Formulation of a Heuristic Rule for Misuse and Anomaly Detection for U2R Attacks in Solaris™ Operating System Environment

Maheshkumar Sabhnani
EECS Dept, University of Toledo
Toledo, Ohio 43606 USA

Gursel Serpen
EECS Dept, University of Toledo
Toledo, Ohio 43606 USA

Abstract

This paper proposes a heuristic rule for detection of user-to-root (U2R) attacks against Solaris™ operating system. Relevant features for developing heuristic rules were manually mined using Solaris™ Basic Security Module audit data. The proposed rule was tested on both DARPA 1998 and 1999 intrusion detection datasets. Results show that all user-to-root attacks exploiting the suid program were detected with 100% probability and with zero false alarms. The rule can detect both successful and unsuccessful U2R attempts against the Solaris™ operating system. The proposed rule is general enough to detect any U2R attack that leverages the buffer overflow technique. Empirical results indicate that the rule also detected novel user-to-root attacks in DARPA 1998 intrusion detection dataset. Hence the rule has the potential and can be used for anomaly detection.

Keywords: Intrusion detection, User-to-root attack, misuse, anamoly, rule-based system, heuristic

1. Introduction

The process in which a normal user gains root (super-user) privileges through an illegal transition is defined as a user-to-root (U2R) attack. To execute a U2R attack, the intruder first somehow gains local access (as a normal user) to the victim machine through one of the password sniffing, dictionary attack, or social engineering, etc. techniques. The intruder then exploits some vulnerability or bug associated with the operating system environment of the machine under attack to perform the transition from user to root level. After acquiring root privileges, the intruder has full control on the victim machine to install backdoor entries for future exploitation, change system files to collect information, and other potentially very damaging actions. Only one process allows legal transition from user to root level in Solaris™ environment, which is the 'su' utility. If a user gains root shell without executing the 'su' command, this means an intrusive U2R transition has taken place [1]. As the damage caused by U2R attacks can be potentially very destructive, it is very important that 100% detection is achieved for attacks in this category.

Common techniques used to execute an illegal U2R transition within Solaris™ operating system environments involve buffer overflow attacks [1]. The attacker intentionally overflows some of the internal program buffers so that they are written onto the system stack. This results in writing over the further instructions that are yet to be executed by the system. By carefully manipulating the data that is written onto the stack, the attacker can execute arbitrary instructions on the system, and gain root access. Although the buffer overflow situation can be easily monitored within the program, some common utility programs like *eject*, *ffbconfig*, *fdformat*, and *ps* were susceptible to these attacks back in 1998. Current versions of these programs are not vulnerable to buffer overflow attacks as the operating systems have been patched since that date. There are many other root-owned utility programs that can be executed by normal users on a machine. It is possible that attackers could exploit similar root-owned programs with the same or a similar bug in future. Hence the immediate need arises for intrusion detection systems with anomaly detection capability to be developed to counter these attacks as well. Such an effort is being made to address this need in this paper: an improved heuristic rule with anomaly detection capability, in a limited context, will be developed for detection of U2R attacks in Solaris™ environment.

Section 2 will present the discussion on audit data file or log utility generated by the Basic Security Module (BSM). Next, 1998 and 1999 Defense Advanced Research Projects Agency (DARPA) BSM datasets (network, control files used for BSM, etc.) will also be briefly discussed in Section 3. Literature survey on detecting U2R attacks against the Solaris™ operating system using the BSM will be presented on Section 4. Section 5 discusses the formation of proposed heuristic rule for U2R attacks. Section 6 presents the performance and results obtained on DARPA 1998 and 1999 datasets. The same section shall also discuss the comparative performance of the proposed rule with others in literature.

2. Basic security module of Solaris™ operating system

Basic Security Module (BSM) is a software utility available with the Solaris™ operating system to assist in collecting security-related information about various events executed on the machine [2]. There are more than 200 kernel and user events that the system executes including *execve, mmap, open, close, munmap, stat,* etc. Each event is associated with various parameters like who executed, execution time, paths, arguments, return values, errors generated, and which process and session was responsible among others. This critical system information is collected by the BSM in chronological order. Further processing is left to the system administrator who can extract relevant information as needed. The data is stored in binary format in a disk file. Two operating system utilities that can process this binary information are 'auditreduce' and 'praudit'. While 'auditreduce' extracts specific information like records during specific time periods, or records of a specific event, 'praudit' converts binary file into a human readable format.

For each event executed on the system, BSM generates one record that contains all the information related to that event. Hence the BSM file on a host is a set of records that indicate all events generated. Audit control files assist in how much information needs to be stored, which users are to be logged, and where the files need to be stored. BSM filenames are automatically generated that help identify the host and the time the file was created and terminated. For example, a typical audit record structure consists of *header* token (marks the beginning of record, event id, event name, and execution date and time), *arg* token (various arguments passed to the event system call), *data* token (other relevant information like path information, attributes of the file accessed), *subject* token (who executed the event, session id, process id, and machine id), *return* token (value returned by the event), and *trailer* token (marks the end of record). A sample EXECVE event BSM record is shown in Figure 1. Further details about the various tokens and fields can be obtained from the relevant Solaris™ system manual [2].

3. DARPA 1998 and 1999 intrusion detection evaluation program

In 1998, Defense Advanced Research Projects Agency (DARPA) funded an intrusion detection evaluation program at Lincoln Laboratories of the Massachusetts Institute of Technology [3]. In this program, various attack scenarios were conducted on a simulated network that consisted of Linux, Sun, and Solaris™ operating system victims. Attacks belonging to various categories such as Probing, DoS, U2R, and R2L were simulated, and network and host data was collected. The data sets also consisted of normal network traffic. The training dataset consisted of seven weeks of data while the testing dataset consisted of two weeks of data. Three kinds of data was collected: one by sniffers connected at both inside and outside the network, second was the BSM data on Solaris™ victims, and third was file system dumps compiled at the end of each day.

A total of four U2R attacks were executed against the Solaris™ machines during DARPA 1998 intrusion evaluation program: *eject*, *fdformat*, *ffbconfig*, and *ps* attacks. The 'ps' attack was present only in testing data set to help build anomaly detection systems. All these attacks belong to subcategory buffer overflow attacks exploiting root-owned programs accessible to normal users on the system. New attacks were added in both training and testing phases of the DARPA 1999 intrusion detection evaluation program. Also a new victim operating system, Windows NT [4], was introduced. Each day's BSM audit file consisted of hundreds of sessions and thousands of processes executed and the typical file size was on average 100 MB. These two datasets are publicly accessible and can be downloaded [5], [6].

4. Literature survey: Detecting U2R attacks against Solaris™ using BSM

Ghosh and Michael [7] implemented three machine learning algorithms, which included the Elman recurrent neural network [8], string transducer [9], and state tester [10], to model the normal behavior of programs on a Solaris™ platform. The goal was to implement anomaly detection and hence occurrence of an intrusive activity. Each algorithm used sequences of BSM events generated by a program during its execution. Example system programs that were studied included *admintool*, *ping*, *allocate*, *auditd*, *eject*, *ffbconfig*, and *su*. Dataset used was from the DARPA 1999 program. All three algorithms were tested on two attack categories U2R and R2L. The Elman net performed the best with 100% detection with three false alarms per day while string transducer performed next best with 100% detection at five false alarms per day. State tester detected all user-to-root attacks at around nine false alarms per day.

```
header,167,2,execve(2),,Thu Jul 02 08:02:01 1998, + 232158572 msec
path,/usr/lib/sendmail
attribute,104551,root,bin,8388614,96918,0
exec_args,3,
/usr/lib/sendmail,-oi,charlae@alpha.apple.edu
subject,2066,root,rjm,2066,rjm,317,309,24 0 172.16.114.168
return,success,0
trailer,167
```

Figure 1. Sample BSM record for EXECVE event

Eskin [11] used a statistical classification technique to estimate a probability distribution over the data and applied a statistical test to detect the anomalies. No training was required over the normal data. This anomaly detection technique was used to detect intrusions based on the analysis of process system calls. Two datasets containing system calls data were analyzed. First data was obtained from the BSM portion of DARPA 1999 evaluation. Second dataset was obtained from Stephanie Forrest's group at the University of New Mexico [12]. This method was compared with two popular baseline methods *tslide* and *slide* [12]. The results showed that probability modeling technique performed much better than the baseline techniques for programs that had less than 5% of intrusive data present (like *ps*, *xlock*, *named*, and *ftpd*). A 100% detection rate was achieved with a false alarm rate of less than 0.02% over the total number of sessions. This technique is also discussed in [13].

Emran and Ye [14] developed a multivariate statistical-based anomaly detection algorithm, namely Canberra technique. Datasets used for testing this technique were obtained from DARPA 1998 evaluation program and the Information Systems and Assurance Laboratory of Arizona State University. However only five minutes of BSM data rather than the entire BSM data was employed. Data was in the form of BSM events representing system calls. Performance of the Canberra technique was assessed on number of attacks including password guessing, suspicious program usage, and port scanning. Results showed that the Canberra technique performed very well for both anomaly and misuse detection only when normal records are widely separated in the feature space. In situations where attacks are intermixed with normal sessions, an unacceptably high level of false alarms, in the neighborhood of 30%, was generated.

Lindqvist and Porras [15] developed a signature-analysis engine for computer and network misuse detection using a forward-chaining rule-based expert system, which was implemented using the Production-Based Expert System Toolset (P-BEST). Rules were generated for many attacks, specifically SYN flooding and buffer overflow attacks. BSM data was used to create rules for buffer overflow attacks and TCP data was used for SYN flooding attacks. A single rule was specified using the P-BEST language. A 100% detection rate was achieved for misuse detection of buffer overflow attacks with no false alarms. Authors report that the rule used for buffer overflow attacks is fast enough to be used for real-time detection of these attacks. The processing time is only a few minutes for a typical day's BSM data for Solaris™. The rule failed to show any anomaly detection capability when tested on DARPA 1998 testing dataset: it failed to detect the attack against the *ps* program.

Lee and Stolfo [16] also used BSM data collected during the 1998 DARPA evaluation program to build a misuse detection model. They applied data mining techniques to extract session and event features from the BSM data in the form of records. These records were then labeled as normal or intrusive according the data information provided by the 1998 DARPA evaluation program. Finally, the RIPPER utility was applied to generate rules [17]. Performance of their algorithm on U2R attacks was slightly more than 80% detection at 0% false alarm rate.

The literature survey shows that mainly rule based systems that leveraged BSM information demonstrated desirable performance characteristics for detecting U2R attacks against the Solaris™ operating system. All these systems were built for misuse detection, and a 100% detection rate was achieved with very low false alarm rates for known attacks. However no mention of anomaly detection capability is made for these systems. Also malicious unsuccessful attempts to execute U2R attacks must also be treated as attacks. This discussion points at possible improvements for a U2R attack detection system that utilizes BSM information in Solaris™ platforms. Therefore it is desirable that a heuristic rule based system should demonstrate a performance with 100% detection and 0% false alarm rates against U2R attacks, while also incorporating some anomaly detection capability. The rule should not only detect all successful attacks, but also detect unsuccessful U2R attack attempts positioned to exploit system security loopholes. The proposed rule should also be conceived to be general enough to detect any U2R attack that leverages the buffer overflow technique: this requirement has the potential to induce anomaly detection capability although it is likely to be somewhat limited in scope. Furthermore the rule should

not be computationally complex and require minimal processing time, and hence offer the potential to be used for real-time detection of U2R attacks since U2R attacks are potentially very dangerous.

This paper proposes a heuristic rule to detect illegal transitions from user level to root (super-user) level through leveraging programs having suid bit set within Solaris™ operating system environments. The aim is to develop a rule such that a 100% detection rate, 0% false alarm rate, and anomaly detection are achieved. The proposed heuristic rule will be targeted for use by a host-based intrusion detection system against user-to-root (U2R) attacks. Domain knowledge pertaining to U2R attacks and Basic Security Module (BSM) information will be used in formulating the heuristic rule. Specifically, an audit log of all system events occurring on a host, which is generated by the BSM utility, shall be used towards developing the rule. The proposed heuristic rule will be conceived in a generic format to facilitate a possible port to other operating systems with tools similar to the BSM. This rule will be tested on the DARPA 1998 and 1999 intrusion detection datasets.

5. Heuristic rule formulation for U2R attacks

The literature suggests that expert systems using heuristic rules perform best on U2R attacks using the BSM audit data. This section explains rule formulation process for U2R attacks using the buffer overflow technique. First BSM audit data was manually mined to extract features that can be analyzed to detect U2R attacks. Then using the DARPA 1998 and 1999 datasets, thresholds were set using statistical analysis of datasets so that U2R attacks can be detected.

Generally, exploiting an internal buffer of a system program is the main source for a U2R attack. Such a program that is amenable to exploitation through its internal buffer is executed at the shell prompt by the attacker. There are around 229 BSM events available for audit within the Solaris™ operating system. Only a few are responsible directly for user level commands issued to the kernel from the shell prompt. It is important to find those event(s) that correspond to the user commands. It was observed that all the commands executed by the user were audited contained EXECVE BSM event as the starting event during logging. This event is responsible for storing command name and various parameters passed during execution of a process. The event then maps the command to a system call, which executes the requested process. Hence any command executed by a user can be directly identified by analyzing the various EXECVE events occurring in the BSM audit file.

All U2R attacks, executed on Solaris™ machines during DARPA 1998 evaluation program, belonged to buffer overflow category. These attacks were *eject*, *fdformat*, *ffbconfig*, and *ps*. Every program has a 'set-user-ID' (suid) bit associated with it. The owner of a program, who is the root (R), can set this bit so that when other users execute the program, they obtain privileges of R for the complete duration of the execution. Some root-owned system processes have this bit set because these programs can execute/call other child processes that cannot execute unless the parent process has root privileges. The potential problem is with having the suid bit set. Assume that user N executes an attack through a process owned by a user R having suid bit set. Now if N is able to stop the process before it finishes execution, N shall have privileges of R since suid bit was set. All buffer overflow attacks use this mechanism. Attacker executes a root-owned program with suid bit set and passes a large argument to it such that one or more internal buffers overflow. Whenever an internal buffer overflows by default, the system stack present in memory containing next instructions is overwritten by the overflowing data. Hence by carefully overwriting the system stack, an attacker can execute arbitrary instructions as a root since during the execution of program the attacker has root privileges. One of the common instructions is to generate a shell with root privileges. This dynamics is common for all buffer overflow attacks.

Once the common mechanism for leveraging a buffer overflow attack is defined, the next step is to identify those tokens in the BSM data that can help extract relevant information. The header token will indicate the total size (*length*) of event and hence suggest whether a large argument was passed to the program. It will also tell the time and date of execution (*time*), and EXECVE event. The 'path' token specifies which program was executed (*path*). The 'attribute' token points to whether suid bit (*suid*) is set and also indicates the owner of the process. The 'exec_args' token will designate how many (*arg_num*) and which arguments (*content*) are passed to the program. Subject token will indicate the effective user id (*euid*), the real user id (*ruid*), and the session number (*session*) for the current EXECVE event. Hence the rule to detect buffer overflow attack using these BSM features can be proposed as follows:

> "*A user can execute a U2R attack by passing a large argument to a root-owned program whose suid bit is set to 1.*"

The rule can be restated in BSM terminology as follows:

> "*An attempt for a U2R attack occurs when an EXECVE event with large 'length' and small number of arguments, as represented by 'arg_num', executes a root-owned program whose suid is set to 1.*"

Consequently, in a formal notation with BSM terminology, the rule can be restated as:

(event = EXECVE) ∧
(euid != ruid) ∧
(owner = "root") ∧
(suid = 1) ∧
(length > 400) ∧
(arg_num > 6) ⇒ Buffer overflow attack

The threshold for *length* was set to 400 by comparing normal EXECVE event sizes from the DARPA 1998 dataset. Only few non-attack records had *length* greater than 400 but those records required much larger value for parameter *args_num*. An example program is *sendmail* which requires a large list of email addresses. The observation made on the DARPA 1998 dataset for any normal EXECVE event was that if the *args_num* is less than 6, the *length* cannot be greater than 400.

The proposed rule detects a user attempt to pass a large argument to a root-owned program. It is independent of the program name, size of the arguments passed, and contents of the suspicious argument. Hence any user attempt to pass a large argument to a root-owned program will be detected by this rule. This shows the anomaly detection capability of the proposed rule. Other U2R attacks that are not present in the DARPA 1998 training dataset might have different argument contents and may target some other root-owned program but the signature will be the same and hence the rule should be able to detect the attack. This is an important contribution of the proposed rule. In future, if the rule generates false alarms, then only two parameters need to be adjusted: *length* and *arg_num*. These are tunable parameters and can be changed according to the organizational needs to reduce false alarms. Currently these parameters were set by using statistical analysis of DARPA 1998 datasets.

It is further relevant to note that the proposed rule detects whenever the user passes a large argument to a root-owned program. In patched versions of the Solaris operating system environment, requests to execute most root-owned programs, which were previously vulnerable to buffer overflow attacks prior to those patches, with a large argument are simply ignored and lead to generation of an error message by the operating system. Hence if the root-owned program is not vulnerable to the buffer overflow attack then it will result in an unsuccessful attempt for an U2R attack. The proposed rule will still detect this attempt as the rule just observes passing a large argument. Hence the rule shall detect both successful and unsuccessful buffer overflow attempts against Solaris™ programs. Since the rule detects the attack before the attacker gains root privileges, the rule cannot differentiate between successful and unsuccessful U2R attempts. Additional rules may be needed to detect whether the U2R attack was successful or not by monitoring the change in privilege levels.

6. Performance of proposed heuristic rule

The proposed heuristic rule has been tested on DARPA 1998 and 1999 datasets. The goal was to find out whether a buffer overflow attack has been attempted during a session as captured in the BSM audit file. Generally around 200 sessions were present in a typical day's audit file. The script to check the proposed rule took less than five minutes to process each day's BSM audit file using a computing platform with a 500 MHz Intel processor, 512 MB RAM, and running the Solaris™ operating system.

The proposed rule detected all buffer overflow attacks both in DARPA 1998 and DARPA 1999 datasets. Noting that the proposed rule was created by analyzing DARPA 1998 training dataset only, all attacks were successfully detected in DARPA 1999 datasets as well. There were no false alarms or missed alarms generated: 100% detection and 0% false alarm rates were achieved. Also no unsuccessful U2R attempts are reported in DARPA 1998 datasets. Hence the proposed rule detected all successful U2R attacks against the Solaris™ operating system. Furthermore, the rule did detect a new attack present in DARPA 1998 test data set against the 'ps' program, which was not present in DARPA 1998 training data set. Hence the proposed rule has anomaly detection capabilities even though it might be in a limited context.

During the DARPA 1998 program, four participants (UCSB, IOWA, COLUMBIA and EMERALD) took part in the intrusion detection system evaluation. The results of this evaluation program are discussed in [18]. Only one of the four participants was able to detect all buffer overflow attacks in the test data but with around 100 false alarms per day, which is an unacceptably high number for all practical purposes. Endler [19] proposed a rule to detect buffer overflow attacks by checking whether a user is executing a program having suid bit reset with root privileges. Endler's rule demonstrated a 100% detection rate for U2R attacks but generated many false alarms: there were more than 50 false alarms on the first day of DARPA 1998 training data set. Therefore, Endler's rule fails to compare favorably with the proposed rule in this paper, which detected all buffer overflow attacks with no false alarms both in DARPA 1998 training and testing datasets.

Lindquist [15] used a rule to detect U2R attacks, which was very similar to the proposed rule. They proposed the following:

(event = EXECVE) ∧
(euid != ruid) ∧
(content = "^\\") ∧
(length > 400) ⇒ Buffer overflow attack

Lindquist's rule checked whether the *exec_args* contained the string "^\\", which was present in all U2R attacks against Solaris™ machines. We have written a nawk script, which implemented this rule, to evaluate its performance on the same data sets the proposed rule was assessed. Lindquist's rule detected all attacks in DARPA 1998 training data set. However the same rule failed to detect "new" attacks, which are present in testing but not in training data set (against *ps* program). Hence the rule failed to demonstrate anomaly detection capability. The reason for that was buffer overflow attack through the *ps* program does not contain "^\\" in the *exec_args*. The rule proposed in our paper is capable of anomaly detection since it detected new attacks present in DARPA 1998 testing dataset only. Also Lindquist's rule does not check whether the root owns the program being executed. If not, then there is no U2R attack executed by the attacker: it possibly points at a transition from one user to another user, which is not that dangerous as compared to a U2R transition. Furthermore, Lindquist's rule detected around 80% of attacks in DARPA 1998 testing dataset with no false alarms.

Ghosh [7] used three techniques, Elman neural networks, string transducer, and state tester to detect U2R attacks. All three techniques employed machine learning algorithms, where the normal process behavior was learned and then anomalous behavior was detected to conclude that attack has taken place. All three techniques could detect U2R attacks with 100% detection rate but at the expense of 3 to 5 false alarms per day, which is relatively high for practical purposes.

Lee and Stolfo [16] used data mining techniques on DARPA 1998 training data set to extract features from the BSM data. They employed rule discovery algorithm RIPPER on extracted features to generate rules. The rule generated by RIPPER to detect buffer overflow attack is as follows:

> "*If a shell is executed in the suid set state, then this is a buffer overflow attack.*"

This rule could detect only 80% attacks in the DARPA 1998 testing data set and failed to show anomaly detection.

In summary, the proposed rule can respond to all successful and unsuccessful buffer overflow attacks against Solaris™ operating system present in the DARPA 1998 and 1999 datasets with a 100% detection rate with no false alarms. No other detection model reported in literature could achieve this performance. Results show that the proposed rule could detect novel unknown attacks and hence has the capability for some level of anomaly detection. A potential limitation, which can be easily addressed with a second enhancing heuristic rule, is that the rule cannot differentiate between successful and unsuccessful buffer overflow attempts. Furthermore, it might be necessary to change the thresholds for *length* and *arg_num* parameters based on the organizational requirements to reduce false alarms generated if any.

7. Conclusions

A heuristic rule was proposed to detect buffer overflow attacks against Solaris™ machines. The proposed rule performed very well on both DARPA 1998 and DARPA 1999 datasets with 100% detection and 0% false alarm rates while also demonstrating potential (although limited) anomaly detection capability. The proposed rule is sufficiently generic to detect all U2R attacks that leverage the buffer overflow mechanism against Solaris™ machines. The proposed rule is not complex computationally, and the features required can be easily observed through the BSM audit file. The proposed rule can be used to detect U2R attacks in real-time due to its minimal computational cost. The rule detects both successful and unsuccessful buffer overflow attempts against Solaris™ though the rule cannot differentiate between the two. A second follow-up rule would be needed to detect whether the attacker gained root privileges or not.

Currently, the proposed rule was tested for off-line detection of U2R attacks although real-time execution of the rule does not present any noteworthy computational challenges. However, issues like reading BSM records at real-time, managing BSM files, generating automated responses to attack detection etc. are reserved for future studies and are not addressed in this paper.

8. References

[1] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis*, Massachusetts Institute of Technology, Boston, MA, 1998.

[2] SunSHIELD BSM Guide, http://docs.sun.com/db/-doc/806-1789, 1995, cited April 2003.

[3] R. P. Lippmann, I. Graf, D. Wyschogrod, S. E. Webster, D. J. Weber, and S. Gorton, "The 1998 DARPA/AFRL Off-Line Intrusion Detection Evaluation", *First International Workshop on Recent Advances in*

Intrusion Detection (RAID), Louvain-la-Neuve, Belgium, 1998.

[4] R. P. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation", *Computer Networks*, 2000, Vol. 34 (4), pp. 579-595.

[5] DARPA data set, 1998. http://www.ll.mit.edu/IST/-ideval/data/1998/1998_data_index.html, cited April 2003.

[6] DARPA data set, 1999. http://www.ll.mit.edu/IST/-ideval/data/1999/1999_data_index.html, cited April 2003.

[7] A. K. Ghosh, C. Michael, and M. Schatz, "A Real-Time Intrusion Detection System Based on Learning Program Behavior", *In Proceedings of Recent Advances in Intrusion Detection*, 2000, pp. 93-109.

[8] J. L. Elman, "Finding Structure in Time", *Journal of Cognitive Science*, 1990, Vol. 14, pp. 179-211.

[9] M. Mohri, "Finite-State Transducers in Language and Speech Processing", *Journal of Computational Linguistics*, 1997, Vol. 23 (2), pp. 269-311.

[10] A. P. Kosoresow, and S. A. Hofmeyr, "Intrusion Detection via System Call Traces", *Journal of IEEE Software*, September/October, 1997, Vol. 14 (5), pp. 35-42.

[11] E. Eskin, "Anomaly Detection over Noisy Data using Learned Probability Distributions", *In Proceedings of 17th International Conference on Machine Learning*, San Francisco, CA, 2000, pp. 255-262.

[12] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models", *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Oakland, CA, 1999, pp. 133-145.

[13] E. Eskin, M. Miller, Z. Zhong, G. Yi, W. Lee, S. Stolfo, "Adaptive Model Generation for Intrusion Detection Systems", *In Proceedings of the first ACMCCS Workshop on Intrusion Detection and Prevention*, Athens, Greece, 2000.

[14] S. M. Emran, and N. Ye, "Robustness of Canberra Metric in Computer Intrusion Detection", *Proceedings of the IEEE Workshop on Information Assurance and Security US Military Academy*, West Point, NY, 5-6 June, 2001, pp. 80-84.

[15] U. Lindqvist, and P. Porras, "Detecting Computer and Network Misuse through the Production-based Expert System Toolset (P-{BEST})", *IEEE Symposium on Security and Privacy*, 1999, pp. 146-161.

[16] W. Lee, and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", *ACM Transactions on Information and System Security*, November 2000, Vol. 3 (4), pp. 227-261.

[17] W. W. Cohen, "Fast effective rule induction", *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, Lake Tahoe, CA: Morgan Kaufmann, 1995, pp. 115-123.

[18] R. P. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. Kendall, S. E. Webster, and M. A. Zissman, "Results of the DARPA 1998 Offline Intrusion Detection Evaluation", *RAID 1999 Conference*, West Lafayette, Indiana, Sept. 7-9, 1999.

[19] D. Endler, "Intrusion Detection Applying Machine Learning to Solaris Audit Data", *Proceedings of the 1998 Annual Computer Security Applications Conference*, Los Alamitos, CA, Dec. 1998, pp. 268-279.