, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden.
3. Crosbie M and Spafford E (1995) Applying genetic Programming to Intrusion Detection. In: Proc. of the AAAI Fall Symposium.
4. Forrest S, Hofmeyr SA, Somayaji A and Longstaff TA (1996) A sense of self for Unix processes. In: Proc. 1996 IEEE Symposium on Security and Privacy, Los Alamitos, CA. IEEE Computer Society Press. pp: 120-128.
5. Ghosh AK and Schwartzbard A (1999) A study in using neural networks for anomaly and misuse detection. In Proc. of the 8th USENIX Security Symposium, August 1999.
6. Haystack S. E. Smaha (1988) An intrusion detection system. In: Proc. of the IEEE Fourth Aerospace Computer Security Applications Conference.
7. Ilgun K, Kemmerer RA and Porras PA (1995) State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3), 181-199.
8. Kumar S and Spafford EH (1995) A software architecture to support misuse intrusion detection. In: Proc. of the 18th National Information Security Conference. pp: 194-204.
9. Lane T and Brodley CE (1998) Temporal sequence learning and data reduction for anomaly detection. In: Proceedings of 5th ACM Conference on Computer S Communication Security.
10. Lee W and Stolfo SJ (1998) Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, January 1998.
11. Lee W, Stolfo S- J and Mok KW (1999) A data mining framework for building intrusion detection models. In: Proc. of the 1999 IEEE Symposium on Security and Privacy, May 1999.
12. Lee W, Wei Fan, Matt Miller, Sal Stolfo and Erez Zadok (2000) Toward cost-sensitive modeling for intrusion detection and response. In 1st ACM Workshop on Intrusion Detection Systems.
13. Lippmann R, Pried D, Graf I, Hames J, Kendall K, McClung D, Weber D, Webster S, Wyschogrod D, Cunninghan R and Zissman M (2000) Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In: Proc. of the 2000 DARPA Information Survivability Conference and Exposition.
14. Lunt T (1993) Detecting intruders in computer systems. In Proc. of the 1993 Conference on Auditing and Computer Technology.
15. Lunt T, Tamaru A, Gilham F, Jagannathan R, Neumann P, Javitz H, Valdes A and Garvey T (1992) A real-time intrusion detection expert system (IDES) - final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California.
16. Maxion RA and Tan KMC (2000) Benchmarking anomaly-based detection systems. In: Proc. of the 1st International Conference on Dependable Systems & Networks.
17. Pohlheim H (2005) Genetic and Evolutionary Algorithms: Principles Methods and Algorithms. hup://ww\v.gearbx.cpm/docu/mdex.luml. January.
18. Porras PA and Neumann PG (1997) EMERALD: Event monitoring enabling responses to anomalous live disturbances. In: National Information Systems Security Conference, Baltimore MD.
19. Tacobson V, Leres C and McCanne S (1989) tcpdump. Available via anonymous ftp to ftp.ee.lbl.gov.
20. Warrender C, Forrest S and Pearl Mutter B (1999) Detecting intrusions using system calls: Alternative data models. In: Proc. of the 1999 IEEE Symposium on Security and Privacy.